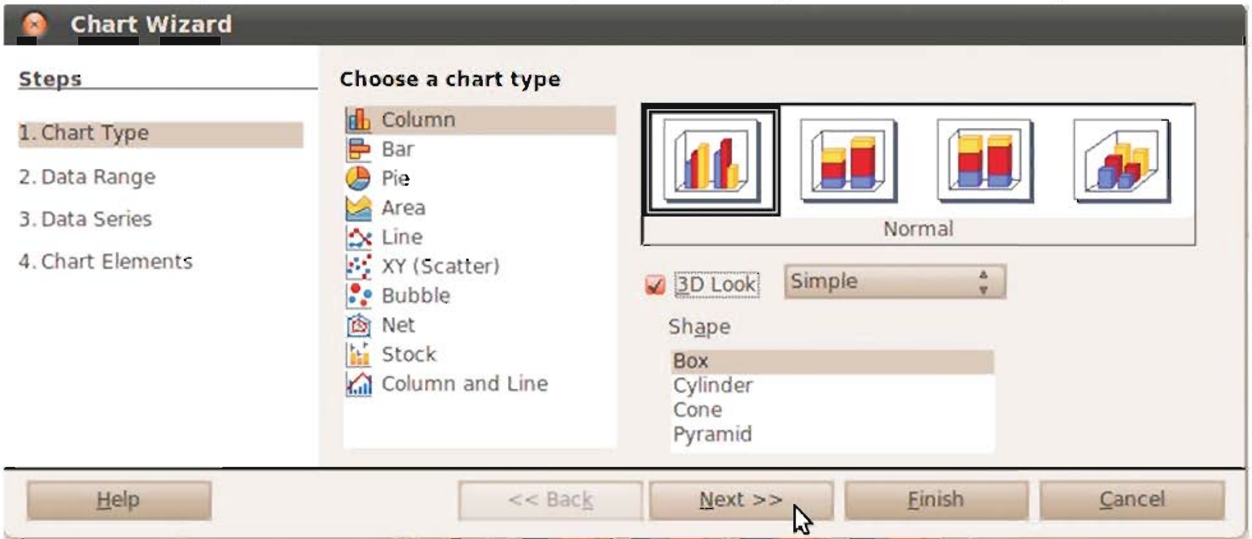
જ્યારે તમે આલેખ (ચાર્ટ) ઉમેરવાનો વિકલ્પ પસંદ કરો છો ત્યારે તમે ચાર્ટ વિઝાર્ડ (chart wizard) જોઈ શકશો. કેલ્સીમાં ચાર્ટ વિઝાર્ડ એ હકીકતમાં સ્પ્રેડશીટ પેકેજમાં ચાર્ટ તૈયાર કરવાની ક્રિયાઓની એક સરળ અને ઉપયોગકર્તાઓ સાથે મૈત્રીસભર પગલાંની એક શ્રેણી છે. કોઈ પણ વ્યક્તિએ માત્ર ડેટાનો વિસ્તાર પસંદ કરવાનો હોય છે અને તે પછી ચાર્ટ વિઝાર્ડ ચાલુ (ઇન્વોક) કરવાનું હોય છે. બાકીનાં પગલાંનું ધ્યાન વિઝાર્ડ લઈ લેશે. ચાર્ટ વિઝાર્ડનું ડાયલૉગ બૉક્સ આકૃતિ 8.2માં દર્શાવ્યા પ્રમાણેનું હોય છે.



આકૃતિ 8.2 : ચાર્ટ વિઝાર્ડ (Chart wizard)

આકૃતિ 8.2માં દર્શાવેલું ચાર્ટ વિઝાર્ડ ચાર પગલાં જણાવી ચાર્ટનો પ્રકાર (chart type), વિગતોનો વિસ્તાર (data range), વિગતોની શ્રેણી (data series) અને ચાર્ટના ઘટકો (chart elements)ની માહિતી લે છે. આ ચાર પગલાંની ચર્ચા નીચે કરેલી છે :

ચાર્ટનો પ્રકાર પસંદ કરવો (Selecting chart type)

આકૃતિ 8.2માં દર્શાવ્યા પ્રમાણે column આલેખ પસંદ કરો. તે આલેખ કેવો દેખાશે તે પણ આપણે અગાઉથી જોઈ શકીએ છીએ. દરેક સમયે આપણે આલેખનો પ્રકાર બદલીએ તે સાથે તે આલેખ કેવો દેખાશે (preview) તે પણ બદલાશે, જે આપણને આલેખ કેવો દેખાશે તેની સુંદર સમજ આપે છે.

ચાર્ટ વિઝાર્ડના મુખ્ય ભાગ નીચે પ્રમાણે છે :

- આલેખ તૈયાર કરવા અનિવાર્ય બધાં પગલાંની યાદી
- આલેખના પ્રકારોની યાદી
- વિવિધ પ્રકારના આલેખ અગાઉથી જોવાની સગવડ (preview)
- દરેક પ્રકારના આલેખના વિકલ્પો અને
- આગળ વધવા માટે અથવા પાછા જઈને બદલવાના વિકલ્પો

ચાર્ટ વિઝાર્ડમાં તમે અન્ય પ્રકારના આલેખ અને તેના પ્રિવ્યૂ જોઈ શકો છો. આલેખ(ચાર્ટ)ના પ્રકારોમાં કૉલમ ચાર્ટ (Column chart), બાર ચાર્ટ (Bar chart), પાઈ ચાર્ટ (Pie chart), લાઇન ચાર્ટ (Line chart), એરિયા ચાર્ટ (Area chart), XY ચાર્ટ, બબલ ચાર્ટ (Bubble chart), નેટ ચાર્ટ (Net chart), સ્ટૉક ચાર્ટ (Stock chart), કૉલમ એન્ડ લાઇન ચાર્ટ (Column and line chart)નો સમાવેશ થાય છે. કેલ્સી 10 પ્રકારના મૂળભૂત આલેખની પસંદગી આપણને આપે છે. એક વખત chart type આપણે પસંદ કરીએ પછી તેના અન્ય વિકલ્પો જેવા કે આલેખનો દેખાવ અને આકાર પસંદ કરી શકો છો. કેલ્સી દરેક પ્રકારના આલેખ માટે થોડા વિકલ્પો પૂરા પાડે છે. તમે જે આલેખની પસંદગી કરો તે પ્રમાણે તેના વિકલ્પો બદલાશે. કોઈ પણ તબક્કે તમે પાછા અગાઉનાં પગલાં ઉપર જઈ શકો અને તમે પસંદ કરેલા વિકલ્પો બદલી શકો છો.

અહીં 3D view box (લંબચોરસ જગ્યા) પણ હોય છે. જો તમે 3D boxને (તેના ઉપર ક્લિક કરીને) પસંદ કરશો તો તમે બદલાયેલ પ્રિવ્યૂ જોઈ શકશો. ફક્ત 3D ને યોગ્ય આલેખ જ (જેવા કે કૉલમ ચાર્ટ, બાર ચાર્ટ, પાઈ ચાર્ટ અને એરિયા ચાર્ટ) 3Dમાં જોવાનો વિકલ્પ આપે છે. તમે આલેખનો આકાર Shape boxમાંથી "Box", "Cylinder", "Cone" અને "Pyramid" પસંદ કરી શકો છો. તમે જે પસંદ કરો તે પ્રમાણે યોગ્ય પ્રિવ્યૂ તમને જોવા મળશે.

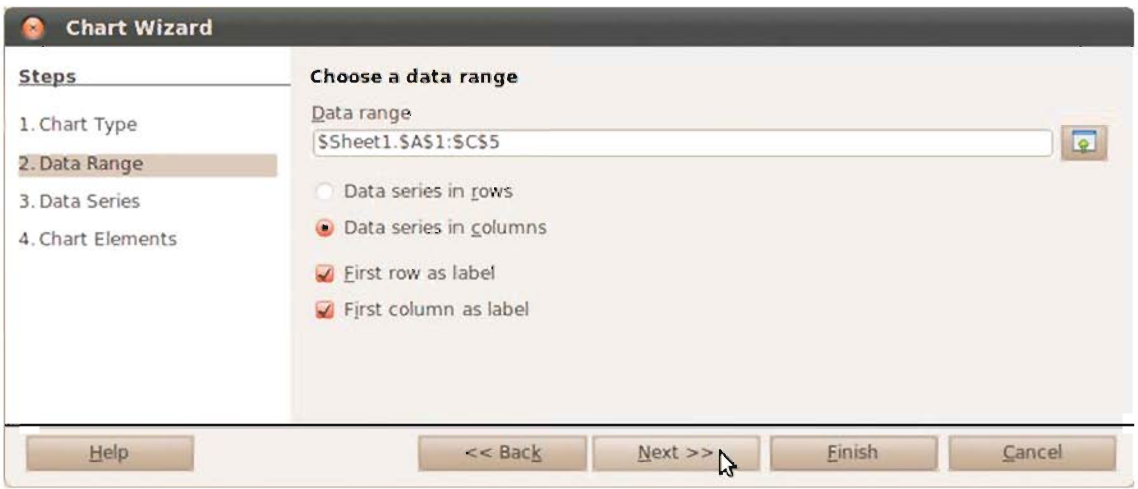
ઉદાહરણ તરીકે, આપણે નીચે જણાવેલા વિકલ્પો પસંદ કરેલા છે :

- ચાર્ટ ટાઈપ - Column
- ચાર્ટ લુક - **3D** સાથે **Simple**
- ચાર્ટ શેઈપ - **Box**

અંતમાં, Next બટન દબાવો. તમે નિરીક્ષણ કરી શકશો કે કેલ્સીએ ચાર્ટ વિઝાર્ડની પશ્ચાદ્‌ભૂમિમાં એક આલેખ બનાવી દીધો છે. આ તબક્કે, જો તમે Finish બટન દબાવશો તો આ જ ડિફૉલ્ટ ચાર્ટ તમે જોઈ શકશો કે જે તમે પછી તેને સુધારી કે ફૉર્મેટ કરી શકો છો. તમે અગાઉના તબક્કામાં જવા માટે Back બટન દબાવી શકો અને તમારા વિકલ્પોમાં ફેરફાર પણ કરી શકો છો.

ડેટા વિસ્તાર (Data range)

એક વખત તમે આલેખનો પ્રકાર નક્કી કરી દો પછી તમારે આલેખ માટે જરૂરી ડેટા આપવા પડે. એક શક્યતા એ છે કે આપણે અહીં કર્યું છે તેમ Insert → Chart આપતાં પહેલાં જ ડેટાનો વિસ્તાર આપી દીધો હોય. જો તેમ ન કર્યું હોય તો આકૃતિ 8.3માં દર્શાવ્યા પ્રમાણે આપણે હજી Data range વિકલ્પ પસંદ કરીને આપી શકીએ છીએ.



આકૃતિ 8.3 : Data rangeની પસંદગી

જો આલેખ ઉમેરતાં પહેલાં આપણે સંપૂર્ણ ડેટા વિસ્તારને બદલે ડેટા વિસ્તારનો અંશ જ પસંદ કરેલો હોય તો કેલ્સી ડેટા વિસ્તાર ઓળખવા પૂરતું ચકોર છે. એટલે કે, ધારો કે તમે ફક્ત એક જ સેલ જેમ કે B3 પસંદ કરેલો છે અને તમે **Insert → Chart** પસંદ કરો છો તો આ સંજોગોમાં કેલ્સી પોતાની જાતે ડેટાનો વિસ્તાર ધારી લેશે. જો કે ડેટા વિસ્તાર ચકાસી લેવી એ એક સારી પદ્ધતિ છે. તમે અન્ય સ્પ્રેડશીટ દસ્તાવેજના હયાત ડેટા અથવા અન્ય સુસંગત (કમ્પેટિબલ) દસ્તાવેજના ડેટાનો પણ નીચે જણાવ્યા પ્રમાણે ઉપયોગ કરી શકો છો.

કેલ્સીના આલેખ નીચે જણાવેલા ડેટા આધારિત હોય છે :

- કેલ્સી સેલના વિસ્તારમાંથી સ્પ્રેડશીટની કિંમતો
- Writer ટેબલમાંથી સેલની કિંમતો. તમે Writer, Draw અથવા Impress જેવા પૅકેજમાં પણ ડેટા દાખલ કરીને પછી તેની નકલ (copy and paste) કેલ્સી સ્પ્રેડશીટમાં કરી શકો છો.
- કોઈ પણ યોગ્ય રીતે ગોઠવેલી શાબ્દિક માહિતીની ફાઈલ (ફૉર્મેટેડ ફાઈલ) અથવા ટેબલ.

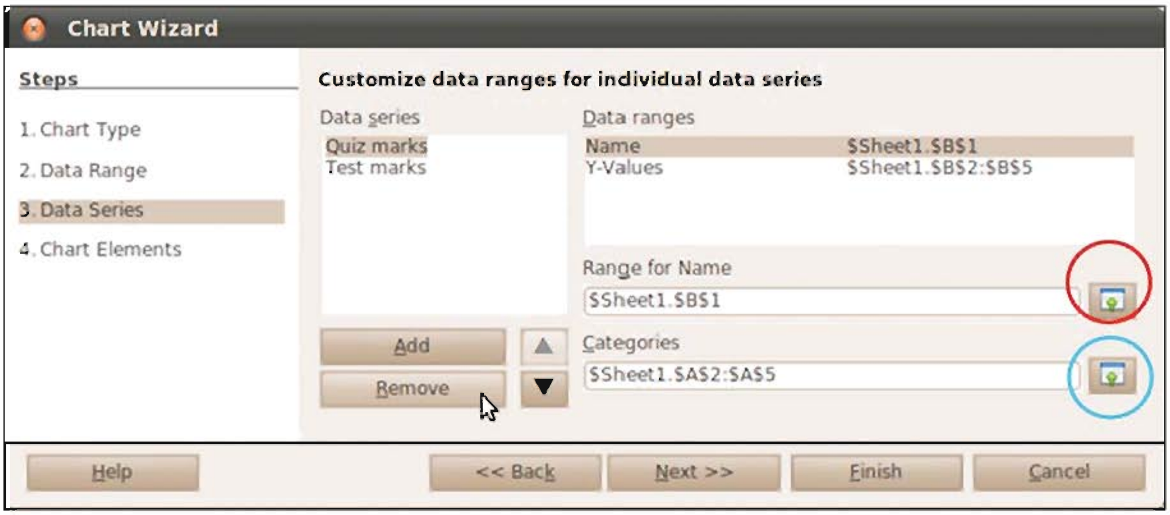
આપણા ઉદાહરણમાં ડેટા આ જ શીટમાંથી આવે છે અને તેની વિસ્તાર A1 થી C5 છે. અહીં નોંધ કરો કે કેલ્સીએ ડેટાનો વિસ્તાર $Sheet1.$A$1:$C$5 લીધો છે. એટલે કે, કેલ્સી વિસ્તારને નિરપેક્ષ સ્થાનાંકમાં ફેરવી દે છે. આથી, જો આલેખને અન્ય કોઈ સ્થાન ઉપર લઈ જવામાં આવે તો પણ ડેટાના વિસ્તારમાં કોઈ ફેરફાર ન થાય (અચલ). તે પછીનું બટન રો અથવા કોલમમાં ડેટા શ્રેણીનું (data series) સ્થાન પૂછે છે. અહીં આપણે આપણા ગુણ કોલમ

(ઊભી હરોળ)માં દાખલ કરેલા છે. આથી આપણે **Data series in column** વિકલ્પ પસંદ કરી શકીએ. આ ઉપરાંત આપણે **First row as label** વિકલ્પ પણ પસંદ કરવો પડશે. જ્યારે ડેટા અનેક ઊભી હરોળોમાં (કૉલમ્સ) હોય અને દરેકમાં શીર્ષક હોય ત્યારે આ વિકલ્પ ઘણો ઉપયોગી બને છે. પહેલી રો (આડી હરોળ) દરેક કૉલમ(ઊભી હરોળ)નાં બધાં શીર્ષક બને છે. આપણે પહેલી હરોળ બધાં શીર્ષક ધરાવે છે તે જણાવીને આપણે કૉલમનાં શીર્ષક આપી શકીએ છીએ. વિદ્યાર્થીના ગુણનાં ઉદાહરણમાં પહેલી આડી હરોળ **Quiz Marks** અને **Test Marks** જેવાં શીર્ષકો ધરાવે છે.

આ જ રીતે, જ્યારે દરેક આડી હરોળમાં શીર્ષક હોય તો આપણે **First column as label** વિકલ્પ પસંદ કરી શકીએ. તમે બંને ચેકબૉક્સ (check boxes) પણ પસંદ કરી શકો.

ડેટા શ્રેણી (Data series)

ચાર્ટ વિઝાર્ડ આલેખ તૈયાર કરવા માટે તમારી અનુકૂળતા પ્રમાણે ડેટાનો વિસ્તાર બનાવવાની સગવડ પૂરી પાડે છે. અહીં, આપણે ચાર્ટ વિઝાર્ડ દ્વારા જાતે જ પસંદ કરેલી કિંમતોનો ઉપયોગ કરીશું. આકૃતિ 8.4માં ચાર્ટ વિઝાર્ડ દ્વારા પસંદ કરેલી ડેટા શ્રેણી દર્શાવેલી છે.



આકૃતિ 8.4 : ડેટા શ્રેણીની પસંદગી

તમે આ ડાયલૉગ બૉક્સ વડે ડેટા શ્રેણી ઉમેરી કે રદ કરી શકો છો. જેમ કે આપણા ઉદાહરણમાં અંતિમ પરીક્ષાના ગુણ ઉમેરવા માટે **Add** બટનનો ઉપયોગ કરીને ડેટાની શ્રેણી ઉમેરી શકાય છે. કોઈ ડેટાની શ્રેણી રદ કરવા માટે પ્રથમ શ્રેણી પસંદ કરો કે જે તમે રદ કરવા ઇચ્છા છો તે પછી **Remove** (આકૃતિ 8.4માં જુઓ માઉસનું તીર) પસંદ કરો. ડાયલૉગ બૉક્સ વિસ્તારનું નામ અને વિસ્તારની કિંમતો આપવાની સગવડ પણ પૂરી પાડે છે. અહીં વિસ્તારનું નામ $B$1 છે અને તેની કિંમત "Quiz marks" છે. તમે **Select** બટન (આકૃતિમાં લાલ લીટી સાથે વર્તુળ દોરેલું છે) પણ વાપરી શકો છો. તે ચાર્ટ વિઝાર્ડને નાનું બનાવી દેશે અને તમને વર્કશીટના વિસ્તારમાં લઈ જશે. તમે તમારી જરૂરિયાત પ્રમાણે યોગ્ય સેલ પસંદ કરી શકો છો. જ્યારે તમે પસંદગીનું કાર્ય પૂરું કરો તે પછી ચાર્ટ વિઝાર્ડનું **Maximize** બટન દબાવો. જ્યારે તમને સેલનાં સ્થાનાંક યાદ ન હોય ત્યારે આ ઉપયોગી બને છે.

આ જ પ્રમાણે તમે ડેટા શ્રેણીની કિંમતો પસંદ કરી શકો છો. આપણા ઉદાહરણમાં ડેટા શ્રેણીની કિંમતો સેલ B2થી B5માં રહેલા quiz marks છે. તમે $Sheet1.$B$2: $B$5 પણ લખી શકો છો.

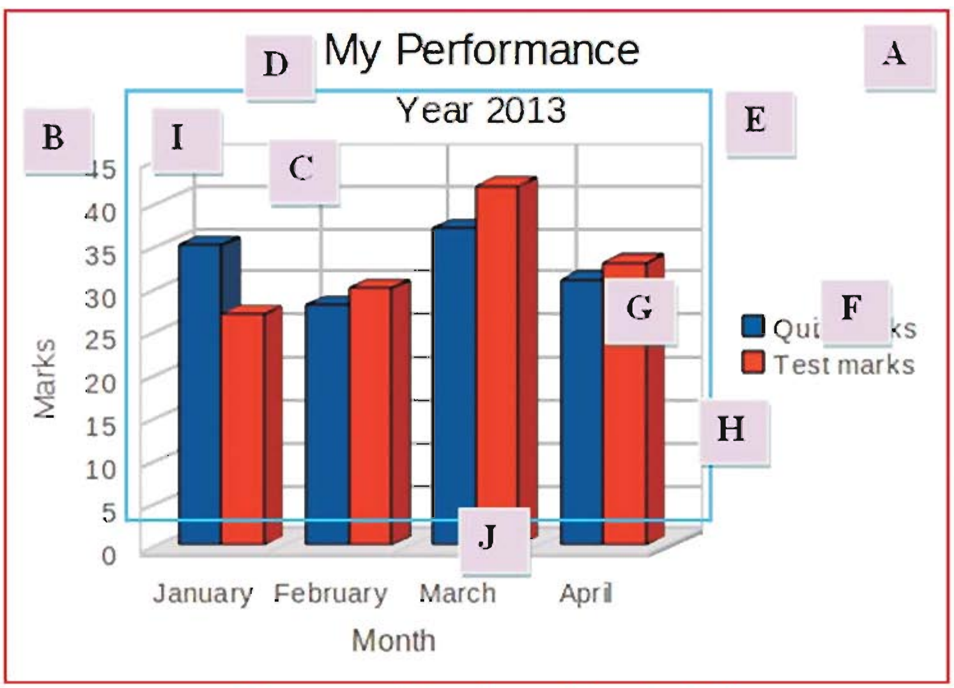
Data seriesના તબક્કામાં તમે ડેટાની categories પણ દાખલ કરી શકો છો. આપણા ઉદાહરણમાં Data series Categories સેલ A2થી A5માં રહેલા મહિનાનાં નામ છે. તમે $Sheet1.$A$2: $A$5 પણ લખી શકો છો. અહીં પણ તમે Select બટન (આકૃતિ 8.4માં વાદળી લીટીથી સાથે ગોળ નિશાની કરેલું) વડે ચાર્ટ વિઝાર્ડને નાનું કરી શકો અને સ્પ્રેડશીટમાંથી કિંમતો સીધી જ પસંદ કરી શકો છો. અંતમાં Maximize બટન દબાવો.

ડેટા એલિમેન્ટ્સ (Data elements)

ચાર્ટ વિઝાર્ડ આલેખના જુદા જુદા ભાગનાં નામ જેમ કે Title, Subtitle, X-axis captions, Legendનું સ્થાન અને Grids પ્રદર્શિત કરવાના વિવિધ વિકલ્પો દાખલ કરવાની સગવડ આપે છે. આકૃતિ 8.5માં ચાર્ટ વિઝાર્ડનો સ્ક્રીન આપેલો છે કે જેમાં ડેટા એલિમેન્ટ્સ સેટ કરવાની સગવડ મળે છે.

આકૃતિ 8.5 : ચાર્ટના ઘટકો

ચાર્ટ વિઝાર્ડના આ તબક્કામાં ચાર્ટ વિશેની અલગ અલગ માહિતી જેવી કે આલેખનું શીર્ષક, આલેખનું પેટા શીર્ષક અને અક્ષ તેમજ લિજેન્ડની માહિતી પૂરી પાડવામાં આવે છે. આકૃતિ 8.5માં દર્શાવ્યા પ્રમાણે **Chart elements**નો પહેલો વિકલ્પ **Title** છે. અહીં તમે આલેખનું મુખ્ય શીર્ષક આપી શકો છો. બીજો વિકલ્પ છે **Subtitle,** જ્યાં તમે આલેખનું પેટા શીર્ષક આપી શકો. શીર્ષક અને પેટા શીર્ષક આલેખ વિસ્તારના સૌથી ઉપરના ભાગમાં રાખવામાં આવે છે. તમે શીર્ષક અને પેટા શીર્ષક આપી દો તે પછી તમે અક્ષ વિશે માહિતી આપી શકો. ત્રણ અક્ષ X-અક્ષ, Y-અક્ષ અને Z-અક્ષનાં શીર્ષકની માહિતી અહીં આપી શકો છો. આ તબક્કાએ તમારે **Display legend** વિકલ્પ વડે લિજેન્ડ પ્રદર્શિત કરવા છે કે નહીં તે પણ પસંદ કરી શકો છો. જો તમે પ્રદર્શિત કરવાનું પસંદ કરો તો તે લિજેન્ડ તમારે આલેખની ડાબી, જમણી, ઉપર કે નીચે ક્યાં પ્રદર્શિત કરવા ઇચ્છો છો તે જણાવવું પડશે. જો કોઈ વિકલ્પ ન આપો તો પૂર્વનિર્ધારિત રીતે આલેખ વિસ્તારમાં આલેખની જમણી બાજુએ લિજેન્ડ દેખાશે.



આકૃતિ 8.6 : સાદો આલેખ અને ચાર્ટ એલિમેન્ટ્સ

બધા વિકલ્પો સાચા આપી દીધા પછી તૈયાર આલેખ માટે તમે Finish બટન દબાવો. હવે ચાલુ શીટમાં તમને આલેખ દેખાશે. તમે આકૃતિ 8.6માં દર્શાવ્યા પ્રમાણે ચાર્ટ ઘટકોનાં લેબલ સાથે આલેખ જોઈ શકશો. આલેખના વિવિધ ભાગની ચર્ચા હવે પછીના વિભાગમાં કરી છે.

આલેખના વિવિધ ઘટકો (Elements of a chart)

આલેખમાં સામાન્ય રીતે આ મુજબના ઘટકો હોય છે : આલેખ વિસ્તાર, આલેખનું શીર્ષક અને પેટા શીર્ષક, કિંમતો સાથે આલેખના અક્ષ, આલેખ અક્ષ ડેટા વિસ્તાર, આલેખ ડેટા વિસ્તાર, આલેખની અક્ષનાં લિજેન્ડ, ડેટા શ્રેણી અને આલેખ જગ્યા (ચાર્ટ ફ્લોર). આપણે અત્યાર સુધીમાં આકસ્મિક કસોટી અને વર્ગ પરીક્ષાઓમાં વિદ્યાર્થીના દેખાવ બાબતનો એક સાદો આલેખ બનાવી દીધો છે. આલેખના કોઈ ચોક્કસ ભાગ વિશે જાણકારી મેળવવા માટે તમે ફક્ત માઉસના તીરને તે જગ્યાએ મૂકો. આથી તે ભાગનું નામ પ્રદર્શિત થશે. ચાલો, આપણે કેલ્સીના આલેખનાં કેટલાક ઘટકોનું નિરીક્ષણ કરીએ. આકૃતિ 8.6માં કેટલાંક લેબલ આલેખના જુદા જુદા ભાગનો નિર્દેશ કરે છે. આ લેબલની સમજૂતી નીચે આપેલી છેઃ

લેબલ A : આલેખ વિસ્તાર (Chart area) : આલેખનો વિસ્તાર લંબચોરસ વિસ્તાર છે જેમાં આલેખનો સમાવેશ થયેલો છે.

લેબલ B : કિંમતો સાથે Y-અક્ષ (Y-axis with values) : આપણે જે ડેટા આપેલો હોય તેના આધારે Y-અક્ષની કિંમતો જાતે જ લેવામાં આવે છે. અહીં આપણે 50થી વધારે ગુણ આપેલા નથી આથી Y-અક્ષ ઉપર 0 થી 45 કિંમત લેવામાં આવી છે.

લેબલ C : અક્ષ ડેટા વિસ્તાર (Data area axis) : અક્ષ ડેટા વિસ્તાર એ ડેટા વિસ્તારની અંદર અક્ષ છે. આ વિકલ્પ પસંદ કરવાથી આલેખની પશ્ચાદ્‌ભૂમિમાં તમે આડી અને ઊભી રેખાઓનું જાળું જોઈ શકો છો. ડેટા વિસ્તારની અંદર અક્ષનો મુખ્ય ઉપયોગ કિંમતો જાણવાનો છે જ્યારે આલેખમાં કિંમતો પ્રદર્શિત ન કરી હોય.

લેબલ D : શીર્ષક અને પેટા શીર્ષક (Title and Subtitle) : આખા આલેખને આપેલું કામ એ આલેખનું શીર્ષક છે. અહીં આપણે My Performance શીર્ષક આપેલું છે. શીર્ષક સાથે આપણે પેટા શીર્ષક પણ આપી શકીએ. પેટા શીર્ષક આપવાનો હેતુ વધારાની માહિતી અથવા આલેખનો ઉપયોગ બતાવવાનો હોય છે. અહીં આપણે Year 2013 પેટા શીર્ષક તરીકે આપેલું છે.

લેબલ E : ડેટા વિસ્તાર (Data area) : આલેખ વિસ્તારની અંદર ડેટા વિસ્તારમાં આલેખ સમાયેલો હોય છે. જે આલેખ તરીકે પણ ઓળખાય છે. આલેખ વિસ્તારમાં રહેલો આલેખ એ મુખ્ય ભાગ છે અને આથી આલેખ વિસ્તારની મધ્યમાં તે રાખવામાં આવે છે. આકૃતિ 8.6માં વાદળી રંગની સીમારેખાથી ઘેરાયેલો વિસ્તાર આલેખ (chart) અથવા ડેટા વિસ્તાર (data area) તરીકે ઓળખાય છે.

લેબલ F : લિજેન્ડ (Legend) : આલેખ એક પ્રકારનું ચિત્રકામ છે. તેમાં ડેટા અને ગ્રાફિક્સને સંબંધિત કેટલીક શાબ્દિક માહિતી હોવી જોઈએ. આકૃતિ 8.6માં ઊભી હરોળો (કૉલમ) વાદળી અને લાલ રંગમાં દર્શાવેલી છે. તમે શીર્ષક, X-અક્ષ અને Y-અક્ષના લેબલ ઉપરથી જાણી શકો કે આ આલેખ જુદા જુદા માસમાં લીધેલી વિવિધ કસોટીઓના ગુણનો છે. જો કે, અહીં એ સ્પષ્ટ નથી કે કયા રંગની કૉલમ કયા ગુણનો નિર્દેશ કરે છે. આ માટેની માહિતીનો સમાવેશ કરવા માટે આલેખમાં લિજેન્ડનો ઉપયોગ થાય છે. અહીં આપણે આકૃતિ 8.6માં વાદળી રંગ આકસ્મિક કસોટીનાં ગુણ અને લાલ રંગ વર્ગ કસોટીના ગુણ દર્શાવવા માટે વાપરેલો છે.

લેબલ G : ડેટા શ્રેણી કૉલમ (Data series column) : આલેખ એક કરતાં વધારે ડેટા શ્રેણીનો હોઈ શકે. દરેક ડેટા શ્રેણી માટે અલગ અલગ રંગ પસંદ કરવામાં આવે છે અને પછી આલેખ તૈયાર થાય છે. ઉદાહરણ તરીકે, કૉલમ ચાર્ટ(column chart)માં દરેક ડેટા શ્રેણી માટે રંગની પસંદગી કરી તે રંગનાં અનેક સ્તંભ જુદા જુદા સમયગાળા કે તબક્કા માટે બનાવવામાં આવે છે. આપણા ઉદાહરણમાં આકસ્મિક કસોટીના ગુણની એક ડેટા શ્રેણી બનાવેલી છે અને વાદળી રંગ આપોઆપ પસંદ થયેલો છે. કેલ્સી દરેક માસના ગુણ દર્શાવીને વાદળી રંગના અનેક સ્તંભ બનાવે છે. આકૃતિ 8.6માં વાદળી રંગના સ્તંભ હકીકતમાં આકસ્મિક કસોટીના ગુણની ડેટા શ્રેણી દર્શાવે છે. આ જ પ્રમાણે, આકૃતિ 8.6માં લાલ રંગના સ્તંભ હકીકતમાં વર્ગપરીક્ષાના ગુણ દર્શાવે છે.

લેબલ H : ચાર્ટ ફ્લોર (Chart floor) : ચાર્ટ ફ્લોર એ એક ક્ષિતિજ સમાંતર સપાટી છે જે આલેખના વિવિધ ભાગનો પાયો છે. કોઈ ચોક્કસ પ્રકારના આલેખ જેમ કે બાર અને કોલમ પ્રકારના આલેખમાં તે ખૂબ ઉપયોગી છે. તમે આકૃતિ 8.6માં ચાર્ટ ફ્લોરને રાખોડી રંગથી ભરેલો જોઈ શકશો.

લેબલ I : ચાર્ટ વૉલ (Chart wall) : ચાર્ટ વૉલ એ ધરતીને કાટખૂણે રહેલી સપાટી છે જે આલેખના Y-અક્ષને ટેકો પૂરો પાડે છે. આકૃતિ 8.6માં દર્શાવ્યા પ્રમાણે તેમાં કોઈ રંગ ભરેલો નથી. જો કે તેમાં તમે તમારી પસંદગીનો રંગ ભરીને તે બદલી શકો છો.

લેબલ J : X-અક્ષ ઉપરનાં લિજેન્ડ (legends on X-axis) : તમને આપેલા એક સાદા આલેખને લો. આલેખના X-અક્ષ ઉપર ફક્ત કિંમતો જ છે. આકૃતિ 8.6માં દર્શાવ્યા પ્રમાણે X-અક્ષ ઉપર "January", "February" વગેરે જેવા મહિનાનાં નામનો ઉપયોગ આપણે કર્યો છે. આ નામ જાતે જ ખુલાસો કે સ્પષ્ટીકરણ કરી દે છે. કોઈ પણ વ્યક્તિ વગર મહેનતે જાણી શકે કે આપણે વર્ષના મહિનાઓ વિશે વાત કરીએ છીએ. જો આપણે વાહનની ઝડપ બાબત જણાવવા ઇચ્છતા હોઈએ તો ? અલબત્ત, તે માટે આપણે 10, 20, 30 વિગેરે લખી શકીએ. પણ એવો કોઈ રસ્તો છે કે આ કિમી / કલાક અથવા મીટર / મિનિટ છે તે જણાવી શકે ? X-અક્ષ ઉપરના આ એકમો અને વસ્તુની શ્રેણી X-અક્ષ ઉપર લિજેન્ડ દ્વારા જણાવી શકાય છે. આ જ પ્રમાણે આપણે Y-અક્ષ ઉપર પણ લિજેન્ડ જણાવી શકીએ છીએ.

આલેખ તૈયાર કરવાનો અન્ય રસ્તો (An alternative way to create a chart)

ઉપર જણાવ્યા તે સિવાય અન્ય વિકલ્પરૂપે તમે સ્ટાન્ડર્ડ ટૂલબાર ઉપરથી ચાર્ટ (chart) આઇકોન પસંદ કરી શકો. જો તમે સ્ક્રીન ઉપર સ્ટાન્ડર્ડ ટૂલબાર જોઈ શકતા ન હો તો આપણે અગાઉ ચર્ચા કરી હતી તે પ્રમાણે View → Toolbars વડે તે ટૂલબારને દશ્યમાન કરી શકો છો. આકૃતિ 8.7માં ટૂલબાર ઉપર ચાર્ટ આઇકોન દર્શાવ્યો છે.

આકૃતિ 8.7 : ચાર્ટ આઇકોન

તમારે ફક્ત ત્રણ કાર્ય કરવાં પડશે :

1. સૌપ્રથમ શીટમાં ડેટા દાખલ કરો.
2. ડેટા પસંદ કરો અને ઉપર જણાવ્યા પ્રમાણે ચાર્ટ આઇકોન દબાવો.
3. હવે **Chart type** પસંદ કરો. તે પછી **Finish** દબાવો. તમે આ તબક્કે અન્ય વિકલ્પો ન આપો.

જ્યારે તમે Finish બટન દબાવો છો, તે સાથે તમારો આલેખ તૈયાર થઈને વર્તમાન શીટ ઉપર જ દેખાશે. તમારા આલેખમાં બધા શીર્ષક અને લિજેન્ડ ન પણ હોઈ શકે. જો કે તમે ઇચ્છો તો તેને ફૉર્મેટ કરીને વધારે આકર્ષક અને અસરકારક હંમેશાં બનાવી શકો છો. નીચેના વિભાગમાં ફૉર્મેટ કરવાના અને આલેખને બદલવાના રસ્તા સમજાવ્યા છે.

આલેખને ફૉર્મેટ કરવો અને બદલવો (Formatting and Modifying Charts)

અનેક સમયે તમારી જરૂરિયાત પ્રમાણે આલેખ તૈયાર કરવાની જરૂર પડે છે. એટલે કે, આલેખને તમારી શૈલી પ્રમાણે તમે બદલવા ઇચ્છો. ચાલો, આપણે આલેખને ફૉર્મેટ (format) કરવાના વિવિધ વિકલ્પોની ચર્ચા કરીએ.

આલેખની જગ્યા અને કદ બદલવા (Modifying the place and size of the chart)

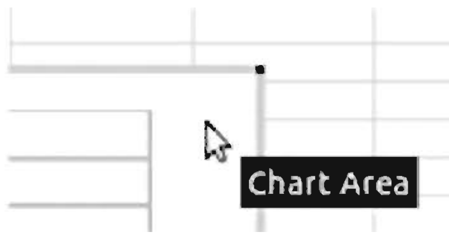
જ્યારે આલેખ બનાવવામાં આવે છે ત્યારે તેનો ચોક્કસ દેખાવ અને જગ્યા નક્કી હોય છે જે સામાન્ય રીતે વર્તમાન વર્કશીટમાં હોય છે. ઘણી વાર તમને પૂર્વનિર્ધારિત જગ્યા પસંદ ન પડે. અમુક સમયે તમને તેનું કદ ન ગમે. તમે આખો આલેખ ફક્ત માઉસનું ડાબું બટન દબાવીને ડ્રેગ કરવાથી (માઉસ વડે તેનાં હૅન્ડલ દ્વારા) તમારી ઇચ્છિત જગ્યાએ ખસેડી શકો છો. તમે તેની નકલ કરી શકો, તમે તે સ્થાન પરથી દૂર કરી શકો (cut) અને અન્ય સ્થાન પર મૂકી શકો છો (paste). એ જ પ્રમાણે, આલેખ પસંદ કરી માઉસ વડે તેના હૅન્ડલ ડ્રેગ કરી તેનું કદ પણ બદલી શકો છો. જુઓ આકૃતિ 8.8.

આકૃતિ 8.8 : આલેખનું કદ બદલવું

અમુક સમયે આલેખ બની ગયા પછી તમારે ડેટાની અમુક કિંમતોમાં ફેરફાર કરવાની જરૂર પડે તો ચિંતા કર્યા વગર કિંમતો બદલી નાખો. આથી આલેખ આપોઆપ બદલાઈ જશે. માઉસ વડે કોઈ સ્પ્રેડશીટમાં આલેખ ઉમેરવામાં આવે અથવા પસંદ કરવામાં આવે તો આલેખને લગતા બટન અને આદેશ માટે કેલ્સી પૅકેજનું ટૂલબાર બદલાઈ જાય છે. જ્યારે સેલમાં ડેટામાં ફેરફાર કરવામાં આવે છે તો આલેખમાં પણ ફેરફાર થાય છે.

આલેખને ફક્ત ફૉર્મેટ કરવો જ શક્ય નથી પણ આલેખમાં સુધારા પણ કરી શકાય છે. નીચેના વિભાગમાં આલેખને ફૉર્મેટ કરવા તેમજ સુધારા કરવાની રીતો વર્ણવી છે :

આલેખનાં કદ અને જગ્યા ઉપરાંત સૌપ્રથમ તમે કદાચ ધ્યાન ઉપર લીધો હશે તે છે તેનો પ્રકાર (type). ધારો કે column પ્રકારના આલેખમાંથી હવે તમે બદલીને line પ્રકારનો ઇચ્છો છો. આ માટે તમારે સૌપ્રથમ એ યાદ રાખવું પડે કે આલેખમાં સુધારા કરવા માટે તેને Edit modeમાં લઈ જવો પડે. આ માટે **આલેખ વિસ્તાર (Chart area)**માં કોઈ પણ જગ્યાએ ડબલ (લેફ્ટ) ક્લિક કરો. આકૃતિ 8.9માં દર્શાવ્યા પ્રમાણે આલેખની આસપાસ રાખોડી (grey) રંગની કિનારી જોવા મળશે. તમે તેનાં જુદા જુદા પ્રાચલો (parameters) બદલવા માટે આલેખ વિસ્તાર (ચાર્ટ એરિયા) ઉપર રાઇટ ક્લિક કરો.



આકૃતિ 8.9 : આલેખ વિસ્તાર (ચાર્ટ એરિયા)

આલેખનો પ્રકાર બદલવો (Modifying Chart Type)

આલેખનો પ્રકાર બદલવા માટે આ પ્રમાણે આદેશ આપો : **Format → Chart Type** આના વિકલ્પ રૂપે, આકૃતિ 8.10માં દર્શાવ્યા પ્રમાણે તમે ફૉર્મેટિંગ ટૂલબાર ઉપર રહેલા **Chart type** આઇકોનને દબાવો.

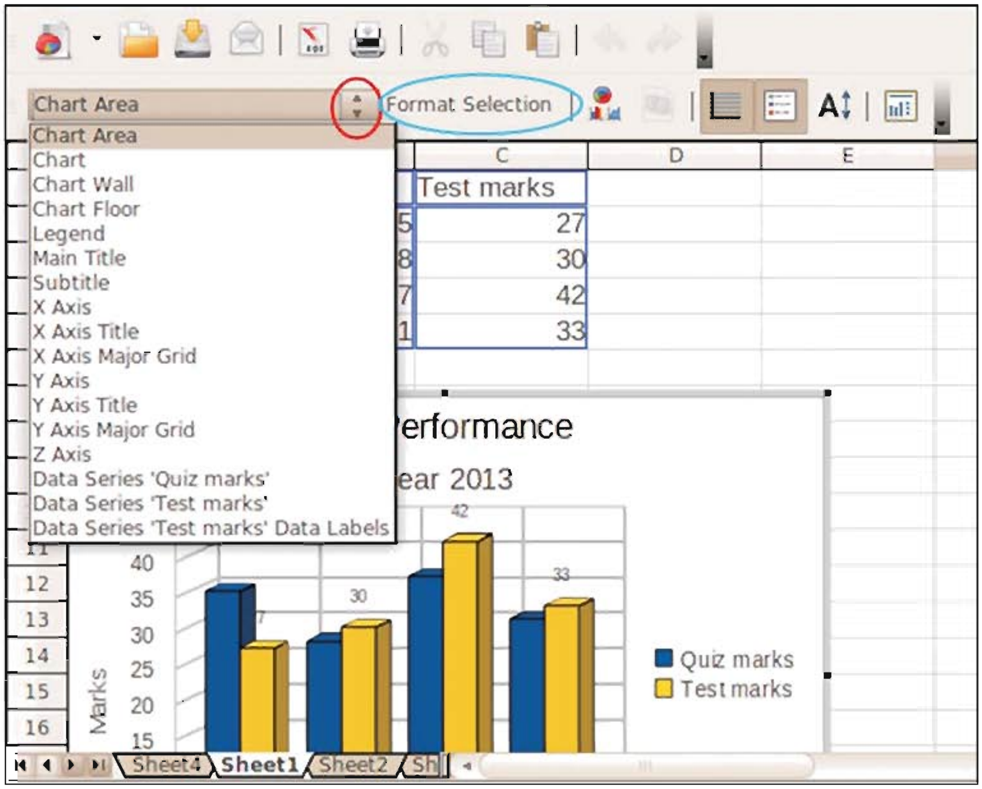
આકૃતિ 8.10 : ચાર્ટ ફૉર્મેટિંગ ટૂલબાર

જ્યારે તમે આલેખ પસંદ કરો છો તે સમયે ચાર્ટ ફૉર્મેટિંગ ટૂલબાર દેખાય છે. ચાર્ટ ફૉર્મેટિંગ ટૂલબાર આલેખના જુદા જુદા ભાગ જેવાં કે આલેખ વિસ્તાર, આલેખની ડેટા શ્રેણી, આલેખનાં અક્ષ, આલેખનાં શીર્ષકો વગેરેને ફૉર્મેટ કરવા (વધારે યોગ્ય અને આકર્ષક બનાવવા) માટે વપરાય છે. આકૃતિ 8.10માં વાદળી રંગની લીટીથી નિશાની કરેલો ત્રીજો આઇકોન આલેખના વિવિધ પ્રકાર માટેનો છે. જ્યારે તમે આ આઇકોન પર ક્લિક કરો છો ત્યારે એક ડાયલૉગ બૉક્સ સ્ક્રીન ઉપર પ્રદર્શિત થાય છે. આ ડાયલૉગ બૉક્સના વિકલ્પો આપણને આલેખના પ્રકાર અને 3D આલેખ જેવો આલેખનો દેખાવ તેમજ બૉક્સ (Box) કે પિરામિડ (Pyramid) જેવા આલેખના આકાર બદલવાની પરવાનગી આપે છે.

તમે સીધા જ આલેખ ઉપર રાઇટ ક્લિક કરીને તેનો પ્રકાર (type) બદલી શકો છો. જ્યારે તમે રાઇટ ક્લિક કરો તે સમયે સ્ક્રીન ઉપર એક નાનું ઊભું મેનૂ પ્રદર્શિત થાય છે જેમાં તે સમયે શક્ય કાર્યોની યાદી હોય છે.

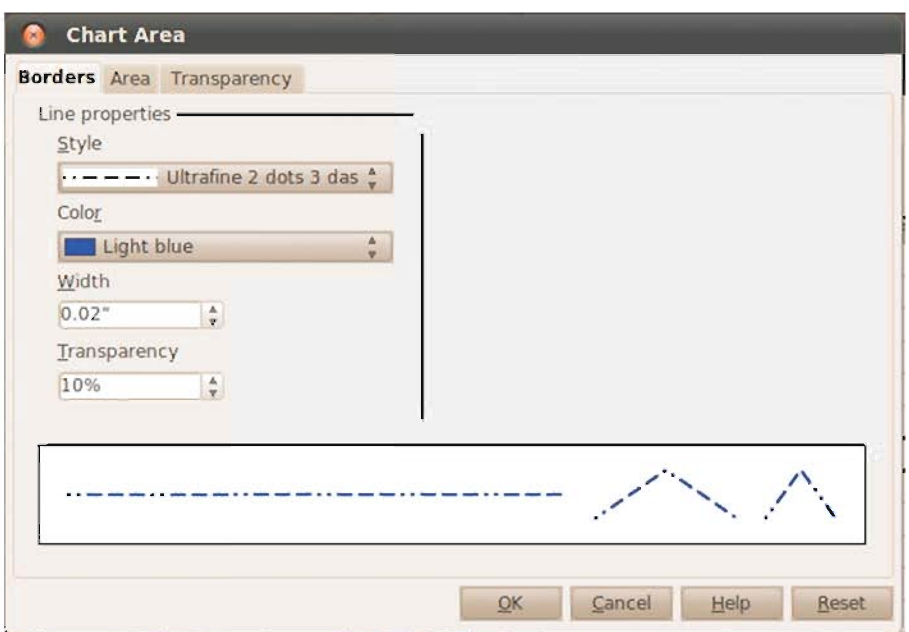
આલેખ વિસ્તારને ફૉર્મેટ કરવો (Formatting the chart area)

ઉપર જણાવ્યા મુજબ આલેખ વિસ્તાર (ચાર્ટ એરિયા) એ આલેખનો એક ભાગ છે. જ્યારે તમે આલેખ ઉપર ક્લિક કરો ત્યારે આકૃતિ 8.11માં દર્શાવ્યા પ્રમાણે આલેખનું ટૂલબાર કાર્યરત થાય છે. તમે જે જે ભાગ ફૉર્મેટ કરી શકો તેની યાદી ટૂલબાર ઉપરનો સૌપ્રથમ આઇકોન રજૂ કરે છે. તમે આ યાદીમાંથી જે કોઈ ભાગ ફૉર્મેટ કરવા ઇચ્છતા હોય તે ચોક્કસ ભાગ પસંદ કરી શકો છો. આ માટે આકૃતિ 8.11માં દર્શાવ્યા પ્રમાણે આ બૉક્સ સાથેના ઉપર અને નીચે દર્શાવતા તીર સાથેના નાના ત્રિકોણોનો ઉપયોગ કરો. આકૃતિ 8.11માં લાલ લીટીથી ચિહ્નિત (encircled) ભાગ જુઓ. એક વખત તમે યોગ્ય ભાગની પસંદગી કરી લો તો પછી **Format Selection** વિકલ્પ પસંદ કરો. આકૃતિ 8.11માં વાદળી રંગનું વર્તુળ જુઓ. આ જ આકૃતિ (8.11)માં આપણે **Chart Area** પસંદ કરેલો છે. જો કે આલેખના અન્ય ભાગોની યાદી પણ તમે જોઈ શકો છો. **Chart area** પસંદ કર્યા પછીનું પગલું **Format Selection** બટન પસંદ કરવાનું છે.



આકૃતિ 8.11 : આલેખ વિસ્તારને ફૉર્મેટ કરવો

આ આપણને યોગ્ય ડાયલૉગ બૉક્સ વડે પસંદ કરેલાં આલેખનાં ભાગને ફૉર્મેટ કરવા દે છે. આકૃતિ 8.12માં દર્શાવેલા ડાયલૉગ બૉક્સ આપણને આલેખના વિસ્તારની હદ (border), આલેખનો વિસ્તાર અને ભરેલા રંગની પારદર્શકતાને ફૉર્મેટ કરવાની મંજૂરી આપે છે. સામાન્ય રીતે આલેખની હદ (સીમા) પૂર્વ નિર્ધારિત રીતે દૃશ્યમાન નથી પણ જો કે તમે તેને દૃશ્યમાન બનાવી શકો છો. આ ઉપરાંત સીમાની ઢબ જેમ કે સળંગ લીટી, ટપકાંવાળી લીટી, જાડી રેખા વગેરે અને સીમાનો રંગ પણ પસંદ કરી શકો છો. આલેખ વિસ્તારને વિવિધ રંગોથી, ગ્રેડિયન્ટથી (પ્રણવતા-gradient) અથવા વિવિધ પારદર્શક્તાના ધોરણથી બિટ મેપ ઈમેજ (Bitmap image) વડે ભરી શકો છો. જો તમે આલેખ વિસ્તારને એકસરખા રંગથી (પારદર્શક્તા સાથે અથવા વગર) ભરવા ઇચ્છતા ન હોય તો કદાચ તમે રૈખિક (linear), ત્રિજ્યાગતિ (radial) અથવા અક્ષીય (axial) દિશાઓમાં રંગ ભરવા વિચારી શકો છો.

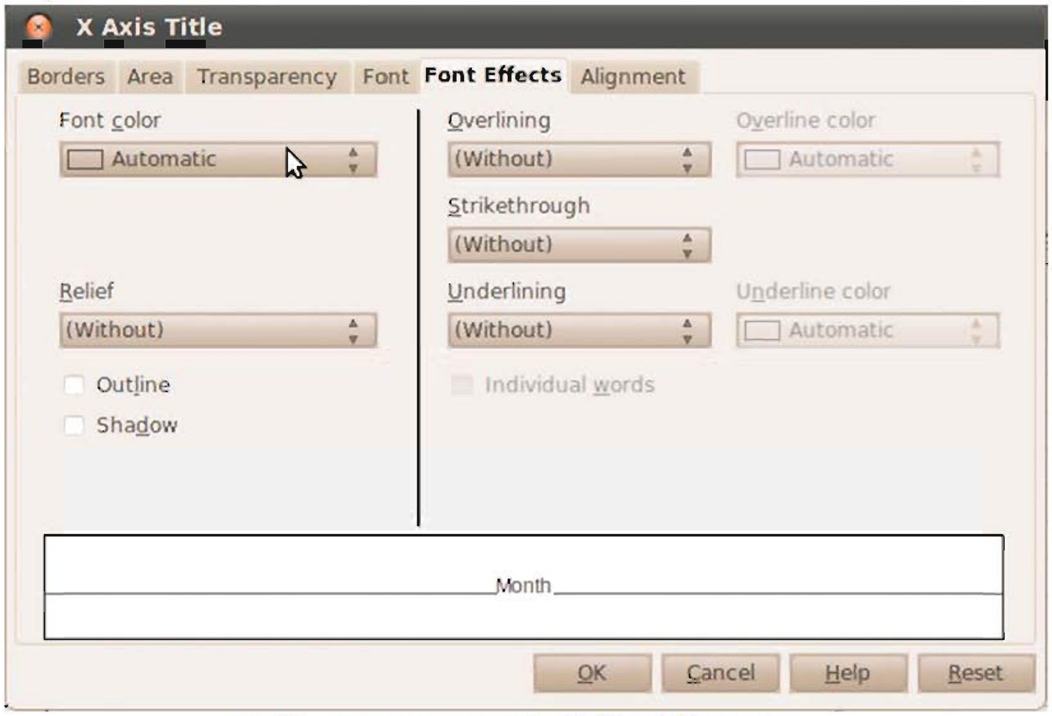


આકૃતિ 8.12 : આલેખ વિસ્તારને બદલવો

અક્ષનાં શીર્ષકોને ફૉર્મેટ કરવા (Formatting titles to the axis)

આલેખને સારી રીતે સમજવા માટે X-અક્ષ અને Y-અક્ષને શીર્ષકો આપવા માટે નીચે જણાવેલી ક્રિયાઓ કરો :

- આલેખ પસંદ કરો.
- આકૃતિ 8.11માં દર્શાવ્યા પ્રમાણે ચાર્ટ ફૉર્મેટિંગ ટૂલબાર તમે જોઈ શકશો. ટૂલબાર ઉપરથી X-અક્ષનું શીર્ષક ફૉર્મેટ કરવા માટેનો આઇકોન પસંદ કરો.



આકૃતિ 8.13 : X-અક્ષનાં શીર્ષકો ફૉર્મેટ કરવા

- હવે **Format Selection** બટન દબાવો.
- આથી તમે સ્ક્રીન ઉપર **X-Axis Titles** નામનું ડાયલૉગ બૉક્સ જોશો. આ ડાયલૉગ બૉક્સ આકૃતિ 8.13માં દર્શાવેલું છે.
- ડાયલૉગ બૉક્સમાં જરૂરી માહિતી ભરો.

આ પ્રકારની ફૉર્મેટિંગની સગવડને કારણે તમે X-અક્ષનાં શીર્ષકોનાં અનેક ભાગોની માહિતી આપી શકો છો જેમ કે અક્ષનાં શીર્ષકના વિસ્તારની સીમારેખા, શીર્ષકના ફૉન્ટ, ફૉન્ટની અસરો અને તેનું એલાઈમેન્ટ (alignment ગોઠવણ).

ડાયલૉગ બૉક્સ સીમારેખાને સેટ કરવાના વિકલ્પો, શીર્ષક વિસ્તારને ભરવા અને ભરેલા વિસ્તારની પારદર્શકતાની કક્ષા નક્કી કરવાની સગવડ પૂરી પાડે છે. અગાઉ જણાવ્યા પ્રમાણે તમે X-અક્ષની કિંમતોના ફૉન્ટ ફૉર્મેટ કરી શકો અને ફૉન્ટની અસરો બદલી પણ શકો છો. **Font** ટૅબ તમને યોગ્ય ફૉન્ટ પસંદ કરવા દે છે. જો તમારા કમ્પ્યૂટરમાં યોગ્ય ફૉન્ટ હોય તો તમે ગુજરાતી અને હિન્દી જેવી પ્રાદેશિક ભાષામાં પણ લેબલ પ્રદર્શિત કરી શકો છો. તમે ફૉન્ટના રંગ, વિવિધ રંગની રેખાઓ વડે શીર્ષકોના ફૉન્ટની રેખા ઢાંકવી, અક્ષર લખીને ઉપર લીટી દોરવી, વિવિધ રંગની રેખાઓ વડે શીર્ષકોના ફૉન્ટ નીચે લીટી દોરવી વગેરે જેવી ફૉન્ટની અસરો સામેલ પણ કરી શકો અને દૂર પણ કરી શકો છો. તમે શીર્ષકના ફૉન્ટને આકર્ષક અને સુશોભિત દેખાવ આપવા માટે Shadow અને Outline અસરો પણ આપી શકો છો. આ ઉપરાંત આકૃતિ 8.13માં દર્શાવ્યા પ્રમાણે તમે ડાયલૉગ બૉક્સના અંતિમ ટૅબ કે જેને **Alignment** નામ આપેલ છે તેને પસંદ કરીને તમે શાબ્દિક લખાણનું વહેણ (flow) તેમજ શાબ્દિક લખાણની ગોઠવણ (alignment) પણ કરી શકો છો. આ જ પ્રમાણે તમે Y-અક્ષનાં શીર્ષકોને પણ ફૉર્મેટ કરી શકો છો.

X-અક્ષને ફૉર્મેટ કરવી (Formatting X-Axis)

આલેખનાં ફૉર્મેટિંગ ટૂલબાર વડે તમે X-અક્ષને પણ ફૉર્મેટ કરી શકો છો. પ્રથમ યોગ્ય ભાગ પસંદ કરો, અહીં X-Axis પસંદ કરો અને પછી **Format Selection** બટન દબાવો. આથી સ્ક્રીન ઉપર તમે એક ડાયલૉગ બૉક્સ જોશો. આ ડાયલૉગ

બૉક્સનો ઉપયોગ કરીને તમે X-અક્ષના માપક્રમ (Scale) માટેના વિકલ્પો સુયોજિત કરી શકો છો કે જે સામાન્ય રીતે આપોઆપ થાય છે. તમે અક્ષનું સ્થાન એટલે કે અન્ય અક્ષ સાથે શરૂઆત, અંત અથવા અન્ય જગ્યાએ ક્યાં છેદાય તે સેટ કરી શકો છો. અક્ષનાં સ્થાન સાથે તમે અક્ષ ઉપર ક્યાં લેબલ ઇચ્છો છો તે પણ જણાવી શકો છો. તમે અક્ષની નજીક, અક્ષની બહાર, અક્ષની શરૂઆતમાં અથવા અક્ષનાં અંતમાં લેબલ રાખી શકો છો.

અનેક સમયે લેબલ ઘણા વર્ણનાત્મક હોય છે જેના કારણે આલેખનાં કદનું સંચાલન મુશ્કેલ બને છે અને એ સમયે તમે ફૉન્ટનું કદ નાનું કરવા ઇચ્છો. જો તમારે લેબલની જરૂર જ બિલકુલ ન હોય તો **Show labels** વિકલ્પ ઉપલબ્ધ છે. તમે ખરાની નિશાની કરી તે વિકલ્પને નાપસંદ કરી શકો, જેતે અક્ષનાં લેબલ કાઢી નાખશે. આ જ પ્રમાણે તમે Y-અક્ષને પણ ફૉર્મેટ કરી શકો.

ચાર્ટ વૉલ ફૉર્મેટ કરવી (Formatting the chart wall)

ચાર્ટ વૉલને ટૂલબાર ઉપરના વિકલ્પો પસંદ કરી અને **Format Selection** બટન પસંદ કરીને ફૉર્મેટ કરી શકાય છે. આ કાર્યથી એક ડાયલૉગ બૉક્સ સ્ક્રીન ઉપર દેખાશે જે આકૃતિ 8.11માં દર્શાવેલ છે તેના જેવું છે. આ ડાયલૉગ બૉક્સ તમને સીમારેખા(બૉર્ડર)ના ગુણધર્મો જેવા કે સીમારેખા દૃશ્યમાન કે અદૃશ્ય રાખવી, સીમારેખાનો પ્રકાર, સીમારેખાનો રંગ, સીમારેખાની જાડાઈ વગેરે બદલવાની સગવડ પૂરી પાડે છે. સીમારેખાના ગુણધર્મો ઉપરાંત આ ડાયલૉગ બૉક્સ તમને વિસ્તારનો રંગ અને પારદર્શકતાનું પ્રમાણ બદલવાની તક પૂરી પાડે છે.

ચાર્ટ ફ્લોરને ફૉર્મેટ કરવી (Formatting the chart floor)

ચાર્ટ ફ્લોર પણ ટૂલબાર ઉપરના વિકલ્પો પસંદ કરીને **Format Selection** બટન પસંદ કરીને બદલી શકાય છે. ચાર્ટ ફ્લોર ફૉર્મેટિંગ વિકલ્પ તમે આલેખ પસંદ કરી, ચાર્ટ ફ્લોર ઉપર રાઇટ ક્લિક કરી અને પોપ-અપ મેનૂમાંથી **Chart floor** વિકલ્પ પસંદ કરીને કરી શકો છો. આ કાર્ય કરવાથી ડાયલૉગ બૉક્સ સ્ક્રીન ઉપર દેખાશે જે તમને આલેખની સીમારેખા, વિસ્તાર અને પારદર્શકતાનું પ્રમાણ જેવા ગુણધર્મો સુધારવાની સગવડ આપે છે.

આલેખનાં લિજેન્ડ ફૉર્મેટ કરવાં (Formatting chart legends)

જો તમે લિજેન્ડ દાખલ કરવાનું સંપૂર્ણપણે ભૂલી ગયા હોય અથવા આલેખ બનાવતા સમયે છોડી દીધા હોય તો તમે આલેખ ઉપર રાઇટ ક્લિક કરી અને **Insert Legend** વિકલ્પ પસંદ કરીને ઉમેરી શકો છો. એક સમયે લિજેન્ડ તમને જોવામાં આવે તે પછી તમે તેને સુધારવાનું વિચારી શકો. આકૃતિ 8.11માં દર્શાવ્યા પ્રમાણે આલેખનાં ટૂલબાર વડે લિજેન્ડને તમે ફૉર્મેટ કરી શકો છો. જ્યારે તમે આલેખનાં ફૉર્મેટિંગ ટૂલબારમાંથી ફૉર્મેટિંગ ચાર્ટ લિજેન્ડ પસંદ કરી Format Selection બટન પર ક્લિક કરો છો ત્યારે હંમેશ મુજબ એક ડાયલૉગ બૉક્સ સ્ક્રીન ઉપર દેખાશે. જો તમારા ધ્યાનમાં આવ્યું હોય તો આલેખનાં લિજેન્ડ પણ અદૃશ્ય સીમારેખા સાથે લંબચોરસમાં ગોઠવાયેલાં હોય છે. આ ડાયલૉગ બૉક્સનો ઉપયોગ કરીને તમે લિજેન્ડ વિસ્તારની સીમારેખા ગોઠવી શકો, વિસ્તારને ફૉર્મેટ કરી શકો અને પારદર્શકતાનું પ્રમાણ બદલી શકો વગેરે. આ વિકલ્પો અગાઉ સમજાવ્યા પ્રમાણે જ કામ કરે છે. આ વિકલ્પો ઉપરાંત વધારાના અન્ય વિકલ્પો જેવા કે ફૉન્ટ, ફૉન્ટની અસરો અને આલેખના સ્થાન વિસ્તારમાં રહેલાં લિજેન્ડની જગ્યા પણ ડાયલૉગ બૉક્સમાં પૂરા પાડવામાં આવ્યા છે. આ વિકલ્પો જરૂરી છે કારણ કે આલેખનાં લિજેન્ડમાં કેટલીક શાબ્દિક માહિતી સાથે થોડાં ગ્રાફિક્સ (ચિત્રો) પણ હોય છે. આ ઉપરાંત લિજેન્ડની શાબ્દિક માહિતી આલેખમાં કઈ જગ્યાએ મૂકવાની છે તે પણ જણાવવું જરૂરી છે. જો કંઈ ન જણાવવામાં આવે તો આલેખની જમણી બાજુની જગ્યાએ રાખવામાં આવે છે પણ જો કે તમે લિજેન્ડને આલેખની ઉપર, આલેખની નીચે અથવા આલેખની ડાબી બાજુએ પણ રાખી શકો છો. તમે કોઈ પણ અનુકૂળ ફૉન્ટને તેના કદ અને દેખાવ જેમ કે ઘાટો, ત્રાંસો કે લખાણ નીચે લીટી દોરેલો પસંદ કરી શકો છો.

ડેટાની શ્રેણી ફૉર્મેટ કરવી (Formatting data series)

ડેટાની શ્રેણી બદલવા માટે તમે આલેખ પસંદ કરી તેના ઉપર રાઇટ ક્લિક કરો. એક ઊભું મેનૂ પ્રદર્શિત થશે. મેનૂમાંથી Format Data Series વિકલ્પ પસંદ કરો અને તે પછી તેને ફૉર્મેટ કરો. આ જ કાર્ય બીજી રીતે કરવા માટે આલેખનાં ફૉર્મેટિંગ ટૂલબાર (આકૃતિ 8.11માં દર્શાવ્યા પ્રમાણે) પરથી Data Series આઇટમ પસંદ કરો અને Format Selection બટન દબાવો. આ તમને ડાયલૉગ બૉક્સ દ્વારા ડેટા શ્રેણીનાં પ્રાચલો (parameters) બદલવાની તક આપશે.

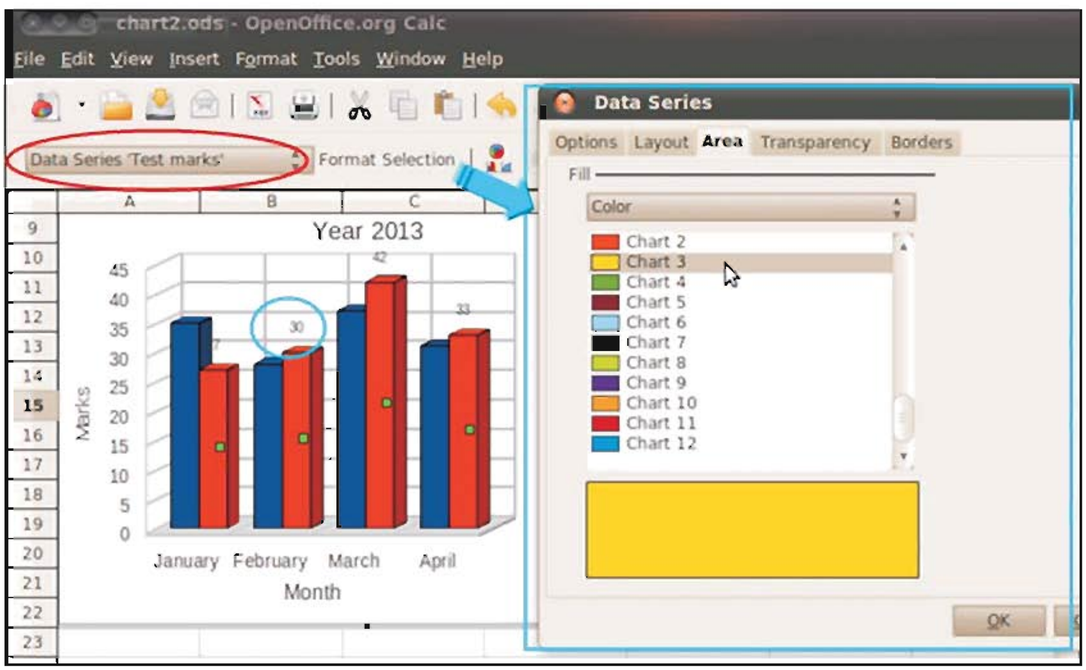
ડેટાની શ્રેણીને ફૉર્મેટ કરવાની અન્ય રીત નીચે વર્ણવી છે :

- આલેખ તૈયાર થયા પછી ડેટા શ્રેણી નિરૂપણ કરતી કોઈ પણ કૉલમ (ઊભી હરોળ) ઉપર ક્લિક કરો. જો તમે એક લાલ રંગની કૉલમ ઉપર ક્લિક કર્યું હશે તો તમે નિરીક્ષણ કરી શકશો કે બધી લાલ રંગની કૉલમ કાર્યરત થયેલી છે. આ ડેટા શ્રેણી રજૂ કરે છે.
- આકૃતિ 8.14માં લાલ રંગના કૉલમવાળી વર્ગપરીક્ષાના ગુણની ડેટા શ્રેણી પસંદ કરેલી પદ્ધતિથી દર્શાવી છે.
- આલેખનો ફૉર્મેટિંગ ટૂલબાર શીટમાં ઉપરની જગ્યાએ દેખાય છે. લાલ વર્તુળમાં આપેલો ભાગ જુઓ. તે **Data series 'Test Marks'** દેખાડે છે.
- **Format Selection** બટન દબાવો. તમે આલેખ સાથે બીજું એક લંબચોરસ ડાયલૉગ બૉક્સ જોઈ શકશો. બહારથી વાદળી રંગની રેખા સાથેનો લંબચોરસ વિસ્તાર જુઓ.
- તમે layout, area filling, transparency level વગેરેમાંથી યોગ્ય ટૅબ (tab) પસંદ કરો.
- આકૃતિ 8.14માં દર્શાવ્યા પ્રમાણે તમે જોઈ શકશો કે આપણે fill colour લાલથી પીળામાં ફેરવીએ છીએ તે પ્રમાણે આલેખના વિસ્તારનો દેખાવ બદલાય છે.

આ જ પ્રમાણે તમે બાકીની ડેટા શ્રેણીને પણ ફૉર્મેટ કરી શકો છો.

ડેટા લેબલ ઉમેરવા (Inserting data labels)

બીજું એક આ જ પ્રકારનું કાર્ય આલેખના દરેક ડેટા કૉલમ સાથે ડેટા લેબલ દાખલ કરવાનું છે. ધારો કે તમે આલેખના કોઈ ભાગ જેમ કે કૉલમ સાથે ખરેખરી કિંમત ઇચ્છો છો તો તમારે ફક્ત ડેટા શ્રેણીને કાર્યરત બનાવી રાઇટ ક્લિક જ કરવાની છે. તે પછી તમે **Insert Data Labels** પસંદ કરો. આકૃતિ 8.14માં તમે જોઈ શકશો કે ડેટા લેબલ અત્યાર સુધીમાં ઉમેરાઈ ગયા છે. આકૃતિ 8.14માં વર્તુળથી નિશાની કરેલી વાદળી રેખા સાથેની સંખ્યા જુઓ. આ જ પ્રમાણે અન્ય ડેટા શ્રેણીને તમે ફૉર્મેટ કરી શકો છો.



આકૃતિ 8.14 : ડેટા શ્રેણીને ફૉર્મેટ કરવી

વાસ્તવિક ઉદાહરણ (Working example)

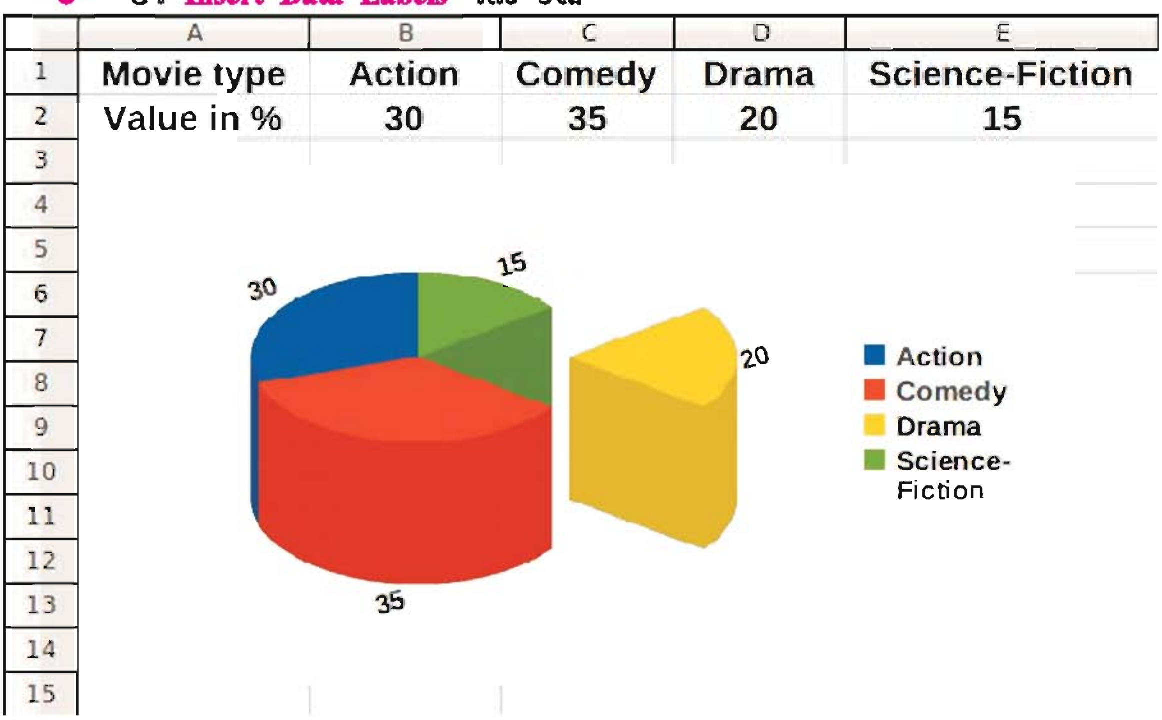
ચાલો, આપણે બીજા પ્રકારનો એક આલેખ તૈયાર કરીએ. નીચે આપેલા ડેટા લો. ધારો કે તે ડેટા ફિલ્મ(movies)ને લગતા છે. તમને બધાને ફિલ્મનો આનંદ માણવો ગમતો હશે. જો કે દરેક વ્યક્તિની અલગ પસંદ અને રુચિ હોય છે. કેટલાકને action મૂવી પસંદ હશે જ્યારે કેટલાકને Comedy પણ Science Fiction ફિલ્મનો એક વિશિષ્ટ પ્રેક્ષકવર્ગ હોય છે. આ સ્થિતિને વર્ણવતો ડેટા કોષ્ટક 8.2માં આપેલો છે.

Movie type	Action	Comedy	Drama	Science-Fiction
Value in %	30	35	20	15

કોષ્ટક 8.2 : અલગ અલગ ફિલ્મની પસંદ

કોષ્ટક 8.2માં આપેલી માહિતીને આધારે આપણે એક આલેખ બનાવીએ. નીચે જણાવેલ કાર્ય કરો :

- કેલ્સી વર્કશીટમાં કોષ્ટક 8.2માં આપેલા ડેટા દાખલ કરો અને યોગ્ય નામ આપી તેનો સંગ્રહ (save) કરો.
- ખાતરી કરો કે ડેટાનો વિસ્તાર A1થી E2 છે.
- **Insert → Chart** આદેશ આપો.
- Chart typeમાંથી **Pie** પ્રકારનો આલેખ પસંદ કરો.
- Lookમાં **3D** અને **Simple** પસંદ કરો.
- **Finish** બટન દબાવો.
- આથી આકૃતિ 8.15માં દર્શાવ્યા મુજબનો આલેખ તમે જોઈ શકશો.
- જો કે તમારા આલેખમાં ડેટા લેબલ દેખાતા નથી તેમજ Drama શ્રેણી બહાર પણ કાઢેલી નથી. ડેટા લેબલ ઉમેરવા તેમજ Dramaની શ્રેણી (પીળા રંગથી નિર્દેશિત) બહાર કાઢવા માટે નીચે આપેલાં પગલાં પ્રમાણે કાર્ય કરો :
- આલેખ પસંદ કરો. આથી તમે રાખોડી રંગની સીમારેખા આલેખની આસપાસ જોઈ શકશો.
- આલેખ ઉપર રાઇટ ક્લિક કરો, જેથી એક ઊભું મેનૂ પ્રદર્શિત થશે.
- હવે **Insert Data Labels** પસંદ કરો.



આકૃતિ 8.15 : પાઈ ચાર્ટ (Pie chart)

- તમે જોઈ શકશો કે આલેખ સાથે કિંમતો દર્શાવાયેલી છે.
- પાઈ ચાર્ટમાંથી એક ભાગ બહાર કાઢવા માટે માઉસ કી વડે ફક્ત તે વિસ્તાર પસંદ કરો. ત્યાર બાદ માઉસ વડે તે ભાગને ઇચ્છિત જગ્યા સુધી ખસેડો (ડ્રેગ કરો).

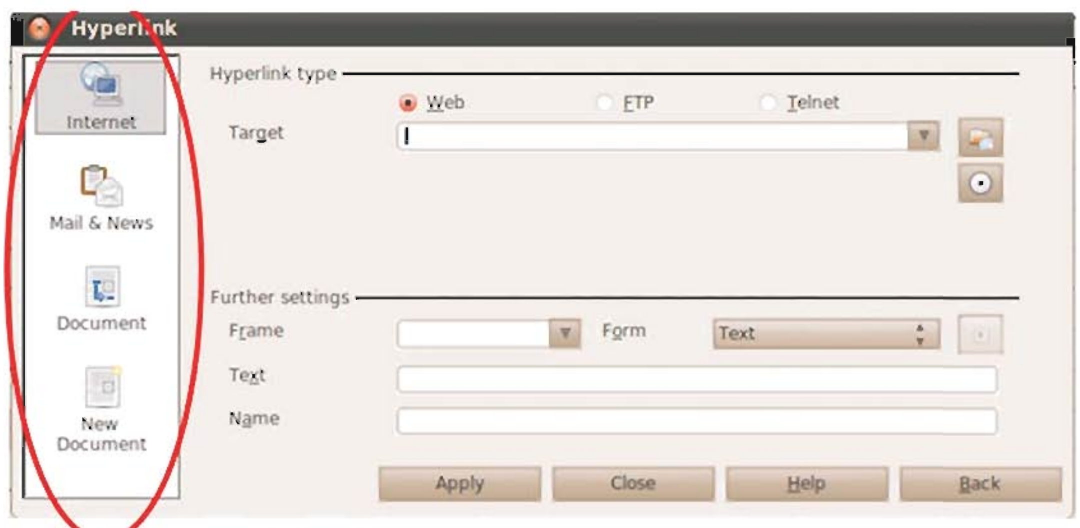
- જો તમે વધારે ફૉર્મેટિંગ કરવા ઇચ્છતા હો તો આલેખ પસંદ કરો. આલેખનું ફૉર્મેટિંગ ટૂલબાર તમને દેખાશે. તમારે જે ભાગનું ફૉર્મેટિંગ કરવું હોય તે પસંદ કરી **Format Selection** આપો. આ સગવડથી તમે આલેખનો પ્રકાર (ટાઇપ), આલેખ વિસ્તાર (એરિયા), ડેટા શ્રેણી, ફૉન્ટ, ફૉન્ટ અસરો, અક્ષનો દેખાવ, લિજેન્ડ, ગ્રીડ, ચાર્ટ ફ્લોર અને વૉલ વગેરેમાં ફેરફાર કરી શકો છો. અહીં આપણે ફૉન્ટને ઘાટા (બોલ્ડ) કરી શકીએ તેમજ ફૉન્ટનું કદ વધારી પણ શકીએ છીએ. આપણે ડેટા લેબલ 15 અંશ વર્તુળાકાર ફેરવેલાં પણ છે.
- આ જ પ્રમાણે તમે આલેખનાં લિજેન્ડ ફૉર્મેટ કરી શકો છો અને લિજેન્ડના ફૉન્ટનું કદ વધારી શકો છો.

આલેખ સાથે હાઇપરલિંક જોડવી (Adding hyperlink to the chart)

હાઇપરલિંક (Hyperlink) એ ડેટાનો સંદર્ભ છે જે પસંદ કર્યા પછી તેને અનુસરી શકાય છે. તમે જાણો છો તે પ્રમાણે હાઇપરલિંક કોઈ આખા દસ્તાવેજ કે દસ્તાવેજના ભાગને ચીંધે છે. એ શાબ્દિક લખાણ કે જે આવા જોડાણ(link) સમાવેશ કરે છે તેને હાઇપરટેક્સ્ટ (hypertext) કહેવામાં આવે છે. એક વાર આલેખ તૈયાર થઈ જાય પછી તમે આખા આલેખને અથવા આલેખના કોઈ ભાગ સાથે હાઇપરલિંક જોડી શકો છો. જ્યારે વપરાશકર્તા હાઇપરલિંક ઉપર ક્લિક કરે છે ત્યારે તે પૂર્વનિર્ધારિત જગ્યાએ વપરાશકર્તાને લઈ જાય છે. તમે વેબ ઉપર રહેલાં કોઈ દસ્તાવેજ, તમારા લોકલ એરિયા નેટવર્ક / કમ્પ્યૂટરના દસ્તાવેજ કે નવા દસ્તાવેજ સાથે હાઇપરલિંક જોડી શકો છો.

કોઈ પણ પ્રકારની હાઇપરલિંક માટે તમારી પાસે આલેખ હોવો જોઈએ. તમારે સૌપ્રથમ આલેખ તૈયાર કરવો જોઈએ. કોઈ વેબ (ઇન્ટરનેટ) દસ્તાવેજ સાથે હાઇપરલિંક બનાવવા માટે નીચે જણાવેલ પગલાંને અનુસરો :

- સૌપ્રથમ ડેટા દાખલ કરો અને યોગ્ય પ્રકારનો આલેખ બનાવો.
- આલેખને ચકાસો અને માઉસ ક્લિક વડે પસંદ કરો.
- **Insert → Hyperlink** આદેશ આપો.
- તે તમને ગંતવ્ય સ્થાનની લિંક માટે ચાર વિકલ્પો આપશે. તમે આલેખ(અથવા પસંદ કરેલા ભાગ)ને Internet, Mail & News, Document અથવા New Document સાથે હાઇપરલિંક કરી શકો છો. આકૃતિ 8.16માં આલેખને જોડવા માટેના ચાર વિકલ્પો (લાલ અંડાકારથી ચિહ્નિત કરેલાં) દર્શાવ્યા છે.



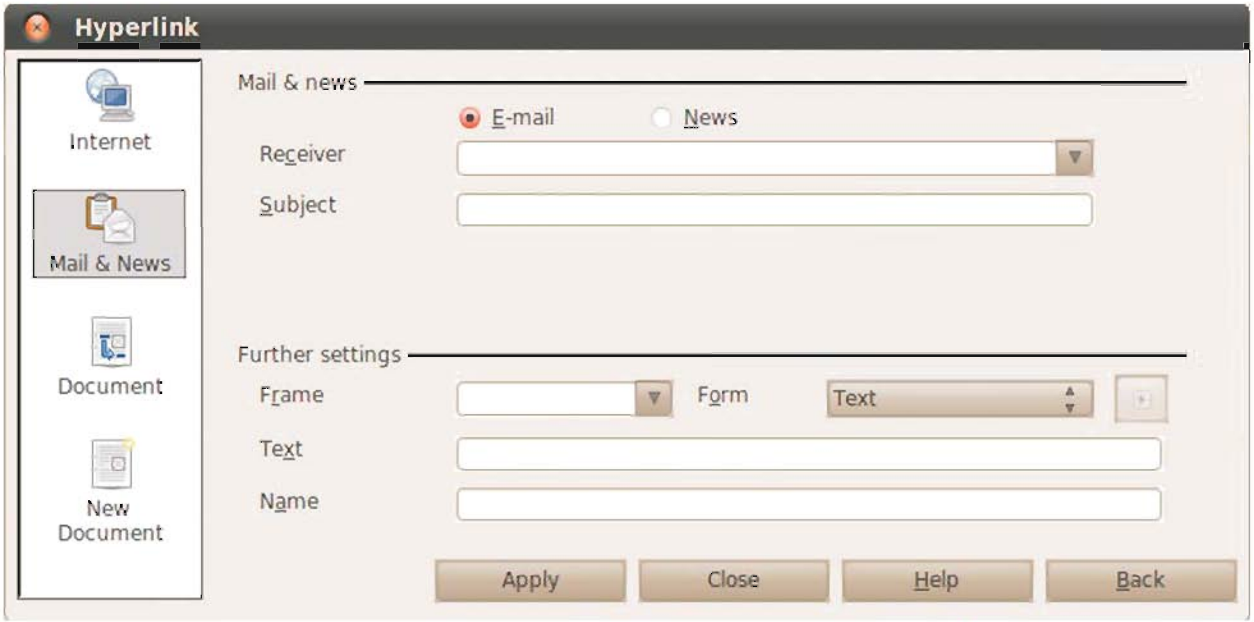
આકૃતિ 8.16 : હાઇપરલિંકના વિકલ્પો

પૂર્વનિર્ધારિત રીતે પહેલો વિકલ્પ **"Internet"** હાઇલાઇટ થયેલો હોય છે અને **Target**માં *Google.com.* જેવું અંતિમ મુકામ URL આપી શકો છો. જો તમારે આ લિંકનો પ્રયોગ કરવો હોય તો ઇન્ટરનેટ જોડાણ હોવું જરૂરી છે. જો બધું બરાબર હોય તો ક્લિક કરવાથી આલેખ તમને ગૂગલનાં મુખ્ય પાના ઉપર લઈ જશે. આના વિકલ્પરૂપે કોઈ પણ પ્રકારની હાઇપરલિંક **Ctrl + Enter** દબાવીને અનુસરી શકાય છે.

તમે બધી જરૂરી માહિતી દાખલ કરી દીધા પછી Apply પસંદ કરી શકો છો. ડાયલૉગ બૉક્સ બંધ કરવા માટે Close બટનનો ઉપયોગ કરો.

મેઈલ સાથે આલેખ જોડવો (Linking a chart to a mail)

જો તમારે મેઈલ ઍન્ડ ન્યૂઝ સાથે આલેખ જોડવો હોય તો **Mail and News** વિકલ્પ પસંદ કરો. તમે એક અલગ ડાયલૉગ બૉક્સ સ્ક્રીન ઉપર જોશો. જ્યારે તમે **Mail & News** સાથે જોડાણ કરો છો ત્યારે આકૃતિ 8.17માં દર્શાવ્યા પ્રમાણેના વિકલ્પો જોવા મળશે.

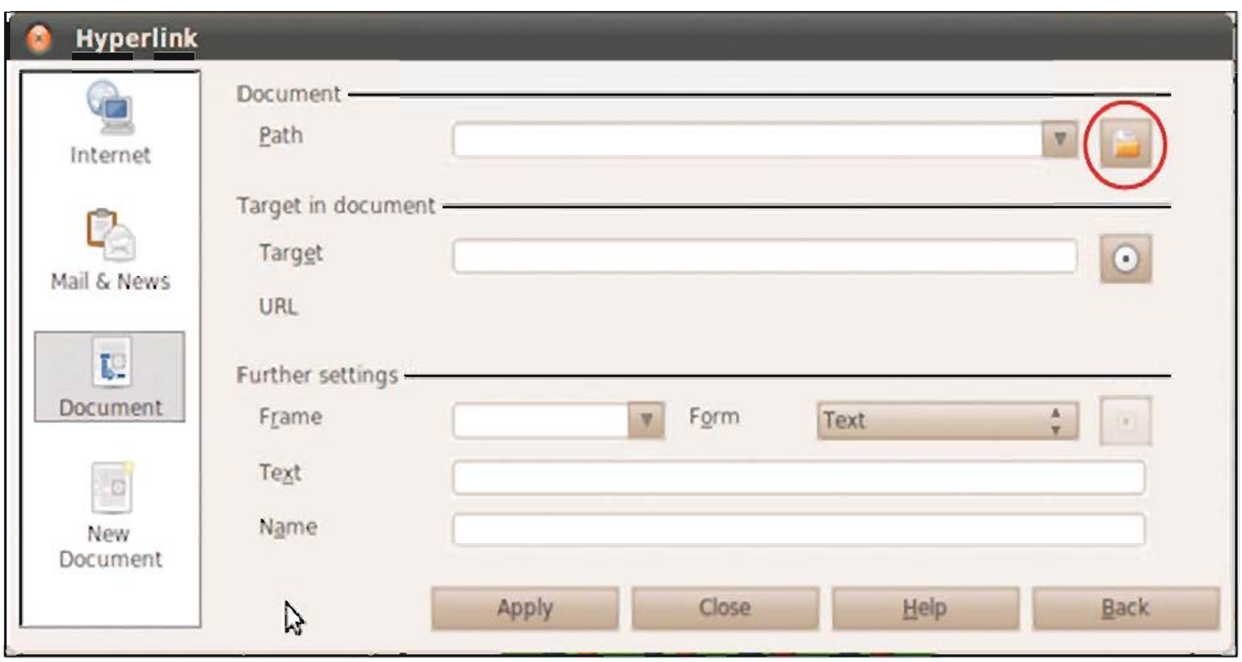


આકૃતિ 8.17 : હાઈપરલિંકમાં Mail and Newsના વિકલ્પો

તમારે મેઈલનાં receiver અને subjectની માહિતી આપવી પડશે. જ્યારે તમે **Apply** વિકલ્પ આપશો ત્યારે તે આલેખ મેળવનારનાં સરનામાં ઉપર ટપાલ માટે કતારમાં હશે. જો કે આ પ્રક્રિયા ઑફલાઈન રહે છે અને મેઈલ ક્લાયન્ટ (mail client) પશ્ચાદ્ભૂમિમાં રાખવાની જરૂર રહે છે. એ જ પ્રમાણે, **News**નાં વિકલ્પ માટે તમારે ન્યૂઝ સર્વર(news server)ની રચના (configure) કરવાની જરૂર પડે છે. અંતમાં **Close** બટન દબાવો.

દસ્તાવેજ સાથે આલેખ જોડવો (Linking a chart to a document)

દસ્તાવેજ સાથે આલેખ જોડવા માટે ત્રીજો વિકલ્પ Document પસંદ કરો. આથી તમે આકૃતિ 8.18માં દર્શાવ્યા પ્રમાણેનો સ્ક્રીન જોશો.



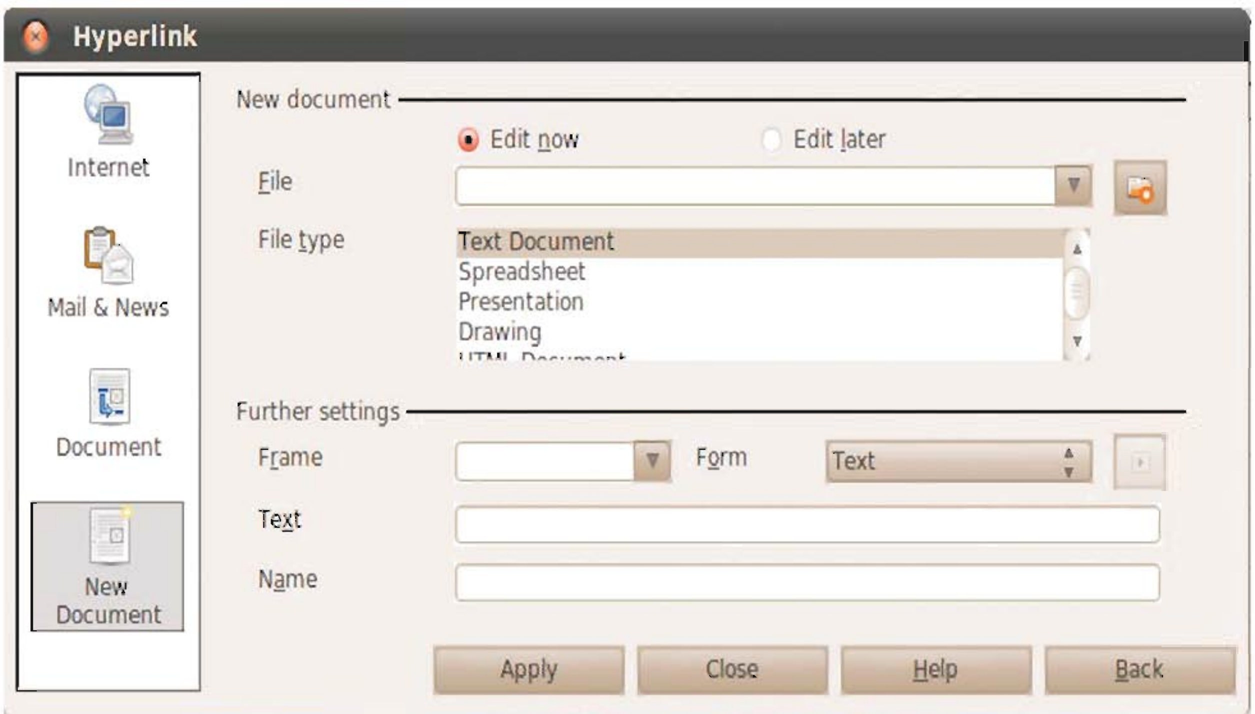
આકૃતિ 8.18 : દસ્તાવેજ સાથે આલેખને જોડવો

જ્યારે તમે નાનો આલેખ બનાવ્યો હોય ત્યારે આ વિકલ્પ ઉપયોગી છે કે જેમાં વધારે સમજણ બીજા દસ્તાવેજમાં આપેલી હોય છે. આવા સંજોગોમાં વધારે માહિતી માટે તમે આલેખને અન્ય દસ્તાવેજ સાથે જોડો. ટૂંકમાં તમે એમ વિચારો કે અમુક દસ્તાવેજ આલેખ સાથે સંબંધિત છે તો તમે આ સગવડ દ્વારા આલેખ સાથે જોડી શકો છો. દસ્તાવેજનો સંપૂર્ણ પાથ (path) અંતિમ સ્થાન સૂચવે છે. જો તમને સંપૂર્ણ પાથ યાદ ન હોય તો તમે pathની બાજુમાં રહેલા શાબ્દિક બૉક્સની પાસેના Browse બટન(લાલ લીટીથી ગોળ ચિહ્નિત) નો ઉપયોગ કરીને દસ્તાવેજને જોઈ શકો છો (બ્રાઉઝ કરી શકો છો). તમે **Target** વિકલ્પનો ઉપયોગ કરીને અંતિમ મુકામના દસ્તાવેજમાં લક્ષ (target) પણ આપી શકો છો.

અંતમાં **Apply** બટન પર ક્લિક કરો. ડાયલૉગ બૉક્સ બંધ કરવા માટે અને કાર્યની સમાપ્તિ માટે **Close** બટન દબાવો.

નવા દસ્તાવેજ સાથે આલેખ જોડવો (Linking a chart to a new document)

નવા દસ્તાવેજ સાથે આલેખને જોડવા માટે ચોથો વિકલ્પ **New Document** પસંદ કરો. તમે **File** અને **File type** આપો. તમે આ વિકલ્પ વડે સ્પ્રેડશીટ, વર્ડ પ્રોસેસર, ડ્રોઈંગ વગેરે જેવી વિવિધ પ્રકારની ફાઈલ બનાવી શકો છો. એક વાર ફાઈલ બની જાય તે પછી તેમાં સુધારા-વધારા કરવા માટે તે ઉપલબ્ધ બની જાય છે અને પૂર્વનિર્ધારિત રીતે **Edit now** વિકલ્પ પસંદ થયેલો છે. જો તમે હમણાં કોઈ પણ પ્રકારના સુધારા કરવા ન ઇચ્છતા હોય તો **Edit later** વિકલ્પ તમે પસંદ કરી શકો છો. ઉપર જણાવેલી સ્થિતિ આકૃતિ 8.19માં દર્શાવેલી છે.



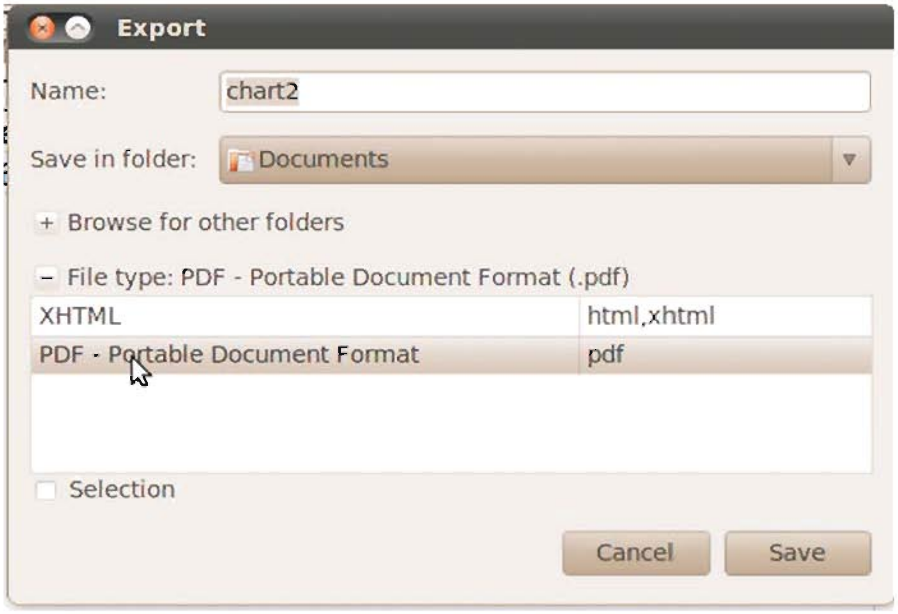
આકૃતિ 8.19 : નવા દસ્તાવેજ સાથે આલેખ જોડવો

અંતમાં, **Apply** વિકલ્પ આપો. ડાયલૉગ બૉક્સ બંધ કરવા માટે **Close** બટનનો ઉપયોગ કરો.

આલેખ નિકાસ કરવો અને અન્ય કાર્યો (Exporting the chart and other operations)

તમે આલેખ સાથેના દસ્તાવેજને PDF અનુલંબન સાથેની સુવાહ્ય (portable) દસ્તાવેજ ફાઈલ તરીકે નિકાસ કરી શકો છો. આ કાર્ય માટે નીચે પ્રમાણે પગલાં ભરો :

- સૌપ્રથમ **File → Export** આદેશ આપો. આથી આકૃતિ 8.20માં દર્શાવ્યા પ્રમાણેનો સ્ક્રીન તમે જોઈ શકશો.

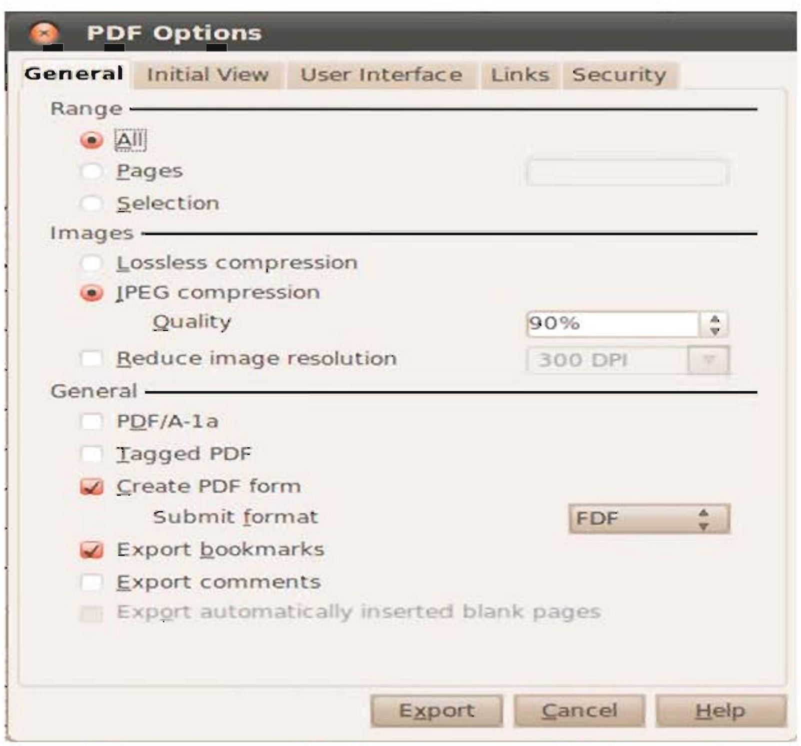


આકૃતિ 8.20 : દસ્તાવેજની નિકાસ કરવી

ફાઈલની નિકાસ XHTML (વેબ પેઇજ) અથવા PDF (પૉર્ટેબલ ડૉક્યુમેન્ટ ફાઇલ) જેવા બે વિકલ્પોમાં કરી શકો તે તમને જોવા મળશે.

- **PDF** પસંદ કરો.
- તમારા ઇચ્છિત ફોલ્ડરમાં ફાઈલનો સંગ્રહ કરો.

જ્યારે તમે તમારી પસંદ PDF તરીકે આપો છો ત્યારે આકૃતિ 8.21માં દર્શાવ્યા પ્રમાણે અનેક વિવિધ પ્રાચલો(પેરામીટર)ની માહિતી લઈ આખરે PDF ફાઈલમાં પરિણમતી ફાઈલ માટેનું ડાયલૉગ બૉક્સ તમે જોશો. આકૃતિ 8.21માં દર્શાવેલા ડાયલૉગ બૉક્સનો ઉપયોગ કરીને વિવિધ વિકલ્પો જેમ કે પાનાંની સંખ્યા, ઈમેજ(ગ્રાફિક્સ)નું ફૉર્મેટ (જો તેમાં સમાવેલ હોય તો) વગેરેને નિયોજિત કરી શકાય છે. જો તમે અમુક પાનાંની જ નિકાસ કરવા માટે પસંદ કરેલાં હોય તો PDFમાં પરિણમતા દસ્તાવેજમાં ફક્ત તે જ પાનાંની નિકાસ થશે. પૂર્વ નિર્ધારિત રીતે તે આખો દસ્તાવેજ હોય છે.



આકૃતિ 8.21 : દસ્તાવેજની PDF સ્વરૂપમાં નિકાસ કરવી

આ ઉપરાંત દસ્તાવેજની બધી ઈમેજ jpeg સ્વરૂપમાં સંગ્રહ થાય છે. અહીં તમારી પાસે નુકસાનરહિત ઈમેજ સંકોચન અને આપેલા ગુણવત્તાના ટકા સાથે **JPEG** ઈમેજ સંકોચનના વિકલ્પો હોય છે. જો તમે પસંદ ન કરેલો હોય તો **Create PDF** વિકલ્પ પસંદ કરો. સામાન્ય રીતે આ બધા વિકલ્પો પૂર્વનિર્ધારિત હોય છે અને કમ્પ્યૂટર (સિસ્ટમ) દ્વારા પોતાની જાતે જ પસંદ કરી લેવામાં આવે છે. તમે Generalના પૂર્વ નિર્ધારિત વિકલ્પોમાં કોઈ ફેરફાર ન પણ કરો. તમે સીધા જ **Export** બટન દબાવો. આ કરવાથી આલેખ સાથેનો દસ્તાવેજ PDF સ્વરૂપમાં નિકાસ થશે.

આલેખની નકલ કરવી (Copying the chart)

આલેખની નકલ અન્ય પ્રોગ્રામ જેવા કે Writer કે Impressમાં કરવા માટે તમારે તે આલેખ પસંદ કરી કૉપી (copy) કરો અને જ્યાં જરૂર હોય તે જગ્યાએ પેસ્ટ (paste) કરો. આલેખની નકલ કરવા માટે નીચે જણાવેલાં સરળ પગલાં પ્રમાણે કાર્ય કરો :

- માઉસ કી વડે આલેખ પસંદ કરો.
- રાઇટ ક્લિક કરીને **Copy** અથવા **Cut** વિકલ્પ આપો.
- તમે **Edit** મેનૂમાં જઈને જરૂરી વિકલ્પ પણ પસંદ કરી શકો છો.
- હવે જેમાં નકલ કરવાની છે તે દસ્તાવેજમાં જાઓ. હવે **Paste** આદેશ આપો.
- જેમાં નકલ કરી તે ફાઈલનો સંગ્રહ (save) કરો.

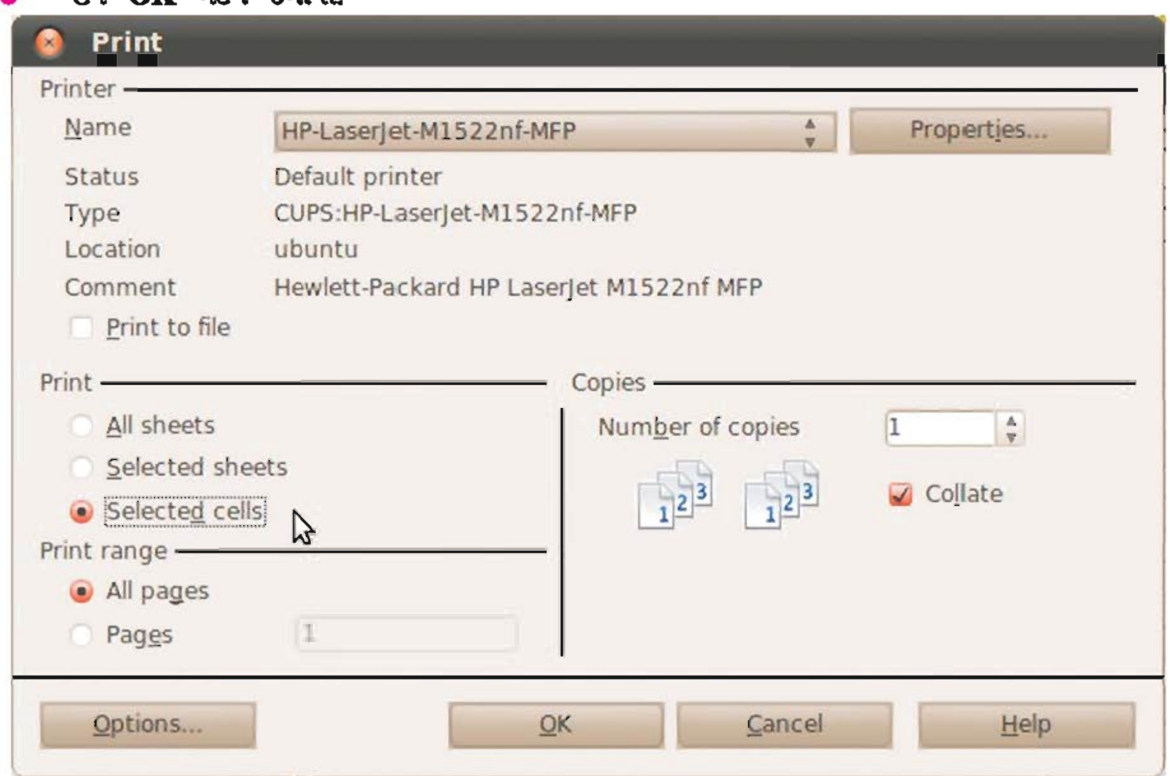
આલેખ દૂર કરવો (Deleting the chart)

આલેખ દૂર કરવા (ડિલીટ કરવા) માટે સૌપ્રથમ તે પસંદ કરો અને Delete બટન દબાવો.

આલેખનું મુદ્રણ કરવું (Printing the chart)

અનેક સમયે તમારે આલેખ અને વર્કશીટની માહિતી છાપવાની જરૂર પડે છે. આલેખ છાપવા માટે નીચે જણાવેલાં સરળ પગલાં પ્રમાણે કાર્ય કરો :

- **File → Print** આદેશ આપો. આથી આકૃતિ 8.22માં દર્શાવ્યા પ્રમાણેનું ડાયલૉગ બૉક્સ તમે જોઈ શકશો.
- ઉપલબ્ધ પ્રિન્ટરની યાદીમાંથી પસંદગી કરો.
- હવે **OK** બટન દબાવો.



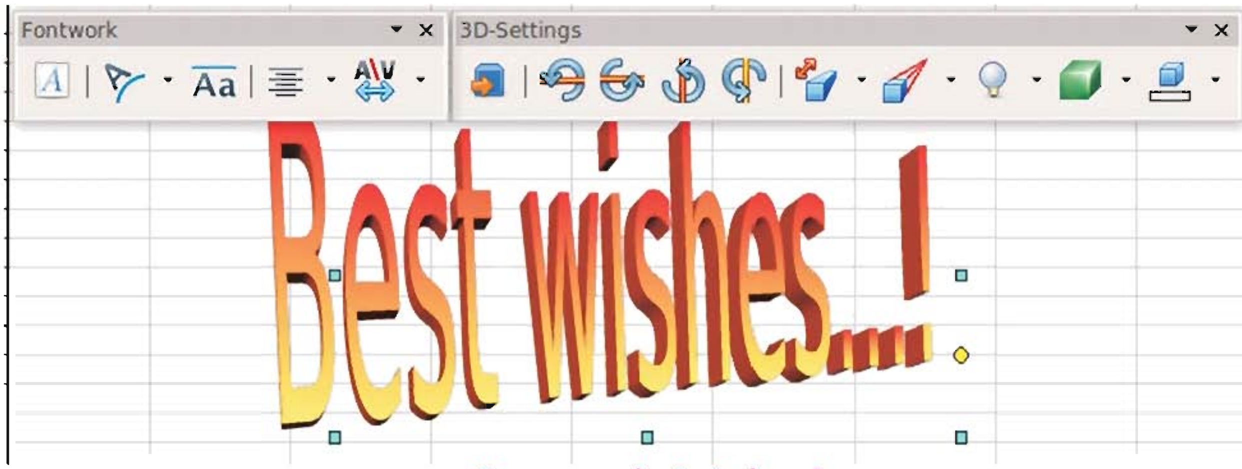
આકૃતિ 8.22 : દસ્તાવેજ છાપવો

અહીં એ નોંધ કરો કે કોઈ પણ દસ્તાવેજ છાપતાં પહેલા તે પાનું છાપ્યા પછી કેવું દેખાશે (પ્રિવ્યૂ preview) તે તપાસી લેવું ઘણું અગત્યનું છે. તે (પ્રિવ્યૂ) તમને તમારો દસ્તાવેજ કેવી રીતે છપાશે તેનું સ્પષ્ટ ચિત્ર આપશે.

અન્ય મલ્ટિમીડીયા ઑબ્જેક્ટ ઉમેરવા (Inserting other multimedia objects)

તમે વર્કશીટમાં આલેખ ઉપરાંત અન્ય કોઈ પણ મલ્ટિમીડિયા ઑબ્જેક્ટ ઉમેરી શકો છો. મલ્ટિમીડીયા ઑબ્જેક્ટમાં ચિત્રો, કેમેરાની ક્લિપ, વીડિયો, મૂવી ક્લિપ, એનિમેશનવાળી ક્લિપ જેમ કે GIF ફાઈલ અને 3D શાબ્દિક માહિતી જેવાં વિવિધ માધ્યમોનો સમાવેશ કરવામાં આવે છે. આ કાર્ય માટે આપણે સામાન્ય રીતે Insert જેવાં સરળ મેનુ આદેશનો ઉપયોગ કરીએ છીએ. એટલે કે, કેમેરા ક્લિપ કે તમારા મોબાઇલ ફોનની ક્લિપ ઉમેરવા માટે તમારે સૌપ્રથમ તમારા કમ્પ્યૂટરને તે એકમ સાથે જોડવાની જરૂર પડે છે. તમે તાર સાથે જોડાણ (USB કનેક્ટર સાથેનો તાર (Cord) મોબાઈલ ફોન કે કેમેરા સાથે હોય તો) અથવા બ્લૂટૂથ ટેકનોલૉજીનો ઉપયોગ કરી શકો છો. તમારા ફોટોગ્રાફ જેવો ઑબ્જેક્ટ તમારા કમ્પ્યૂટરમાં હોય તો તે ઇચ્છિત ફોટોગ્રાફ ઉમેરવા માટે **Insert → Picture** આદેશ આપો. આ જ પ્રમાણે તમે કેલ્સી દસ્તાવેજમાં કોઈ પણ મૂવી કે એનિમેશનવાળી ક્લિપ (અથવા GIF ફાઈલ) ઉમેરી શકો છો.

તમે વર્કશીટમાં 3D શાબ્દિક માહિતી પણ ઉમેરી શકો છો. કેલ્સીમાં 3D શાબ્દિક લખાણ ગ્રાફિકલ ટેક્સ્ટ (graphical text) આર્ટ અથવા ફોન્ટવર્ક (fontwork) તરીકે ઓળખાય છે. આકૃતિ 8.23માં એક ઉદાહરણ આપેલ છે.



આકૃતિ 8.23 : કેલ્સીમાં ફોન્ટવર્ક

ફોન્ટવર્ક (fontwork) ઑબ્જેક્ટ બનાવવા માટે નીચે જણાવેલાં પગલાં પ્રમાણે કાર્ય કરો :

- Drawing ટૂલબાર કે **Fontwork** ટૂલબાર ઉપર રહેલા **Fontwork Gallery** આઇકોન ઉપર ક્લિક કરો.
- જો ડ્રૉઇંગ ટૂલબાર અથવા ફોન્ટવર્ક ટૂલબાર સ્ક્રીન ઉપર દેખાતાં ન હોય તો **View** મેનૂમાંથી **Toolbars** વિકલ્પ પસંદ કરો જેથી ટૂલબાર સ્ક્રીન ઉપર દેખાય (**View → Toolbars**).
- **Fontwork Gallery** ડાયલૉગ બૉક્સમાંથી ફોન્ટવર્ક સ્ટાઇલ નક્કી કરી OK બટન દબાવો. તમે ફોન્ટવર્ક ગૅલેરીમાંથી સ્ટાઇલ (style) વાપરી શકો, Fontwork **shape** વડે તમે આકારમાં ફેરફાર કરી શકો, તમે અક્ષરોની ઊંચાઈ બદલી શકો, તમે શાબ્દિક લખાણની ગોઠવણ (alignment) કરી શકો અને શાબ્દિક લખાણમાં બે અક્ષરો વચ્ચેની જગ્યા સેટ કરી શકો. તમે રેખાનો રંગ, ભરેલો રંગ (fill colour) અને રંગ ભરવાની શૈલી (fill style) જેવાં વધારાનાં ગુણધર્મોમાં ફેરફાર કરવા માટે ડ્રૉઇંગ ટૂલબારનો ઉપયોગ કરી શકો.
- તમારા દસ્તાવેજમાં ફોન્ટવર્ક ઑબ્જેક્ટને ઉમેરી શકો છો. તમારું પોતાનું લખાણ ઉમેરવા માટે ઑબ્જેક્ટ ઉપર ડબલ ક્લિક કરો.
- Edit મોડમાંથી બહાર નીકળવા માટે Esc દબાવો.

એક વાર ફૉન્ટવર્ક ઓબ્જેક્ટ ઉમેરી દીધા પછી તેમાં સુધારા-વધારા કરવા માટે (edit) ફક્ત તેના ઉપર ક્લિક કરો. ફૉન્ટવર્ક ટૂલબાર પ્રદર્શિત થશે. શાબ્દિક લખાણને સુધારો. તમારા સુધારા-વધારા કરવાનું કાર્ય પૂર્ણ થાય તે પછી Esc કી દબાવો.

સારાંશ

આ પ્રકરણમાં આપણે વિવિધ પ્રકારના આલેખો, તેની અગત્ય અને તેના ઉદ્દેશો વિશે શીખ્યા. આપણે કેલ્સી સ્પ્રેડશીટ પૅકેજમાં આલેખ કઈ રીતે ઉમેરી શકાય તેમજ તેમાં કેવી રીતે ફેરફાર કરી શકાય તે બાબત પણ જોયું. આલેખના અનેક ભાગ હોય છે. આ પ્રકરણમાં આલેખના સામાન્ય ભાગની વિસ્તૃત સમજણ આપી. આપણે વધારે અસરકારક રજૂઆત કરવા માટે આલેખના ભાગને કઈ રીતે વધારે સારા બનાવી શકાય અને સુધારી શકાય તે બાબત પણ શીખ્યા. આ પ્રકરણમાં આપણે આલેખના બે પ્રકાર કૉલમ ચાર્ટ અને પાઈ ચાર્ટ બાબત પણ ઊંડાણપૂર્વક ચર્ચા કરી. આપણે આલેખ સાથે હાઇપરલિંક કઈ રીતે જોડી શકીએ તે પણ જોયું. આ ઉપરાંત આ પ્રકરણમાં આલેખને છાપવો, આલેખની નકલ કરવી, આલેખ દૂર કરવો અને આલેખનો સુવાહ્ય દસ્તાવેજ (PDF) તરીકે નિકાસ કરવો તે બાબત પણ જોયું. અંતમાં 3D અને મલ્ટિમીડિયા ઑબ્જેક્ટ સ્પ્રેડશીટ દસ્તાવેજમાં કઈ રીતે ઉમેરી શકાય તે બાબત પણ ચર્ચા કરી.

સ્વાધ્યાય

1. કેલ્સીમાં આલેખ બનાવવાના ફાયદા સમજાવો.
2. આલેખના જુદા જુદા પ્રકારો વિશે એક એક લીટીમાં સમજૂતી આપો.
3. કેલ્સીમાં આલેખ કઈ રીતે ઉમેરી શકાય તે સમજાવો.
4. ચાર્ટ વિઝાર્ડના વિવિધ ભાગની યાદી બનાવો.
5. આલેખના વિવિધ ભાગની યાદી સાથે દરેકનું એક એક લીટીમાં વર્ણન કરો.
6. એક વખત આલેખ તૈયાર થઈ જાય પછી તેનો પ્રકાર કઈ રીતે બદલી શકાય ?
7. આલેખના અક્ષરને ફૉર્મેટ કઈ રીતે કરી શકાય ?
8. આલેખ સાથે હાઇપરલિંક જોડી શકાય ? કઈ રીતે ?
9. નીચેનામાંથી સાચો વિકલ્પ પસંદ કરો :

(1) આલેખ છાપવા માટે નીચેનામાંથી કયો વિકલ્પ વપરાય છે ?

(a) Insert → Chart (b) File → View
(c) File → Print (d) View → Chart

(2) કેલ્સીમાં બનાવેલા આલેખમાં કેટલા અક્ષ હોઈ શકે ?

(a) બે (b) ત્રણ (c) બે અથવા ત્રણ (d) ચાર

(3) કેલ્સીમાં 3D શાબ્દિક લખાણ દાખલ કરવા માટે નીચેનામાંથી શું વપરાય છે ?

(a) ફૉન્ટવર્ક (b) આર્ટવર્ક
(c) ડ્રૉઇંગવર્ક (d) ગ્રાફવર્ક

(4) આલેખને જોડવા માટે નીચેનામાંથી શું વપરાય છે ?

(a) એક્ટિવલિંક (b) હાઇપરલિંક
(c) ડ્રૉઇંગલિંક (d) ક્નેક્શન લિંક

(5) નીચેનામાંથી કયો વિકલ્પ આલેખનો પ્રિવ્યૂ દેખાડે છે ?

(a) XHTML તરીકે સંગ્રહ (b) પેઇજ પ્રિવ્યૂ
(c) આલેખની નિકાસ (d) આપેલ કોઈ પણ નહીં

(6) ડેટા વિસ્તારમાં ઊભી હરોળ (કૉલમ) અથવા આડી હરોળ(રો)ને અચળ બનાવવા માટે - નિરપેક્ષ બનાવવા માટે ક્યું ચિહ્‌ન વપરાય છે ?

(a) # (b) $ (c) & (d) %

(7) નીચેનામાંથી શેની સાથે કેલ્સી આલેખને જોડી શકાય છે ?

(a) હયાત દસ્તાવેજ સાથે (b) નવા દસ્તાવેજ સાથે
(c) વેબ પેઇજ સાથે (d) આપેલા તમામ વિકલ્પો

પ્રાયોગિક સ્વાધ્યાય

1. અંગ્રેજી, વિજ્ઞાન, ગણિત વગેરે વિષયોમાંથી તમારા ગુણ શોધો. દરેક વિષયમાં વર્ગના સૌથી વધારે ગુણ શોધો. દરેક વિષયના મહત્તમ ગુણ સાથે તમારા ગુણની સરખામણી કરતો કૉલમ ચાર્ટ બનાવો.
2. તમારા પડોશમાં દરેક વ્યક્તિ ક્યું છાપું વાંચે છે તેનું એક નાનું સર્વેક્ષણ કરો. ડેટા ભેગો કરો, કોરા પત્રક ઉપર ગોઠવણી કરો અને તેના આધારે એક 3D પાઈ ચાર્ટ બનાવો.
3. કોઈ એક ગાણિતિક પદાવલિ જેમ કે Y= 2*X લો. નીચે જણાવ્યા પ્રમાણે Xની જુદી જુદી કિંમત લો :

Value of X	Value of Y
1	?
3	?
5	?
8	?
6	?

નીચે જણાવ્યા પ્રમાણે કરો :

(a) કેલ્સી વર્કશીટમાં કિંમત દાખલ કરો.

(b) સૂત્ર વાપરીને Xની દરેક કિંમત માટે Yની કિંમત શોધો.

(c) X અને Y કિંમતો સાથે લઈને લાઈન ચાર્ટ બનાવો.

(d) તમને ઈચ્છા થાય તે પ્રમાણે આલેખને ફૉર્મેટ કરો.

4. નીચે જણાવેલ કોઈ પરિવારનો અંદાજપત્ર સંબંધિત ડેટા લો :

Category	July
Provision	1000
Milk	500
Vegetables	500
Servant	800
Electricity	1500
Gas	500
Fees	1500
Newspaper	100
Cable	200
Mobile	200

નીચે જણાવ્યા પ્રમાણે કરો :

(a) આ ડેટા સ્પ્રેડશીટમાં દાખલ કરો અને પાઈ ચાર્ટ બનાવો.

(b) આલેખ તૈયાર થઈ ગયા પછી તમારી પસંદ પ્રમાણે ફૉર્મેટ કરો. દા.ત. કોઈ ભાગનો રંગ બદલો, ફૉન્ટની શૈલી બદલો, લિજેન્ડ ઉમેરો / દૂર કરો, આલેખમાં ડેટા લેબલ ઉમેરો, આલેખમાંથી થોડા ખંડ બહાર કાઢો.

(c) XHTML ફાઈલ તરીકે આલેખની નિકાસ કરો. વેબ બ્રાઉઝર ખોલો અને જુઓ કે તે આલેખ કેવો દેખાય છે.

સમસ્યા અને સમસ્યાનું નિરાકરણ

દરેક મનુષ્ય પોતાની રોજિંદી જિંદગીમાં અનેક સમસ્યાઓનું નિરાકરણ લાવે છે. ચાલો, આપણે આવી એક ક્રિયાની ચર્ચા કરીએ. ધારો કે તમે તમારા મિત્રો સાથે સંતાકુકડી રમત (Hide and seek) રમો છો. આ રમતમાં આપણી સમસ્યા એ છે કે આપણા છુપાયેલા મિત્રની જગ્યા શોધવાની છે. આ કાર્ય ફક્ત તો જ થઈ શકે જો આપણે તેની છુપાવાની જગ્યા વિશે આગાહી કરી શકીએ.

ચાલો, આપણે બીજી એક સમસ્યા લઈએ, ધારો કે આપણે એક ગાણિતિક સમીકરણનો ઉકેલ લાવવો છે જેમાં આપણને $2x + 4 = 0$ સમીકરણ આપેલું છે તો xની શું કિંમત હોઈ શકે ? આ સમસ્યાના જવાબ માટે આપણે સમીકરણને $2x = -4$ની રીતે ગોઠવણી કરવી પડશે, અને પછી $x = -4/2$, આથી અહીં જવાબ -2 મળશે.

ચાલો, આપણે અન્ય ઉદાહરણ લઈએ. આપણે અંગ્રેજી શબ્દ "Eloquent" નો અર્થ શોધવો છે. આ માટે આપણે એક અંગ્રેજી ભાષાનો શબ્દકોશ જોઈશે. શબ્દકોશમાં હજારો શબ્દો અને તેના અર્થ હોય છે. છતાં આપણે ઘણી ઝડપથી ઈચ્છિત શબ્દનો અર્થ શોધી કાઢીએ છીએ. અત્યાર સુધીમાં ક્યારેય પણ તમે વિચાર્યું છે કે આ કાર્ય તમે આટલી ઝડપથી કઈ રીતે કરી શકો છો? હકીકતમાં શબ્દકોશમાં શબ્દો ક્રમિક રીતે ગોઠવાયેલા હોય છે તેનો આપણે ફાયદો લઈએ છીએ. આપણે જે શબ્દો "E" થી શરૂ ન થતા હોય તેને ઝડપથી કાઢી નાખીએ છીએ અને "E" થી શરૂ થાય તેવા શબ્દો શોધવાનું શરૂ કરીએ છીએ, એ જ રીતે બીજો અક્ષર "*l*" હોય તેવા શબ્દો શોધવા માટે કાઢી નાખવાની પદ્ધતિ (elimination method) નો ઉપયોગ કરીએ છીએ. આપણે આ કાઢી નાખવાની પદ્ધતિનો જ ઉપયોગ બાકીના અક્ષરો શોધવામાં ચાલુ રાખીએ છીએ કે જ્યાં સુધી આપણો ઈચ્છિત શબ્દ મળી ન જાય.

ઉપરના ઉદાહરણથી આપણને ખ્યાલ આવ્યો કે સમસ્યા કેવી હોઈ શકે ? સમસ્યાઓનું નિરાકરણ માનસિક શક્તિઓ અથવા યંત્રોના કાર્ય (કોઈ પદ્ધતિસરના કાર્ય) વપરાશકર્તાનું ઈચ્છિત પરિણામ મેળવવા તરફ લઈ જાય છે. ઉપર ચર્ચા કરેલાં ઉદાહરણો દર્શાવે છે કે કેટલીક સમસ્યાઓના પરિણામની આગાહી ચોક્કસ રીતે કરી શકાય છે જ્યારે કેટલીક સમસ્યાઓના ચોક્કસ પરિણામની આગાહી કરવી ઘણી મુશ્કેલ છે. આ રીતે સમસ્યાઓનું બે પ્રકારમાં વર્ગીકરણ કરી શકાય : સારી રીતે વ્યાખ્યાયિત કરેલી અને સ્પષ્ટ રૂપરેખા વગરની સમસ્યાઓ. ઉપર જણાવેલાં ઉદાહરણોમાં બીજો અને ત્રીજો દાખલો એ સ્પષ્ટ રીતે વર્ણવેલી સમસ્યાનો છે જ્યારે પહેલો દાખલો એ સ્પષ્ટ રૂપરેખા વગરનો છે. અહીં તમે નોંધ કરશો કે સ્પષ્ટ સમસ્યાઓનાં લક્ષ્ય ચોક્કસ અને સ્પષ્ટ હોય છે અને આથી આપણે સમસ્યાના ઉકેલનાં તબક્કા (પગલાં) પણ સ્પષ્ટ રીતે વર્ણવી શકીએ છીએ. એ પણ હકીકત છે કે કમ્પ્યૂટર ફક્ત સ્પષ્ટ રૂપરેખા ધરાવતી સમસ્યાઓનો જ ઉકેલ લાવે છે અને આથી આપણે આ પ્રકરણમાં ફક્ત સ્પષ્ટ રૂપરેખા ધરાવતી (well defined) સમસ્યાઓ બાબત જ ચર્ચા કરીશું.

કમ્પ્યૂટરનાં ક્ષેત્રમાં કોઈ પણ આપેલી સમસ્યાઓનો ઉકેલ એ કમ્પ્યૂટરને આપેલી ક્રમિક સૂચનાઓ જ છે. કમ્પ્યૂટર અતિ સરળતાથી ઘણી ગૂંચવાડાભરી વિવિધ પ્રકારની સમસ્યાઓને ઉકેલે છે. સમસ્યાના ઉકેલ માટે કમ્પ્યૂટરને સૂચનાઓનો સંપૂર્ણ ગણ આપવો પડે છે. આ સૂચનાઓ કમ્પ્યૂટરને દરેક તબક્કે શું કરવાનું છે તે જણાવે છે. એક વસ્તુ ખાસ યાદ રાખો કે કમ્પ્યૂટર જાતે સમસ્યાનો ઉકેલ લાવી શકતું નથી પણ તે સમસ્યાના ઉકેલમાં ફક્ત મદદરૂપ થાય છે. નીચે જણાવેલાં પગલાંને અનુસરવાથી આપણે કોઈ પણ સમસ્યાનો ઉકેલ લાવી શકીએ છીએ :

1. સમસ્યાને વ્યાખ્યાયિત કરો.
2. ઈનપુટ, આઉટપુટ અને સમસ્યાના અવરોધોને ઓળખો.
3. સમસ્યાના ઉકેલના અલગ અલગ વિકલ્પો શોધો.
4. ઉપર જણાવેલા વિવિધ વિકલ્પોની યાદીમાંથી સારામાં સારો વિકલ્પ પસંદ કરો.
5. આપણે નક્કી કરેલા વિકલ્પ માટે વિગતવાર ક્રમિક સૂચનાઓ તૈયાર કરો.
6. આપણે તૈયાર કરેલા સૂચનાઓના ગણ દ્વારા પરિણામની ગણતરી કરો.
7. આપણે મેળવેલો જવાબ સાચો છે કે કેમ તે ચકાસો.

જે વ્યક્તિએ ઉકેલ મેળવવો છે તેણે પગલાં 1થી 5 કરવાનાં છે જ્યારે પગલાં 6 અને 7 કમ્પ્યૂટર વડે થશે.

ધારો કે આપણે કોઈ આપેલી સંખ્યા એકી છે કે બેકી તે શોધવું છે. આ પ્રશ્નના ઉત્તર માટે નીચે આપેલી સૂચનાઓનો ઉપયોગ કરી શકાય :

1. સંખ્યાનું ઇનપુટ લો.
2. તે સંખ્યાને 2 વડે ભાગો અને શેષ શોધો.
3. જો શેષ 1 રહે તો આપેલી સંખ્યા એકી છે નહીંતર તે સંખ્યા બેકી છે.

આ સમસ્યાનો સર્વસામાન્ય ઉકેલ નીચે જણાવેલી ત્રણ રીતો વડે મેળવી શકાય :

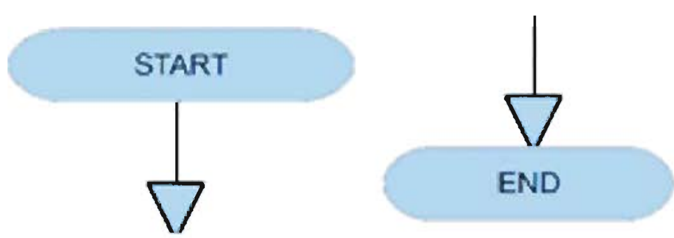
1. સુડો કોડ (Pseudo code) - અનુકરણ કોડ
2. ફ્લોચાર્ટ (Flowchart) - ક્રમદર્શી રેખાચિત્ર
3. અલ્ગોરિધમ (Algorithm) - કલનવિધિ

આપેલી સંખ્યા એકી છે કે બેકી તે જાણવા માટે ઉપર જણાવેલાં ત્રણ પગલાં એ સુડો કોડ (Pseudo code) છે. સુડો એટલે આભાસી અથવા અનુકરણ કરેલો (simulated). આપણે આમાંનો બીજો અર્થ અહીં સ્વીકારીશું કારણ કે અહીં આપણા સંદર્ભમાં તે વધારે બંધબેસતો શબ્દ છે. આપણે આ પ્રશ્નના ઉકેલ માટે ઉપર જણાવેલાં ત્રણ પગલાંને અનુકરણ કરેલો કોડ કહી શકીએ. ચાલો, હવે અન્ય બે રીતો વિશે પણ જાણીએ.

ફ્લોચાર્ટ (Flowchart)

ફ્લોચાર્ટ એ એક કાર્યરીતિ છે જેમાં સમસ્યાના ઉકેલ માટે યંત્ર દ્વારા કરવામાં આવતા દરેક કાર્યને ચિત્રાત્મક સ્વરૂપે રજૂ કરવામાં આવે છે. જુદા જુદા કાર્યો માટે વિવિધ ચિહ્નોનો ઉપયોગ ફ્લોચાર્ટ બનાવવામાં થાય છે. આ ચિહ્નોને ફ્લોચાર્ટનાં ઘટકો કહેવામાં આવે છે. આપણી પાસે પ્રક્રિયાના દરેક કાર્ય માટે અલગ અલગ (અજોડ) ચિહ્ન છે. ચાલો, હવે આપણે વધારે વપરાતા ઘટકો અને તેનાં ચિહ્નો બાબત ચર્ચા કરીએ.

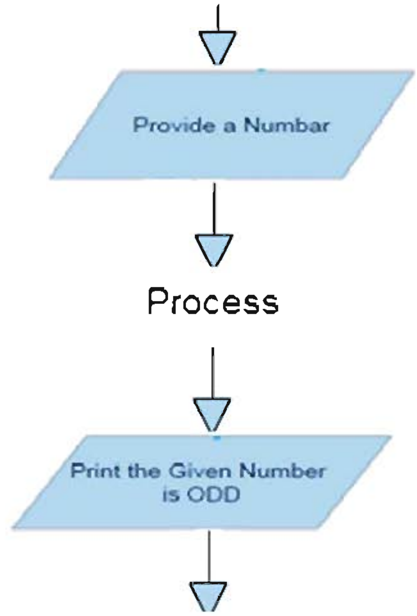
આરંભ અને અંત (Start and End) : આરંભ અને અંત ઘટકો ફ્લોચાર્ટની શરૂઆત અને અંત બતાવવા માટે વપરાય છે. તે આકૃતિ 9.1માં દર્શાવ્યા પ્રમાણે અંડાકાર હોય છે. આ ચિહ્નોને અંતિમ ચિહ્નો (ટર્મિનલ સિમ્બોલ) પણ કહેવામાં આવે છે.



આકૃતિ 9.1 : ટર્મિનલ સિમ્બૉલ

કોઈ પણ ફ્લોચાર્ટમાં આ ચિહ્ન બે વાર વપરાય છે : ફ્લોચાર્ટની શરૂઆતમાં તેમજ તેના અંતમાં.

ઇનપુટ - આઉટપુટ (Input-Output) : દરેક સમસ્યાને ઇનપુટ (નિવેશ)ની જરૂર હોય છે, આ નિવેશ ઉપર પ્રક્રિયા કરવામાં આવે છે અને તે આઉટપુટ (નીપજ-ઉત્પાદન) તૈયાર કરે છે. ફરી ઉપર જણાવેલો પ્રશ્ન 3 લો, અહીં આપણે એક સંખ્યાના ઇનપુટની જરૂર છે, આ આપેલી સંખ્યાને 2 વડે ભાગવાની અને શેષ શોધવાની એ પ્રક્રિયા છે અને આપણે આપેલી સંખ્યા એકી છે કે બેકી તે નિર્ણય એ આપણું આઉટપુટ છે. આથી આપણે એક ઇનપુટ અને એક આઉટપુટ દર્શાવવા માટે ચિહ્ન જોઈશે. ફ્લોચાર્ટમાં ઇનપુટ અને આઉટપુટ દર્શાવવા માટે આકૃતિ 9.2માં દર્શાવ્યા પ્રમાણે સમાંતર ચતુર્ભુજ વપરાય છે.

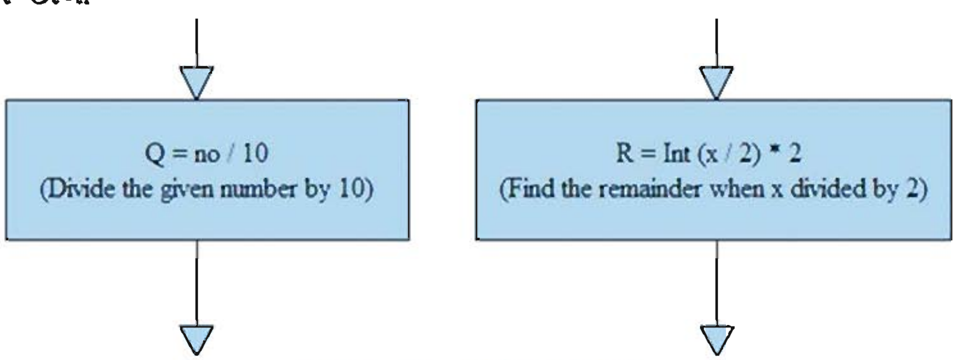


આકૃતિ 9.2 : ઈનપુટ-આઉટપુટ માટેનું ચિહ્ન

તીર (ઍરો - Arrow) : આકૃતિ 9.1 અને આકૃતિ 9.2નું નિરીક્ષણ કરો તો જણાશે કે આપણે ચિહ્નની અંદર જતું તેમજ ચિહ્નમાંથી બહાર આવતું તીર (ઍરો) બતાવેલું છે. કાર્ય જે ક્રમમાં કરવાનું હોય તે દર્શાવવા માટે તીરનો ઉપયોગ થાય છે. સામાન્ય રીતે તે ચિહ્નથી શરૂ થાય છે અને બીજા ચિહ્ન ઉપર તેનો અંત આવે છે. આ રીતે એક તીર એક ચિહ્નમાંથી નીકળે છે અને એક તીર તે ચિહ્નમાં અંદર જાય છે.

અહીં નોંધ કરશો કે start ચિહ્નમાંથી ફક્ત બહાર જતું તીર છે જ્યારે End ચિહ્નમાં ફક્ત તેની અંદર જતું તીર છે.

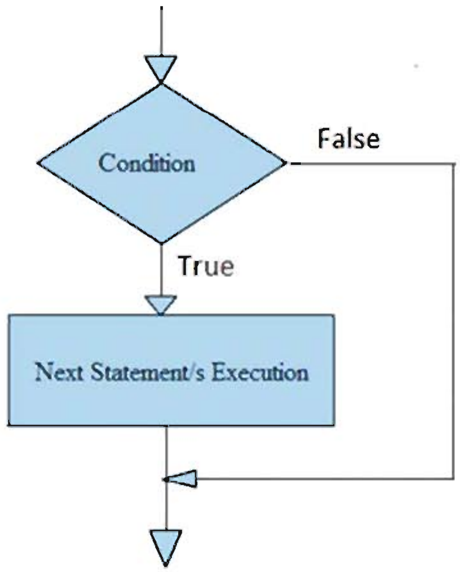
પ્રક્રિયા (Process) : કોઈ પણ સમસ્યાના ઉકેલમાં પ્રક્રિયા (process) એ મુખ્ય ભાગ છે. હકીકતમાં પ્રક્રિયા એ ક્રિયાઓની એક શ્રેણી છે. પ્રક્રિયાને દર્શાવવા માટે આકૃતિ 9.3માં દર્શાવ્યા પ્રમાણે એક લંબચોરસનું ચિહ્નનો આપણે ઉપયોગ કરીએ છીએ.

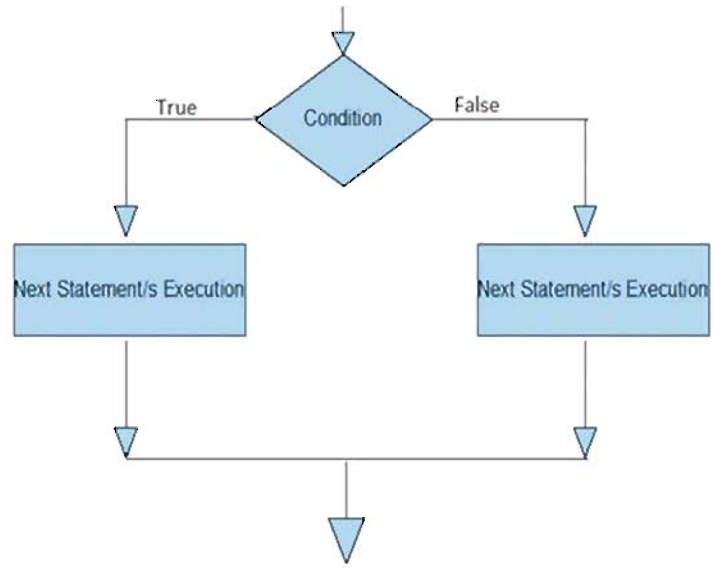


આકૃતિ 9.3 : પ્રક્રિયાનું ચિહ્ન

સામાન્ય રીતે કમ્પ્યૂટરની પ્રક્રિયા એ અંકગણિતનું કોઈ કાર્ય અથવા તાર્કિક ક્રિયા હોય છે. અંકગણિતનું કાર્ય એટલે સરવાળા, બાદબાકી, ગુણાકાર અથવા ભાગાકાર. તાર્કિક કાર્ય સામાન્ય રીતે નિર્ણય લેવામાં મદદરૂપ થાય છે જ્યારે તે કોઈ પ્રશ્નના હા કે ના પ્રકારના જવાબ માટે વપરાય છે. દાખલા તરીકે, આપણે પ્રશ્ન પુછીએ કે 10 એ 5 કરતાં મોટો છે ? આ પ્રશ્નનો જવાબ 'હા' અથવા સાચો છે.

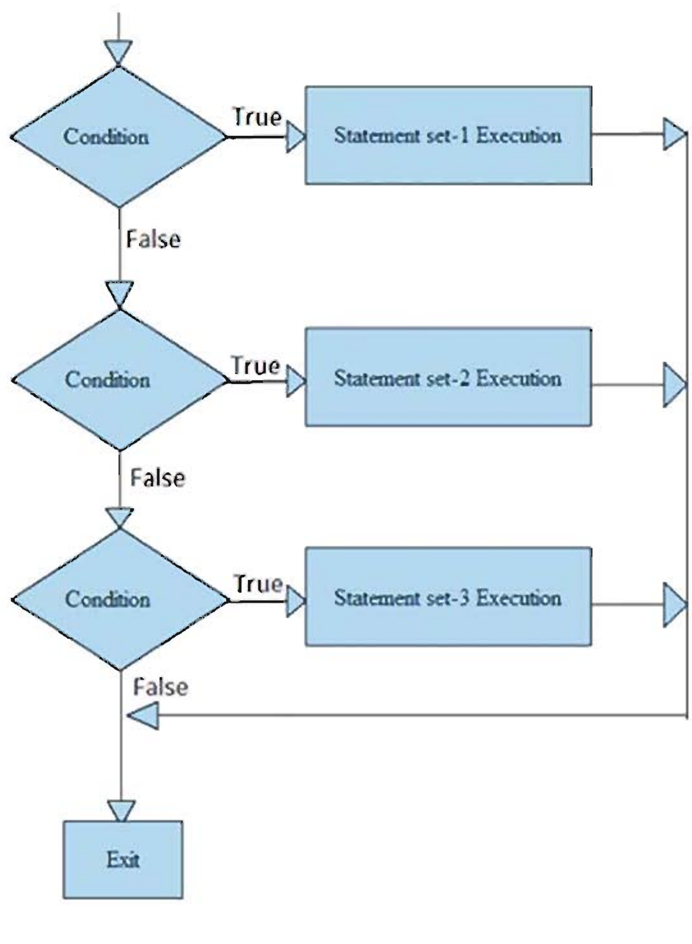
નિર્ણય (Decision) : તાર્કિક નિર્ણયની પ્રક્રિયાને આકૃતિ 9.4માં દર્શાવ્યા પ્રમાણે એક હીરા આકારનાં ચિહ્નથી દર્શાવવામાં આવે છે. તેને પરીક્ષણ ચિહ્ન (ટેસ્ટ સિમ્બોલ) પણ કહેવામાં આવે છે. જ્યારે આપણે ઉકેલ માટેની સામાન્ય શ્રેણી બદલવા ઈચ્છતા હોય (જુઓ આકૃતિ 9.4) અથવા નિર્ણયનાં પરિણામનાં આધારે કોઈ ચોક્કસ વિધાનનો અમલ કરવાની જરૂર હોય (જુઓ આકૃતિ 9.5) ત્યારે નિર્ણયપેટી (decision box) વપરાય છે.





આકૃતિ 9.4 : નિર્ણય ચિહ્ન **આકૃતિ 9.5 : જુદાં જુદાં પરિણામોનો અમલ**

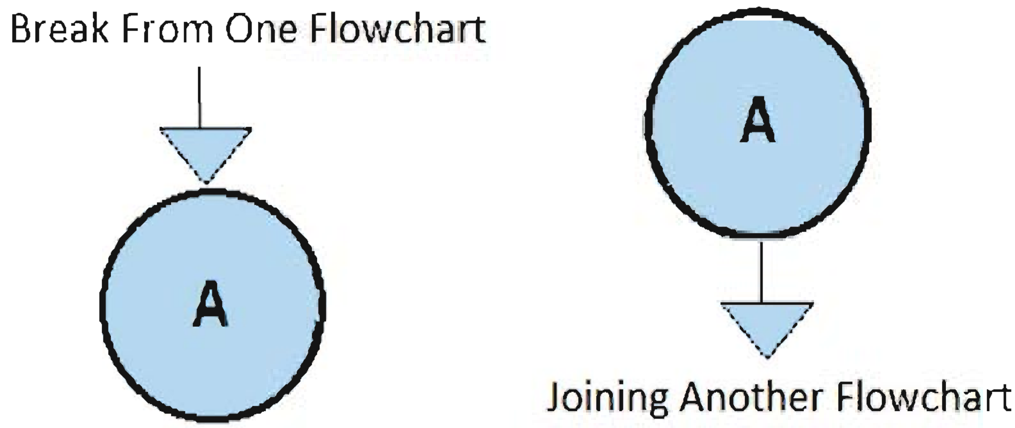
અમુક સમયે નિર્ણય લેવામાં બે કરતાં વધારે વિકલ્પોની આપણને જરૂર પડે છે. આવા સમયે આકૃતિ 9.6માં દર્શાવ્યા પ્રમાણે એક કરતાં વધારે નિર્ણયપેટીઓને જોડીને જરૂરી વિકલ્પ બનાવવમાં આવે છે.



આકૃતિ 9.6 : એક કરતાં વધારે નિર્ણયોનું જોડાણ

આકૃતિ 9.6માં દર્શાવ્યા પ્રમાણે સૌપ્રથમ શરત તપાસવામાં આવે છે અને જો તેનું પરિણામ ખરું આવે તો પહેલાં વિધાનોના ગણનો અમલ થશે. જો પરિણામ ખોટું આવે તો બીજી શરત તપાસવામાં આવશે. આ પ્રક્રિયા આપેલી બધી શરતો માટે ફરીફરીને કરવામાં આવશે. અહીં એ નોંધ કરો કે જ્યારે કોઈ એક શરતનું પરિણામ ખરું આવે તો તે પછીની શરત તપાસવામાં આવતી નથી.

અનુસંધાન ચિહ્ન Connector : અનુસંધાન ચિહ્ન દર્શાવવા માટે વર્તુળનું ચિહ્ન વપરાય છે. અમુક સમયે એ શક્ય છે કે ફ્લોચાર્ટ એટલો મોટો બની જાય કે તે એક પાનામાં સમાવી ન શકાય અથવા બે પ્રક્રિયાને તીર વડે જોડવી શક્ય ન બને. આવી પરિસ્થિતિમાં બે ભાગને જોડવા માટે અનુસંધાન ચિહ્ન (કનેક્ટર) વાપરવામાં આવે છે. જોડાણ માટે ઓછામાં ઓછા બે વર્તુળ જરૂરી છે; આકૃતિ 9.7માં દર્શાવ્યા પ્રમાણે એક તીર વર્તુળની અંદર જતું અને બીજું તીર વર્તુળમાંથી બહાર આવતું. જુદા જુદા અનુસંધાન ચિહ્ન ઓળખવા માટે અલગ અલગ અક્ષરો જોડીમાં વપરાય છે.



આકૃતિ 9.7 : અનુસંધાન ચિહ્ન (કનેક્ટર)

ફ્લોચાર્ટનાં ઉદાહરણો :

અત્યાર સુધીમાં આપણે જાણ્યું કે ફ્લોચાર્ટ શું છે અને તેમાં વપરાતા ચિહ્નો બાબત જાણકારી મેળવી. ચાલો, હવે આપણે ફ્લોચાર્ટ બનાવીને કેટલીક સમસ્યાઓનો ઉકેલ લાવીએ.

ઉદાહરણ 1 : જમીન ઉપર લાદી લગાવવાની કિંમત 10 રૂ. ચો. ફીટ છે. હવે ધારો કે આપણી સમસ્યા લંબચોરસ આકારના નગરના સભાગૃહમાં જમીન ઉપર લાદી લગાવવાની કુલ કિંમત શોધવાની છે.

જવાબ :

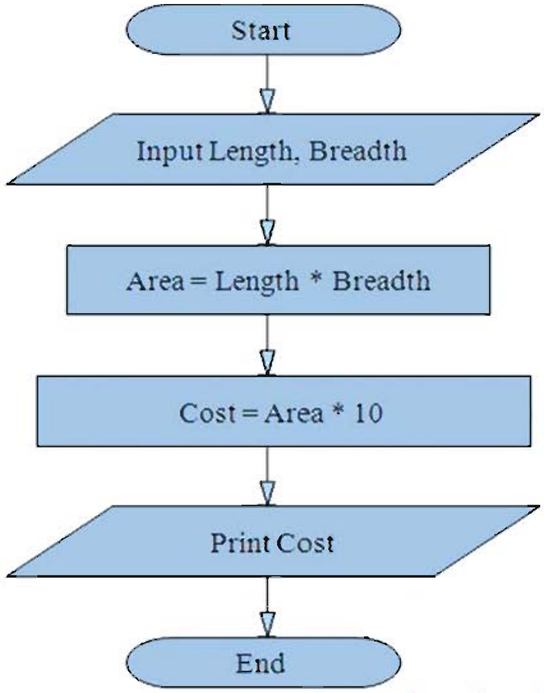
જમીન ઉપર લાદી લગાવવાનો કુલ ખર્ચ શોધવા માટે આપણે પહેલાં જમીનનું ક્ષેત્રફળ શોધવું પડે. આપણે જાણીએ છીએ કે લંબચોરસનું ક્ષેત્રફળ એ તેની લંબાઈ અને પહોળાઈનો ગુણાકાર છે. આથી આપણે જો સભાગૃહની લંબાઈ અને પહોળાઈ જાણતા હોઈએ તો લંબચોરસનું ક્ષેત્રફળ શોધી શકીએ. જમીનનું ક્ષેત્રફળ શોધવા માટે આ મુજબ સૂત્ર આપી શકાય : ક્ષેત્રફળ = લંબાઈ × પહોળાઈ

હવે જમીન ઉપર લાદી લગાવવાની કુલ કિંમત શોધવા માટે આપણે જમીનનું ક્ષેત્રફળ અને એક ચો. ફીટ લગાવવાની કિંમતનો ગુણાકાર કરવો પડે. કુલ કિંમત શોધવાનું સૂત્ર : કુલ કિંમત = ક્ષેત્રફળ × 10.

અહીં સમસ્યાના ઉકેલ માટે આપણે ચાર ચલ ક્ષેત્રફળ, લંબાઈ, પહોળાઈ અને કિંમતની જરૂર પડે છે. હજી આગળ વધતાં પહેલાં આપણે પહેલા શબ્દ ચલની વ્યાખ્યા કરીએ, કારણ કે તે બધા ફ્લોચાર્ટમાં વપરાય છે.

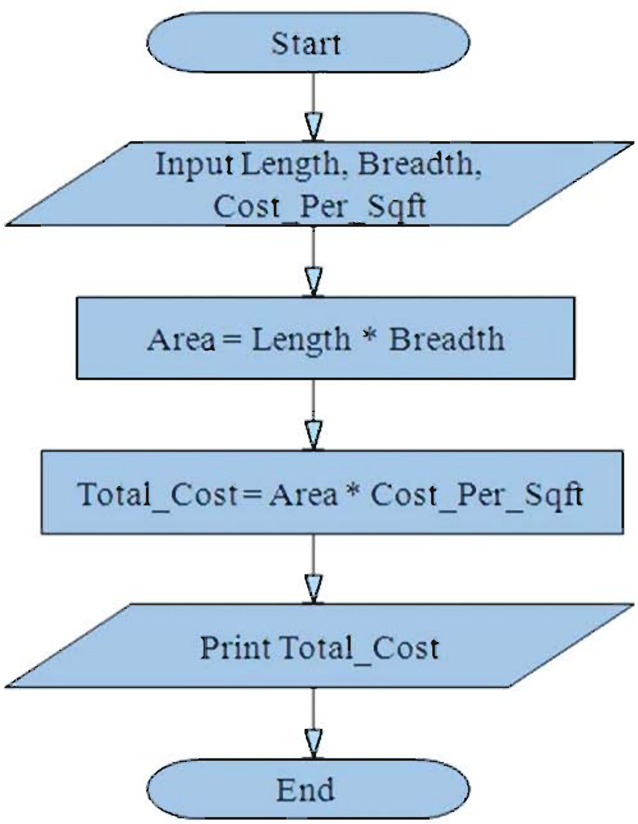
ચલ (Variable) : ચલ એ એક એન્ટિટી (entity) છે જેની કિંમત પ્રક્રિયાના અમલ દરમિયાન બદલી શકાય છે અને તેને કિંમત આપી શકાય છે.

જમીન ઉપર લાદી લગાવવાની કિંમત શોધવાનો ફ્લોચાર્ટ આકૃતિ 9.8માં દર્શાવ્યો છે.



આકૃતિ 9.8 : લાદી લગાવવાની કિંમત શોધવાનો ફ્લોચાર્ટ

પદાવલિ કિંમત = ક્ષેત્રફળ × 10માં કિંમત 10 જુઓ, અહીં 10 એ અચળ (કિંમત) કહેવાય છે. આ એક એવી એન્ટિટી છે, જેની કિંમત એક વાર આપીએ તે પછી આખી પ્રક્રિયા દરમિયાન તેમાં ફેરફાર થતો નથી અને તે સ્થાયી રહે છે. આકૃતિ 9.8માં દર્શાવેલા ફ્લોચાર્ટ દ્વારા આપણે જે ઉકેલ મેળવીએ છીએ તે ફક્ત એ કિસ્સામાં જ વાપરી શકાય જ્યારે એક ચો. ફીટ લાદી લગાવવાની કિંમત રૂ. 10 હોય. જો લાદી લગાવવાની કિંમતમાં ફેરફાર થાય તો ઉપર જણાવેલો ઉકેલ ન ચાલે. ઉપર જણાવેલી સમસ્યાનો સર્વસામાન્ય ઉકેલ આકૃતિ 9.9માં આપેલો છે.



આકૃતિ 9.9 : લાદી લગાવવાનો ખર્ચ શોધવાનો સુધારાયેલ ફ્લોચાર્ટ

આકૃતિ 9.9નું નિરીક્ષણ કરો, અહીં આપણે ચારને બદલે પાંચ ચલનો ઉપયોગ કર્યો છે. અહીં Cost_Per_Sqft ચલનો ઉપયોગ એક ચો. ફીટ લાદી લગાવવાની કિંમતનો સંગ્રહ કરવા માટે કર્યો છે. દરેક સમયે આપણે તેની કિંમતમાં ફેરફાર કરી શકીએ અને આપણે જરૂરી આઉટપુટ મેળવી શકીએ. કોષ્ટક 9.1માં આવાં કેટલાક આઉટપુટ આપેલાં છે.

Length	Breadth	Area=Length*Breadth	Cost_Per_Sqft	Total_Cost=Area*Cost_Per_Sqft
50	50	2500	10	25000
25	75	1875	10	18750
45	35	1575	20	31500

કોષ્ટક 9.1 : આકૃતિ 9.9માં દર્શાવેલા ફ્લોચાર્ટનું આઉટપુટ

ઉદાહરણ 2 : ધારો કે તમારા શહેરનાં રમતગમત સંકુલમાં એક સુંદર અને વર્તુળ ક્રિકેટનું મેદાન છે. તેના સત્તાધીશો આ મેદાનની ફરતે વાડ બનાવવા ઇચ્છે છે. આ ઉપરાંત તેઓ સંપૂર્ણ મેદાનને લૉન (વ્યવસ્થિત રીતે કાપેલા અને સુંવાળા ઘાસવાળી હરિયાળી જમીન)થી ઢાંકવા ઇચ્છે છે. સત્તાધીશો જાણવા માગે છે કે કેટલા મીટરની વાડ જોઈશે ? તેઓ એ પણ જાણવા ઇચ્છે છે કે મેદાનનું કુલ ક્ષેત્રફળ કેટલું છે કે જેને લૉન વડે ઢાંકવાની જરૂર છે.

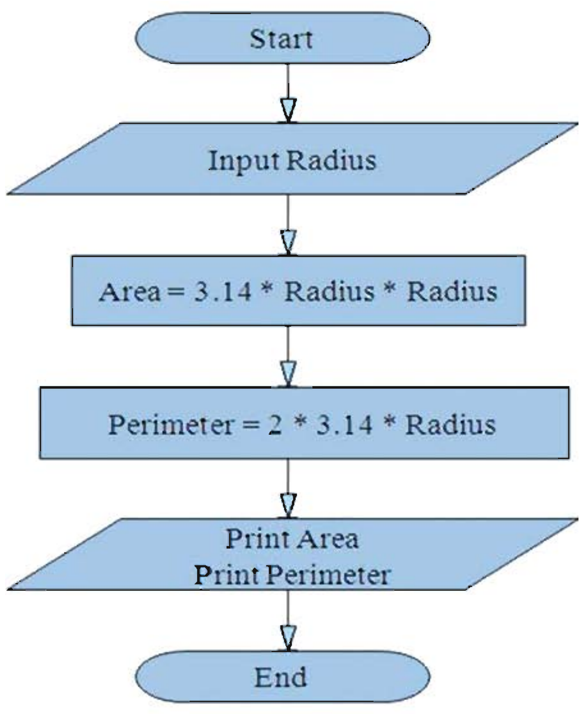
જવાબ :

આ પ્રશ્નના ઉકેલ માટે આપણે મેદાનનું ક્ષેત્રફળ અને તેની પરિમિતિ શોધવી પડશે. ગોળ મેદાનનું ક્ષેત્રફળ શોધવા માટેનું સૂત્ર ક્ષેત્રફળ = $\pi * R^2$ અને મેદાનની પરિમિતિ = $2 * \pi * R$ (અહીં πની કિંમત 3.14 અને R એ મેદાનની ત્રિજ્યા છે)

આથી, આ પ્રશ્નના ઉકેલ માટે ત્રણ ચલ Radius (ત્રિજ્યા), Area (ક્ષેત્રફળ) અને Perimeter (પરિમિતિ) આપણને જરૂર પડશે. ફ્લોચાર્ટમાં આપણે ક્ષેત્રફળની ગણતરી અને મેદાનની પરિમિતિ શોધવા માટે જે સૂત્ર વાપરીશું તે નીચે આપેલાં છે.

Area = 3.14 * Radius * Radius (ક્ષેત્રફળ = 3.14 * ત્રિજ્યા * ત્રિજ્યા)

અને Perimeter = 2 * 3.14 * Radius (પરિમિતિ = 2 * 3.14 * ત્રિજ્યા)

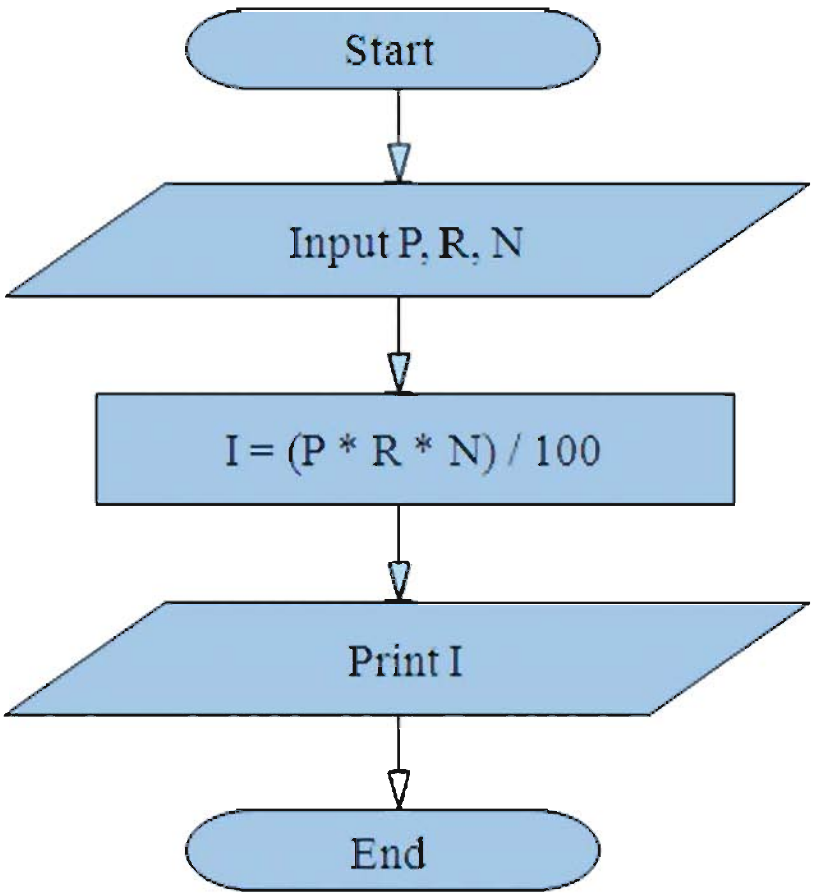


આકૃતિ 9.10 : વર્તુળનું ક્ષેત્રફળ અને પરિમિતિ શોધવાનો ફ્લોચાર્ટ

ઉદાહરણ 3 : જહાનવીએ બેન્કમાંથી 6 વર્ષની 11.5 ટકાના દરથી 35,000 રૂ.ની લોન લીધી. તેણીએ બેન્કને કેટલું સાદું વ્યાજ ચૂકવવું પડશે તેની ગણતરી કરો.

જવાબ

કોઈ લોનની મુદલ રકમ 'P', વ્યાજનો દર 'R' અને 'N' વર્ષ માટેના સાદા વ્યાજ 'I' ની ગણતરી માટેનું સૂત્ર **I = (P*R*N)/100** છે. આ પ્રશ્નના ઉકેલ માટે આપણે ચાર ચલ I, P, R, N અને અચળ કિંમત 100ની જરૂર પડશે. આ પ્રશ્નનો ઉકેલ આકૃતિ 9.11માં ફ્લોચાર્ટના રૂપમાં દર્શાવ્યો છે.



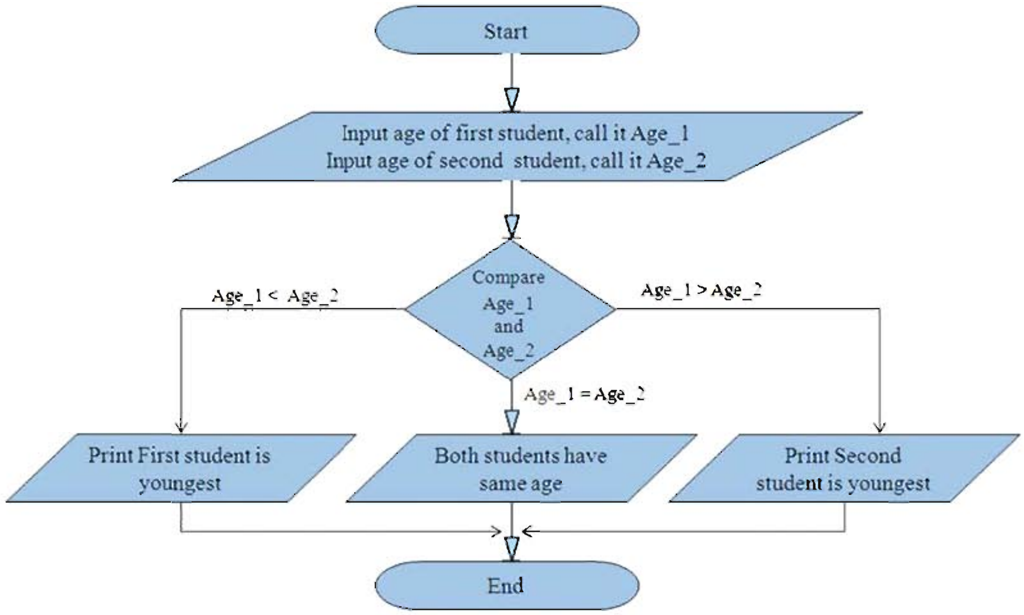
આકૃતિ 9.11 : સાદા વ્યાજની ગણતરી માટેનો ફ્લોચાર્ટ

ઉદાહરણ 4 : ધારો કે બે વિદ્યાર્થીઓમાંથી તમારે નાની ઉંમરનો વિદ્યાર્થી શોધવો છે, આ કિસ્સામાં વિદ્યાર્થીની ઉંમર એ નિવેશ (ઇનપુટ) હશે.

જવાબ :

ઉપરના પ્રશ્નનો ઉકેલ નીચે જણાવેલી ક્રિયાઓ કરવાથી મળશે :

સૌપ્રથમ બંને વિદ્યાર્થીઓની ઉંમરનું ઇનપુટ (નિવેશ) લો. તે ઉંમર Age_1 અને Age_2 ચલમાં રાખો. હવે Age_1 અને Age_2ની કિંમતની સરખામણી કરો. જો બંને કિંમત સરખી હોય તો આપણી પાસે સરખી ઉંમરના બંને વિદ્યાર્થી છે અને આથી બંને નાનામાં નાના છે. જો Age_1ની કિંમત Age_2 કરતાં ઓછી હોય તો પહેલો વિદ્યાર્થી નાનો છે, નહીંતર બીજો વિદ્યાર્થી. આ પ્રશ્નના ઉકેલનો ફ્લોચાર્ટ આકૃતિ 9.12માં આપેલો છે.



આકૃતિ 9.12 : બે વિદ્યાર્થીઓમાંથી નાનો વિદ્યાર્થી શોધવાનો ફ્લોચાર્ટ

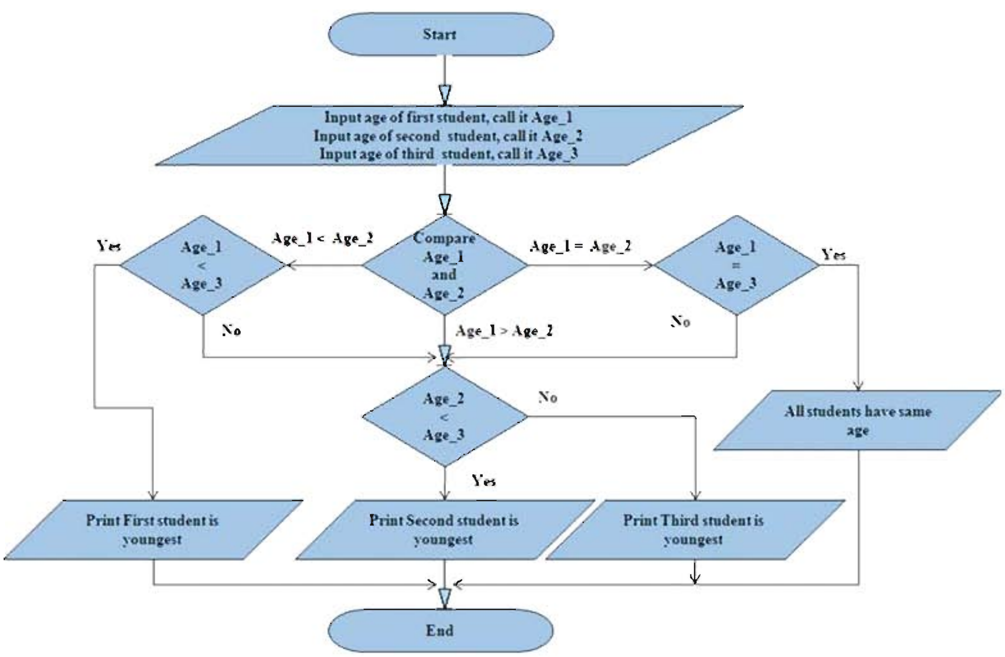
ઉદાહરણ 5 : હવે આપણે ત્રણ વિદ્યાર્થીઓમાંથી સૌથી નાનો વિદ્યાર્થી શોધવાનો પ્રયત્ન કરીએ.

જવાબ :

સૌપ્રથમ ત્રણ વિદ્યાર્થીઓની ઉંમરનું ઇનપુટ લો અને તે ચલ Age_1, Age_2 અને Age_3માં રાખો. જો ત્રણે કિંમત એકસરખી હોય તો આપણી પાસે ત્રણે સરખી ઉંમરના વિદ્યાર્થી છે. આથી ત્રણે વિદ્યાર્થીઓને નાનામાં નાના ગણવામાં આવે.

જો એમ ન હોય તો Age_1 અને Age_2ની સરખામણી કરો. જો Age_1ની કિંમત Age_2 કરતાં ઓછી હોય તો Age_1 ને Age_3 સાથે સરખાવો અને હજી Age_1 એ Age_3 કરતાં ઓછી હોય તો પહેલો વિદ્યાર્થી નાનામાં નાનો છે.

ઉપર જે સરખામણી કરી તેમાં જો Age_2 એ Age_1 કરતાં ઓછી હોય તો Age_2ને Age_3 સાથે સરખાવો અને જો Age_2 એ Age_3 કરતાં ઓછી હોય તો બીજો વિદ્યાર્થી નાનામાં નાનો છે. આ સિવાય ત્રીજો વિદ્યાર્થી નાનામાં નાનો છે. આ પ્રશ્નના ઉકેલનો ફ્લોચાર્ટ આકૃતિ 9.13માં આપેલો છે.



આકૃતિ 9.13 : ત્રણ વિદ્યાર્થીઓમાંથી નાનામાં નાનો વિદ્યાર્થી શોધવાનો ફ્લોચાર્ટ

ઉદાહરણ 6 : ચાલો, ઉપરના ઉદાહરણમાં હજી પણ સુધારો કરીએ અને વર્ગના બધા વિદ્યાર્થીઓમાંથી નાનામાં નાનો વિદ્યાર્થી શોધીએ. આ પ્રશ્નના ઉકેલ માટે નીચે જણાવેલાં પગલાંનો ઉપયોગ કરી શકાય :

પગલું 1 : પહેલો વિદ્યાર્થી લો અને તેની કોઈ પણ ઉંમર હોય, ધારો કે તે MINIMUM છે.

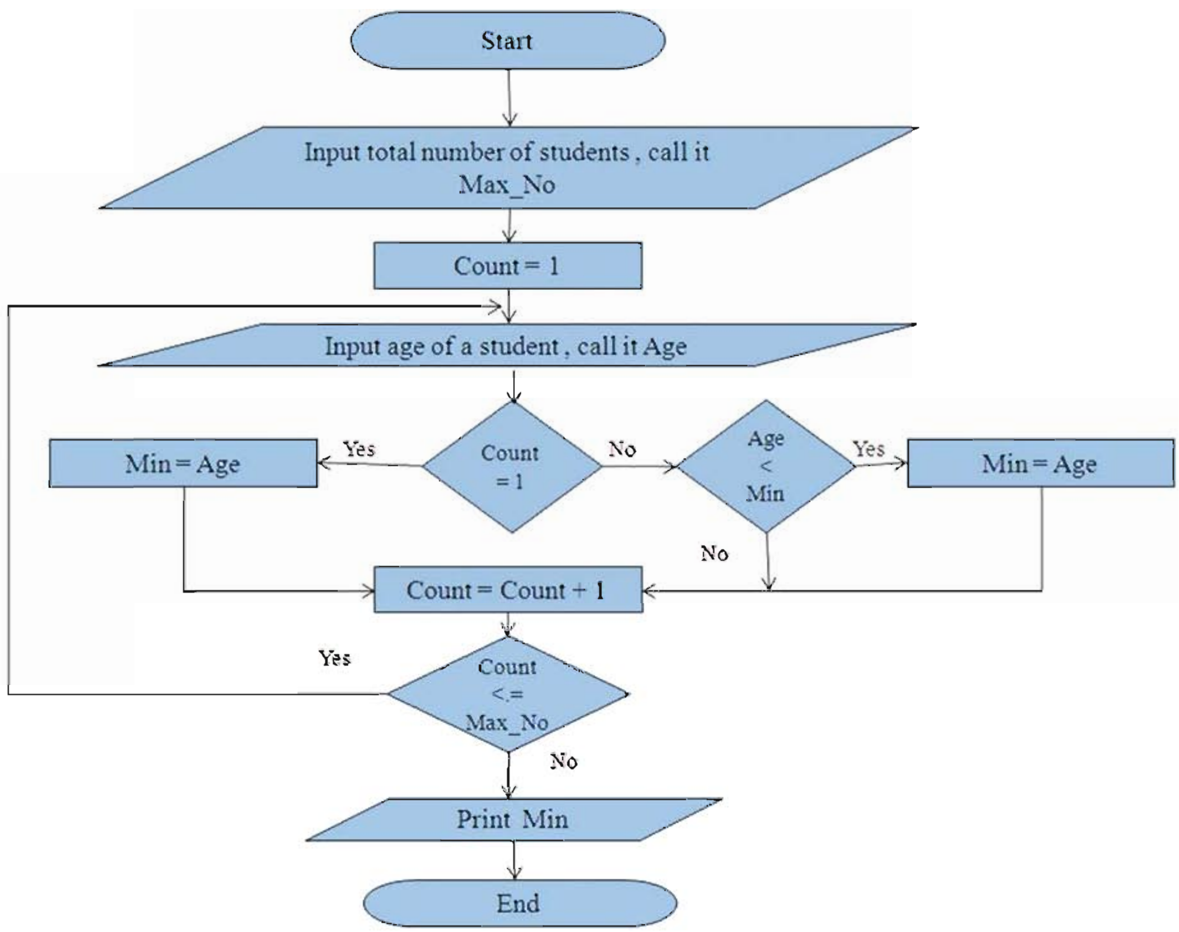
પગલું 2 : હવે બીજો વિદ્યાર્થી પસંદ કરો અને તેની ઉંમર MINIMUM જણાવેલ તેની સાથે સરખાવો.

પગલું 3 : જો નવા વિદ્યાર્થીની ઉંમર ઓછી હોય તો તેની ઉંમર MINIMUM બનાવીશું અને અગાઉના વિદ્યાર્થીની ઉંમર રદ કરીશું.

પગલું 4 : જો બંને વિદ્યાર્થીઓની ઉંમર એકસરખી હોય તો નવા વિદ્યાર્થીની ઉંમરની અવગણના કરવામાં આવશે અને અગાઉના વિદ્યાર્થીની ઉંમર હજી પણ MINIMUM છે.

પગલું 5 : આપણે ઉપર જણાવેલાં પગલાં 3 અને 4નું પુનરાવર્તન ચાલુ રાખીશું કે જ્યાં સુધી બધા વિદ્યાર્થીઓમાંથી નાનામાં નાનો વિદ્યાર્થી આપણને મળી જાય. જ્યારે અમુક વિધાનોને એક કરતાં વધારે વખત પુનરાવર્તિત કરવામાં આવે તેને લૂપ (Loop) કહેવામાં આવે છે.

આ પ્રશ્નના ઉકેલનો ફ્લોચાર્ટ આકૃતિ 9.14માં આપેલો છે.



આકૃતિ 9.14 : વર્ગના બધા વિદ્યાર્થીઓમાંથી સૌથી યુવાન શોધવાનો ફ્લોચાર્ટ

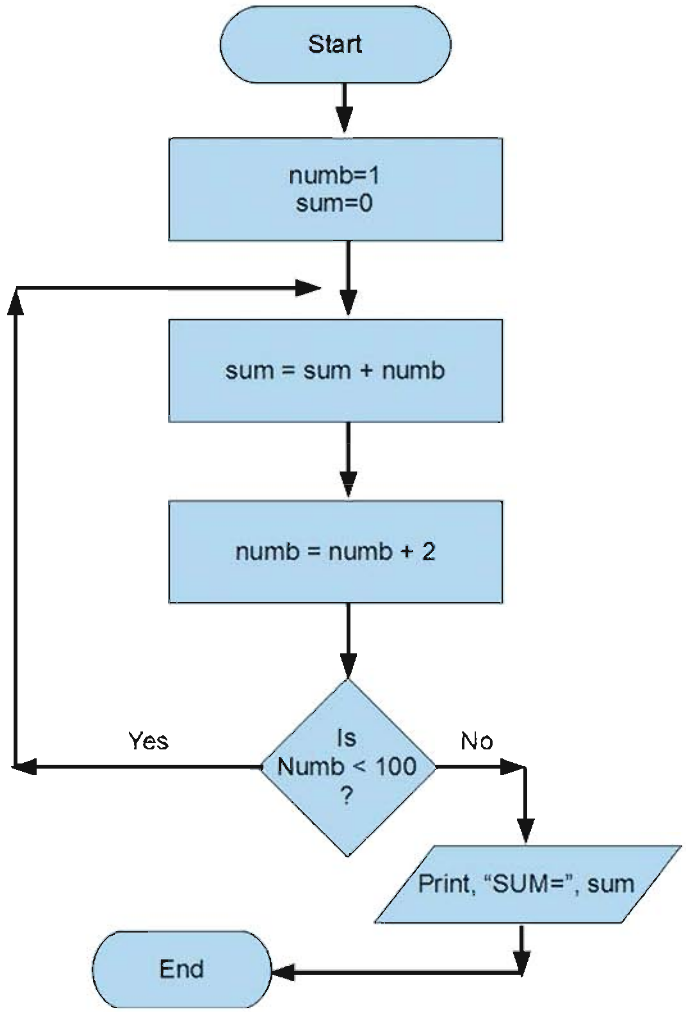
ઉદાહરણ 7 : ચાલો, હવે આપણે એક ગણિતનો પ્રશ્ન ઉકેલીએ. ધારો કે આપણે પહેલી 50 એકી સંખ્યા (odd numbers) ઓનો સરવાળો કરવા ઇચ્છીએ છીએ. એટલે કે આપણે 1 + 3 + 5 + 7 + ... + 99નો સરવાળો કરવા ઇચ્છીએ છીએ.

જવાબ :

ધારો કે તમારી પાસે બે બૉક્સ (પેટીઓ) છે, એક ડાબા હાથમાં અને બીજી જમણા હાથમાં. હવે ડાબી બાજુના બૉક્સમાં એક લખોટી મૂકો અને હવે તે બૉક્સને જમણા હાથમાં રહેલા બૉક્સમાં ખાલી કરો. હવે ડાબા હાથમાં રહેલા બૉક્સમાં ત્રણ લખોટી મૂકો અને તેને જમણા હાથમાં રહેલા બૉક્સમાં ખાલી કરો (બદલી કરો). આથી જમણા હાથમાં રહેલા બૉક્સમાં

કેટલી થશે ? જમણા હાથમાં રહેલા બૉક્સમાં હવે કુલ 1 + 3 લખોટીઓ હશે. જો આપણે આ પ્રક્રિયા 5, 7, ... 99 લખોટીઓ સાથે ચાલુ રાખીશું તો જમણા હાથમાં રહેલા બૉક્સમાં 1 + 3 + 5 + 7 + ... + 99 લખોટીઓ હશે. અહીં એ નોંધ કરો કે આપણે ધારેલું કે જમણા હાથનું બૉક્સ શરૂઆતમાં ખાલી છે.

આ ક્રિયાને અનુરૂપ, આપણે બે ચલ numb અને sum લઈએ. numb ચલની શરૂઆતની કિંમત 1 અને sumની કિંમત 0 (ખાલી બૉક્સની જેમ) રાખીએ. હવે લૂપના દરેક સમયે numb ચલની કિંમત 2થી વધારવામાં આવે છે અને તેની કિંમત sumમાં ઉમેરવામાં આવે છે. આથી numb ચલની કિંમત બદલાતી રહેશે અને તે 1, 3, 5, ... અને છેલ્લે 99 બનશે. આથી લૂપનો અંત આવે ત્યાં સુધી વિધાન sum = sum + numbથી numbની કિંમત sumમાં ઉમેરાતી રહેશે. આ પ્રશ્નનો ફ્લોચાર્ટ આકૃતિ 9.15માં આપેલો છે.



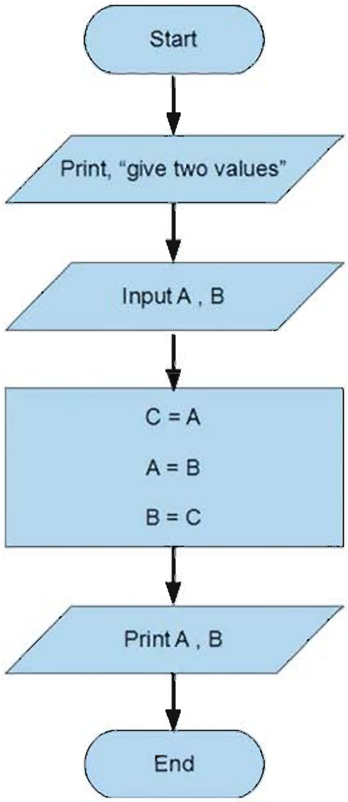
આકૃતિ 9.15 : 1થી 99 સુધીની એકી સંખ્યાઓના સરવાળાનો ફ્લોચાર્ટ

ઉદાહરણ 8 : હવે આપણે બે ચલની કિંમતની અદલાબદલી કરવાની બે જુદી જુદી રીતો જોઈએ.

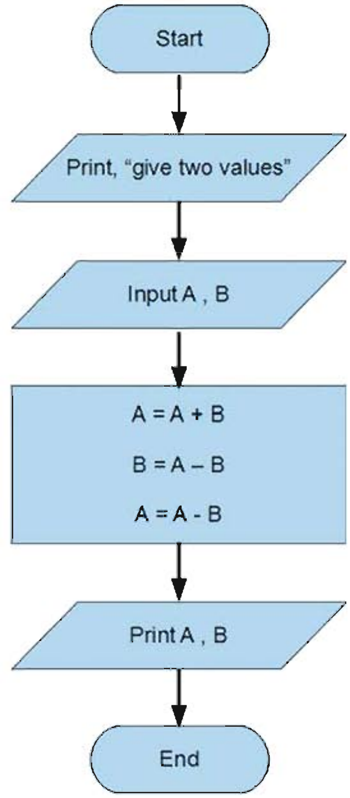
જવાબ :

બે ચલ A અને B લો. જો A અથવા Bને કોઈ અન્ય કિંમત વડે બદલવામાં આવશે તો તેમાં શરૂઆતમાં રહેલી કિંમત આપણે ગુમાવીશું. આથી આપણે એક વધારાનો ચલ Cનો ઉપયોગ કરીશું. Aની કિંમત જળવાઈ રહે તે માટે તેની કિંમત પહેલાં આપણે Cમાં રાખીએ. તે પછી Bની કિંમત આપણે Aમાં લઈ જઈએ અને ત્યાર બાદ Aની સાચવેલી કિંમત (જે Cમાં રાખેલી છે) Bમાં લઈ જઈએ. આકૃતિ 9.16માં આ ઉકેલનો ફ્લોચાર્ટ આપેલો છે.

આકૃતિ 9.17માં આ જ પ્રશ્નના ઉકેલનો ફ્લોચાર્ટ આપેલો છે, પણ તેમાં વધારાના ચલનો ઉપયોગ થતો નથી. ખરેખર કિંમત બદલવા માટે સરવાળા અને બાદબાકીનો ઉપયોગ કરેલો છે.



આકૃતિ 9.16 : વધારાના ચલનો ઉપયોગ કરીને કિંમતની અદલાબદલી



આકૃતિ 9.17 : વધારાના ચલ વિના કિંમતની અદલાબદલી

ફ્લોચાર્ટ(Flowchart)ના ફાયદાઓ અને ગેરફાયદાઓ (Advantages and Disadvantages of Flowchart)

જ્યારે આપણે કોઈ પણ પ્રકારનાં ટૂલ્સ (tools) વાપરીએ છીએ, ત્યારે તેના કેટલાક ફાયદાઓ અને ગેરફાયદાઓ હોય છે. ફ્લોચાર્ટ તેમાં અપવાદ નથી. તેના કેટલાક ફાયદાઓ અને મર્યાદાઓ છે.

ફ્લોચાર્ટના ફાયદા :

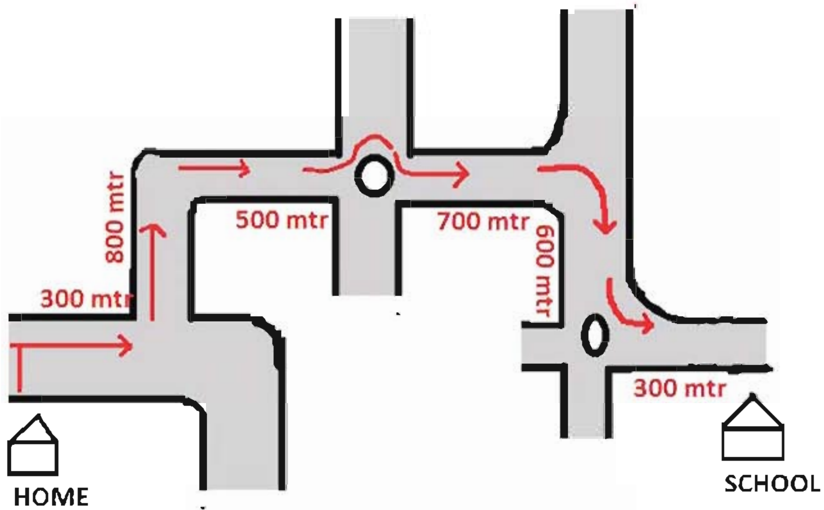
1. તે સમસ્યાનો ઉકેલ ચિત્રાત્મક સ્વરૂપે રજૂ કરતો હોવાથી સમજવો ખૂબ સરળ છે.
2. તે કમ્પ્યૂટરની સૂચનાઓ લખવામાં ખૂબ જ અનુકૂળ મદદ પૂરી પાડે છે.
3. સમસ્યાના ઉકેલના પગલાંની સમીક્ષા કરવામાં અને સુધારવામાં પ્રોગ્રામરને સહાયભૂત બને છે.
4. સમસ્યાના ઉકેલની જુદી જુદી રીતો વિશે ચર્ચા કરવામાં તે મદદરૂપ બને છે, જેને કારણે અંતે તેની સરખામણી ભૂલરહિત કરવાની સગવડ તે પૂરી પાડે છે.

ફ્લોચાર્ટના ગેરફાયદા :

1. મોટી અને ગૂંચવાડાભારી સમસ્યાનો ફ્લોચાર્ટ દોરવાનું કામ ઘણો સમય માગી લે છે અને તે થકાવટવાળું કામ છે.
2. ફ્લોચાર્ટમાં અનેક ચિહ્નો હોય છે અને આથી પ્રોગ્રામના તર્કમાં કોઈ પણ ફેરફાર કરવાથી અનેક સમયે નવો ફ્લોચાર્ટ દોરવાની જરૂર પડે છે.
3. ફ્લોચાર્ટમાં ડેટા કેટલા વિગતવાર જણાવવા તેના કોઈ ધોરણ નથી. આથી એકસરખા તર્કવાળી સમસ્યાના ફ્લોચાર્ટ તેની અંદરની માહિતીથી અલગ પડે છે.

અલગોરિધમ (Algorithm) (કલનવિધિ)

તમારો મિત્ર તમારા ઘેર આવી ગયો છે અને હવે તમારી નિશાળમાં આવવાની ઇચ્છા રાખે છે. તે તમને ફોન કરે છે અને તમારી નિશાળ સુધી પહોંચવાની દિશા પૂછે છે. સ્કૂલ કઈ રીતે પહોંચી શકાય તેની સૂચના આપવા માટે તમારા મગજમાં પહેલાં કોઈ નિયત માર્ગ હોય એ જરૂરી છે. ધારો કે, આકૃતિ 9.18માં દર્શાવ્યા પ્રમાણેનો નિયત માર્ગનો નકશો તેને મદદ કરવા માટે છે.



આકૃતિ 9.18 : નિયત માર્ગનો નકશો

જવાબ :

આકૃતિ 9.18માં આપેલાં નિયત માર્ગના નકશાનો ઉપયોગ કરીને તમે તમારા મિત્રને નીચે જણાવ્યા પ્રમાણે એક પછી એક પગલાંનું માર્ગદર્શન આપી શકશો :

પગલું 1 : મારા ઘરથી શરૂ કરી પહેલાં જમણી બાજુ ફરો અને 300 મીટર સીધા ચાલો.

પગલું 2 : હવે ડાબી બાજુ ફરો અને 800 મીટર સીધા જાઓ.

પગલું 3 : હવે જમણી બાજુ ફરો અને 500 મીટર ચાલો આથી ચાર રસ્તા પર એક સર્કલ આવશે.

પગલું 4 : સર્કલ પર વળાંક ન લો અને સીધા એ જ રસ્તા પર બીજા 700 મીટર ચાલો જેના પર તમે ચાલતા આવ્યા હતા.

પગલું 5 : હવે જમણી બાજુ ફરો અને 600 મીટર ચાલ્યા બાદ ફરી ચાર રસ્તા વચ્ચે અંડાકાર આકારના ફુવારા પર તમે પહોંચશો.

પગલું 6 : આ જંક્શનથી ડાબી બાજુ ફરો અને 300 મીટર ચાલો.

પગલું 7 : હવે તમે શેરીની જમણી બાજુએ મારી નિશાળ જોઈ શકશો.

ઉપરનાં ઉદાહરણ પરથી આપણે કહી શકીએ કે પગલાંઓની શ્રેણીને અનુસરવાથી તમારી નિશાળે પહોંચવાનો પ્રશ્ન ઉકેલી શકીએ છીએ. કમ્પ્યૂટર સાયન્સમાં અલગોરિધમ કોઈ ચોક્કસ સમસ્યાના ઉકેલ માટેની ક્રમશઃ કાર્યપ્રણાલીનો નિર્દેશ કરે છે. અલગોરિધમ અંગ્રેજી જેવી પ્રાકૃતિક ભાષામાં લખવામાં આવે છે. એક વાર અલગોરિધમ લખવામાં આવે પછી તેના ઉપરથી કમ્પ્યૂટર પ્રોગ્રામ લખવાનું અતિ સરળ બની જાય છે. ચાલો, હવે થોડાક અલગોરિધમ લખવાનું શીખીએ.

ઉદાહરણ 1 :

1થી 100ની અંદર 11 વડે વિભાજિત સંખ્યાઓનો સરવાળો શોધવાનો અલગોરિધમ લખો.

અલગોરિધમ :

પગલું 1 : આરંભ

પગલું 2 : Num અને Sum બે ચલ લો.

પગલું 3 : શરૂઆતમાં કિંમત આપો. Num = 1 અને Sum = 0.

પગલું 4 : Num ચલ 11 વડે ભાગી શકાય તેમ છે કે કેમ તે તપાસો. આ માટે Numને 11 વડે ભાગીને શેષ શોધવી પડશે. આ શેષને અન્ય ચલ (ધારો કે) Rમાં સંગ્રહ કરો.

R = Num – int(Num/11) * 11

પગલું 5 : જો Rની કિંમત શૂન્ય ન હોય તો પગલું 7 પર જાઓ.

પગલું 6 : Sum = Sum + Num

પગલું 7 : Num = Num + 1

પગલું 8 : જો Num ની કિંમત 100 થી ઓછી હોય તો પગલું-4 પર જાઓ.

પગલું 9 : Sum પ્રિન્ટ કરો.

પગલું 10 : અંત

સમજૂતી :

શેષ શોધવા માટે આપણે Numમાં સંગ્રહ કરેલી સંખ્યાને 11 વડે ભાગીએ છીએ અને પછી તેનો પૂર્ણાંક ભાગ લઈએ છીએ. દા.ત. જો Numની કિંમત 39 હોય તો તેને 11 વડે ભાગવાથી આપણે 3.54545455 મેળવીએ. જો જવાબનો પૂર્ણાંક ભાગ લઈએ તો તે ફક્ત 3 છે. દશાંશ ચિહ્ન પછીનો ભાગ કાઢી નાખવામાં આવે છે. જો પૂર્ણાંક ભાગને 11 સાથે ગુણાકાર કરવામાં આવે તો આપણે 33 મેળવીશું. જો આ 33ને મૂળ સંખ્યા 39માંથી બાદ કરવામાં આવે તો આપણે કિંમત 6 મેળવીશું. આ કિંમત શેષ છે. અહીં એ નોંધ કરશો કે int નામની પ્રક્રિયા કે જે સંખ્યાનો પૂર્ણાંક ભાગ આપે છે, તેની આપણે ધારણા કરેલી છે.

પગલાં 1થી 4 જાતે સમજૂતી આપે તેવાં છે. પગલું 5 ઉપર જણાવેલી તક્નિક વાપરીને શેષની કિંમત તપાસે છે. જો શેષ શૂન્ય ન હોય તો Numની કિંમત 11 વડે વિભાજિત નથી અને આપણે તે પછીની સંખ્યા લઈશું. જો Numની કિંમત 11 વડે વિભાજિત હોય તો આપણે પગલાં 6 ઉપર જઈશું. આપણે Numની કિંમત Sumમાં ઉમેરીશું અને પછી પગલાં 7નો અમલ કરીશું. પગલાં 8માં આપણે Num કિંમત 100 છે કે ઓછી તેની તપાસ કરીશું. આ તપાસ લૂપની બહાર નીકળવા માટે વપરાય છે. આપણે આ રીતે જ્યાં સુધી Num ની કિંમત 100 કરતાં ઓછી હશે ત્યાં સુધી ઉપરની ક્રિયા ચાલુ રાખીશું. અંતમાં આપણે Sum ની કિંમત છાપીશું.

ઉદાહરણ 2 : આપેલી મુદ્દલ રકમ, સમયગાળો અને વ્યાજ દર પરથી વાર્ષિક વ્યાજ ગણવામાં આવે ત્યારે ચક્રવૃદ્ધિ વ્યાજની ગણતરી કરવા માટેનો અલગોરિધમ લખો.

અલગોરિધમ :

આપણે ચક્રવૃદ્ધિ વ્યાજની ગણતરી માટે નીચેનું સૂત્ર વાપરીએ છીએ :

$$CI = P * (1 + R/100)^N - P$$

અહીં P-મુદ્દલ રકમ, R-વ્યાજનો દર અને N-વર્ષમાં સમયગાળો

પગલું 1 : આરંભ

પગલું 2 : ચાર ચલ P, N, R અને CI લો.

પગલું 3 : P, N અને Rની કિંમત આપો.

પગલું 4 : સૂત્ર $CI = P * (1 + R/100)^N - P$ વાપરી ચક્રવૃદ્ધિ વ્યાજ ગણો.

પગલું 5 : CIમાં સંગ્રહ કરેલી કિંમત છાપો.

પગલું 6 : અંત

ઉદાહરણ 3 :

કોઈ કર્મચારીના કામના કલાકોના આધારે અઠવાડિક પગાર શોધો. આ પ્રશ્નના જવાબ માટે આપણે કર્મચારીને એક કલાકનો કેટલો પગાર ચૂકવાય છે તે જાણવું પડે.

અલગોરિધમ :

અઠવાડિક પગાર ચૂકવવા માટે આપણે બે કિંમતના ઇનપુટ (નિવેશ) લેવા પડે : કર્મચારીએ કેટલાં કલાક કામ કર્યું તે અને દરેક કલાકનો પગાર.

પગલું-1 : આરંભ

પગલું-2 : ત્રણ ચલ No_Of_Hrs_Worked, Pay_Per_Hour અને Weekly_Pay લો.

પગલું-3 : No_Of_Hrs_Worked અને Pay_Per_Hourની કિંમત આપો.

પગલું-4 : નીચેનું સૂત્ર વાપરી અઠવાડિક પગારની ગણતરી કરો :

Weekly_Pay = No_Of_Hrs_Worked * Pay_Per_Hour

પગલું-5 : ચલ Weekly_Payમાં સંગ્રહ કરેલી કિંમત છાપો.

પગલું-6 : અંત

સારાંશ

આ પ્રકરણમાં આપણે સમસ્યાના ઉકેલ માટે બે વધારે વપરાતી તકનિક ફ્લોચાર્ટ અને અલગોરિધમ વિશે શીખ્યા. આપણે ફ્લોચાર્ટમાં વપરાતી વિવિધ સંજ્ઞાઓ વિશે પણ જાણ્યું. સમસ્યાના ઉકેલ માટે ફ્લોચાર્ટ વાપરવો કે અલગોરિધમ તેના નિર્ણય માટે વપરાશકર્તાની જરૂરિયાતો બાબત જાણકારી મેળવી.

સ્વાધ્યાય

1. ફ્લોચાર્ટના મૂળભૂત અંગો કયા કયા છે ?
2. કયા પ્રકારની સમસ્યાનો ઉકેલ કમ્પ્યૂટર વડે લાવી શકાય ?
3. કયા પ્રકારનાં કાર્યો એ કમ્પ્યૂટરની પ્રક્રિયાના ભાગ છે ?
4. અંકગણિતનાં કાર્યોમાં કયા કયા કાર્યોનો સમાવેશ થાય છે ?
5. નિર્ણય પેટી(decision box)નો ઉપયોગ કયા પ્રકારનાં કાર્ય દર્શાવવા માટે થાય છે ?
6. ચલને વ્યાખ્યાયિત કરો.
7. અલગોરિધમ એટલે શું ? તે ફ્લોચાર્ટથી કઈ રીતે જુદો પડે છે ?
8. આપેલા વિકલ્પોમાંથી ખરો વિકલ્પ પસંદ કરો :

(1) ફ્લોચાર્ટની શરૂઆત દર્શાવવા માટે કઈ સંજ્ઞાનો ઉપયોગ થાય છે ?

(a)

(b)

(c)

(d)

(2) નીચેનામાંથી શું સમસ્યાના ઉકેલ માટે યોગ્ય ક્રમમાં સૂચનાઓનો નિર્દેશ કરે છે ?

(a) અલગોરિધમ (b) ફ્લોચાર્ટ (c) શ્રેણી (d) રોડમેપ (નકશો)

(3) નીચેનામાંથી કયું ચિહ્ન ફ્લોચાર્ટમાં શરત તપાસવા માટે વપરાય છે ?

(a) હીરો (ડાયમંડ) (b) વર્તુળ (સર્કલ) (c) તીર (ઍરો) (d) ચોરસ (સ્ક્વેર)

(4) નીચેનામાંથી કયું ચિહ્ન ફ્લોચાર્ટમાં આઉટપુટ દર્શાવવા માટે વપરાય છે ?

(a) ચોરસ (સ્ક્વેર) (b) વર્તુળ (સર્કલ)

(c) સમાંતર ચતુર્ભુજ (પૅરેલેલોગ્રામ) (d) ત્રિકોણ (ટ્રાયઍંગલ)

(5) નીચેનામાંથી કયું ચિહ્ન ફ્લોચાર્ટમાં અંત દર્શાવવા માટે વપરાય છે ?

(a) વર્તુળ (b) હીરો

(c) ગોળ ખૂણાવાળો લંબચોરસ (રાઉન્ડેડ રેક્ટેન્ગલ)(d) ચોરસ

(6) નીચેનામાંથી કયું વિધાન અલગોરિધમ અને ફ્લોચાર્ટનો હેતુ જણાવે છે ?

(a) સ્મૃતિ(મેમરી)ની ક્ષમતા જાણવા

(b) અંક પદ્ધતિનો આધાર (base) ઓળખવા

(c) આઉટપુટ(નિર્ગમ)ને પ્રિન્ટર ઉપર મોકલવું

(d) સમસ્યાને સંપૂર્ણપણે અને સ્પષ્ટપણે જણાવવી

(7) નીચેનામાંથી કઈ સમસ્યાના ઉકેલની તકનિક નથી ?

(a) સુડો કોડ (b) ફ્લોચાર્ટ (c) અલગોરિધમ (d) શ્રેણી

(8) નીચેનામાંથી સમસ્યાના ઉકેલની ચિત્રાત્મક રજૂઆત કઈ છે ?

(a) સુડો કોડ (b) ફ્લોચાર્ટ

(c) અલગોરિધમ (d) કમ્પ્યૂટર પ્રોગ્રામ

(9) ફ્લોચાર્ટમાં તીરનું ચિહ્ન શું બતાવવા માટે વપરાય છે ?

(a) કાર્યના પ્રવાહની દિશા (b) કાર્યનો ક્રમ

(c) કાર્યની શરૂઆત (d) કાર્યની પૂર્ણાહુતિ

(10) નીચેનામાંથી શું કોઈ પણ કાર્યનો મુખ્ય ભાગ (core part) છે ?

(a) ઇનપુટ (નિવેશ) (b) આઉટપુટ (નિર્ગમ) (c) પ્રક્રિયા (d) અલગોરિધમ

(11) નીચેનામાંથી કયું ચિહ્ન પ્રક્રિયા દર્શાવે છે ?

(a) લંબચોરસ (b) ચોરસ (c) વર્તુળ (d) હીરો

(12) નીચેનામાંથી ફ્લોચાર્ટમાં જુદા જુદા અનુસંધાન ચિહ્નની જોડી માટે શું વપરાય છે ?

(a) તીર (b) મૂળાક્ષરો કે બીજા અક્ષર

(c) વર્તુળ (d) હીરાનું ચિહ્ન

પ્રાયોગિક સ્વાધ્યાય

નીચેનાં કાર્યો માટે ફ્લોચાર્ટ દોરો અને અલગોરિધમ લખો :

1. આપેલા મીટરને સેન્ટિમીટરમાં ફેરવો.
2. આપેલા સેન્ટિગ્રેડને ફેરનહીટમાં ફેરવો.
3. આપેલી બે સંખ્યામાંથી નાની સંખ્યા શોધો.
4. આપેલી બે સંખ્યામાંથી નાની સંખ્યા શોધો. એ એકી સંખ્યા છે કે બેકી સંખ્યા તે તપાસો.
5. આપેલી ત્રણ સંખ્યામાંથી લઘુતમ સંખ્યા શોધો.
6. આપેલી કોઈ સંખ્યા અન્ય કોઈ આપેલી સંખ્યાથી વિભાજિત છે કે કેમ તે તપાસો.
7. ત્રણ સંખ્યાની સરેરાશ શોધો.
8. કોઈ વિદ્યાર્થીના કુલ ગુણને આધારે શોધો કે તે વિદ્યાર્થી પાસ છે કે નાપાસ (જો કુલ ગુણ 35થી ઓછા હોય તો વિદ્યાર્થીને નાપાસ જાહેર કરો).
9. કોઈ વિદ્યાર્થીના કુલ ગુણને આધારે વિદ્યાર્થીએ કઈ શ્રેણી મેળવી છે તે શોધો. (જો વિદ્યાર્થીના કુલ ગુણ 35થી ઓછા હોય તો વિદ્યાર્થીને D શ્રેણી મળે છે. જો કુલ ગુણ 35 અને 60 વચ્ચે હોય તો C શ્રેણી અને જો કુલ ગુણ 60 અને 70 વચ્ચે હોય તો B શ્રેણી અને 70 કરતાં વધારે ગુણ હોય તો A શ્રેણી મેળવે છે.)
10. X ₹ મુદ્દલ રકમની લોન, R% વ્યાજનો દર અને N વર્ષના સમયગાળાનાં ચક્રવૃદ્ધિ વ્યાજની ગણતરી કરો.
11. આપેલી સંખ્યાના અંકોનો સરવાળો કરો. ઉદાહરણ તરીકે, જો આપેલી સંખ્યા 8327 હોય તો આઉટપુટ (8 + 3 + 2 + 7) = 20 થશે.

સી ભાષાનો પરિચય

નવમા પ્રકરણમાં આપણે ફ્લોચાર્ટ અને અલગોરિધમ વિશે અભ્યાસ કર્યો. આપણે અલગોરિધમની રચના પણ શીખ્યા. ફ્લોચાર્ટ અને અલગોરિધમ એ કોઈ પણ પ્રોગ્રામના ઉકેલ માટેનાં પાયારૂપ પગલાં છે. ત્યાર બાદ આ ફ્લોચાર્ટ કે અલગોરિધમને પ્રોગ્રામમાં ફેરવવામાં આવે છે. પ્રોગ્રામ લખવા માટે અનેક ભાષાઓ ઉપલબ્ધ છે. 'સી' આવી જ એક ભાષા છે. આ પ્રકરણમાં આપણે 'સી' ભાષા વિશે પરિચય મેળવીશું.

પ્રોગ્રામિંગ ભાષાની જરૂરિયાત (Need of programming language)

આપણને સૌપ્રથમ પ્રશ્ન એ થાય કે, પ્રોગ્રામિંગ ભાષાની જરૂરિયાત શા માટે છે ? પહેલેથી આપણે જેને જાણીએ છીએ તેવી અંગ્રેજી કે ગુજરાતી ભાષાનો ઉપયોગ કરીને પ્રોગ્રામ શા માટે લખી ન શકાય ? જવાબ એ છે કે, આ ભાષાઓમાં વાક્યોના એક જ અર્થ હોતા નથી. આથી તે કમ્પ્યૂટર સાથે સુનિશ્ચિત રીતે કાર્ય કરી શકે નહીં. કમ્પ્યૂટર દ્વારા અપેક્ષિત કાર્ય કરાવવા માટે દરેક વાક્ય અર્થપૂર્ણ અને સુનિશ્ચિત હોવું જરૂરી છે.

પ્રોગ્રામિંગ ભાષાનો ઉપયોગ કરી લખવામાં આવેલી સૂચનાઓનો માત્ર એક જ અર્થ હોય છે. તે પૂર્વનિર્ધારિત નિયમોનો ગણ ધરાવે છે. આ નિયમો પ્રોગ્રામિંગ ભાષા માટે વાક્યરચના(syntax)ની રચના કરે છે. આમ, પ્રોગ્રામિંગ ભાષા શીખવી એટલે એવી નવી વાક્યરચના શીખવી, જેના દ્વારા કમ્પ્યૂટરને આપવામાં આવનાર સૂચનાઓ નિશ્ચિતપણે રજૂ કરી શકાય. આ કાર્ય કોઈ પણ નવી ભાષા શીખતી વખતે તેના વ્યાકરણને શીખવા સમાન છે.

અનુવાદકની જરૂરિયાત (Need of translator)

આપણા દ્વારા બોલાયેલી ભાષાને કે પ્રોગ્રામિંગ ભાષાને પણ કમ્પ્યૂટર સમજી શકતું નથી. તે માત્ર 0 અને 1ની ભાષા સમજે છે. આવી મુશ્કેલીનો સામનો આપણે ક્યારેક આપણા જીવનમાં પણ કરતા હોઈએ છીએ. માનો કે X અને Y બે વ્યક્તિઓ છે, જે એકબીજાની ભાષા જાણતી નથી. વ્યક્તિ-X ગુજરાતી ભાષા જાણે છે જ્યારે વ્યક્તિ-Y હિન્દી ભાષા જાણે છે. આ બે વ્યક્તિઓ કેવી રીતે એકબીજા સાથે વાતચિત કરશે ? આના વિકલ્પ તરીકે એક ત્રીજી વ્યક્તિ-Z જરૂરી છે, જે આ બંને ભાષાની જાણકાર હોય. હવે, વ્યક્તિ-Xને વ્યક્તિ-Y સુધી કોઈ સંદેશ પહોંચાડવો હશે ત્યારે, તે પ્રથમ ગુજરાતી ભાષામાં વ્યક્તિ-Zને સંદેશો કહેશે. વ્યક્તિ-Z તે સંદેશાનો હિન્દીમાં અનુવાદ કરશે અને તે સંદેશો વ્યક્તિ-Yને પહોંચાડશે. વ્યક્તિ-Y જ્યારે વ્યક્તિ-X સાથે વાતચીત કરવા ઇચ્છે ત્યારે આ જ પ્રક્રિયાનો વિપરીત ક્રમમાં અમલ કરવામાં આવશે. અહીં વ્યક્તિ-Zને અનુવાદક (translator) તરીકે ઓળખવામાં આવે છે. તથા એક ભાષાને અન્ય ભાષામાં ફેરવવાની ક્રિયાને અનુવાદ (translation) કહેવામાં આવે છે.

આપણી ભાષા નહીં સમજી શકવાની કમ્પ્યૂટરની સમસ્યાનું નિરાકરણ અનુવાદક (translator) પ્રકારના સૉફ્ટવેર પ્રોગ્રામ દ્વારા મેળવવામાં આવે છે. આ અનુવાદકને કંપાઇલર (compiler) તરીકે ઓળખવામાં આવે છે. અનુવાદની પ્રક્રિયા આ પ્રકરણના અંતમાં સમજાવવામાં આવેલી છે.

પ્રોગ્રામ અને પ્રોગ્રામની લાક્ષણિકતાઓ (Program and characteristics of program)

કમ્પ્યૂટર પાસેથી કોઈ પૂર્વ વ્યાખ્યાયિત કાર્ય કરાવવા માટે આપવી પડતી ચોક્કસ અને નિશ્ચિત સૂચનાઓના સમૂહને પ્રોગ્રામ કહેવામાં આવે છે. પસંદ કરેલ ભાષા દ્વારા આ સૂચનાઓને ક્રમબદ્ધ રીતે લખવાની પ્રક્રિયાને પ્રોગ્રામિંગ તરીકે ઓળખવામાં આવે છે.

એક સારો પ્રોગ્રામ નીચે જણાવેલ લાક્ષણિકતાઓ ધરાવતો હોવો જોઈએ :

1. નિશ્ચિત પગલાં પછી પ્રોગ્રામનો અંત આવે તે જરૂરી છે.
2. પ્રોગ્રામમાં આવેલ સૂચનાઓ ચોક્કસપણે વ્યાખ્યાયિત થવી જોઈએ, એટલે કે તે એકથી વધુ અર્થ ધરાવતી ન હોવી જોઈએ.
3. તમામ સૂચનાઓ અસરકારક હોવી જોઈએ, એટલે કે તેનો અમલ યોગ્ય રીતે થવો જોઈએ.

4. પ્રોગ્રામ શૂન્ય કે વધુ ઇનપુટ લઈ શકે.

5. પ્રોગ્રામ એક કે વધુ આઉટપુટ આપી શકે.

તમે જોઈ શકો છો કે, આ લાક્ષણિકતાઓ અલગોરિધમની લાક્ષણિકતાઓને મળતી આવે છે. ઉદાહરણ 10.1 એક નમૂનારૂપ સી પ્રોગ્રામ દર્શાવે છે. ઉદાહરણ 10.1 માટેનું કોડ લિસ્ટિંગ અને પરિણામ આકૃતિ 10.1માં દર્શાવ્યા છે. SciTE એડિટરના ઉપયોગની કાર્ય પદ્ધતિ આપણે આ પ્રકરણના અંતમાં શીખીશું.

```
10_1.c - SciTE
File Edit Search View Tools Options Language Buffers Help
1 10_1.c

/* Example 1: My first C program */

#include <stdio.h>
int main( )
{
    printf("Welcome to the world of C programming using Scite \n");
    return 0;
}
```

```
>gcc -pedantic -Os -std=c99 10_1.c -o 10_1
>Exit code: 0
>./10_1
Welcome to the world of C programming using Scite
>Exit code: 0
```

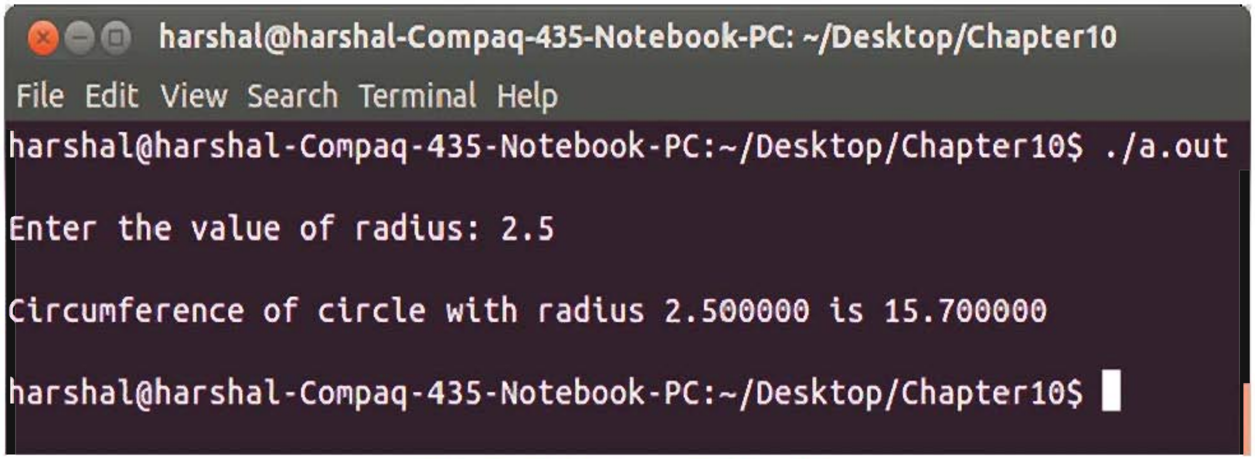
આકૃતિ 10.1 : પ્રથમ 'સી' પ્રોગ્રામ

આ પ્રોગ્રામ દ્વારા "Welcome to the world of C programming using Scite" સંદેશ દર્શાવવામાં આવશે. અહીં main()ને ઉપયોગકર્તા દ્વારા નિર્મિત (user defined) વિધેય કહેવામાં આવે છે જ્યારે printf( ) એ આંતર પ્રસ્થાપિત (inbuilt) કે લાઇબ્રેરી વિધેય છે. ઉપયોગકર્તા દ્વારા તેની જરૂરિયાત અનુસાર રચવામાં આવેલા વિધેયને ઉપયોગકર્તા નિર્મિત વિધેય (User Defined Funcion) કહે છે. ઉપયોગકર્તા પોતાના પ્રોગ્રામમાં ઉપયોગ કરી શકે તેવા 'સી' ભાષામાં પહેલેથી ઉપલબ્ધ વિધેયને આંતરસ્થાપિત કે લાઇબ્રેરી વિધેય તરીકે ઓળખવામાં આવે છે.

સી ભાષાના વિવિધ ઘટકોની વિસ્તૃત ચર્ચા શરૂ કરતાં પહેલાં આકૃતિ 10.2માં દર્શાવેલ પ્રોગ્રામનો અભ્યાસ કરીએ જેનું પરિણામ આકૃતિ 10.3માં દર્શાવ્યું છે.

```
10_2.c - SciTE
File Edit Search View Tools Options Language Buffers Help
1 10_2.c

/* Example 2: Program to calculate Circumference of a circle */

#include <stdio.h>
#define PI 3.14
int main( )
{
    float radius, circumference;
    printf("\nEnter the value of radius: ");
    scanf("%f", &radius);
    circumference = 2 * PI * radius;
    printf("\nCircumference of circle with radius %f is %f \n \n", radius, circumference);
    return 0;
}
/* End of program */
```

આકૃતિ 10.2 ઉદાહરણ 10.2નું કોડ-લિસ્ટિંગ



આકૃતિ 10.3 : ઉદાહરણ 10.2નું પરિણામ

સમજૂતી

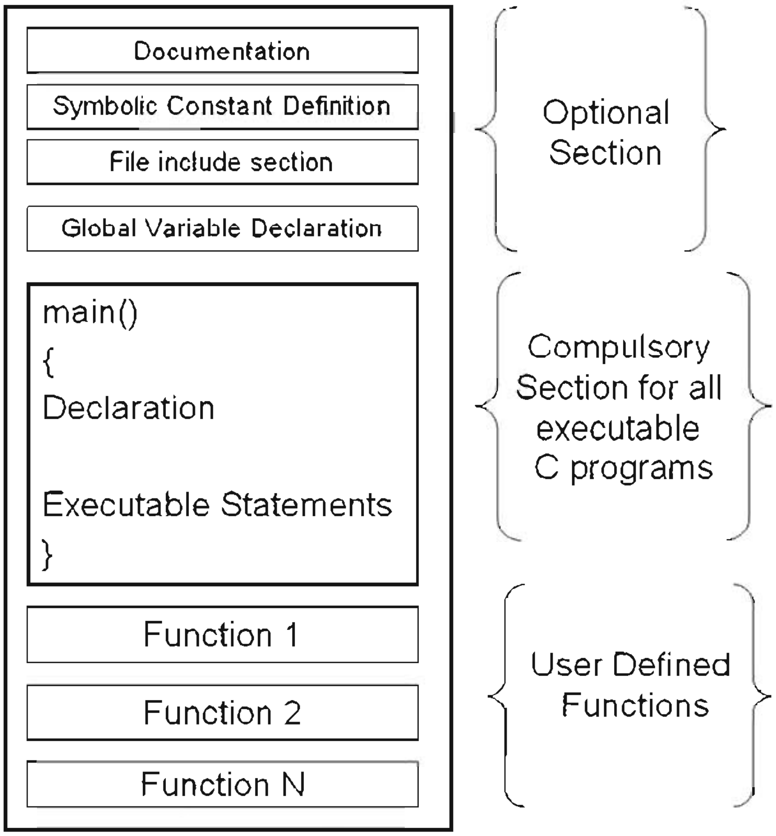
અહીં આપેલ પ્રોગ્રામનો ઉપયોગ કરી આપેલ ત્રિજ્યાને આધારે વર્તુળનો પરિઘ શોધી શકાય છે.

- પ્રથમ વિધાન એવી નોંધ (comment) છે, જે પ્રોગ્રામના ઉદ્દેશ્યનો નિર્દેશ કરે છે.
- બીજા વિધાન દ્વારા કંપાઇલરને "stdio.h" નામની હેડર ફાઈલની વિગતોનો સમાવેશ કરવાની સૂચના આપવામાં આવી છે. પ્રોગ્રામમાં ઉપયોગમાં લેવામાં આવેલા આંતર પ્રસ્થાપિત વિધેય વિશેની માહિતી આ ફાઈલમાં આપેલી હોય છે.
- ત્રીજું વિધાન PI નામના સાંકેતિક અચળ (Symbolic Constant)ને વ્યાખ્યાયિત કરે છે, જેની કિંમત 3.14 છે. આ વિધાનને પ્રી-પ્રોસેસર ડાઈરેક્ટિવ (pre-processor directive) તરીકે ઓળખવામાં આવે છે. પ્રોગ્રામનો અમલ કરતાં પહેલાં આ વિધાન દ્વારા તમામ PI શબ્દને 3.14 કિંમત સાથે બદલવાની સૂચના આપવામાં આવે છે.
- ચોથું વિધાન એ ઉપયોગકર્તા દ્વારા વ્યાખ્યાયિત વિધેય main() છે, જે અમલ થઈ શકે તેવા (executable) તમામ પ્રોગ્રામ માટે ફરજિયાત છે. અહીં આ વિધેય પૂર્ણાંક સંખ્યા પરત કરતું હોવાથી int main() સ્વરૂપે લખવામાં આવ્યું છે.
- પાંચમું વિધાન main( ) વિધેયની શરૂઆત છે.
- છઠ્ઠું વિધાન radius અને circumference નામના બે ચલને વ્યાખ્યાયિત કરે છે. આ ચલ અપૂર્ણાંક સંખ્યાઓનો સંગ્રહ કરવા સક્ષમ છે. સી ભાષા જુદી જુદી કિંમતોનો સંગ્રહ કરવા માટે સક્ષમ છે. સી ભાષામાં જુદી જુદી કિંમતોનો સંગ્રહ કરવા માટે સક્ષમ એવા ઘટકોને ચલ (variable) તરીકે ઓળખવામાં આવે છે.
- સાતમું વિધાન સંદેશ છાપવા માટે printf વિધાનનો ઉપયોગ કરે છે.
- આઠમા વિધાનમાં આંતર પ્રસ્થાપિત વિધેય scanf દ્વારા ઉપયોગકર્તા પાસેથી ત્રિજ્યા (radius) મેળવવામાં આવે છે.
- નવમું વિધાન એક સી એક્સપ્રેશન (C expression) છે, તે વર્તુળનો પરિઘ ગણવા માટેનું સમીકરણ છે.
- દસમું વિધાન ત્રિજ્યા (radius) અને પરિઘ(circumference)ની કિંમતો સાથે સ્ક્રીન પર સંદેશ દર્શાવે છે.
- અગિયારમું વિધાન વિધેયમાંથી સામાન્ય બર્હિગમન (exit) પૂરું પાડે છે. જો આ વિધાન લખવામાં ન આવે તો ચેતવણીનો સંદેશ - "Function should return a value" દર્શાવવામાં આવે છે. પ્રોગ્રામના કાર્યમાં તે કોઈ અસર કરતું નથી, પરંતુ સામાન્ય બર્હિગમન એ હંમેશા એક સારો વિકલ્પ છે.
- બારમું વિધાન main( ) વિધેયનો અંત દર્શાવે છે.

આકૃતિ 10.1 અને 10.2માં દર્શાવ્યા મુજબ main( ) વિધેય સી પ્રોગ્રામના સંપૂર્ણ ભાગનું બંધારણ કરે છે. વધુમાં આપણે એમ કહી શકીએ કે અમલ-યોગ્ય તમામ સી પ્રોગ્રામમાં ઉપયોગકર્તા દ્વારા વ્યાખ્યાયિત એવું એક main( ) નામનું વિધેય હોવું જરૂરી છે. દરેક સી પ્રોગ્રામનું અમલીકરણ main( ) વિધેયના ઉઘડતા છગડિયા કૌંસ ({)થી શરૂ કરવામાં આવે છે.

સી પ્રોગ્રામનું માળખું (Structure of C program)

અગાઉ આપણે જોઈ ગયા તે મુજબ સી પ્રોગ્રામ એ વિધેય નામે ઓળખાતા સમૂહોનો ગણ છે. વિધેય એ એક કે વધુ વિધાનોનો સમૂહ છે, જેના દ્વારા પૂર્વનિર્ધારિત કાર્યોનો અમલ કરી શકાય છે. આકૃતિ 10.4માં સંપૂર્ણ સી પ્રોગ્રામનું માળખું દર્શાવ્યું છે. આ વિભાગોની વિસ્તૃત ચર્ચા કરીએ.



આકૃતિ 10.4 : સંપૂર્ણ સી પ્રોગ્રામનું માળખું

દસ્તાવેજીકરણ વિભાગ (Documentation section)

આ એક મરજિયાત વિભાગ છે. નામ દર્શાવે છે તે મુજબ આ વિભાગનો ઉપયોગ દસ્તાવેજીકરણ માટે કરવામાં આવે છે. તે /* અને */ ની વચ્ચે આવરીને લખવામાં આવે છે. કોઈ પણ સ્થાને /* અને */ ની વચ્ચે આવરીને લખવામાં આવેલા લખાણને 'નોંધ' (comment) માનવામાં આવે છે. કૉમેન્ટ પર સી કમ્પાઈલર દ્વારા કોઈ પ્રક્રિયા કરવામાં આવતી નથી અને તેને જેમની તેમ છોડી દેવામાં આવે છે.

પ્રોગ્રામનો હેતુ, લેખકનું નામ, પ્રોગ્રામ લખ્યાની તારીખ વગેરે જેવી માહિતી દર્શાવવા માટે સામાન્ય રીતે આ વિભાગનો ઉપયોગ કરવામાં આવે છે. સી પ્રોગ્રામમાં કોઈ પણ સ્થાને કૉમેન્ટ ઉમેરી શકાય છે. કૉમેન્ટ ઉમેરવી એ હંમેશાં એક સારી ટેવ ગણાય છે, કારણકે તેના દ્વારા પ્રોગ્રામની વાંચનક્ષમતા અને સમજણ વધે છે.

સાંકેતિક અચળની વ્યાખ્યા (Symbolic constant definition)

આકૃતિ 10.2ના ઉદાહરણમાં દર્શાવ્યા મુજબ પ્રોગ્રામમાં PI નામના સાંકેતિક અચળનો ઉપયોગ કરવામાં આવ્યો છે. પ્રોગ્રામમાં આ પ્રકારના સાંકેતિક અચળનો ઉપયોગ કરવા માટે અચળની આગળ #define ઉમેરવું જરૂરી છે. અહીં, #defineને પ્રી-પ્રોસેસર ડાઇરેક્ટિવ (pre-processor directive) તરીકે ઓળખવામાં આવે છે. તે પ્રોગ્રામમાં આવતા તમામ સાંકેતિક અચળોને આપવામાં આવેલી કિંમત સાથે બદલવાની સૂચના કંપાઇલરને પૂરી પાડે છે. સામાન્ય રીતે સાંકેતિક અચળોને કૅપિટલ અક્ષરો દ્વારા વ્યાખ્યાયિત કરવામાં આવે છે. આમ કરવાથી તેને સામાન્ય ચલ કરતાં અલગ પાડી શકાય છે.

ફાઈલ ઉમેરણ વિભાગ (File include section)

સી ભાષા આંતરપ્રસ્થાપિત કે લાઇબ્રેરી વિધેયો પૂરા પાડે છે. pow( ), sqrt( ) વગેરે તેનાં ઉદાહરણ છે. આ વિધેયોને પૂર્વનિશ્ચિત ઉદેશો હોય છે, જેમ કે, pow( )નો ઉપયોગ x ની આપેલ ઘાત જેટલી કિંમત શોધવા માટે થાય છે તથા આપેલ ઘાત જેટલી કિંમત

શોધવા માટે થાય છે તથા આપેલ સંખ્યાનું વર્ગમૂળ શોધવા માટે sqrt( ) વિધેયનો ઉપયોગ કરવામાં આવે છે. જો જરૂર જણાય તો આપણે પ્રોગ્રામમાં આ વિધેયોનો ઉપયોગ કરી શકીએ છીએ. આ વિધેયોનો ઉપયોગ કરવા માટે તેની માહિતી ધરાવતી ફાઈલોને પ્રોગ્રામમાં ઉમેરવી પડે છે. સી ભાષામાં આ બધી ફાઈલોને હેડર ફાઈલ (header file) તરીકે ઓળખવામાં આવે છે. હેડર ફાઈલનું અનુલંબન ".h" હોય છે. પ્રોગ્રામમાં હેડર ફાઈલ ઉમેરવા માટે #include <filename.h> વાક્ય રચનાનો ઉપયોગ કરવામાં આવે છે. સી ભાષામાં ઉપલબ્ધ અનેક હેડર ફાઈલની યાદી તથા તેના ઉપયોગ અંગેની માહિતી પરિશિષ્ટ IV માં આપવામાં આવી છે.

વૈશ્વિક ચલ ઘોષણા વિભાગ (Global variable declaration section)

સી ભાષામાં ચલનો અમલ તેના વિસ્તાર (scope) પ્રમાણે કરવામાં આવે છે. ઉઘડતા અને બંધ થતા છગડિયા કૌંસ ({}) દ્વારા સી ચલનો વિસ્તાર નક્કી કરવામાં આવે છે. છગડિયા કૌંસમાં વ્યાખ્યાયિત કરેલા ચલને સ્થાનિક ચલ (local variable) તરીકે ઓળખવામાં આવે છે. જેમ કે, ઉદાહરણ 10.2માં ઉપયોગમાં લેવામાં આવેલા radius અને circumference ચલ main() વિધેયના સ્થાનિક ચલ છે. આ પ્રકારના ચલનો ઉપયોગ તેના વિસ્તારની બહાર કરી શકાતો નથી. જો આપણે ચલનો ઉપયોગ તેના વિસ્તારની બહાર પણ તમામ વિધેયોમાં કરવા ઇચ્છતા હોઈએ તો, તે પ્રકારના ચલને વૈશ્વિક ચલ (global variable) કહે છે. આ પ્રકારના ચલને તમામ વિધેયની પહેલા વ્યાખ્યાયિત કરવામાં આવે છે.

main વિધેય (main function)

તમામ સી પ્રોગ્રામમાં main() નામનું વિધેય આવેલું હોય છે. આ વિધેયથી સી પ્રોગ્રામનું અમલીકરણ શરૂ કરવામાં આવે છે. પ્રોગ્રામનું નિયંત્રણ સૌપ્રથમ આ વિધેયને મોકલવામાં આવે છે અને ત્યાંથી અન્ય ક્રિયાઓનો અમલ કરવામાં આવે છે.

ઉપયોગકર્તા નિર્મિત વિધેય (User defined functions)

સી ભાષા કોઈ પણ પ્રોગ્રામને નાના-નાના વિભાગોમાં વહેંચવાની સુવિધા પૂરી પાડે છે. આ વિભાગોને વિધેય કહે છે. આપણા પ્રોગ્રામમાં જરૂરી એવા તમામ અતિરિક્ત વિધેયને આ વિભાગમાં વ્યાખ્યાયિત કરવામાં આવે છે. આકૃતિ 10.5માં દર્શાવેલા ઉદાહરણ 10.3માં ઉપયોગકર્તા દ્વારા નિર્મિત એવા બે વિધેયોનો ઉપયોગ કરવામાં આવ્યો છે.

10_3.c - SciTE

File Edit Search View Tools Options Language Buffers Help

1 10_3.c

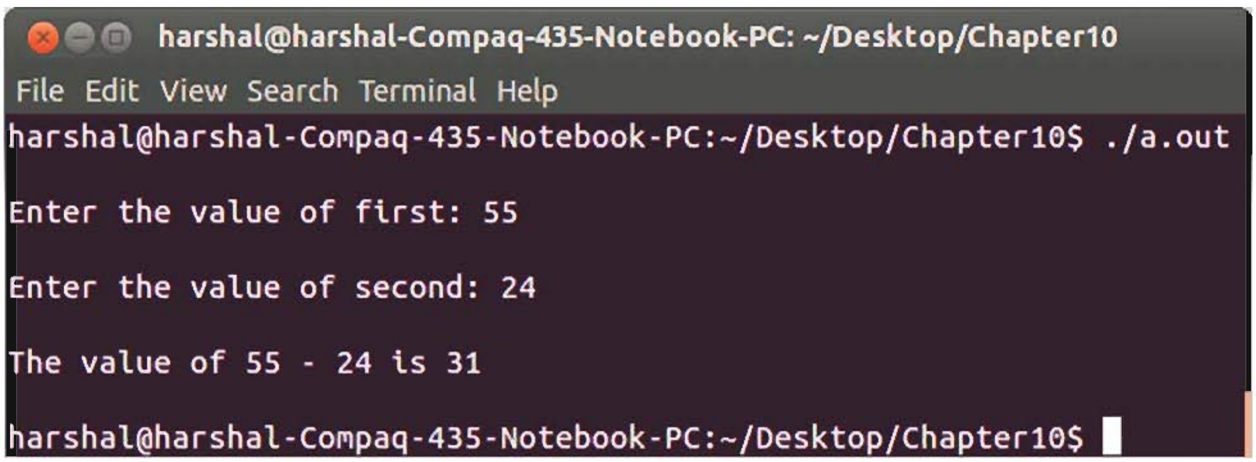
```
/* Example 3: Program to subtract two numbers */

#include <stdio.h>

/* Prototype statement of function sub */
int subtract(int, int);

int main ( )
{
    int first, second, difference;
    printf("\nEnter the value of first: ");
    scanf("%d", &first);
    printf("\nEnter the value of second: ");
    scanf("%d", &second);
    difference = subtract(first, second);
    printf("\nThe value of %d - %d is %d \n \n", first, second, difference);
    return;
}
/* End of main program */

/* Beginning of user defined functions other than main( ) */
int subtract(int first, int second)
{
     int result;
     result = first - second;
     return(result);
}
/* End of function subtract( ) */
```

આકૃતિ 10.5 : ઉદાહરણ 10.3નું કોડ લિસ્ટિંગ

સમજૂતી

આ પ્રોગ્રામ આપેલ બે સંખ્યાઓની બાદબાકી કરવા માટે ઉપયોગી છે. નાનો પ્રોગ્રામ હોવા છતાં તેને ઉપયોગકર્તા દ્વારા નિર્મિત બે વિધેયમાં વિભાજિત કરવામાં આવ્યો છે. પ્રથમ વિધેય main() બાદબાકી કરવા માટેની બે કિંમતો સ્વીકારે છે અને બીજું વિધેય subtract() એ બાદબાકીની ક્રિયા ખરેખર પાર પડે છે.

અહીં એ જોઈ શકાય છે કે subtract() વિધેયનો ઉપયોગ કરતાં પહેલાં તેની પ્રતિકૃતિ (prototype) રજૂ કરવામાં આવી છે. જો ઉપયોગકર્તા દ્વારા નિર્મિત વિધેયને main() વિધેયની નીચે લખવામાં આવ્યું હોય તો સી ભાષામાં આ જરૂરી છે. main() વિધેય ત્રણ પૂર્ણાંક ચલ ઘોષિત કરે છે અને તેમાંથી બે ચલની કિંમત ઉપયોગકર્તાને પૂછે છે. ત્યાર પછી difference = subtract(first, second) વિધાનનો ઉપયોગ કરી આ કિંમતોને subtract વિધેયને મોકલવામાં આવે છે. આ first અને second ને subtract() વિધેયના પ્રાચલ (parameters) કહે છે. આ વિધાનના અમલથી પ્રોગ્રામનું નિયંત્રણ subtract() વિધેયને મોકલવામાં આવે છે. આ વિધેય ત્યાર પછી કિંમતોની બાદબાકી કરી result નામના ચલમાં પરિણામનો સંગ્રહ કરે છે. ત્યાર બાદ result ચલની કિંમત main() વિધેયના વિધાનને પરત કરવામાં આવે છે અને difference નામના ચલમાં તેનો સંગ્રહ કરવામાં આવે છે. અંતમાં main() વિધેય પરિણામને સ્ક્રીન પર દર્શાવી પ્રોગ્રામ પૂરો કરે છે. ઉદાહરણ 10.3નું પરિણામ આકૃતિ 10.6માં દર્શાવ્યું છે.



આકૃતિ 10.6 : ઉદાહરણ 10.3નું પરિણામ

જો કે આકૃતિ 10.4માં દર્શાવેલું સી પ્રોગ્રામનું માળખું તમામ પ્રોગ્રામરો દ્વારા ઉપયોગમાં લેવામાં આવતું પ્રમાણભૂત માળખું નથી. આકૃતિ 10.1માં દર્શાવ્યા મુજબ એ પણ શક્ય છે કે આમાંના કેટલાક વિભાગો પ્રોગ્રામમાં ઉપયોગમાં લેવાયા ન હોય તથા કેટલાક વિભાગોના સ્થાનની અદલાબદલી પણ શક્ય છે. જેમ કે, કોડ લિસ્ટિંગ 10.1માં દર્શાવ્યું છે. તે મુજબ ઉપયોગકર્તા દ્વારા વ્યાખ્યાયિત વિધેયના વિભાગને main() વિધેયના વિભાગ સાથે બદલવામાં આવ્યો છે. અહીં ઉપયોગકર્તા દ્વારા નિર્મિત વિધેયને main() વિધેયની પહેલા ઉમેરવામાં આવ્યું છે.

```
/* Example 4 : Program to subtract two numbers */

#include <stdio.h>

/* Beginning of user defined functions other than main( ) */
int subtract(int first, int second)
{
        int result;
        result = first - second;
return(result);
}
/* End of function subtract( ) */
```

```
/* Beginning of function main() */
int main ( )
{
        int first, second, difference;
        printf("\nEnter the value of first: ");
        scanf("%d", &first);
        printf("\nEnter the value of second: ");
        scanf("%d", &second);
        difference = subtract(first, second);
        printf("\nThe value of %d - %d is %d \n \n", first, second, difference);
        return;
}
/* End of main program */
```

કોડ-લિસ્ટિંગ 10.1 : ઉદાહરણ 10.4નો કોડ

બે તફાવતોને બાદ કરતાં આ પ્રોગ્રામ ઉદાહરણ 10.3ને સમાન છે. પ્રથમ તફાવત એ છે કે subtract() વિધેયને main() વિધેયની પહેલાં વ્યાખ્યાયિત કરવામાં આવ્યું છે અને બીજું એ કે પ્રોટોટાઇપ વિધાનનો ઉપયોગ કરવામાં આવ્યો નથી. ઉદાહરણ 10.3માં difference = subtract (first, second); વિધાન દ્વારા નિયંત્રણ પ્રવાહ main()ની નીચેની તરફ મોકલવામાં આવે છે જ્યારે ઉદાહરણ 10.4માં નિયંત્રણ પ્રવાહ main()ની ઉપરની તરફ જાય છે.

સી ભાષાનાં મૂળાક્ષર (C Character set)

તમને યાદ હશે કે જ્યારે આપણે શાળામાં કોઈ નવી ભાષાનો અભ્યાસ કરતા હતા ત્યારે સૌપ્રથમ આપણે તેના મૂળાક્ષરો શીખ્યા હતા. કોઈ પણ ભાષામાં મૂળાક્ષરો શબ્દોના બંધારણમાં સહાયક બને છે. સી ભાષાને પણ પોતાના મૂળાક્ષરોનો ગણ (character set) છે. આ મૂળાક્ષરોને ચાર વર્ગોમાં વહેંચી શકાય :

1. અક્ષરો (Letters)
2. અંકો (Digits)
3. વ્હાઇટ સ્પેસ (White space)
4. વિશિષ્ટ ચિહ્નો (Special characters)

આ વર્ગોમાં આવતા મૂળાક્ષરો કોષ્ટક 10.1માં દર્શાવ્યા છે :

Letters	Digits	White spaces
A to Z a to z	0 to 9	Blank Space
		Form feed
		Horizontal tab
		New line
		Vertical tab

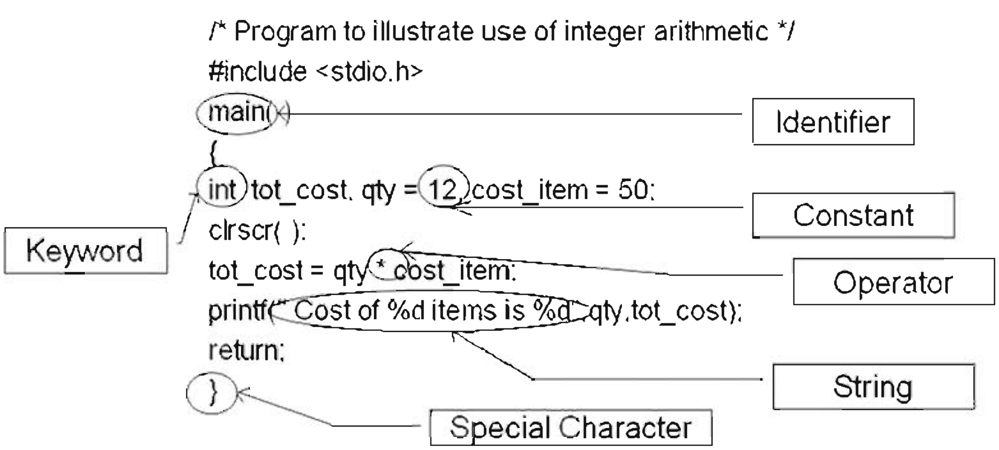
Special Characters			
Operator	**Use**	**Operator**	**Use**
&	Ampersand	>	Greater than
'	Apostrophe	#	Hash sign
*	Asterisk	<	Less than
@	At the rate	-	Minus
\	Backward slash	( )	Parenthesis left / right
{ }	Braces left / right	%	Per cent
[ ]	Bracket left / right	.	Period
^	Caret	+	Plus
:	Colon	?	Question mark
,	Comma	"	Quotation mark
$	Dollar	;	Semi colon
=	Equal to	~	Tilde
!	Exclamation	_	Under score
/	Forward slash	|	Vertical bar

કોષ્ટક 10.1 : સી ભાષાના મૂળાક્ષરો

કોષ્ટક 10.1 માં દર્શાવ્યા મુજબ આપણે અંગ્રેજી ભાષાના કેટલાક અક્ષરોનો ઉપયોગ કર્યો છે. આથી હવે આપણે આ શબ્દોનો સી ભાષાની વાક્યરચનામાં ઉપયોગ કરતાં શીખીશું.

સી શબ્દો (C Words)

કોષ્ટક 10.1માં દર્શાવેલા મૂળાક્ષરોનો ઉપયોગ કરી સી ભાષામાં શબ્દોની રચના કરવામાં આવે છે. આ શબ્દોનો ઉપયોગ કરી સી વિધાન બનાવવામાં આવે છે. આમ, તાર્કિક રીતે ક્રમબદ્ધ લખવામાં આવેલાં સી વિધાનોના સમૂહને સી પ્રોગ્રામ તરીકે ઓળખવામાં આવે છે. સી ભાષામાં આવેલા શબ્દને ટોકન (token) કહે છે. કોષ્ટક 10.1માં આપવામાં આવેલો દરેક અક્ષર પોતે એક ટોકન છે. મૂળભૂત રીતે સી ભાષામાં છ પ્રકારના ટોકન આપવામાં આવે છે : કી-વર્ડ, આઇડેન્ટિફાયર, અચળ, સ્ટ્રિંગ, ઑપરેટર અને વિશિષ્ટ ચિહ્નો. સી પ્રોગ્રામમાં આ ટોકનનો ઉપયોગ આકૃતિ 10.7માં દર્શાવ્યો છે.



આકૃતિ 10.7 : સી પ્રોગ્રામમાં શબ્દો (token)નો ઉપયોગ

કી-વર્ડ (Key word)

આપણે સૌએ ક્યારેક તો શબ્દકોશનો ઉપયોગ કર્યો જ છે. જે-તે ભાષામાં આવેલા કોઈ પૂર્વવ્યાખ્યાયિત શબ્દનો અર્થ શોધવા માટે આપણે શબ્દકોશનો ઉપયોગ કરીએ છીએ. સી ભાષા પણ આવા પૂર્વવ્યાખ્યાયિત શબ્દો ધરાવે છે. ચોક્કસપણે કહીએ તો ANSI C ધારાધોરણ 32 પૂર્વવ્યાખ્યાયિત શબ્દોને સમર્થન આપે છે. સી ભાષામાં આપેલા આ પૂર્વવ્યાખ્યાયિત શબ્દોને કી-વર્ડ તરીકે ઓળખવામાં આવે છે. દરેક કી-વર્ડ સાથે એક સુનિશ્ચિત અર્થ સંકળાયેલો છે. ANSI Cમાં ઉપલબ્ધ કી-વર્ડની યાદી કોષ્ટક 10.2માં આપવામાં આવી છે.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

કોષ્ટક 10.2 : ANSI સી કી-વર્ડ

અભ્યાસક્રમના હવે પછીના પ્રકરણમાં આ કી-વર્ડ અને તેના ઉપયોગો વિશે ચર્ચા કરવામાં આવી છે.

આઇડેન્ટિફાયર (Identifier)

ઉપયોગકર્તા દ્વારા સી મૂળાક્ષરોનો ઉપયોગ કરીને બનાવવામાં આવેલ શબ્દોને આઇડેન્ટિફાયર કહે છે. તેમાં અક્ષરો, અંકો અને કેટલાંક વિશિષ્ટ ચિહ્નોનો સમાવેશ કરી શકાય છે. "difference", "main()", "PI", "float" વગેરે આઇડેન્ટિફાયરનાં કેટલાંક ઉદાહરણ છે. અહીં "difference" ચલનું નામ છે, "main()" વિધેયનું નામ છે, "PI" સાંકેતિક અચળનું નામ છે અને "float" એ પૂર્વવ્યાખ્યાયિત શબ્દ છે. હવે, આગળ વધતાં પહેલાં 'ચલ' શું છે તે વિશે જોઈએ.

ચલ (Variable)

સી પ્રોગ્રામ સામાન્ય રીતે ઇનપુટ સ્વીકારે છે, આ ઇનપુટ ડેટાના સ્વરૂપમાં આવે છે. ડેટાનો સંગ્રહ અને ઉપયોગ કરવા માટે આપણે મેમરીની જગ્યાનો ઉપયોગ કરીએ છીએ. મેમરીમાં આપેલ ડેટા તેના પર કરવામાં આવતી પ્રક્રિયાને આધારે બદલાય છે. મેમરી જગ્યામાં આપેલ ડેટાને ચલ (variable) તરીકે ઓળખાતા એક નામ દ્વારા નિર્દેશિત કરવામાં આવે છે. પોતાને બદલી શકવાની ક્ષમતા ધરાવતા ડેટાને ચલ કહે છે. સી ભાષામાં ચલનું નામ વ્યાખ્યાયિત કરવા કેટલાક નિયમોનું પાલન કરવું જરૂરી છે. આ નિયમો નીચે દર્શાવ્યા છે :

1. ચલનું નામ કી-વર્ડના નામ જેવું ન હોવું જોઈએ.
2. ચલના નામમાં મૂળાક્ષરો, અંકો અને અંડરસ્કોરનો ઉપયોગ કરી શકાય છે. અન્ય કોઈ સ્પેશિયલ કેરેક્ટર માન્ય નથી.
3. ચલના નામનો પ્રથમ અક્ષર મૂળાક્ષર અથવા અંડરસ્કોર હોય તે જરૂરી છે.
4. ANSI ધારાધોરણ પ્રમાણે ચલના નામની મહત્તમ લંબાઈ 31 અક્ષર છે. જો કે, તે કમ્પાઇલર પર આધારિત છે.
5. ચલના નામ કેસ-સેન્સિટિવ છે, એટલે કે num, nuM, nUM, nUm અને Num આ તમામ જુદા જુદા

ચલનાં કેટલાંક માન્ય અને અમાન્ય નામ કોષ્ટક 10.3માં દર્શાવ્યાં છે :

માન્ય
Double → Double એ કી-વર્ડ doubleને સમાન નથી.
INTEREST → કેપિટલ અક્ષરો સી મૂળાક્ષરોનો ભાગ છે.
total_quantity → અન્ડરસ્કોર અને અક્ષરો માન્ય છે.
mark100 → પ્રથમ અક્ષર આલ્ફાબેટ હોવાથી માન્ય છે.
_file → પ્રથમ અક્ષર અન્ડરસ્કોર તરીકે શક્ય છે.
અમાન્ય
char → કી-વર્ડનો ઉપયોગ માન્ય નથી.
Total Value → ખાલી જગ્યા માન્ય નથી.
total&value → વિશિષ્ટ અક્ષરો માન્ય નથી.
10mark → પ્રથમ અક્ષર અન્ડરસ્કોર કે આલ્ફાબેટ જોઇએ.

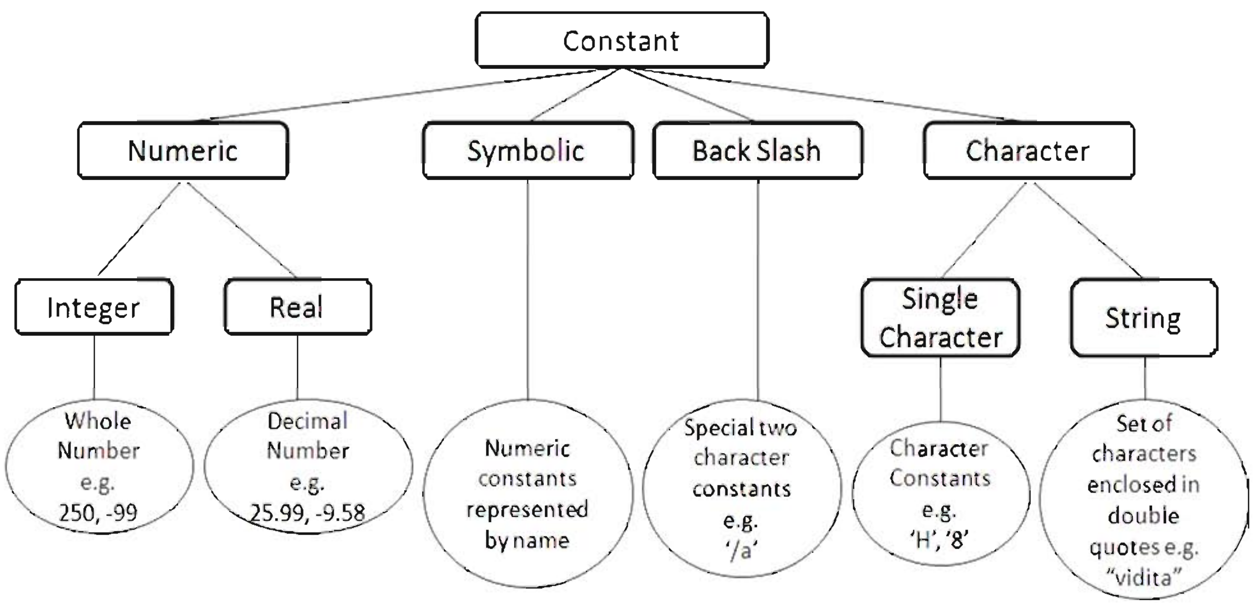
કોષ્ટક 10.3 : સી ચલનાં ઉદાહરણ

અચળ (Constant)

પ્રોગ્રામના અમલીકરણ દરમિયાન જે ઘટકની કિંમત બદલી શકાતી નથી તેને અચળ તરીકે ઓળખવામાં આવે છે. સી અચળોને આકૃતિ 10.7માં દર્શાવ્યા મુજબ જુદા જુદા વર્ગોમાં વહેંચી શકાય છે. આ અચળોની વિસ્તૃત ચર્ચા કરીએ.

સાંખ્યિક અચળ (Numeric constant)

આંકડાકીય વિગતોનો સંગ્રહ કરતા અચળને સાંખ્યિક અચળ કહે છે. સાંખ્યિક અચળોને બે વર્ગોમાં વિભાજિત કરી શકાય : પૂર્ણાંક (integer) અચળ અને અપૂર્ણાંક (real) અચળ



આકૃતિ 10.7 : સી અચળો

પૂર્ણાંક અચળ (Integer constant)

પૂર્ણાંક અચળ એટલે દશાંશ ચિહ્ન વગરની સંખ્યાઓ (whole numbers). પૂર્ણાંક અચળો ત્રણ પ્રકારના છે : દશાંકી (decimal), સોળઅંકી (hexadecimal) અને અષ્ટાંકી (octal). જુદી જુદી અંક-પદ્ધતિઓનાં ઉદાહરણ પરિશિષ્ટ I માં આપવામાં આવ્યાં છે.

દશાંકી અચળો 0 થી 9 અંકોનો ઉપયોગ કરે છે. આ અંકોની આગળ મુનરી ધન (plus) કે ઋણ (minus) નિશાની ઉમેરી શકાય છે.

સોળઅંકી અચળો 0 થી 9 અંકો અને A થી F અક્ષરોનો ઉપયોગ કરે છે. અહીં A થી F અક્ષરો અનુક્રમે 10થી 15 સંખ્યાઓનો નિર્દેશ કરે છે. સોળઅંકી સંખ્યાનો આધાર 16 છે. આ કિંમતોનો સી ભાષામાં ઉપયોગ કરવામાં આવે ત્યારે તેને દશાંકી સંખ્યાથી અલગ પાડવા તે સંખ્યાની આગળ 0x અથવા 0X ઉમેરવામાં આવે છે.

અષ્ટાંકી અચળો 0 થી 7 અંકોનો ઉપયોગ કરે છે. આ અંક-પદ્ધતિનો આધાર 8 છે. આ કિંમતોનો સી ભાષામાં ઉપયોગ કરવામાં આવે ત્યારે દશાંકી સંખ્યાથી અલગ પાડવા તેની આગળ 0 ઉમેરવામાં આવે છે. કોષ્ટક 10.4 કેટલાક માન્ય અને અમાન્ય પૂર્ણાંક અચળ દર્શાવે છે.

માન્ય
40000 → માન્ય દશાંકી ધન પૂર્ણાંક સંખ્યા
–120 → માન્ય દશાંકી ઋણ પૂર્ણાંક સંખ્યા
79999L → માન્ય દશાંકી ધન લૉંગ અપૂર્ણાંક સંખ્યા
0xB5 → માન્ય સોળઅંકી સંખ્યા
0X79 → માન્ય સોળઅંકી સંખ્યા
045 → માન્ય અષ્ટાંકી સંખ્યા
અમાન્ય
25,000 → અલ્પવિરામ માન્ય નથી.
–100.0 → તે અપૂર્ણ અચળ છે.
Rs79999 → અક્ષરો માન્ય નથી.
0xG → G સોળઅંકી અંકમાં ન હોઈ શકે.
0X8,9 → અલ્પવિરામ માન્ય નથી.
096 → 9 અષ્ટાંકી સંખ્યાના ભાગ સ્વરૂપે ન હોઈ શકે.

કોષ્ટક 10.4 : પૂર્ણાંક અચળ

અપૂર્ણ અચળ (Real constant)

અપૂર્ણાંક ભાગ ધરાવતા દશાંકી અંક અપૂર્ણ અચળ તરીકે ઓળખાય છે. 10.50, –65.75 વગેરે અપૂર્ણ અચળનાં ઉદાહરણ છે. અપૂર્ણ અચળોની રજૂઆત વૈજ્ઞાનિક સ્વરૂપે પણ કરી શકાય છે. આ પ્રકારની રજૂઆત માટે મેન્ટિસા (mantissa) અને એક્સ્પોનન્ટ (exponent)નો ઉપયોગ કરવામાં આવે છે. ઉદાહરણ તરીકે, 25.75 સંખ્યાને વૈજ્ઞાનિક સ્વરૂપમાં 0.2575e2 અથવા 0.2575E2 તરીકે રજૂ કરી શકાય છે. અહીં 0.2575 ને મેન્ટિસા અને 2ને એક્સ્પોનન્ટ કહે છે. વળી, "e" અને "E" સરખા છે. કોષ્ટક 10.5માં કેટલાક માન્ય અને અમાન્ય અપૂર્ણ અચળની યાદી આપી છે.

માન્ય
250.00 → માન્ય દશાંકી અપૂર્ણાંક સંખ્યા
295 → માન્ય દશાંકી સંખ્યા, આપોઆપ 295.00 માં રૂપાંતરણ પામશે.
-15.55 → માન્ય ઋણ દશાંકી સંખ્યા
-20.5e5 → માન્ય વૈજ્ઞાનિક કિંમત
15E-2 → માન્ય વૈજ્ઞાનિક કિંમત
અમાન્ય
19,800.00 → અલ્પવિરામ માન્ય નથી.
$786 → વિશિષ્ટ ચિહ્ન માન્ય નથી.
-9.4e5.0 → એક્સ્પોનન્ટ પૂર્ણાંક હોવો જોઇએ.
51E -2 → ખાલી જગ્યા માન્ય નથી.

કોષ્ટક 10.5 : અપૂર્ણ અચળ

અક્ષર પ્રકારના અચળ (Character constant)

નામ પ્રમાણે આ અચળમાં અક્ષરોનો સમાવેશ કરવામાં આવે છે. સી ભાષામાં બે પ્રકારના અક્ષર અચળો ઉપલબ્ધ છે : એક અક્ષર સ્વરૂપે અચળ અને સ્ટ્રિંગ અચળ

એક અક્ષર સ્વરૂપે અચળ (Single character constant)

આ પ્રકારના અચળમાં એક અક્ષરને એક અવતરણ ચિહ્ન (single quote) માં આવરીને રજૂ કરવામાં આવે છે. 'H', 'v', '7', '$' વગેરે એક અક્ષર સ્વરૂપે રજૂ કરવામાં આવેલા અચળનાં કેટલાંક ઉદાહરણ છે. અહીં દરેક અક્ષર અચળ ASCII (American Standard Code for Information Interchange) નામે ઓળખાતી કિંમત સાથે જોડાયેલ હોય છે. ASCII કિંમતો અને તેની સાથે જોડાયેલા જુદા જુદા અક્ષરોની વિગતો પરિશિષ્ટ II માં આપવામાં આવી છે.

સ્ટ્રિંગ અચળ (String constant)

સ્ટ્રિંગ અચળને બે અવતરણ ચિહ્નો(double quotes)માં આવરીને લખવામાં આવેલા અક્ષરોના સમૂહ દ્વારા રજૂ કરવામાં આવે છે. "C Language", "Bye", "V" વગેરે સ્ટ્રિંગ અચળનાં કેટલાંક ઉદાહરણ છે. સ્ટ્રિંગ અચળ સાથે કોઈ ASCII કિંમત જોડાયેલી હોતી નથી. અહીં, સ્ટ્રિંગ અચળ "V" અને અક્ષર સ્વરૂપે રહેલ અચળ 'V' સરખાં નથી. સ્ટ્રિંગ "V" માટે બે મેમરી-સ્થાન આપવામાં આવે છે, જ્યારે 'V' અક્ષરને માત્ર એક મેમરી-સ્થાન આપવામાં આવે છે. કારણ કે, સી ભાષામાં સ્ટ્રિંગનો અંત નલ અક્ષર '\0' થી થાય છે.

બૅક સ્લૅશ અક્ષરો (Back slash characters)

નામ અનુસાર 'સિંગલ કૅરેક્ટર અચળ' એક અક્ષરનો ઉપયોગ કરે છે જ્યારે સ્ટ્રિંગ અનેક અક્ષરોનો ઉપયોગ કરે છે. સી ભાષા એક અન્ય પ્રકારના વિશિષ્ટ અચળ પૂરાં પાડે છે જે બે અક્ષરોનો ઉપયોગ કરે છે. આ અચળોને બૅક સ્લૅશ અક્ષરો (back slash characters) અથવા એસ્કેપ સીક્વન્સ (escape sequence) કહે છે. પ્રથમ અક્ષર હંમેશા બૅક સ્લૅશ "\" હોવાને કારણે આ અચળોને બૅક સ્લૅશ અક્ષરો તરીકે ઓળખવામાં આવે છે તથા આ અચળોના પરિણામસ્વરૂપે વ્હાઇટ સ્પેસ દર્શાવાતી હોવાને કારણે તેને 'એસ્કેપ સીક્વન્સ' પણ કહે છે. સિંગલ કૅરેક્ટર અચળની જેમ આ અચળોની સાથે પણ એક ASCII કિંમત સંકળાયેલી હોય છે. સી ભાષામાં ઉપલબ્ધ બૅક સ્લૅશ અચળો અને તેના ઉપયોગો તથા ASCII કિંમતોની યાદી કોષ્ટક 10.6માં આપવામાં આવી છે.

Back Slash Character	Use	ASCII Value
\0	નલ કિંમત ઉમેરવા માટે	0
\a	શ્રવણીય એલર્ટ ઉમેરવા માટે	7
\b	બેકસ્પેસ ઉમેરવા માટે	8
\t	આડી ટેબ ઉમેરવા માટે	9
\n	નવી લીટી ઉમેરવા માટે	10
\v	ઊભી ટેબ ઉમેરવા માટે	11
\f	ફોર્મ ફીડ ઉમેરવા માટે	12
\r	કેરેજ રીર્ટન ઉમેરવા માટે	13
\"	બે અવતરણ ચિહ્ન ઉમેરવા માટે	34
\'	એક અવતરણ ચિહ્ન ઉમેરવા માટે	39
\?	પ્રશ્નાર્થ ચિહ્ન ઉમેરવા માટે	63
\\	બેક સ્લેશ ઉમેરવા માટે	92

કોષ્ટક 10.6 બેક સ્લૅશ અક્ષરો

એસ્કેપ સિકવન્સનો ઉપયોગ મુખ્યત્વે ગોઠવણી(formatting)ના હેતુસર કરવામાં આવે છે. ઉદાહરણ તરીકે '\n' નો ઉપયોગ કરી ઇનપુટ કે આઉટપુટ સમયે નવી લીટી ઉમેરી શકાય છે. આ અક્ષરોનો ઉપયોગ તમે પુસ્તકમાં અનેક જગ્યાએ જોઈ શકશો.

સાંકેતિક અચળ (Symbolic Constant)

સાંકેતિક આઇડેન્ટિફાયરની મદદથી ઘણા આંકડાકીય અને અક્ષરીય અચળોને વ્યાખ્યાયિત કરી શકાય છે. આ પ્રકારના અચળને સાંકેતિક અચળ કહે છે. સાંકેતિક અચળોને વ્યાખ્યાયિત કરવા માટે નીચે જણાવેલ વાક્યરચનાનો ઉપયોગ કરવામાં આવે છે :

#define identifier value

અહીં, #defineને પ્રી-પ્રોસેસર ડાઇરેક્ટિવ (pre-processor directive) તરીકે ઓળખવામાં આવે છે. જે અચળ વ્યાખ્યાયિત કરવાનો છે તેનું સાંકેતિક નામ 'આઇડેન્ટિફાયર' છે અને value એ સ્વયં અચળ છે. સાંકેતિક અચળનાં કેટલાંક ઉદાહરણ નીચે દર્શાવેલાં છે :

```
#define PI 3.14
#define MAXVALUE 100
#define f float
```

અહીં પ્રથમ વિધાન "PI" નામના સાંકેતિક અચળને વ્યાખ્યાયિત કરે છે જેની કિંમત અપૂર્ણાંક "3.14" છે. બીજું વિધાન "MAXVALUE" નામના સાંકેતિક અચળને વ્યાખ્યાયિત કરે છે જેની કિંમત પૂર્ણાંક 100 છે અને અંતિમ વિધાન "f" નામના સાંકેતિક અચળને વ્યાખ્યાયિત કરે છે, જેની કિંમત સી ભાષામાં આવેલ કી-વર્ડ "float" છે.

પ્રોગ્રામમાં ઉપયોગમાં લેવામાં આવેલા તમામ સાંકેતિક અચળોને તેની વ્યાખ્યામાં આપેલ કિંમત સાથે બદલવાની સૂચના પ્રી-પ્રોસેસર ડાઇરેક્ટિવ વિધાન કંપાઇલરને આપે છે. એક ગોલક (Sphere)ની સપાટીનું ક્ષેત્રફળ શોધવા માટેનો પ્રોગ્રામ ઉદાહરણ 10.5માં દર્શાવ્યો છે, જે પ્રી-પ્રોસેસર ડાઇરેક્ટિવનો ઉપયોગ કરે છે. ઉદાહરણ 10.5નું કોડ લિસ્ટિંગ આકૃતિ 10.8માં દર્શાવ્યું છે તથા આકૃતિ 10.9 પ્રોગ્રામનું પરિણામ દર્શાવે છે.

```
/* Example 5: Program to find surface area of a sphere */

#include <stdio.h>

/* Definition of a symbolic constant */

#define F float
#define P printf
#define S scanf
#define PI 3.14

int main( )
{
    F radius, sarea;
    P("\nEnter the value of radius: ");
    S("%f", &radius);
    sarea = 4 * PI * radius * radius;
    P("\nSurface Area of sphere with radius %.2f is %.2f\n\n",radius,sarea);
    return 0;
}
/* End of program */
```

આકૃતિ 10.8 : ઉદાહરણ 10.5નું કોડ લિસ્ટિંગ

```
harshal@harshal-Compaq-435-Notebook-PC:~/Desktop/Chapter10$ ./a.out

Enter the value of radius: 2.5

Surface Area of sphere with radius 2.50 is 78.50

harshal@harshal-Compaq-435-Notebook-PC:~/Desktop/Chapter10$
```

આકૃતિ 10.9 : ઉદાહરણ 10.5નું પરિણામ

સમજૂતી

આ પ્રોગ્રામમાં ચાર સાંકેતિક અચળો વ્યાખ્યાયિત કર્યા છે : 'F' કી-વર્ડ 'float' માટે, 'P' વિધેય 'printf' માટે, 'S' વિધેય 'scanf' માટે તથા 'PI' અપૂર્ણ કિંમત '3.14' માટે. main() વિધેયનું પ્રથમ વિધાન F ડેટા પ્રકાર માટે બે ચલ વ્યાખ્યાયિત કરે છે. F એ floatનો નિર્દેશ કરતું હોવાથી ખરેખર અહીં બે float ચલને વ્યાખ્યાયિત કરવામાં આવે છે. ત્રીજું વિધાન Pનો ઉપયોગ કરી સંદેશ દર્શાવે છે. ત્યારપછી Sનો ઉપયોગ કરી ત્રિજ્યાની કિંમત વાંચવામાં આવે છે અને ગોલક (Sphere)ના સપાટીનાં ક્ષેત્રફળની ગણતરી કરવામાં આવે છે. અંતમાં ફરી Pનો ઉપયોગ કરી સંદેશ સાથે સપાટીનું ક્ષેત્રફળ દર્શાવવામાં આવે છે.

સી પ્રોગ્રામમાં યાદ રાખવાના મુદ્દા (Points to be remember in C program)

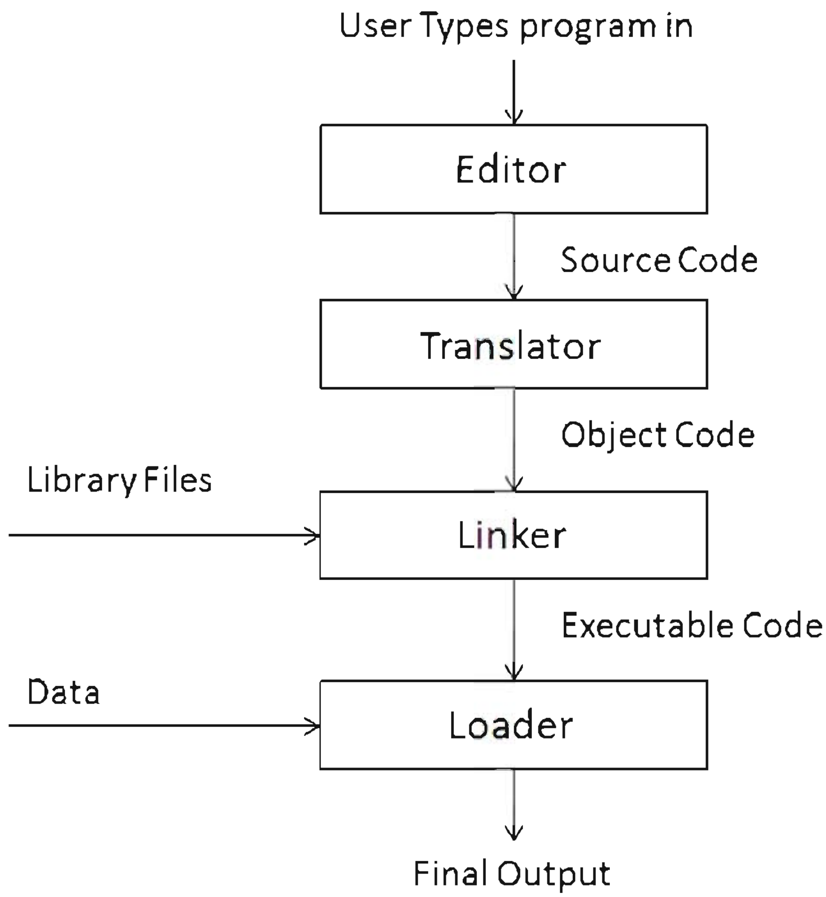
સી ભાષામાં પ્રોગ્રામિંગ સરળ હોવા છતાં, તેનો મહાવરો જરૂરી છે. તમામ સી પ્રોગ્રામરોએ નીચેના વિભાગમાં આપવામાં આવેલ મુદ્દા યાદ રાખવા જોઈએ :

- હેડર ફાઈલને main() વિધેયની પહેલાં ઉમેરવી જોઈએ.
- અમલ કરી શકાય તેવા કોઈ પણ સી પ્રોગ્રામમાં હંમેશા main() વિધેય હોય છે.

- સી પ્રોગ્રામનું અમલીકરણ main() પછીના ઉઘડતા છગડિયા કૌંસ ({)થી શરૂ કરવામાં આવે છે.
- સી પ્રોગ્રામમાં સરખાં નામ ધરાવતાં બે વિધેય માન્ય નથી.
- સામાન્ય રીતે, સી પ્રોગ્રામ નાના (small) અક્ષરો દ્વારા લખવામાં આવે છે. આઇડેન્ટિફાયર કેસ-સેન્સિટિવ છે.
- સી પ્રોગ્રામના બે શબ્દો વચ્ચે ઓછામાં ઓછી એક ખાલી જગ્યા હોવી જરૂરી છે.
- સામાન્ય રીતે સી ભાષામાં વિધાનોને અર્ધવિરામ (semicolon) દ્વારા પૂરાં કરવામાં આવે છે.
- ખાલી જગ્યા ઉમેરી શકાય તેવા તમામ સ્થાને નોંધ (comment) ઉમેરી શકાય છે.
- દરેક ઉઘડતા છગડિયા કૌંસ({)ને સંબંધિત પૂરો થતો છગડિયો કૌંસ (}) હોવો અનિવાર્ય છે.

સી પ્રોગ્રામનું અમલીકરણ (Execution of C program)

અત્યાર સુધીમાં આપણે ઘણા સી પ્રોગ્રામ અને તેનાં પરિણામ જોયાં. હવે આપણે સી પ્રોગ્રામનો અમલ કરતાં શીખીએ. પ્રત્યક્ષ અનુભવ કરતાં પહેલાં આ માટે જરૂરી એવાં સોપાન સમજીએ. આકૃતિ 10.9(a)માં આ સંપૂર્ણ પ્રક્રિયા દર્શાવી છે.

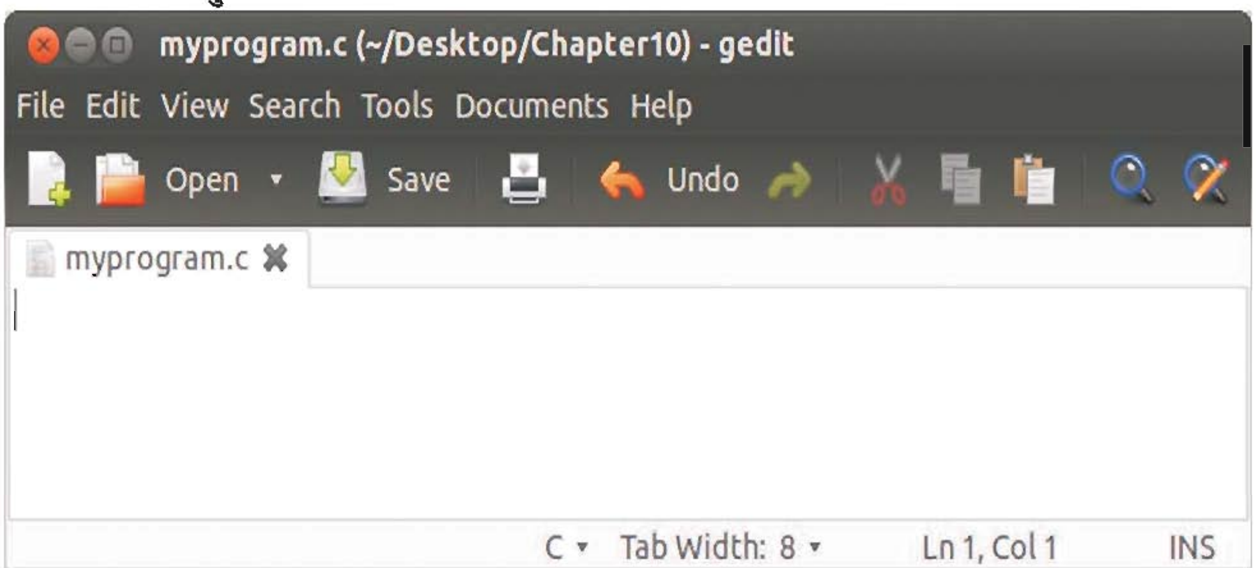


આકૃતિ 10.9 (a) : સી પ્રોગ્રામનાં અમલીકરણનાં પગલાં

આકૃતિ 10.9(a) માં દર્શાવ્યા મુજબ, ટેક્સ્ટ એડિટરનો ઉપયોગ કરી પ્રોગ્રામ લખવો એ સૌ પ્રથમ સોપાન છે. ટેક્સ્ટ એડિટરની મદદથી લખવામાં આવેલા પ્રોગ્રામને સોર્સ કોડ (source code) અથવા સોર્સ પ્રોગ્રામ (source program) કહે છે. સી ભાષામાં લખવામાં આવેલ સી ફાઈલનું અનુલંબન ".c" છે. ત્યાર પછી કંપાઈલરની મદદથી પ્રોગ્રામને સંકલિત (compile) કરવામાં આવે છે. કંપાઈલર એ એક અનુવાદક (translator) છે, જે સોર્સ પ્રોગ્રામને ઑબ્જેક્ટ કોડ (object code) કે ઑબ્જેક્ટ પ્રોગ્રામ (object program) નામે ઓળખાતા મશીન ભાષાના કોડમાં રૂપાંતરિત કરે છે. ઑબ્જેક્ટ કોડના જુદાં જુદાં સ્વરૂપો ઉપલબ્ધ છે. ત્યાર પછી લિંકર (linker) નામે ઓળખાતા પ્રોગ્રામની મદદથી ઑબ્જેક્ટ કોડ સાથે લાઈબ્રેરી ફાઈલનું જોડાણ (linking) કરી એક્ઝિક્યુટેબલ પ્રોગ્રામ (executable program) કે એક્ઝિક્યુટેબલ કોડ (executable code) તૈયાર કરવામાં આવે છે. અંતમાં, પરિણામ દર્શાવવા માટે આ એક્ઝિક્યુટેબલ કોડને જરૂરી વિગતો સાથે લોડર (loader) નામના પ્રોગ્રામની મદદથી મેમરીમાં મૂકવામાં આવે છે.

આ પુસ્તકમાં આપણે gcc નામના કંપાઇલરનો ઉપયોગ કર્યો છે. gcc કંપાઇલર Free Software Foundation દ્વારા પૂરું પાડવામાં આવ્યું છે. તે યુનિક્સ / લિનક્સ આધારિત ANSI સી કમ્પાઇલર છે. સામાન્ય રીતે તેને કમાન્ડ લાઇન દ્વારા ઉપયોગમાં લેવામાં આવે છે, પરંતુ તેને SciTE જેવા ટેક્સ્ટ એડિટર અથવા Eclipse જેવા ઇન્ટિગ્રેટેડ ડેવલપમેન્ટ એન્વાયર્મેન્ટ (Integrated Development Environment - IDE) સાથે પણ સાંકળી શકાય છે. મોટાભાગના તમામ લિનક્સ-પ્રસ્થાપનો સાથે તે પૂર્વસ્થાપિત હોય છે, જેથી આપણે તેનો ઉપયોગ કોઈ જ મુશ્કેલી વગર સરળતાથી કરી શકીએ છીએ.

ચાલો, આપણે કમ્પાઇલરનો ઉપયોગ કરીએ. સૌ પ્રથમ ટેક્સ્ટ એડિટરનો ઉપયોગ કરી પ્રોગ્રામ લખવાની જરૂર પડશે. સી પ્રોગ્રામ લખવા માટે કોઈ પણ ટેક્સ્ટ એડિટરનો ઉપયોગ કરી શકાય. લિનક્સ ઑપરેટિંગ સિસ્ટમમાં vi, gedit, emacs અને બીજા ઘણા ટેક્સ્ટ એડિટર ઉપલબ્ધ છે. તમને અનુકૂળ હોય તેવું કોઈ એક ટેક્સ્ટ એડિટર પસંદ કરો. એકમાત્ર વાત અહીં ધ્યાનમાં રાખવી જોઇએ કે બનાવવામાં આવેલી ફાઈલનો .c અનુલંબન આપી સંગ્રહ કરવો જરૂરી છે. અક્ષર 'c' નાના (small) અક્ષરોમાં લખાયેલ હોય તે પણ જરૂરી છે. આપણે સી પ્રોગ્રામની રચના કરવા માટે gedit નામના ટેક્સ્ટ એડિટરનો ઉપયોગ કરીએ. ટર્મિનલ ખોલી, તેના પ્રોમ્પ્ટ પર gedit myprogram.c ટાઇપ કરો. આમ કરવાથી આકૃતિ 10.10માં દર્શાવ્યા મુજબનું એક ખાલી એડિટર ખુલશે.



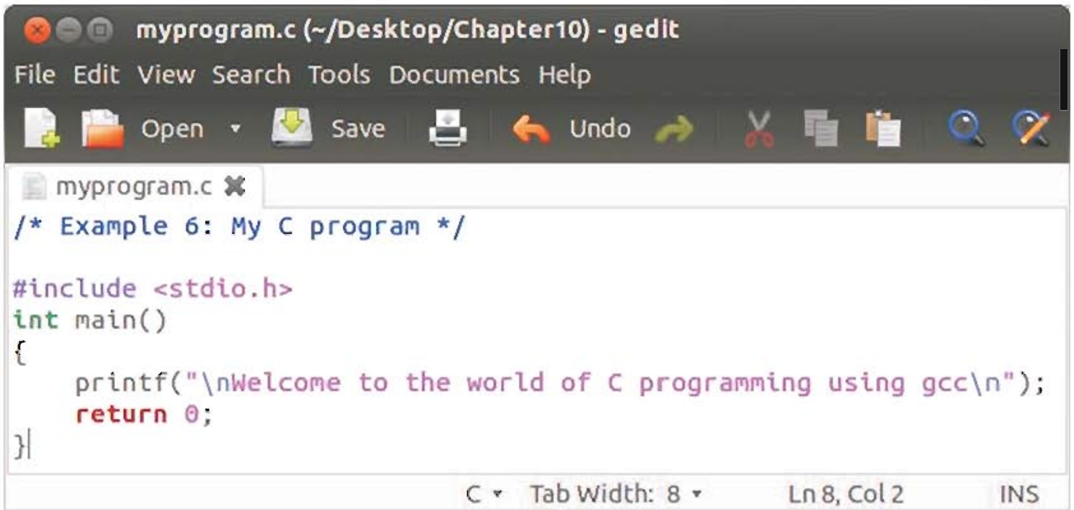
આકૃતિ 10.10 : ખાલી એડિટર સ્ક્રીન

આ ખાલી એડિટર સ્ક્રીનમાં હવે કોડ લિસ્ટિંગ 10.2માં આપવામાં આવેલ વિગતો ઉમેરો.

```
/* Example 6 : My C program */

#include <stdio.h>
int main()
{
    printf("\nWelcome to the world of C programming using gcc\n");
     return 0;
}
```

કોડ લિસ્ટિંગ 10.2 : ઉદાહરણ 10.6નો કોડ

પ્રોગ્રામ તૈયાર થઈ ગયા પછી એડિટર આકૃતિ 10.11માં દર્શાવ્યા મુજબ દેખાશે. ફાઈલનો સંગ્રહ કરી એડિટર બંધ કરો.



આકૃતિ 10.11 : ઉદાહરણ 10.6ના કોડ લિસ્ટિંગ સાથે એડિટર

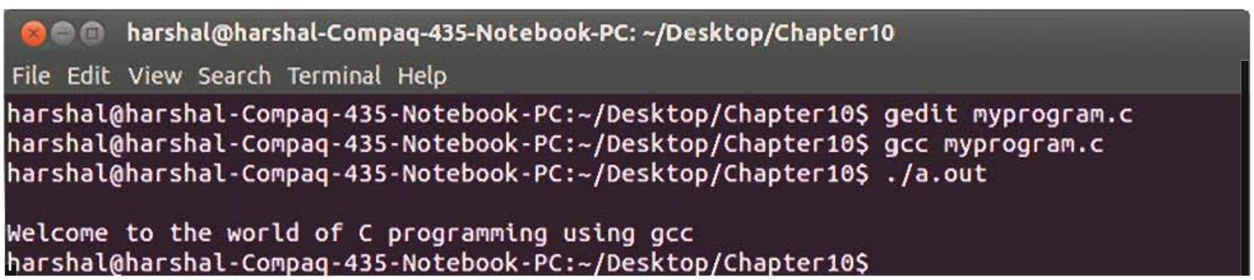
હવે આપણે તેનું કમ્પાઇલેશન કરવા તૈયાર છીએ. ટર્મિનલમાં પાછા ફરી નીચે જણાવેલ કમાન્ડ ટાઇપ કરો :

$ gcc myprogram.c

જો પ્રોગ્રામમાં કોઈ ક્ષતિ નહીં હોય તો નવા પ્રોમ્પ્ટની લીટી દર્શાવાશે. આનો અર્થ એ કે કમ્પાઇલેશનની ક્રિયા સફળતાપૂર્વક પૂરી થઈ છે અને સોર્સ ફાઈલનો સંગ્રહ કરવામાં આવ્યો છે તે જ ડિરેક્ટરીમાં a.out પૂર્વ નિર્ધારિત નામ સાથે એક એક્ઝિક્યુટેબલ ફાઈલની રચના કરવામાં આવી છે. જો પ્રોગ્રામમાં ક્ષતિ હશે તો તે દર્શાવતો સંદેશ સ્ક્રીન પર પ્રદર્શિત કરવામાં આવશે. પ્રોગ્રામમાં ક્ષતિ હોય તો તેને દૂર કરી પ્રોગ્રામને પુનઃ કમ્પાઇલ કરવો જરૂરી છે. પ્રોગ્રામનું પરિણામ જોવા માટે નીચે જણાવેલ કમાન્ડ ટાઇપ કરી એન્ટર કી દબાવો :

$ ./a.out

આ કમાન્ડ દ્વારા myprogram.c પ્રોગ્રામનું પરિણામ આકૃતિ 10.12માં દર્શાવ્યા મુજબ રજૂ કરવામાં આવશે :



આકૃતિ 10.12 : ઉદાહરણ 10.6નું પરિણામ

તમામ સી પ્રોગ્રામનું કમ્પાઇલેશન a.out નામની પૂર્વનિર્ધારિત ફાઈલની રચના કરતું હોવા છતાં, પરિણામી ફાઈલનું નામ આપવું એ વધુ સારી ટેવ ગણાય છે. આમ કરવાથી, આપેલ નામ સાથેની પરિણામી ફાઈલ એકવાર બનાવ્યા પછી જ્યાં સુધી મૂળ સોર્સકોડમાં ફેરફાર ન થાય ત્યાં સુધી તેનો અનેક વાર પુનઃઉપયોગ પણ કરી શકાય છે. નીચે જણાવેલ કમાન્ડ પરિણામી ફાઈલને આપવામાં આવતું નામ દર્શાવે છે :

$ gcc -o myprogram.o myprogram.c

આ કમાન્ડમાં પ્રથમ આર્ગ્યુમેન્ટ myprogram.o પરિણામી ફાઈલનું નામ રજૂ કરે છે. જ્યારે બીજી આર્ગ્યુમેન્ટ સોર્સ ફાઈલનું નામ દર્શાવે છે. પ્રથમ આર્ગ્યુમેન્ટ સાથે .o અનુલંબનનો ઉપયોગ કરવામાં આવ્યો છે તે જુઓ. આ માત્ર નિર્દેશ છે કે આપેલ ફાઈલ પરિણામી (output) ફાઈલ છે. કમ્પાઇલરને આ અનુલંબનની જરૂર નથી, માટે myprogram.oના સ્થાને માત્ર myprogram પણ લખી શકાય છે. આપેલ પ્રોગ્રામનો સફળતાપૂર્વક અમલ થશે તો કમ્પાઇલર હવે a.outને બદલે myprogram.o નામની ફાઈલ બનાવશે. હવે આપણે નીચેના કમાન્ડનો અમલ કરી પ્રોગ્રામનું પરિણામ મેળવી શકીશું :

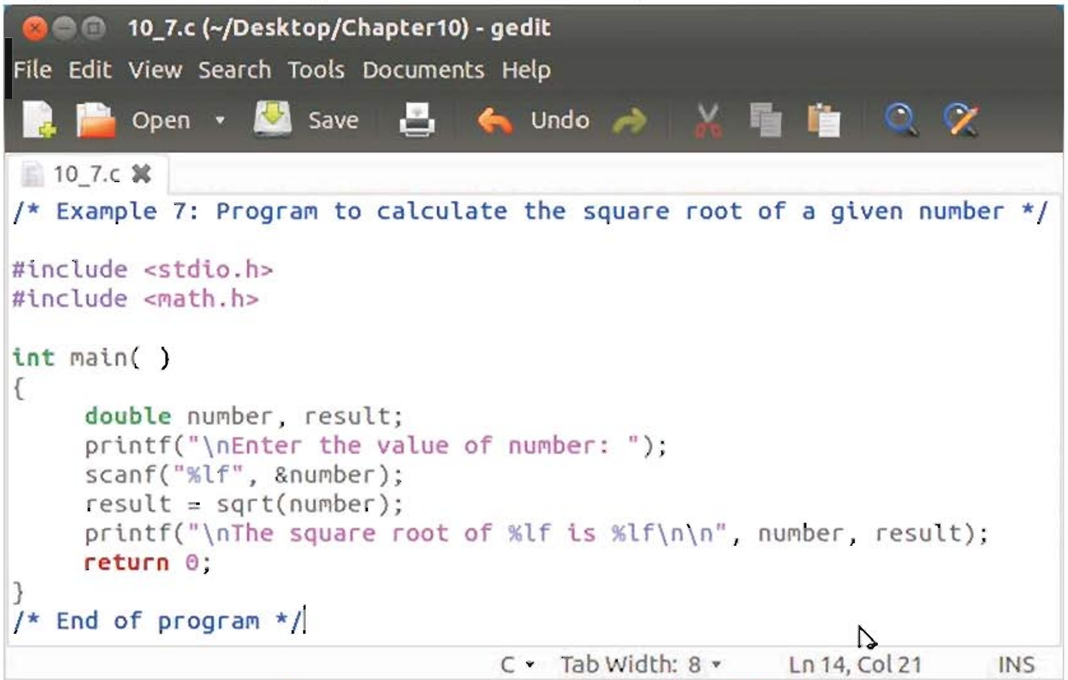
$ ./myprogram.o

gcc કમાન્ડ સાથે અન્ય ઘણા પ્રાચલોનો ઉપયોગ પણ કરી શકાય છે. gcc કમાન્ડ વિશેની વિસ્તૃત મદદ મેળવવા માટે નીચે જણાવેલ કમાન્ડનો ઉપયોગ કરી શકાય છે :

$ man gcc

હવે આપણે અન્ય એક પ્રોગ્રામ લખીને કમ્પાઈલ કરવાનો પ્રયત્ન કરીએ.

ઉદાહરણ 10.7માં દર્શાવેલ પ્રોગ્રામ આંતરપ્રસ્થાપિત વિધેયનો ઉપયોગ કરી ઉપયોગકર્તાએ આપેલ સંખ્યાનું વર્ગમૂળ ગણી આપે છે. આકૃતિ 10.3માં આ ઉદાહરણનું કોડ લિસ્ટિંગ આપવામાં આવ્યું છે.

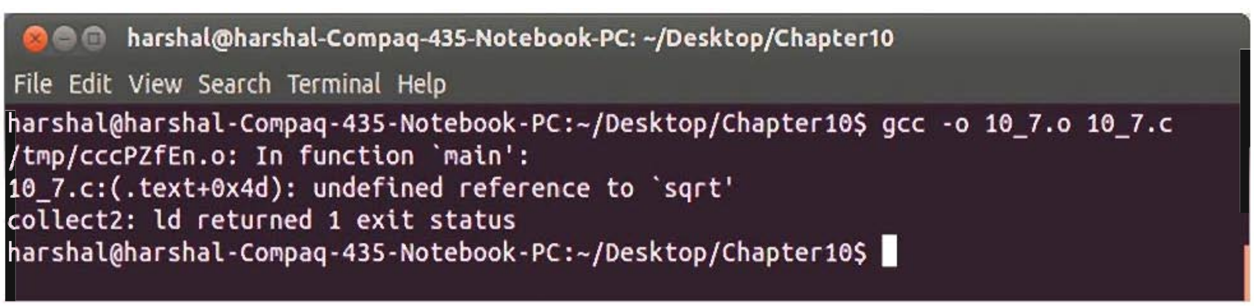


આકૃતિ 10.13 : ઉદાહરણ 10.7નું કોડ લિસ્ટિંગ

gccનો ઉપયોગ કરી આ પ્રોગ્રામને કંપાઈલ કરવાનો પ્રયત્ન કરીએ. ટર્મિનલ ખોલી કમાન્ડ પ્રોમ્પ્ટ પર નીચે આપેલો કમાન્ડ ટાઈપ કરો :

$ gcc -o 10_7.o 10_7.c

આ gcc કમાન્ડનું પરિણામ આકૃતિ 10.14માં દર્શાવ્યું છે.

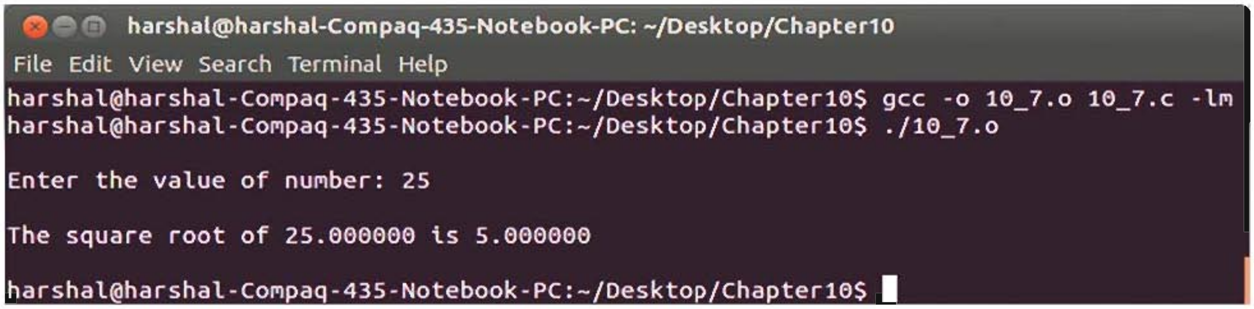


આકૃતિ 10.14 : ઉદાહરણ 10.7નું પરિણામ

અહીં એ બાબતની નોંધ કરો કે, પરિણામમાં પ્રોમ્પ્ટના સ્થાને "undefined reference to 'sqrt'" સંદેશ દર્શાવવામાં આવ્યો છે. પ્રોગ્રામમાં sqrt() નામના આંતરપ્રસ્થાપિત વિધેયનો ઉપયોગ કર્યો હોવાથી કમ્પાઈલેશન સમયે તેની સાથે ગાણિતિક લાઈબ્રેરી જોડવી જરૂરી છે. નીચે આપેલ કમાન્ડ દ્વારા ગાણિતિક લાઈબ્રેરી જોડી શકાશે.

$ gcc -o 10_7.o 10_7.c -lm

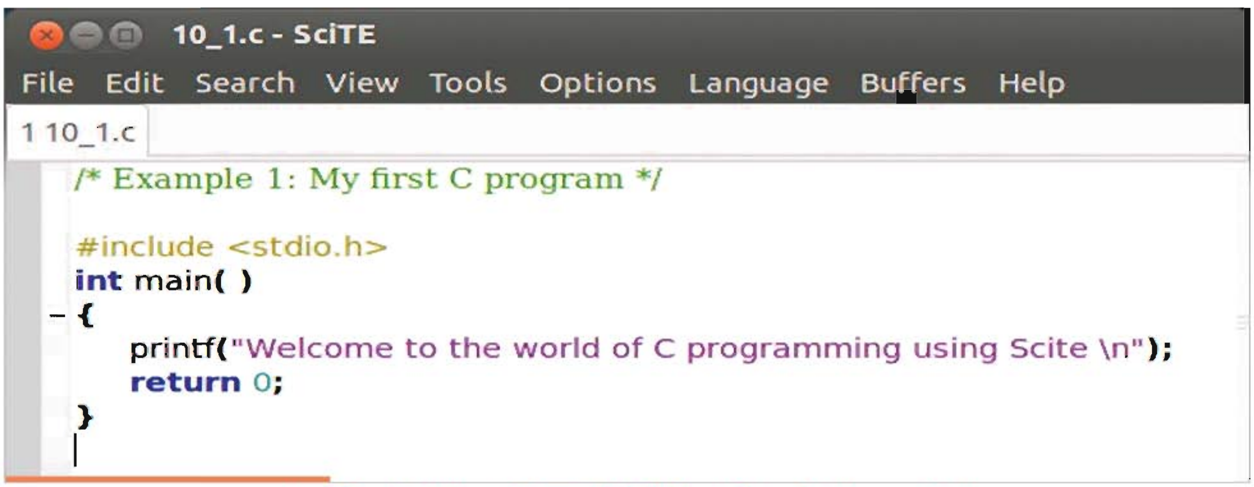
આ કમાન્ડ આપવાથી પ્રોગ્રામ કોઈ પણ ભૂલના સંદેશ વગર કમ્પાઈલ થઈ જશે. હવે પ્રોમ્પ્ટ પર ./10_7.o ટાઈપ કરી પરિણામ જુઓ. આકૃતિ 10.15માં સાચું પરિણામ દર્શાવ્યું છે.



આકૃતિ 10.15 : ઉદાહરણ 10.7નું સાચું પરિણામ

અહીં દર્શાવેલી પદ્ધતિ લિનક્સ ઑપરેટિંગ સિસ્ટમમાં સી પ્રોગ્રામ લખવા અને કમ્પાઇલ કરવા માટેની સૌથી પાયારૂપ પદ્ધતિ છે. અન્ય કોઈ પણ સાદા ટેક્સ્ટ એડિટરનો ઉપયોગ કરવાને બદલે આ પુસ્તકમાં આપણે પ્રોગ્રામ લખવા અને અમલ કરવા માટે SciTE નામના ટેક્સ્ટ એડિટરનો ઉપયોગ કરીશું. SciTE એક જ વિન્ડોમાં પ્રોગ્રામને કંપાઇલ અને અમલ કરવાની સુવિધા પૂરી પાડે છે. ઉપયોગકર્તા પાસેથી ઇનપુટ મેળવવાના હોય તે પ્રકારના પ્રોગ્રામ SciTEમાં લખી શકાય છે, પરંતુ તેનો અમલ કરવા માટે આપણે ટર્મિનલનો ઉપયોગ કરીશું.

તમારા કમ્પ્યૂટરમાં યોગ્ય સ્થાન પરથી SciTE ટેક્સ્ટ એડિટર ખોલો. આપણે અગાઉના ઉદાહરણમાં SciTE એડિટર વિન્ડોનો દેખાવ જોયો છે. આકૃતિ 10.1માં દર્શાવ્યા મુજબ SciTE એડિટરની ખાલી સ્ક્રીનમાં ઉદાહરણ 10.1માં આપેલ લખાણ ટાઇપ કરો. ફાઈલને 10_1.c. નામ આપી સંગ્રહ કરો. આ માટે Ctrl + S અથવા **File → Save**નો ઉપયોગ કરી શકાય. હવે વિન્ડો આકૃતિ 10.16માં દર્શાવ્યા મુજબના દેખાવ જેવી લાગશે.



આકૃતિ 10.16 : SciTE વિન્ડોમાં લખેલ ઉદાહરણ 10.1

એકવાર પ્રોગ્રામ લખાઈ જાય અને સંગ્રહ થઈ જાય પછી તેમાં જો કોઈ વાક્યરચનાની ભૂલ (syntax errors) આવેલી હોય તો તે શોધવાની જરૂર પડશે. આ ભૂલ શોધવા માટે પ્રોગ્રામને કંપાઇલ કરવો પડે. આ માટે Ctrl + F7 કી અથવા Tools → Compile વિકલ્પ પસંદ કરો. જો પ્રોગ્રામમાં કોઈ ક્ષતિ નહીં હોય તો આકૃતિ 10.17માં દર્શાવ્યા મુજબની સ્ક્રીન રજૂ કરવામાં આવશે.

આકૃતિ 10.17 : કમ્પાઈલેશનની સફળતાનો સંદેશ

હવે પ્રોગ્રામનો અમલ કરી શકાશે. આ માટે F5 કી દબાવો અથવા Tools → Go વિકલ્પ પસંદ કરો. આમ કરવાથી આકૃતિ 10.18માં દર્શાવ્યા મુજબ સોર્સકોડની વિન્ડો સાથે આઉટપુટ વિન્ડો દર્શાવવામાં આવશે.

આકૃતિ 10.18 : બે વિન્ડો સાથેનું SciTE એડિટર

અહીં એ જોઈ શકાય છે કે આપણે કરેલી બંને પ્રવૃત્તિઓનું પરિણામ દર્શાવવામાં આવે છે. જ્યારે પ્રોગ્રામને કમ્પાઇલ કરવામાં આવ્યો હતો ત્યારે નીચે આપેલ પરિણામ પ્રથમ દર્શાવવામાં આવ્યું હતું :

> gcc -pedantic -Os 10_1.c -o 10_1.o –std=c99

> Exit code : 0

અહીં પરિણામ સ્વરૂપે મળેલી 10_1.o ફાઈલ અમલ થઈ શકે તે પ્રકારની નથી. જ્યારે Goનો ઉપયોગ કરવામાં આવશે ત્યારે નીચે જણાવેલ પરિણામ મેળવી શકીશું :

>gcc –pedantic –Os –std=c99 10_1.c –o 10_1

>Exit code : 0

>./10_1

Welcome to the world of C programming using Scite

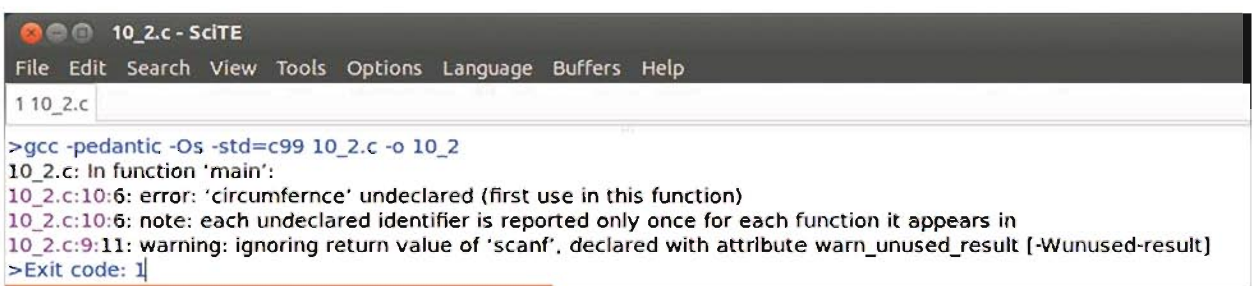
>Exit code : 0

અહીં ફાઈલ પર બે તબક્કામાં પ્રક્રિયા કરવામાં આવે છે. પ્રથમ તબક્કામાં ફાઈલને કંપાઇલ કરવામાં આવે છે અને 10_1 નામની એક્ઝિક્યુટેબલ ફાઈલ બનાવવામાં આવે છે. આ પ્રક્રિયા પ્રથમ બે લીટી દ્વારા રજૂ કરવામાં આવી છે. બીજા તબક્કામાં (ત્રીજી લીટીમાં દર્શાવ્યા મુજબ) ./10_1 કમાન્ડનો ઉપયોગ કરી આઉટપુટ ફાઈલનો અમલ કરવામાં આવ્યો છે. છેલ્લી બે લીટી પરિણામ દર્શાવે છે.

નોંધ : > નિશાની પછી આવેલું લખાણ એ કમ્પાઇલર દ્વારા કરવામાં આવેલી પ્રક્રિયા છે, જ્યારે ઉપયોગકર્તા માટે સ્ક્રીન પર દર્શાવવાના પરિણામની આગળ > નિશાની ઉમેરવામાં આવતી નથી.

F8 કી દબાવીને અથવા તો **View → Output** વિકલ્પનો ઉપયોગ કરીને આઉટપુટ વિન્ડો દર્શાવવી કે અદૃશ્ય બનાવવી પણ શક્ય છે. અગાઉનાં તમામ પરિણામોને દૂર કરવા Shift + F5 કી અથવા **Tools → Clear Output** વિકલ્પનો ઉપયોગ કરી શકાય છે.

આ ઉદાહરણમાં આપણે એવો પ્રોગ્રામ પસંદ કરેલો જે યોગ્ય રીતે કાર્ય કરી શકે, માટે કમ્પાઇલેશનની પ્રક્રિયા દરમિયાન કોઈ ભૂલનો સંદેશ દર્શાવવામાં આવ્યો નહીં. હવે આપણે એ જોઈએ કે જો ખોટો પ્રોગ્રામ ઉમેરવામાં આવે તો શું થાય ? ધારો કે, એડિટરમાં આપણે ઉદાહરણ 10.2ને થોડા ફેરફાર સાથે ઉમેર્યું છે. "circumference = 2 * PI * radius;" વિધાન લખવાને બદલે આપણે ટાઇપિંગ ભૂલ કરીને "circumfernce = 2 * PI * radius;" લખ્યું છે. હવે જો આપણે Ctrl + F7 અથવા F5 કી દબાવીએ તો આકૃતિ 10.19માં દર્શાવ્યા મુજબની સ્ક્રીન રજૂ કરવામાં આવશે.



આકૃતિ 10.19 : ભૂલ ધરાવતા પ્રોગ્રામનું કમ્પાઇલેશન

સંપૂર્ણ ચિત્ર યોગ્ય રીતે દર્શાવી શકાય તેમ ન હોવાથી અહીં માત્ર આઉટપુટ વિન્ડો જ દર્શાવવામાં આવી છે. માત્ર પરિણામ જોવા માટે Option → Vertical Split વિકલ્પ પસંદ કરી શકાય છે. આકૃતિ 10.19માં ક્ષતિઓ (errors) દર્શાવેલી જોઈ શકાય છે. આકૃતિમાં આપેલી "10_2.c:10:6: error: 'circumfernce' undeclared (first use in this function)" લીટી જુઓ. તે દર્શાવે છે કે circumfernce નામનો ચલ પ્રોગ્રામમાં ઘોષિત કરવામાં આવ્યો નથી. આ ભૂલ સુધારી ફરી પ્રોગ્રામને કમ્પાઇલ કરવાની પ્રક્રિયાનો અમલ કરો. હવે, F5 કી દબાવ્યા પછી કમ્પાઇલરને કોઈ ક્ષતિ નહીં મળે તો તે પ્રોગ્રામનો અમલ કરવાનો પ્રયત્ન કરશે. જો કે, આપણે પ્રોગ્રામમાં scanf() વિધેયનો ઉપયોગ કર્યો હોવાથી આ કિસ્સામાં કોઈ પરિણામ દર્શાવવામાં આવશે નહીં. ઉબુન્ટુની હાલમાં ઉપયોગમાં લેવામાં આવેલી SciTEની આવૃત્તિમાં ક્ષતિ (bug) હોવાનું જણાય છે, કારણ કે તે ઉપયોગકર્તા પાસેથી ઇનપુટ માટેની રાહ જોતું નથી. કોઈ અનુલંબન વગર ફાઈલના નામનો ઉપયોગ કરી આપણે ટર્મિનલ દ્વારા પ્રોગ્રામનો અમલ કરી શકીએ છીએ.

સી ભાષાનો ઇતિહાસ (History of C)

સી પ્રોગ્રામ વિશેનો અભ્યાસ કર્યા બાદ, હવે તેના ઇતિહાસ વિશે સંક્ષિપ્ત માહિતી મેળવીએ. સી ભાષાના મૂળ ઈ. સ. 1972માં બેલ લૅબોરેટરી (Bell laboratory)માં નંખાયાં હતાં. સી ભાષાની રચનાનું શ્રેય ડેનિસ એમ. રિચી (Dennis M. Ritchie)ને આપી શકાય. *Basic Combined Programming Language* (BCPL) નામે ઓળખાતી ભાષા પરથી સી ભાષાની રચના કરવામાં આવી હતી. સી ભાષાની રચનાનો હેતુ એક પુષ્ટ (robust) સિસ્ટમ સૉફ્ટવેર બનાવવાનો હતો. પરંતુ પછીના વર્ષોમાં તે પ્રોગ્રામરોની પ્રિય ભાષા બની રહી અને તમામ પ્રકારના સૉફ્ટવેર બનાવવા માટે તેનો ઉપયોગ થવા લાગ્યો. માટે જ તેને General Purpose Programming Language તરીકે ઓળખવામાં આવી.

ઈ. સ. 1972માં રચિત આ ભાષાને 1989માં American National Standard Institute (ANSI) ધારાધોરણ દ્વારા પ્રમાણભૂત કરવામાં આવી. ત્યાર પછી તેને આન્સી સી (ANSI C) તરીકે ઓળખવામાં આવી. આજે આ ધારાધોરણને જુદી જુદી ઑપરેટિંગ સિસ્ટમ અને કમ્પાઇલર દ્વારા સમર્થન આપવામાં આવે છે.

સી એક સંરચિત (structured) ભાષા છે. તેમાં પ્રોગ્રામને વિધેય નામે ઓળખાતા નાના વિભાગોમાં વહેંચવાની સુવિધા છે. એકવાર આ વિધેયની રચના કર્યા બાદ તેનો પુનઃ ઉપયોગ શક્ય છે. આવા વિધેયનો સમૂહ સી પ્રોગ્રામની રચના કરે છે. જ્યારે કોઈ ક્ષતિનું નિવારણ કરવાનું હોય ત્યારે આ અભિગમ દ્વારા સમગ્ર પ્રોગ્રામને બદલે માત્ર એક વિધેય ઉપર ધ્યાન કેન્દ્રિત કરી શકાય છે. સામાન્ય રીતે સી ભાષામાં લખવામાં આવેલા કોઈ પણ પ્રોગ્રામને નહિવત્ ફેરફારો સાથે અન્ય ઑપરેટિંગ સિસ્ટમ કે કમ્પાઇલર ધરાવતાં જુદાં જુદાં મશીન પર સરળતાથી ચલાવી શકાય છે. આ ગુણ સી ભાષાને પૉર્ટેબલ (portable) બનાવે છે. કેટલાક સી ભાષાને મીડલ લેવલ લૅંગ્વેજ (middle level language) માને છે તો કેટલાક માટે તે હાયર લેવલ લૅંગ્વેજ (higher level language) છે. એક પ્રોગ્રામરને જરૂરી એવી તમામ સુવિધાઓ કોઈ પણ પ્રકારે સી ભાષાના ઉપયોગથી ઉપલબ્ધ છે.

સારાંશ

આ પ્રકરણમાં આપણે સી પ્રોગ્રામમાં બંધારણ વિશે અભ્યાસ કર્યો. સી પ્રોગ્રામના વિભાગ સ્વરૂપે ઉપયોગમાં લઈ શકાતા વિવિધ ઘટકો વિશે જાણકારી મેળવી. ત્યાર પછી આપણે સી ભાષાના મૂળાક્ષરો વિશે પણ અભ્યાસ કર્યો, જે અક્ષરો, અંકો, વ્હાઇટ સ્પેસ અને વિશિષ્ટ ચિહ્નો એમ ચાર વિભાગમાં વહેંચાયેલા છે. આપણે આ મૂળાક્ષરોનો ઉપયોગ કરી સી ભાષામાં શબ્દોની રચના કરતા શીખ્યા. ત્યારબાદ આપણે સી પ્રોગ્રામની રચના અને અમલીકરણનાં પગલાં જોયાં. પ્રોગ્રામને કંપાઇલ તથા અમલ કરવા માટે gcc કમ્પાઇલરનો ઉપયોગ કર્યો. અંતમાં આપણે સી પ્રોગ્રામ લખવા અને તેનો અમલ કરવા SciTE ટેક્સ્ટ એડિટરનો ઉપયોગ કરતા શીખ્યા.

શિક્ષકોને સૂચના

SciTE ટેક્સ્ટ એડિટરની ચર્ચા અહીં એમ માનીને કરવામાં આવી છે કે આ એડિટર કમ્પ્યૂટરમાં પ્રસ્થાપિત છે અને તેનો શૉર્ટકટ ઉપલબ્ધ છે. SciTE એડિટરમાં સી પ્રોગ્રામ લખવાનું શરૂ કરતાં પહેલાં એ નિશ્ચિત કરી લેવું જરૂરી છે કે તેમાં ઓછામાં ઓછું એકવાર Language → C/C++ વિકલ્પની પસંદગી કરેલી હોય. આમ કરતી વખતે સી પ્રોગ્રામ લખતી વખતે કી-વર્ડને પ્રકાશિત (highlight) કરી દર્શાવવામાં આવે છે.

આ ઉપરાંત F5 કે Tools → Go વિકલ્પના ઉપયોગથી SciTE માં જો સીધો જ પ્રોગ્રામનો અમલ કરાવવા ઇચ્છતા હોઈએ તો પ્રોપર્ટિઝ ફાઈલમાં નીચે જણાવેલ લીટી ઉમેરવી જરૂરી બનશે :

command.go.needs.*.c=gcc $(ccopts) –std=c99 $(FileNameExt) –o $(FileName)

નીચે દર્શાવેલ પગલાં દ્વારા પ્રોપર્ટી ફાઈલ બદલી શકાશે :

1. તમારા કમ્પ્યૂટરમાં cpp.properties ફાઈલનું સ્થાન શોધી કાઢો. (અમારા કમ્પ્યૂટરમાં તે /usr/share scite/ cpp.properties છે.)

2. ટર્મિનલ ખોલી તેમાં sudo edit your_file_path/cpp.properties ટાઈપ કરી એન્ટર કી દબાવો.

3. અહીં એડમિનિસ્ટ્રેટરનો પાસવર્ડ પૂછવામાં આવશે.

4. gedit વિન્ડોમાં cpp.properties ફાઈલ ખુલશે. કોષ્ટક 10.7માં દર્શાવ્યા મુજબનો કોડ ફાઈલમાં શોધી કાઢો.

```
ccopts=-pedantic -Os
cc=g++ $(ccopts) -c $(FileNameExt) -o $(FileName).o
ccc=gcc $(ccopts) -c $(FileNameExt) -o $(FileName).o

make.command=make
command.compile.*.c=$(ccc) -std=c99
command.build.*.c=$(make.command)
command.build.*.h=$(make.command)
#command.go.*.c=./a.out
command.go.*.c=./$(FileName)
```

કોષ્ટક 10.7 : cpp.propertiesમાં શોધવાનો કોડ

તમારા કમ્પ્યૂટરમાં છેલ્લી બે લીટી આ જ પ્રમાણેની છે તેની ખાતરી કરો. જો તે જુદી હોય તો તેને કોષ્ટક 10.7માં દર્શાવ્યા મુજબ લખો. આ એક એવી વ્યવસ્થા છે કે જેમાં SciTEનો ઉપયોગ કરતી વખતે એક્ઝિક્યુટેબલ (આઉટપુટ) ફાઈલનું નામ સોર્સ ફાઈલના નામનો ઉપયોગ કરીને આપવામાં આવશે.

5. આ ફાઈલમાં નીચેની લીટી ઉમેરી ફાઈલનો સંગ્રહ કરો :

```
# To make the Go command both compile (if needed) and execute, use this setting :
command.go.needs.*.c=gcc $(ccopts) -std=c99 $(FileNameExt) -o $(FileName)
```

6. gedit અને ટર્મિનલ વિન્ડો બંધ કરો. SciTE એડિટર હવે તૈયાર છે.

આ પ્રકરણમાં આપવામાં આવેલી સ્ક્રીન એ નમૂનારૂપ સ્ક્રીન છે. શાળામાં ઉપલબ્ધ ઉબુન્ટુની આવૃત્તિ મુજબ તે અલગ હોઈ શકે છે. પરંતુ સ્ક્રીનનું કાર્ય એકસમાન જ રહે છે.

સ્વાધ્યાય

1. સી પ્રોગ્રામની લાક્ષણિક્તાની યાદી બનાવી સમજૂતી આપો.
2. main( ) વિધેયનું મહત્ત્વ સમજાવો.
3. સી પ્રોગ્રામમાં 'ફાઈલ ઈન્ક્લુડ' વિભાગનો હેતુ શું છે ?
4. આઈડેન્ટિફાયર એટલે શું ? સી પ્રોગ્રામમાં તે કેવી રીતે ઉપયોગી છે ?
5. ચલ એટલે શું ? ચલ વ્યાખ્યાયિત કરવાના નિયમો જણાવો.
6. એક અક્ષર અને સ્ટ્રિંગ અચળ વચ્ચેનો તફાવત જણાવો.

7. નીચેનાં વિધાનો સાચાં છે કે ખોટાં તે જણાવો :

(a) સી પ્રોગ્રામને ".h" અનુલંબન આપવામાં આવે છે.

(b) સામાન્ય રીતે સી વિધાનો અલ્પવિરામથી પૂરાં કરવામાં આવે છે.

(c) Amount એ ચલનું યોગ્ય નામ છે.

(d) #define PI 3.24 દ્વારા સી પ્રોગ્રામમાં PI નામની ફાઈલ ઉમેરવામાં આવશે.

(e) "X" એ એક અક્ષરનું યોગ્ય ઉદાહરણ છે.

8. આપેલ વિકલ્પોમાંથી સાચો વિકલ્પ પસંદ કરો :

(1) સી પ્રોગ્રામ ફાઈલને નીચેનામાંથી કયું અનુલંબન આપવામાં આવે છે ?

(a) c (b) h (c) s (d) t

(2) સી મૂળાક્ષરોને કેટલા વિભાગમાં વર્ગીકૃત કરી શકાય ?

(a) 0 (b) 2 (c) 4 (d) 8

(3) ‘‘=’’ નિશાની સી મૂળાક્ષરોના કયા વર્ગમાં સમાવિષ્ટ છે ?

(a) અક્ષરો (b) ખાલી જગ્યા

(c) વિશિષ્ટ ચિહ્નો (d) અંક

(4) સી ભાષામાં નીચેનામાંથી કયો શબ્દ યોગ્ય કી-વર્ડ છે ?

(a) ofsize (b) sizeof (c) forsize (d) sizefor

(5) સી ભાષા માટે નીચેનામાંથી કયો ચલ અયોગ્ય છે ?

(a) Register (b) RegIster (c) registre (d) register

(6) સી ભાષા માટે નીચેનામાંથી કયો પૂર્ણ અચળ અયોગ્ય છે ?

(a) OxG (b) OxA (c) OxB (d) OxD

(7) સી ભાષા માટે નીચેનામાંથી કયો અપૂર્ણ અચળ યોગ્ય છે ?

(a) –2.0.5e5 (b) –20.5e5.5 (c) –20.5e5 (d) –20.5e.5

(8) સી ભાષા માટે નીચેનામાંથી કયો અચળ એક અક્ષર રજૂ કરે છે ?

(a) 'a' (b) '\a' (c) "a" (d) a અને b બંને

(9) સી ભાષામાં નીચેનામાંથી શું વ્યાખ્યાયિત કરવા માટે *#define* પ્રી-પ્રોસેસર ડાઈરેક્ટિવનો ઉપયોગ કરી શકાય ?

(a) સ્ટ્રિંગ અચળ (b) સાંકેતિક અચળ

(c) પૂર્ણ અચળ (d) એક અક્ષર

(10) નીચેનામાંથી કઈ કી દ્વારા પ્રોગ્રામનો સીધો જ અમલ કરી શકાય છે ?

(a) F7 (b) F9 (c) F5 (d) F8

પ્રાયોગિક સ્વાધ્યાય

1. તમારું નામ, શાળાનું નામ, ધોરણ અને શાળાનું સરનામું સ્ક્રીનની મધ્યમાં દર્શાવવા માટેનો સી પ્રોગ્રામ બનાવો.

```
***********************
* Name :               *
* School Name :        *
* Standard :           *
* Address :            *
*                      *
***********************
```

2. સ્ક્રીન પર તમારી અટકનો પ્રથમ અક્ષર દર્શાવવા માટેનો સી પ્રોગ્રામ બનાવો. ઉદાહરણ તરીકે જો તમારી અટકનો પ્રથમ અક્ષર P હોય તો નીચે મુજબ પરિણામ દર્શાવવું જોઈએ.

```
******
*     *
*     *
******
*
*
*
```

3. તમારી પસંદગીનો શુભેચ્છા સંદેશ સ્ક્રીન પર દર્શાવવા માટેનો સી પ્રોગ્રામ લખો. ઉદાહરણ તરીકે, જો તમે કોઈને દિવાળી પર શુભેચ્છા પાઠવવા ઈચ્છા હો તો "Wishing you a Happy and Prosperous Diwali" સંદેશ દર્શાવો.

સી ભાષામાં ડેટા પ્રકાર, પ્રક્રિયકો અને પદાવલિઓ

આ પહેલાના પ્રકરણમાં આપણે કેટલાક સરળ સી પ્રોગ્રામ જોયા અને સી ભાષાના મૂળાક્ષરો તથા ટોકન વિશે પણ અભ્યાસ કર્યો. તેમાનાં એક ટોકન 'આઇડેન્ટિફાયર'(Identifier)ની પણ આપણે ચર્ચા કરેલી. તમામ સી પ્રોગ્રામમાં આઇડેન્ટિફાયરનો ઉપયોગ કરવામાં આવે છે. ધારો કે, પ્રોગ્રામમાં આપણે date નામનો આઇડેન્ટિફાયર વ્યાખ્યાયિત કરવો છે, જે તારીખની કિંમતનો સંગ્રહ કરવા સક્ષમ હોય. આ આઇડેન્ટિફાયરમાં કઈ કિંમતનો સંગ્રહ થવો જોઈએ ? આપણે તેમાં 12.50 કિંમત ઉમેરી શકીએ ? તો જવાબ છે, 'ના'. આ માટે યોગ્ય કિંમત 12 કે 13 જેવી 1થી 31 વચ્ચેની કોઈ પણ પૂર્ણાંક સંખ્યા હશે. આ ઉદાહરણ દર્શાવે છે કે, માત્ર આઇડેન્ટિફાયર નહીં, તેમાં સંગ્રહવામાં આવેલી કિંમત પણ મહત્ત્વની છે. સી ભાષા કેટલાક કી-વર્ડનો ઉપયોગ કરી આઇડેન્ટિફાયરમાં સંગૃહીત કિંમતનો પ્રકાર નિશ્ચિત કરે છે. આ કી-વર્ડ નિશ્ચિત પ્રકારના ડેટા માટે રચવામાં આવેલા હોવાથી તેને ડેટા પ્રકાર (Data type) કહે છે. સી ભાષામાં ઉપલબ્ધ જુદાં જુદાં ડેટા પ્રકારો વિશે આપણે આ પ્રકરણમાં ચર્ચા કરીશું.

ડેટા પ્રકાર શું છે ? (What is Data Type ?)

આઇડેન્ટિફાયરમાં સંગ્રહ કરી શકાય તે કિંમતના પ્રકારને ડેટા પ્રકાર (Data type) તરીકે ઓળખવામાં આવે છે. જો આઇડેન્ટિફાયર dateમાં 12 સંખ્યાનો સંગ્રહ કરવામાં આવે તો તેનો ડેટા પ્રકાર પૂર્ણાંક છે તેમ કહેવાય. આ જ રીતે આઇડેન્ટિફાયર amountને 99.50 કિંમત આપેલ હોય તો તેનો ડેટા પ્રકાર અપૂર્ણાંક છે તેમ કહેવાય.

સી ભાષા વિગતને તેની કિંમત સાથે જોડવા માટે કી-વર્ડનો ઉપયોગ કરે છે. આ કી-વર્ડ બે વસ્તુઓ વ્યાખ્યાયિત કરે છે : આઇડેન્ટિફાયરમાં સંગૃહીત કિંમતનો પ્રકાર અને આઇડેન્ટિફાયર દ્વારા જરૂરી એવી મેમરી જગ્યા. સી ભાષામાં દરેક ડેટા પ્રકારને નિશ્ચિત મેમરી સ્થાન આપવામાં આવે છે. તેનો નિર્દેશ બાઇટ(byte)માં કરવામાં આવે છે. 8 બીટ(Bit)ના સમૂહને 1 બાઇટ કહે છે.

સી ભાષાના મૂળભૂત ડેટા પ્રકાર (Basic Data types of C)

સી ભાષા પૂર્ણાંક (integer), અપૂર્ણાંક (decimal) અને અક્ષર(character)ના નામે ઓળખાતા ડેટા પ્રકારને સમર્થન આપે છે. આ ડેટા પ્રકારને અનુક્રમે int, float અને char કી-વર્ડ દ્વારા રજૂ કરવામાં આવે છે. સી ભાષામાં આ ઉપરાંત void નામનો એક અન્ય પ્રાથમિક ડેટા પ્રકાર પણ આપવામાં આવ્યો છે. આઇડેન્ટિફાયરને તેના ડેટા પ્રકાર સાથે સાંકળવા માટે નીચે આપેલ વાક્યરચનાનો ઉપયોગ કરવામાં આવે છે.

Data type identifier 1 [, identifier 2, identifier 3, ……, identifier n];

અહીં, ચોરસ કૌંસમાં આપવામાં આવેલ લખાણ વૈકલ્પિક છે. હવે આપણે તમામ મૂળભૂત ડેટા પ્રકારો વિશે માહિતી મેળવીએ.

પૂર્ણાંક (Integer)

આ પહેલાના મુદ્દામાં આપણે જોયું કે "date" આઇડેન્ટિફાયરને 12 કિંમત આપી શકાય છે. અહીં 12 એ પૂર્ણાંક સંખ્યા છે. ધન કે ઋણ સંપૂર્ણ સંખ્યા કે જેમાં અપૂર્ણાંક ભાગ આપવામા આવ્યો નથી તેને પૂર્ણાંક કહે છે. –99, 12, –10, 900, 30000 વગેરે પૂર્ણાંક સંખ્યાનાં ઉદાહરણો છે. પૂર્ણાંક પ્રકારના ચલની ઘોષણા માટે નીચે જણાવેલ વાક્યરચનાનો ઉપયોગ કરી શકાય છે :

int identifier 1, [identifier 2, identifier 3, ……, identifier n];

આ વિધાનને સી ભાષામાં ઘોષણા વિધાન (declaration statement) કહે છે. ઘોષણા વિધાનનાં કેટલાંક ઉદાહરણ નીચે મુજબ છે :

int roll_number;

int date, month, year;

પ્રથમ વિધાન "roll_number" નામ સાથે એક આઇડેન્ટિફાયરની ઘોષણા કરે છે, જે પૂર્ણાંક સંખ્યાનો સંગ્રહ કરવા સક્ષમ છે. બીજું વિધાન પૂર્ણાંક સંખ્યાનો સંગ્રહ કરી શકે તેવા ત્રણ આઇડેન્ટિફાયર "date", "month" અને "year"ની ઘોષણા કરે છે.

ANSI સીમાં int ડેટા પ્રકાર 4 બાઇટ જેટલા મેમરી સ્થાનનો ઉપયોગ કરે છે. તેનો વિસ્તાર –2147483648થી +21474836411 છે. અત્યાર સુધીમાં વ્યાખ્યાયિત કરેલા તમામ ચલ ચિહ્નિત (નિશાનીવાળા –signed) છે એટલે કે તેમાં ધન અને ઋણ બંને પ્રકારની કિંમતોનો સંગ્રહ કરી શકાય છે. ક્યારેક આપણને ઋણ સંખ્યાઓની બિલકુલ જરૂર ન હોય એમ પણ બને. અહીં આપણે ઘોષિત કરેલા આ એક પણ ચલમાં ઋણ સંખ્યાનો સંગ્રહ કરી શકાય તેમ નથી. સી ભાષામાં ચલ સાથે માત્ર ધન સંખ્યાઓને જ સાંકળી શકાય તે પણ શક્ય છે. આ માટે unsigned કી-વર્ડનો ઉપયોગ કરવામાં આવે છે. ઉપરના ઉદાહરણને હવે નીચે દર્શાવ્યા મુજબ સુધારી શકાય :

```
unsigned int roll_number;

unsigned int date, month, year;
```

આ વિધાનો દર્શાવે છે કે 'roll_number', 'date', 'month' અને 'year' પૂર્ણાંક પ્રકારના આઇડેન્ટિફાયર છે, જે માત્ર ધન સંખ્યાઓનો સંગ્રહ કરવાની ક્ષમતા ધરાવે છે. પૂર્ણાંક ડેટા પ્રકારની સંક્ષિપ્ત માહિતી કોષ્ટક 11.1માં આપવામાં આવી છે.

Data Type	Range	Bytes required	Example
int	–2147483648 to +2147483647	4	int balance_amount;
unsigned int	0 to 4294967295	4	unsigned int counter;

કોષ્ટક 11.1 : પૂર્ણાંક ડેટા પ્રકાર

કોષ્ટક 11.1માં આપેલ કિંમતનો વિસ્તાર ઉપયોગમાં લેવામાં આવેલા બાઇટની સંખ્યા પર અવલંબે છે. ચિહ્નિત (signed) પૂર્ણાંક માટે કિંમતોનો વિસ્તાર $-2^{(n-1)}$ to $+ 2^{(n-1)} - 1$ દ્વારા ગણી શકાય છે. અહીં "n" એ જરૂરી બીટની સંખ્યાનો નિર્દેશ કરે છે. આ જ રીતે અચિહ્નિત (unsigned) પૂર્ણાંક માટે વિસ્તારની ગણતરી 0થી 2^{n-1} દ્વારા કરી શકાય છે. પૂર્ણાંક ડેટા પ્રકારની આગળ long કી-વર્ડ ઉમેરવાથી પૂર્ણાંક સંખ્યાઓનો વિસ્તાર વધારી શકાય છે. intની આગળ long ઉમેરવાથી ચલ માટે 8 બાઇટ જેટલું મેમરી સ્થાન આરક્ષિત કરવામાં આવે છે. આ પ્રકારના ચલને નીચે જણાવેલ વાક્યરચના દ્વારા વ્યાખ્યાયિત કરી શકાય :

```
long int population;
```

પૂર્ણાંક ડેટા પ્રકારના જુદા જુદા પ્રકારનાં સ્વરૂપો જોયા પછી હવે એક નમૂનારૂપ પ્રોગ્રામનો અભ્યાસ કરીએ. પૂર્ણાંક ડેટા પ્રકારના ઉપયોગ દર્શાવતા પ્રોગ્રામનું કોડ લિસ્ટિંગ આકૃતિ 11.1માં આપેલ છે.

```
11_1.c - SciTE
File Edit Search View Tools Options Language Buffers Help
1 11_1.c

/* Example 1: Program to illustrate use of integer data type */

#include <stdio.h>
int main( )
{
    int mark = -10;
    unsigned int date = 30;
    long int population = 42949672950;
    printf(" Mark = %d", mark);
    printf("\n Date = %u", date);
    printf("\n Population = %ld\n\n", population);
    return 0;
}
/* End of Program */
```

આકૃતિ 11.1 : પૂર્ણાંક ડેટા પ્રકારનો ઉપયોગ દર્શાવતો પ્રોગ્રામ

સમજૂતી (Explanation)

આકૃતિ 11.1માં આવેલા દરેક વિધાનની સમજૂતી મેળવીએ. ઉઘડતા છગરિયા કૌંસ પછીનું પ્રથમ વિધાન "mark" નામના પૂર્ણાંક ચલને ઘોષિત કરે છે. આ જ વિધાનમાં ચલની કિંમત –10 પણ આપી છે. બીજું વિધાન "date" નામના પૂર્ણાંક ચલની ઘોષણા કરી તેમાં કિંમત 30 આપે છે. ત્રીજું વિધાન "population" નામના ચલને ઉચ્ચ વિસ્તાર સાથે ઘોષિત કરી તેમાં 42949672950 કિંમતનો સંગ્રહ કરે છે. પછીના ત્રણ વિધાનો દ્વારા આ ત્રણ ચલની કિંમતોને સ્ક્રીન પર દર્શાવવામાં આવે છે. ઉદાહરણ 11.1નું પરિણામ આકૃતિ 11.2માં દર્શાવ્યું છે.

```
11_1.c - SciTE
File Edit Search View Tools Options Language Buffers Help
1 11_1.c

>gcc -pedantic -Os -std=c99 11_1.c -o 11_1
>Exit code: 0
>./11_1
 Mark = -10
 Date = 30
 Population = 42949672950

>Exit code: 0
```

આકૃતિ 11.2 : ઉદાહરણ 11.1નું પરિણામ

અપૂર્ણાંક (Real)

આપણને કેટલીકવાર પૂર્ણાંકને બદલે અપૂર્ણાંક સંખ્યાઓનો ઉપયોગ કરવાની જરૂર પડે છે. ઉદાહરણ તરીકે, કોઈ વ્યક્તિએ ચુકવવાની રકમ 95 કે 95.50 હોઈ શકે. આ પરિસ્થિતિમાં int ડેટા પ્રકાર કાર્ય કરી શકતો નથી. સી ભાષામાં અપૂર્ણાંક સંખ્યાઓનો ઉપયોગ કરવા માટે float કી-વર્ડ દ્વારા વ્યાખ્યાયિત કરવામાં આવતો ડેટા પ્રકાર આપવામાં આવ્યો છે. તે 4 બાઇટ જેટલા સંગ્રાહક સ્થાનનો ઉપયોગ કરે છે. અપૂર્ણાંક સંખ્યાઓ દશાંશચિહ્ન પછી 6 અને દશાંશચિહ્ન પહેલાં 7 અંકો જેટલી ચોકસાઈ ધરાવે છે. જો એથી વધુ ચોકસાઈની જરૂર હોય તો floatના સ્થાને double કી-વર્ડનો ઉપયોગ કરવામાં આવે છે. તે float ડેટા પ્રકારનું વિસ્તરણ છે. double કિંમતો 8 બાઇટનો ઉપયોગ કરે છે અને દશાંશ પછી 16 તથા દશાંશ પહેલાં 17 અંકોની ચોકસાઈ ધરાવે છે. અપૂર્ણાંક ચલનાં ઉદાહરણ નીચે મુજબ છે :

```
float amount_to_pay;
double balance_amount;
```

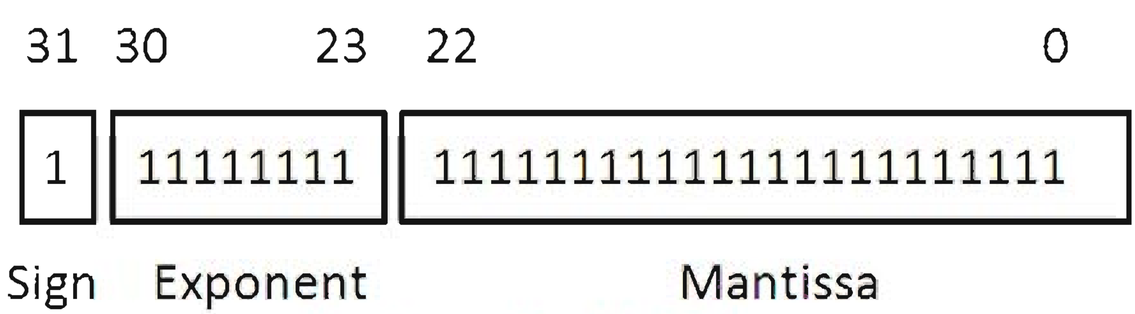
પ્રથમ વિધાન અપૂર્ણાંક સંખ્યાનો સંગ્રહ કરવા માટે સક્ષમ એવા "amount_to_pay" નામના ચલને ઘોષિત કરે છે. બીજું વિધાન "balance_amount" નામના ચલને ઘોષિત કરે છે જે વધુ વિસ્તાર અને ચોકસાઈ સાથે અપૂર્ણાંક સંખ્યાનો સંગ્રહ કરવા સક્ષમ છે.

double કી-વર્ડની આગળ long શબ્દ પણ ઉમેરી શકાય છે. આમ કરવાથી doubleનો વિસ્તાર વધારી શકાય છે. float ડેટા પ્રકારની સંક્ષિપ્ત સમજૂતી કોષ્ટક 11.2માં આપવામાં આવી છે.

Data Type	Range	Significant Digits	Bytes required	Example
float	+/–3.4e–38 to +/–3.4e+38	6	4	float price;
double	+/–1.7e-308 to +/–1.7e+308	16	8	double tot_price;
long double	+/–3.4e–4932 to +/–1.1e+4932	16	16	long double credit;

કોષ્ટક 11.2 : float ડેટા પ્રકાર

કોષ્ટક 11.2માં વિસ્તારને વૈજ્ઞાનિક સ્વરૂપમાં દર્શાવ્યો છે. સી ભાષા વૈજ્ઞાનિક સ્વરૂપે રહેલી સંખ્યાને પણ સમર્થન આપે છે. ઉદાહરણ તરીકે 95.50 સંખ્યાને 0.9550e2 સ્વરૂપમાં પણ રજૂ કરી શકાય છે. અહીં 0.9550ને મેન્ટિસા (mantissa) તરીકે તથા 2ને એક્સ્પોનન્ટ (exponent) તરીકે ઓળખવામાં આવે છે. અપૂર્ણ સંખ્યા ધન કે ઋણ હોઈ શકે છે. આકૃતિ 11.4માં અપૂર્ણાંક સંખ્યાની મેમરી રજૂઆત દર્શાવી છે.



આકૃતિ 11.3 : અપૂર્ણાંક સંખ્યાની મેમરી રજૂઆત

હવે, float, double અને long double ડેટા પ્રકારનો ઉપયોગ દર્શાવતું એક ઉદાહરણ જોઈએ. આકૃતિ 11.4માં float ડેટા પ્રકારને સ્પષ્ટ કરતા પ્રોગ્રામનું કોડ લિસ્ટિંગ દર્શાવ્યું છે.

```
11_2.c - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 11_2.c

/* Example 2: Program to illustrate use of real data type */

#include <stdio.h>
int main( )
{
    /* First three statements declares and initializes of variables first, second and third */

    float first = 98765432109876543 21.987654321;
    double second = 98765432109876543 21.987654321;
    long double third = 98765432109876543 21.987654321;

    printf(" First = %f", first);
    printf("\n Second = %lf", second);
    printf("\n Third = %Lf\n\n", third);
    return 0;
}
/* End of Program */
```

આકૃતિ 11.4 : float ડેટા પ્રકારને સ્પષ્ટ કરતો પ્રોગ્રામ

સમજૂતી (Explanation)

ઉઘડતા છગડિયા કૌંસ પછીનાં ત્રણ વિધાનો "first", "second" અને "third" નામના ત્રણ ચલને ઘોષિત કરે છે. આ દરેક ચલમાં અપૂર્ણ સંખ્યા 98765432109876543 21.987654321નો સંગ્રહ કરવામાં આવ્યો છે. પછીનાં ત્રણ વિધાનો આ ચલની કિંમતોને સ્ક્રીન પર દર્શાવે છે. આકૃતિ 11.5માં ઉદાહરણ 11.2નું પરિણામ દર્શાવ્યું છે.

```
11_2.c - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 11_2.c

>gcc -pedantic -Os -std=c99 11_2.c -o 11_2
>Exit code: 0
>./11_2
 First = 98765435164048 75264.000000
 Second = 98765432109876 55168.000000
 Third = 98765432109876 55168.000000

>Exit code: 0
```

આકૃતિ 11.5 : ઉદાહરણ 11.2નું પરિણામ

અહીં જોઈ શકાય છે કે, ચલ 'first', 'second' અને 'third'ની કિંમતો અનુક્રમે 98765435164048725264.000000, 987654321098765168.000000 અને 987654321098765168.000000 દર્શાવવામાં આવી છે. પરિણામને ધ્યાનથી જોતાં જાણી શકાય છે કે "first" ચલની મૂળ કિંમત સાથે પ્રથમ માત્ર 7 અંકોની સમાનતા છે. એ જ પ્રમાણે "second" અને "third" ચલના પ્રથમ માત્ર 15 અંકો જ સમાન છે. આમ થવાનું કારણ એ છે કે float ડેટા પ્રકાર દશાંશ પછી 6 અને દશાંશ પહેલાં 7 અંકોની ચોકસાઇ ધરાવે છે. આ જ રીતે double અને long float ડેટા પ્રકાર દશાંશ પછી 15 અને દશાંશ પહેલાં 16 અંકોની ચોકસાઈ ધરાવે છે.

અપૂર્ણાંક સંખ્યાઓનો ઉપયોગ કેટલીકવાર ચોકસાઈને હાનિ પહોંચાડે છે. તેથી આપણે અપેક્ષિત નિશ્ચિત પરિણામ મેળવી શકતા નથી. માટે જ પરિણામની વધુ ચોકસાઈ માટે ઉચ્ચ પરિશુદ્ધતા (precision) ધરાવતા ડેટા પ્રકારનો ઉપયોગ હિતાવહ છે. અપૂર્ણાંક અંકોનો ઉપયોગ કરવાનો લાભ એ છે કે તે પૂર્ણાંક અંકોની સરખામણીમાં કિંમતોનો વ્યાપક વિસ્તાર રજૂ કરે છે.

અક્ષર (character)

ઉપરના બંને કિસ્સાઓમાં આપણે આંકડાકીય વિગતોનો સંગ્રહ કર્યો. પરંતુ ધારો કે આપણે પુરુષ માટે 'M' (male) અને સ્ત્રી માટે 'F' (female) જેવી જાતિવાચક સંજ્ઞા અથવા તો શહેરનાં નામ જેવી વિગતોનો સંગ્રહ કરવાની જરૂર હોય તો ? અહીં આપણી પાસે માત્ર મૂળાક્ષરો છે. આ પ્રકારની કિંમતોનો સંગ્રહ int કે float દ્વારા કરી શકાતો નથી. આ પ્રકારની કિંમતોનો સંગ્રહ કરવા માટે char કી-વર્ડનો ઉપયોગ કરવામાં આવે છે. તે 1 બાઈટ જેટલા મેમરી સ્થાનનો ઉપયોગ કરે છે. દરેક અક્ષર ASCII (American Standard Code for Information Interchange) નામે ઓળખતાં પૂર્ણાંક સાથે સંકળાયેલ હોય છે. ASCII કિંમતોની વિગતો પરિશિષ્ટ-II માં આપવામાં આવી છે. પૂર્વ નિર્ધારિત રીતે char અચિહ્નિત(Unsigned) છે. ચિહ્નિત (signed) char પણ શક્ય છે. કોષ્ટક 11.3માં char ડેટા પ્રકાર વિશે ટૂંકી સમજૂતી આપવામાં આવી છે.

Data Type	Range	Bytes required	Example
char	–128 to + 127	1	char gender;
unsigned char	0 to 255	1	unsigned char choice;

કોષ્ટક 11.3 : char ડેટા પ્રકાર

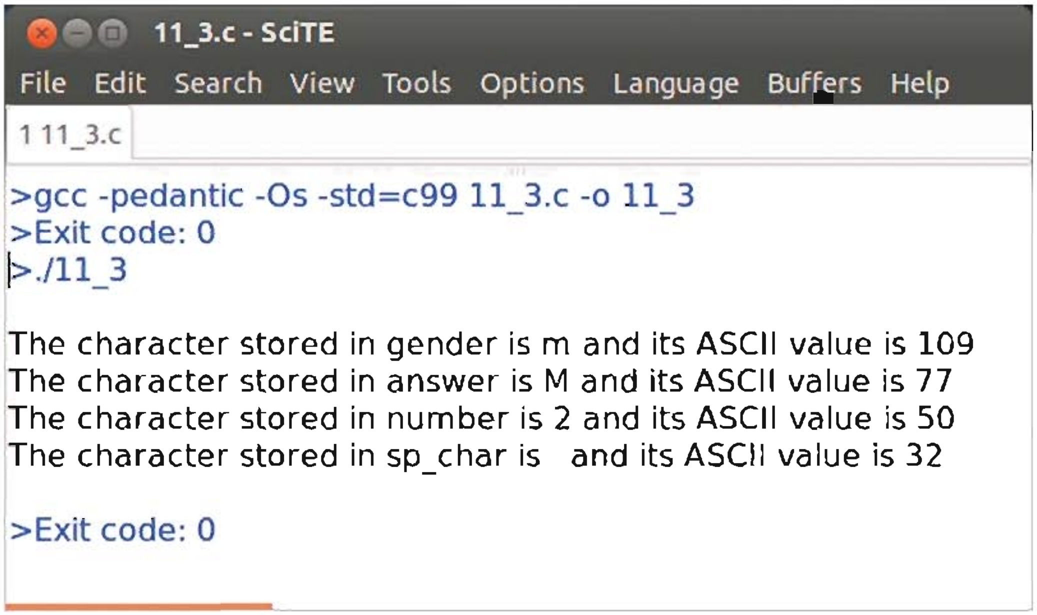
char ડેટા પ્રકારને સ્પષ્ટ કરતા પ્રોગ્રામનું કોડ લિસ્ટિંગ આકૃતિ 11.6માં દર્શાવ્યું છે.

```
11_3.c - SciTE
File Edit Search View Tools Options Language Buffers Help
1 11_3.c

/*Example 3: Program to illustrate use of character data type */

#include <stdio.h>
int main( )
{
    char gender = 'm';
    char answer = 'M';
    char number = '2';
    char sp_char = ' ';
    printf("\nThe character stored in gender is %c and its ASCII value is %d", gender, gender);
    printf("\nThe character stored in answer is %c and its ASCII value is %d", answer, answer);
    printf("\nThe character stored in number is %c and its ASCII value is %d", number, number);
    printf("\nThe character stored in sp_char is %c and its ASCII value is %d\n\n", sp_char, sp_char);
    return 0;
}
/* End of Program */
```

આકૃતિ 11.6 : char ડેટા પ્રકાર દર્શાવતો પ્રોગ્રામ

સમજૂતી (Explanation)

ઉઘડતા છગડિયા કૌંસ પછીનાં પ્રથમ ચાર વિધાનો અનુક્રમે "gender", "answer", "number" અને "sp_char" નામના ચાર ચલ ઘોષિત કરી તેને અનુક્રમે 'm', 'M', '2' અને ' ' (ખાલી જગ્યા) આપશે. પછીનાં ચાર વિધાનો આ ચલમાં આવેલી કિંમતોને તેની ASCII કિંમત સાથે સ્ક્રીન પર દર્શાવશે. ઉદાહરણ 11.3નું પરિણામ આકૃતિ 11.7માં દર્શાવ્યું છે.



આકૃતિ 11.7 : ઉદાહરણ 11.3નું પરિણામ

સ્મૉલ m અને કૅપિટલ Mની ASCII કિંમતો વચ્ચેના તફાવતની નોંધ કરો. આપણે જોઈ શકીએ છીએ કે અંકોનો ઉપયોગ અક્ષરોના સ્વરૂપે પણ કરી શકાય છે. સંખ્યા 2નો અક્ષર તરીકે ઉપયોગ કરવા માટે તેને '2' સ્વરૂપે લખવામાં આવે છે. અહીં એ પણ જુઓ કે sp_char ચલમાં આવેલ કિંમત દૃશ્યમાન ન હોવા છતાં તેનો આસ્કી અંક દૃશ્યમાન છે.

ખાલી વિગતોનો ગણ (Empty data set)

સી ભાષામાં void કી-વર્ડ દ્વારા એક વિશિષ્ટ ડેટા પ્રકાર પૂરો પાડવામાં આવે છે. આ ડેટા પ્રકાર કોઈ કિંમતનો સંગ્રહ કરતો નથી માટે તેનો 'ખાલી ગણ' તરીકે નિર્દેશ કરવામાં આવે છે. તેનો ઉપયોગ મુખ્યત્વે વિધેયની પરત કિંમત દર્શાવવા માટે કરવામાં આવે છે. આગળના પ્રકરણમાં જણાવ્યું તે મુજબ સી પ્રોગ્રામ વિધેયોનો સમૂહ છે. સી પ્રોગ્રામના વિધેય કોઈ કિંમત પરત કરી શકે છે. જો આપણે ઇચ્છતા હોઈએ કે વિધેય કોઈ કિંમત પરત ન કરે તો તેના નામની પહેલાં void કી-વર્ડ ઉમેરવામાં આવે છે.

આગળનાં તમામ ઉદાહરણોમાં પ્રોગ્રામનું અંતિમ વિધાન return 0; છે. આ વિધાન કંપાઈલરને પ્રોગ્રામમાંથી બહાર નીકળવાનો નિર્દેશ કરે છે. જો આ વિધાન ન લખવામાં આવે તો ચેતવણીનો સંદેશ "Function should return a value" રજૂ કરવામાં આવે છે. આ સંદેશ ન આવે તે માટે main( )ની આગળ void ઉમેરી શકાય છે.

ચલને કિંમત આપવી (Assigning values to variable)

પ્રોગ્રામના અમલીકરણ દરમિયાન એક વાર ઘોષિત કર્યા બાદ ચલમાં કિંમતો આપી શકાય છે. આ કિંમતો ચલના ડેટા પ્રકારને અનુરૂપ હોવી જોઈએ. નીચેની વાક્યરચના દ્વારા ચલમાં કિંમત આપી શકાય છે :

Variable = Value;

એક જ વિધાનમાં ચલની ઘોષણા કરી તેમાં કિંમત આપવી પણ શક્ય છે. આમ કરવા માટે નીચેની વાક્યરચનાનો ઉપયોગ કરી શકાય :

data type variable = value;

અત્યાર સુધી આ પ્રકરણમાં ચર્ચવામાં આવેલાં તમામ ઉદાહરણોમાં ચલનો પ્રારંભ આ તકનિકથી કરવામાં આવ્યો છે.

ઉપયોગકર્તા દ્વારા નિર્મિત ડેટા પ્રકાર (User defined data type)

સી ભાષા પરિવર્તનક્ષમ (flexible) છે. તે ઉપયોગકર્તાને હયાત ઘટકોમાંથી નવા ઘટકો બનાવવાની સુવિધા આપે છે તથા મૂળભૂત ડેટા પ્રકારનો ઉપયોગ કરી નવા ડેટા પ્રકારની રચના કરી શકાય છે. સી ભાષામાં ઉપયોગકર્તા દ્વારા વ્યાખ્યાયિત ડેટા પ્રકારની રચના કરવા માટે typedef અને enum નામના બે કી-વર્ડનો ઉપયોગ કરવામાં આવે છે. બંને પ્રકારના કી-વર્ડ દ્વારા ડેટા પ્રકાર તરીકે ઉપયોગમાં લઈ શકાય તેવા ચલ વ્યાખ્યાયિત કરી શકાય છે.

ટાઈપ ડેફિનેશન (Type definition)

ટાઈપ ડેફિનેશન દ્વારા ડેટા પ્રકાર વ્યાખ્યાયિત કરવા માટેની વાક્યરચના નીચે પ્રમાણે છે :

typedef datatype variable;

અહીં, datatype એ int કે float જેવા ઉપલબ્ધ ડેટા પ્રકારનો નિર્દેશ કરે છે. ઉપલબ્ધ ડેટા પ્રકારને આપવામાં આવનાર નવા નામનો નિર્દેશ variable દ્વારા કરવામાં આવે છે. ખરી રીતે જોતાં, સી ભાષા નવો ડેટા પ્રકાર રચવાની અનુમતિ આપતું નથી. તેને બદલે તે ઉપલબ્ધ ડેટા પ્રકારને ઉપનામ (alias) આપવાની સુવિધા ધરાવે છે. આપણે સામાન્ય રીતે જીવનમાં પણ ઉપનામોનો ઉપયોગ કરીએ છીએ. વ્યક્તિને આપવામાં આવતું હુલામણું નામ તેનું ઉદાહરણ છે. typedefના કેટલાંક ઉદાહરણ નીચે આપેલાં છે :

typedef char alpha;

typedef double twice;

પ્રથમ વિધાન char ડેટા પ્રકારને "alpha" ઉપનામ અપાશે જ્યારે બીજા વિધાન દ્વારા double ડેટા પ્રકારને "twice" ઉપનામ આપવામાં આવશે. આ ઉપનામ આપ્યા બાદ, નીચેનાં વિધાનો દ્વારા અક્ષર માટેનો "choice" નામનો ચલ તથા "amount" નામનો ''ડબલ'' પ્રકારનો ચલ ઘોષિત કરી શકાય.

alpha choice;

twice amount;

ઈન્યુમરેટેડ ડેટા પ્રકાર (Enumerated data type)

સી ભાષામાં enum કી-વર્ડ દ્વારા ઈન્યુમરેટેડ ડેટા પ્રકારની રચના કરી શકાય છે. enumનો ઉપયોગ કરવા માટેની વાક્યરચના નીચે મુજબ છે :

enum identifier {value 1, value 2, value 3, …………, value n};

અહીં enum કી-વર્ડ છે. Identifier એ નવો ડેટા ટાઈપ વ્યાખ્યાયિત કરવા માટે ઉપયોગમાં લેવામાં આવનાર નામ છે. Value 1થી Value n એ આંકડાકીય અચલો 0, 1, 2, …. વગેરેને આપવામાં આવેલ નામ છે. enum દ્વારા વ્યાખ્યાયિત કરવામાં આવેલા ચલમાં છગડિયા કૌંસમાં આપવામાં આવેલી કોઈ પણ એક કે વધુ કિંમત ઉમેરી શકાય છે. enum ના ઉપયોગનું ઉદાહરણ નીચે મુજબ છે :

enum money {rupee, dollar, pound, yen};

enum money currency;

currency = dollar;

પ્રથમ વિધાન ઈન્યુમરેટેડ ડેટા પ્રકાર "money" વ્યાખ્યાયિત કરે છે. બીજા વિધાન દ્વારા money પ્રકારનો ઉપયોગ કરી "currency" નામનો ચલ ઘોષિત કરવામાં આવ્યો છે. છેલ્લા વિધાનમાં currency ચલને "dollar" કિંમત આપવામાં આવી છે જે ''1''ને સમકક્ષ છે. અહીં, rupee, dollar, pound અને yen અનુક્રમે 0, 1, 2 અને 3 આંકડાકીય અચળો રજૂ કરે છે. આ આંકડાકીય અચળો કંપાઈલર દ્વારા આપોઆપ ઉમેરવામાં આવે છે. નવી કિંમતો ઉમેરી આ અચળોને બદલી પણ શકાય છે. ઉદાહરણ તરીકે જો આપણે rupeeને 10, dollarને 50, poundને 75 અને yenને 100 કિંમત આપવી હોય તો પ્રથમ વિધાનને enum money {rupee = 10, dollar = 50, pound = 75, yen = 100}; સ્વરૂપે લખી શકાય. હવે currency = dollar વિધાન currency = 50ને સમકક્ષ બનશે.

ઉપયોગકર્તા દ્વારા નિર્મિત ડેટા પ્રકારોના ઉપયોગથી સી પ્રોગ્રામની વાચનક્ષમતામાં વૃદ્ધિ કરી શકાય છે. તે ડેટા પ્રકારને અર્થપૂર્ણ નામ સાથે વ્યાખ્યાયિત કરવાની સુવિધા પણ આપે છે.

તારવેલા ડેટા પ્રકાર (Derived data type)

આપણે અગાઉ જોઈ ગયા તેમ સી વિસ્તારી શકાય તેવી (Extensible) ભાષા છે. આપણી ઘણી જરૂરિયાતો માટે

મૂળભૂત ડેટા પ્રકારો પૂરતા બની રહે તેમ બને, પરંતુ જટિલતાને કારણે દરેક સમયે તેનો ઉપયોગ કરવો યોગ્ય નથી. ઉદાહરણ તરીકે, ધારો કે આપણે 15 વિદ્યાર્થીઓના વર્ગ(grades)નો સંગ્રહ કરવા ઇચ્છીએ છીએ. આ માટે આપણે 15 ચલ ઘોષિત કરવાની જરૂર પડે. 15 ચલની સારસંભાળ પ્રોગ્રામની જટિલતામાં વધારો કરે. આ જ રીતે જો આપણે 15 વિદ્યાર્થીઓ માટે તેમના અનુક્રમ, જાતિ, નામ અને સરનામા જેવી સંપૂર્ણ વિગતનો સંગ્રહ કરવો હોય તો વધારાના ચલની જરૂર પડે. અહીં ચલના જૂથ કદાચ મુશ્કેલી ઊભી કરે. તેના નિવારણ માટે સી ભાષા ડેટા પ્રકાર તારવવાની સુવિધા પૂરી પાડે છે. એરે (array), સ્ટ્રક્ચર (structure), યુનિયન (union) અને પોઇન્ટર (pointer) તારવેલાં ડેટા પ્રકારનાં ઉદાહરણ છે.

એરે (Array)

સી ભાષામાં એરે નામના ડેટા બંધારણની રચના શક્ય છે. આ ડેટા બંધારણમાં એકસમાન લાક્ષણિકતા ધરાવતા ચલના સમૂહનો સમાવેશ કરી શકાય છે. એરે વ્યાખ્યાયિત કરવા માટેની વાક્યરચના નીચે મુજબ છે :

data type variable[size];

અહીં datatype એ ડેટાનો કોઈ પણ મૂળભૂત પ્રકાર હોઈ શકે છે. Variable એ એરેનું નામ છે અને size એ એરેમાં આવેલા ઘટકોની કુલ સંખ્યા છે. ઉદાહરણ તરીકે, **char name_of_subject[10];** સી વિધાન name_of_subject નામનો એક 10 કદ ધરાવતો એરે વ્યાખ્યાયિત કરશે. આનો અર્થ એ થાય કે, "name_of_subject" નામનો ચલ વ્યાખ્યાયિત કરી તેમાં 10 અક્ષરોનો સમૂહ સંગ્રહ કરી શકાશે. અહીં એરેનો દરેક ઘટક અક્ષર પ્રકારનો રહેશે.

સી ભાષામાં **char name_of_subject[ ] = "C Language";** પ્રકારનું વિધાન પણ હોઈ શકે છે. અહીં એરેની લંબાઈ કંપાઈલર દ્વારા આપોઆપ નિશ્ચિત કરી લેવામાં આવશે. આપેલ કિસ્સામાં એરેની લંબાઈ 11 હશે. પ્રકરણ 15માં એરે વિશે વિસ્તૃત ચર્ચા કરવામાં આવી છે.

સી ભાષામાં સ્ટ્રક્ચર, યુનિયન અને પોઇન્ટર જેવા અન્ય તારવેલા ડેટા પ્રકાર પણ ઉપલબ્ધ છે. આ તમામ ડેટા પ્રકારોની ચર્ચા આ પુસ્તકની મર્યાદા બહાર છે.

પ્રક્રિયક અને પદાવલિ (Operators and Expression)

અત્યાર સુધી આપણે સી ભાષામાં ઉપલબ્ધ ડેટા પ્રકારો વિશે જાણકારી મેળવી. હવે આપણે સી ભાષામાં ઉપલબ્ધ જુદા જુદા પ્રક્રિયકો વિશે અભ્યાસ કરીએ. પ્રક્રિયકો અને ઑપરેન્ડ (operand)ના ઉપયોગ દ્વારા પદાવલિ કેવી રીતે બનાવી શકાય તેનો પણ આપણે અભ્યાસ કરીશું. અંતમાં આપણે જોઈશું કે પદાવલિઓનું મૂલ્યાંકન કેવી રીતે કરવામાં આવે છે.

પ્રક્રિયકો (Operators)

શાળામાં પ્રાથમિક ગણિતનો અભ્યાસ કરતી વખતે આપણે બે સંખ્યાઓના સરવાળા કે બાદબાકી જેવી પ્રક્રિયાઓ કરેલી, જેમકે, 5 + 3 અને 9 - 7. અહીં 5, 3, 9 અને 7 અચળો છે જ્યારે '+' અને '–' નિશાનીઓ આ અચળો પરની પ્રક્રિયા વ્યાખ્યાયિત કરે છે.

ઑપરેન્ડ પર કરવામાં આવનાર પ્રક્રિયાને વ્યાખ્યાયિત કરતી નિશાનીને પ્રક્રિયક (operators) તરીકે ઓળખવામાં આવે છે. સી ભાષામાં પ્રક્રિયકોનું વર્ગીકરણ નીચે આપેલ આઠ વર્ગોમાં કરી શકાય :

1. ગાણિતિક પ્રક્રિયકો (Arithmetic Operators)
2. નિરૂપક પ્રક્રિયકો (Assignment Operators)
3. સંબંધસૂચક પ્રક્રિયકો (Relational Operators)
4. વધારા / ઘટાડા સૂચક પ્રક્રિયકો (Increment and Decrement Operators)
5. શરતી પ્રક્રિયકો (Conditional Operator)
6. તાર્કિક પ્રક્રિયકો (Logical Operators)
7. બિટવાઇઝ પ્રક્રિયકો (Bitwise Operators)
8. વિશિષ્ટ પ્રક્રિયકો (Special Operators)

આ તમામ પ્રક્રિયકો વિશે ચર્ચા કરીએ.

ગાણિતિક પ્રક્રિયકો (Arithmetic Operators)

આંકડાકીય ગણતરીઓ કરવા માટે સી ભાષામાં '+', '–', '*', '/' અને '%' પ્રક્રિયકો આપવામાં આવ્યા છે. ઘાતાંક માટેનો કોઈ પ્રક્રિયક સી ભાષામાં ઉપલબ્ધ નથી. કોષ્ટક 11.4માં આ પ્રક્રિયકો તેમના ઉપયોગ સાથે દર્શાવ્યા છે.

પ્રક્રિયક	ઉપયોગ
+	બે સંખ્યાઓનો સરવાળો અથવા યુનરી ધન
-	બે સંખ્યાઓની બાદબાકી અથવા યુનરી ઋણ
*	બે સંખ્યાઓનો ગુણાકાર
/	બે સંખ્યાઓનો ભાગાકાર કરી ભાગફળ રૂપે પરિણામ
%	બે સંખ્યાઓનો ભાગાકાર કરી શેષ રૂપે પરિણામ

કોષ્ટક 11.4 : આંકડાકીય પ્રક્રિયકો

કોષ્ટક 11.4માં આપવામાં આવેલ પ્રથમ ચાર પ્રક્રિયકો વિશે આપણે માહિતગાર છીએ. સી ભાષા મૉડ્યૂલો (Modulo) નામે ઓળખાતા એક અતિરિક્ત પ્રક્રિયક '%'ની સુવિધા પૂરી પાડે છે. બે પૂર્ણાંક સંખ્યાઓના ભાગાકારથી મળતી શેષ ગણવા માટે આ પ્રક્રિયકનો ઉપયોગ કરવામાં આવે છે. અહીં એ નોંધ લેવી જરૂરી છે કે આ પ્રક્રિયકનો ઉપયોગ અપૂર્ણ (float) સંખ્યાઓ સાથે શક્ય નથી.

હવે આપણે આ પ્રક્રિયકોનો ઉપયોગ કેવી રીતે કરવો તે જોઈએ. ઉદાહરણ સ્વરૂપે આપેલ નીચેનું સી વિધાન જુઓ :

total_cost = quantity * cost_item;

આ સી વિધાન સી પદાવલિ(C expression)ની રચના કરે છે. અહીં 'total_cost', 'quantity' અને 'cost_item'ને સી ચલ તરીકે ઓળખવામાં આવે છે. અહીં જોઈ શકાય છે કે પદાવલિમાં બે પ્રક્રિયકો અને ત્રણ ઑપરેન્ડ આવેલાં છે. આ પદાવલિનું મૂલ્યાંકન કરવામાં આવે ત્યારે બે પ્રક્રિયાઓ થાય છે. પ્રથમ, quantity અને cost_item ચલમાં રહેલી કિંમતોનો ગુણાકાર કરવામાં આવે છે અને બીજું, ગુણાકારથી મળેલા પરિણામનો total_cost નામના ચલમાં સંગ્રહ કરવામાં આવે છે.

પદાવલિમાં ઉપયોગમાં લેવામાં આવેલા ઑપરેન્ડને આધારે ગણતરીને ત્રણ વર્ગોમાં વહેંચી શકાય : પૂર્ણાંક ગણતરી (integer arithmetic), અપૂર્ણાંક ગણતરી (real arithmetic) અને મિશ્ર પ્રકારની ગણતરી (mixed mode arithmetic).

પૂર્ણાંક ગણતરી (Integer Arithmetic)

જો પદાવલિમાં ઉપયોગમાં લેવામાં આવેલ ઑપરેન્ડ ધન કે ઋણ પૂર્ણ સંખ્યાઓ હોય તો તેને પૂર્ણાંક ગણતરી (integer arithmetic) તરીકે ઓળખવામાં આવે છે. અહીં, પદાવલિને પૂર્ણાંક પદાવલિ (integer expression) કહે છે. પૂર્ણાંક પદાવલિનું પરિણામ હંમેશાં પૂર્ણાંક મળે છે.

હવે, કુલ કિંમત અને વસ્તુની સંખ્યા આપવામાં આવી હોય ત્યારે વસ્તુની એકમ કિંમત શોધવા માટેનો પ્રોગ્રામ જોઈએ. પૂર્ણાંક ગણતરી રજૂ કરતો આ પ્રોગ્રામ આકૃતિ 11.8માં દર્શાવ્યો છે.

```
11_4.c - SciTE
File Edit Search View Tools Options Language Buffers Help
1 11_4.c

/* Example 4: Program to illustrate use of integer arithmetic */

#include <stdio.h>
int main( )
{
   /* Use of typedef to create new data type name */
    typedef int whole_number;

    /* Declaration of variables using new data type name */
   whole_number total_cost = 650 , quantity = 24, cost_item;

    /* C expression for calculating cost per item */
    cost_item = total_cost / quantity;

    printf("Cost of one item is %d\n\n", cost_item);
    return 0;
}
/* End of Program */
```

આકૃતિ 11.8 : પૂર્ણાંક ગણતરી દર્શાવતો પ્રોગ્રામ

સમજૂતી (Explanation)

અહીં પ્રથમ વિધાન દ્વારા typedef કી-વર્ડનો ઉપયોગ કરી int માટેનું ઉપનામ આપવામાં આવ્યું છે. ત્યાર પછી નવા ઉપનામનો ઉપયોગ કરી "total_cost", "quantity" અને "cost_item" નામના ત્રણ પૂર્ણાંક ચલને વ્યાખ્યાયિત કરવામાં આવ્યા છે. "total_cost" ચલમાં 650 અને "quantity" ચલમાં 24 કિંમત પણ ઉમેરવામાં આવી છે. પછીનું વિધાન "total_cost" નો "quantity" વડે ભાગાકાર કરી પરિણામનો "cost_item" ચલમાં સંગ્રહ કરે છે. ત્યાર પછી "cost_item" ચલની કિંમતને સંદેશ સાથે દર્શાવવામાં આવી છે. ઉદાહરણ 11.4નું પરિણામ આકૃતિ 11.9માં દર્શાવ્યું છે.

```
11_4.c - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 11_4.c

>gcc -pedantic -Os -std=c99 11_4.c -o 11_4
>Exit code: 0
>./11_4
Cost of one item is 27

>Exit code: 0
```

આકૃતિ 11.9 : ઉદાહરણ 11.4નું પરિણામ

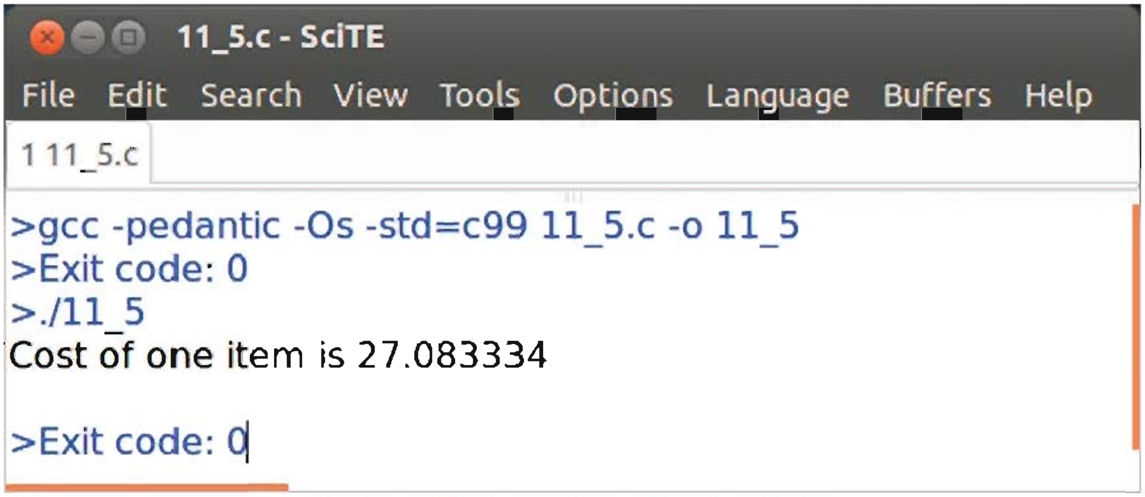
પારંપરિક ગણિતશાસ્ત્ર મુજબ 650/24નું પરિણામ 27.083333333 મળવું જોઈએ. પરંતુ પરિણામમાં "cost_item" ચલની કિંમત 27 જોઈ શકાય છે. આ ઉપરથી કહી શકાય કે પૂર્ણાંક ગણતરીનું પરિણામ હંમેશા પૂર્ણાંક સંખ્યામાં મળશે. પારંપરિક ગણિત કરતાં સી ભાષામાં ભાગાકાર માટેના પ્રક્રિયકનો ઉપયોગ અલગ છે. પારંપરિક ગણિતમાં 5/10 નું પરિણામ મળે, જે અહીં 0 મળશે.

અપૂર્ણાંક ગણતરી (Real Arithmetic)

આપેલ પદાવલિમાં ઉપયોગમાં લેવામાં આવેલા પ્રક્રિયકો જો અપૂર્ણાંક હોય તો તેને અપૂર્ણાંક ગણતરી (real arithmatic) કહે છે. અપૂર્ણાંક ગણતરીનું પરિણામ હંમેશા દશાંશચિહ્ન સાથેની કિંમતો દ્વારા દર્શાવવામાં આવે છે. પૂર્ણાંક ગણતરીને અપૂર્ણાંક ગણતરીમાં ફેરવવા માટે આકૃતિ 11.8માં દર્શાવેલ પ્રોગ્રામમાં ફેરફાર કરીએ. સુધારેલો પ્રોગ્રામ આકૃતિ 11.10માં અને તેનું પરિણામ આકૃતિ 11.11માં દર્શાવ્યું છે.

```
11_5.c - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 11_5.c

/* Example 5: Program to illustrate use of float arithmetic */

#include <stdio.h>
int main( )
{
    /* Use of typedef to create new data type name */
    typedef float decimal;

    /* Declaration of variables using new data type name */
    decimal total_cost = 650 , quantity = 24, cost_item;

    /* C expression for calculating cost per item */
    cost_item = total_cost / quantity;

    printf("Cost of one item is %f\n\n", cost_item);
    return 0;
}
/* End of Program */
```

આકૃતિ 11.10 : અપૂર્ણાંક ગણતરીનું ઉદાહરણ દર્શાવતો પ્રોગ્રામ



આકૃતિ 11.11 : ઉદાહરણ 11.5નું પરિણામ

સમજૂતી (Explanation)

આ ઉદાહરણ એક અપવાદ સિવાય ઉદાહરણ 11.4 જેવું જ છે. અહીં ચલને float સ્વરૂપે ઘોષિત કર્યા છે. "total_cost" અને "quantity" ચલને આપવામાં આવેલી કિંમતો પૂર્ણાંક હોવા છતાં આંતરિક રીતે તેનો સંગ્રહ અપૂર્ણાંક (float) તરીકે થયેલ છે. માટે, પરિણામ પણ અપૂર્ણાંક સંખ્યામાં છે.

મિશ્ર પ્રકારની ગણતરી (Mixed Mode Arithmetic)

આ પ્રકારની ગણતરીમાં પૂર્ણાંક તેમજ અપૂર્ણાંક બંને પ્રકારના ઑપરેન્ડનો ઉપયોગ કરી શકાય છે. અહીં પરિણામનો આધાર ચલને આપવામાં આવેલ કિંમતોના પ્રકાર પર છે. result = 25.75 * 5; જેવી પદાવલિ સી ભાષામાં લખવી શક્ય હોવા છતાં તેનું સીધું મૂલ્યાંકન શક્ય નથી. ઉપયોગમાં લેવામાં આવેલાં તમામ ઑપરેન્ડ સમાન ડેટા પ્રકાર ધરાવતા હોય તો જ પદાવલિનું મૂલ્યાંકન શક્ય બને છે. મિશ્ર પ્રકારની ગણતરી કરી શકાતી હોવાથી સી ભાષા આ મુશ્કેલીનું નિવારણ કરે છે. સી ભાષામાં નિમ્ન સ્તરના ડેટા પ્રકારનું ઉચ્ચ ડેટા પ્રકારમાં આપોઆપ રૂપાંતરણ કરવામાં આવે છે.

નિરૂપક પ્રક્રિયકો (Assignment Operators)

અત્યાર સુધી ચર્ચેલાં ઉદાહરણોમાં આપણે total_cost = quantity * cost_item; કે date = 30; જેવાં વિધાનોનો ઉપયોગ કર્યો છે. અહીં "=" નિશાનીને નિરૂપક પ્રક્રિયક (Assignment Operator) કહે છે. કોઈ અચળ કિંમત કે પદાવલિના પરિણામનું ચલમાં નિરૂપણ કરવા માટે તેનો ઉપયોગ કરવામાં આવે છે.

સી ભાષામાં શૉર્ટહૅન્ડ (short hand) પ્રક્રિયક નામે ઓળખાતાં પ્રક્રિયકો પણ પૂરાં પાડવામાં આવ્યાં છે. ગાણિતિક પ્રક્રિયકોની પાછળ "=" નિશાની ઉમેરવાથી આ પ્રક્રિયક મેળવી શકાય છે. તેનો ઉપયોગ કરવા માટેની વાક્યરચના નીચે મુજબ છે :

variable op= constant value; અથવા *variable op= expression;*

અહીં "op" એ ગાણિતિક પ્રક્રિયક છે. જ્યારે "op="ને શૉર્ટહૅન્ડ પ્રક્રિયક તરીકે ઓળખવામાં આવે છે. શૉર્ટહૅન્ડ પ્રક્રિયકનાં કેટલાંક ઉદાહરણ નીચે આપ્યાં છે :

first -= 1; આ વિધાન first = first - 1;ને સમકક્ષ છે.

first += 3; આ વિધાન first = first + 3;ને સમકક્ષ છે.

first *= (second + third); આ વિધાન first = first * (second + third);ને સમકક્ષ છે.

શૉર્ટહૅન્ડ પ્રક્રિયક સરળ ઉપયોગિતા પૂરી પાડે છે. સરળ હોવા છતાં તેના સક્ષમ ઉપયોગ માટે થોડો મહાવરો જરૂરી છે.

સંબંધસૂચક પ્રક્રિયકો (Relational Operators)

સંબંધસૂચક પ્રક્રિયકો સમાન પ્રકારના બે ઑપરેન્ડને સરખાવવાની સુવિધા પૂરી પાડે છે અને સામાન્ય રીતે પ્રોગ્રામના અમલીકરણનો પ્રવાહ બદલવા માટે તેનો ઉપયોગ કરવામાં આવે છે. ઉદાહરણ તરીકે, આપણે ગ્રાહકને તેણે ખરીદેલી વસ્તુની રકમ તપાસી તે પ્રમાણે વળતર આપવા ઈચ્છીએ છીએ. જો કિંમત કોઈ નિશ્ચિત રકમથી વધુ હોય તો જ ગ્રાહકને વળતર આપવાનું છે. આ ઉદાહરણને સી ભાષામાં *if (total_purchase > 10000) discount = 500;* તરીકે રજૂ કરી શકાય.

સરખામણી કરવા માટે સી ભાષા છ સંબંધસૂચક પ્રક્રિયકો પૂરાં પાડે છે. તમામ સંબંધસૂચક પ્રક્રિયકોની યાદી અને તેના ઉપયોગો કોષ્ટક 11.5માં દર્શાવ્યા છે.

પ્રક્રિયક	ઉપયોગ
= =	બે ઑપરેન્ડની સમાનતા (equality) ચકાસશે.
! =	બે ઑપરેન્ડની અસમાનતા (non equality) ચકાસશે.
>	બે ઑપરેન્ડમાંથી મોટી (greater) કિંમત ચકાસશે.
<	બે ઑપરેન્ડમાંથી નાની (smaller) કિંમત ચકાસશે.
>=	બે ઑપરેન્ડમાંથી મોટી કિંમત અથવા સમાનતા (greater or equality) ચકાસશે.
<=	બે ઑપરેન્ડ માટે નાની કિંમત અથવા સમાનતા (smaller or equality) ચકાસશે.

કોષ્ટક 11.5 : સંબંધસૂચક પ્રક્રિયકો

અહીં એ નોંધવું જરૂરી છે કે બે ઑપરેન્ડની સમાનતા ચકાસવા માટે "="ને બદલે "= =" પ્રક્રિયકનો ઉપયોગ થાય છે. કારણ કે, સી ભાષામાં "=" પ્રક્રિયકનો ઉપયોગ નિરૂપક પ્રક્રિયક તરીકે કરવામાં આવે છે. સંબંધસૂચક પ્રક્રિયકોના ઉપયોગ માટે નીચેની વાક્યરચનાનો ઉપયોગ કરવામાં આવે છે :

expression-1 Rop expression-2

અહીં "Rop" એટલે સંબંધસૂચક પ્રક્રિયક (relational operator) તથા expression-1 અને expression-2 ગાણિતિક પદાવલિ, ચલ કે અચળ હોઈ શકે છે. "if" જેવા નિર્ણય માળખાં તથા "for", "while" અને "do..while" જેવા નિયંત્રણ માળખાં સાથે સંબંધસૂચક પ્રક્રિયકોનો ઉપયોગ કરવામાં આવે છે. આ માળખાઓ વિશે પ્રકરણ-13 અને 14માં વિસ્તૃત ચર્ચા કરવામાં આવી છે. સંબંધસૂચક પ્રક્રિયકનો ઉપયોગ કર્યો હોય તેવા ઘણા પ્રોગ્રામ આ પુસ્તકમાં જોઈ શકાશે.

વધારા / ઘટાડા સૂચક પ્રક્રિયકો (Increment / Decrement Operators)

પ્રોગ્રામમાં આવેલ ચલની કિંમતમાં કેટલીકવાર 1 જેટલો વધારો કે ઘટાડો કરવાની જરૂર પડતી હોય છે. સામાન્ય રીતે વિધાનોના ગણને પુનરાવર્તિત કરવાના હોય ત્યારે આ પ્રકારની સ્થિતિ ઉદ્ભવે છે. અહીં લૂપના અમલની સંખ્યાની દેખરેખ રાખે એવા ચલની જરૂર પડે છે. આ માટે શૉર્ટહેન્ડ પ્રક્રિયકો (short hand operators)નો ઉપયોગ કરી શકાય છે. ઉદાહરણ તરીકે આ માટે variable += 1 અથવા variable – = 1 વાક્યરચનાનો ઉપયોગ કરી શકાય.

સી ભાષા આ કાર્ય માટે બે વિશિષ્ટ યુનરી (unary) પ્રક્રિયકો "++" અને "– –" પૂરાં પાડે છે. આ પ્રક્રિયકોને એક જ ઑપરેન્ડની જરૂર પડે છે. "++"ને વધારા સૂચક (Increment) પ્રક્રિયક અને "– –" ને ઘટાડા સૂચક (Decrement) પ્રક્રિયક તરીકે ઓળખવામાં આવે છે. આ પ્રક્રિયકોના ઉપયોગ માટેની વાક્યરચના નીચે મુજબ છે :

++ variable; અથવા variable ++

– – variable; અથવા variable – –

અહીં, "++" અને "– –"નો ઉપયોગ ચલની આગળ કે પાછળ ઉમેરીને કરવામાં આવે છે. ચલની આગળ ઉમેરવામાં આવે ત્યારે તેને 'પ્રી-ઇન્ક્રિમેન્ટ' (pre-increment) અથવા પ્રી-ડિક્રિમેન્ટ (pre-decrement) પ્રક્રિયક તરીકે ઓળખવામાં આવે છે. અન્યથા તેને ચલની પાછળ ઉમેરવામાં આવે તો પોસ્ટ ઇન્ક્રિમેન્ટ (post increment) અથવા પોસ્ટ ડિક્રિમેન્ટ (post decrement) પ્રક્રિયક કહેવાય છે.

આ પ્રક્રિયકોના પરિણામ સ્વરૂપે ચલની કિંમતમાં 1નો વધારો કે ઘટાડો કરવામાં આવે છે. આ વધારા કે ઘટાડાની અસર પ્રોગ્રામમાં કરવામાં આવેલ પ્રક્રિયકની રીત પર અવલંબે છે. ધારો કે વિધાન નીચે મુજબ લખ્યાં છે :

int first = 15, second = 20, result;

result = first + second++;

અહીં, "result" ચલની કિંમત 35 થશે, 36 નહીં. જ્યારે બીજા વિધાનને અંતે "second" ચલની કિંમત 21 બનશે. તેની પહેલાંની કિંમતમાં 1 ઉમેરવામાં આવશે.

અહીં એ નોંધવું જરૂરી છે કે પોસ્ટ ઇન્ક્રીમેન્ટ પ્રક્રિયકનો ઉપયોગ કરવામાં આવે ત્યારે પદાવલિનું મૂલ્યાંકન કરવા માટે ચલની જૂની કિંમતનો ઉપયોગ કરવામાં આવેશે અને ત્યાર પછી ચલની કિંમતમાં 1 નો વધારો કરવામાં આવે છે.

આ જ રીતે જો નીચેનાં વિધાનો લખવામાં આવે,

int first = 15, second = 20, result;

result = first + ++second;

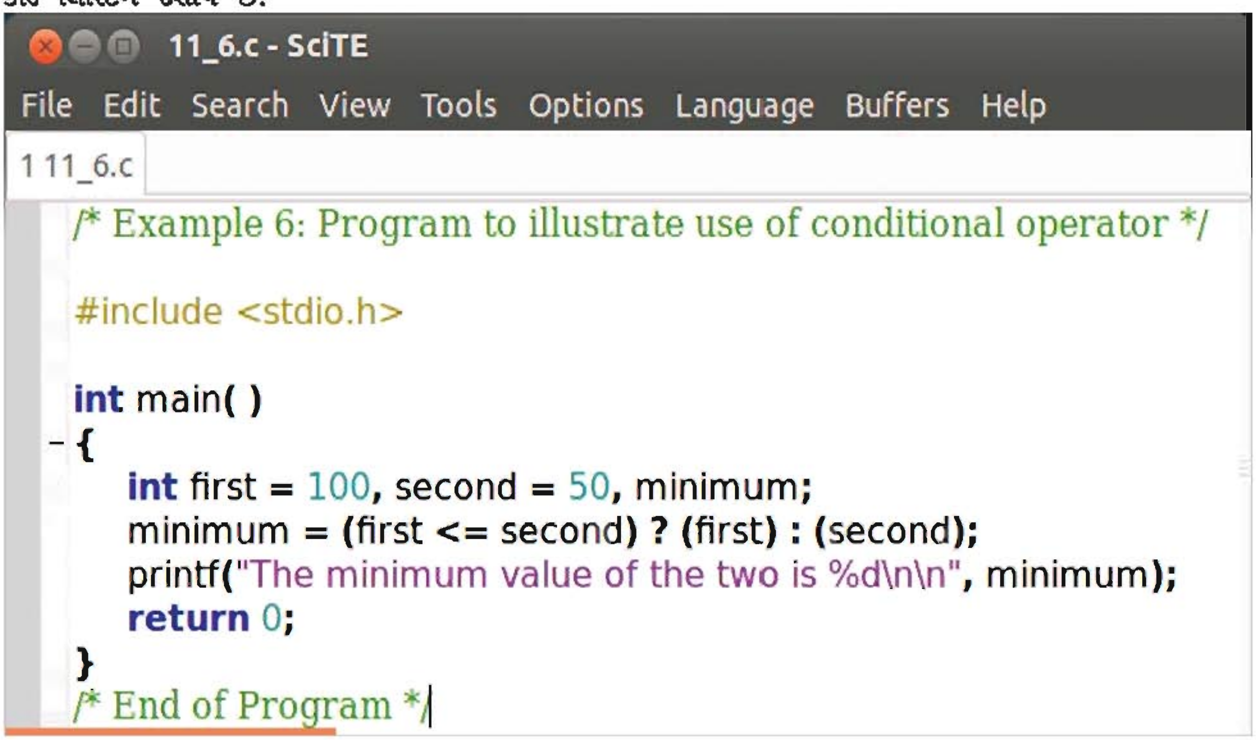
તો "result" ચલની પરિણામી કિંમત 36 હશે. જ્યારે બીજા વિધાનના અંતે "second" ચલની કિંમત 21 બનશે. અહીં એ નોંધવું જરૂરી છે કે, પ્રી-ઇન્ક્રિમેન્ટ પ્રક્રિયકનો ઉપયોગ કરવામાં આવે ત્યારે સૌ પ્રથમ ચલની કિંમતમાં 1નો વધારો કરવામાં આવે છે તથા ત્યાર પછી આ નવી કિંમતનો ઉપયોગ કરી પદાવલિનું મૂલ્યાંકન કરવામાં આવે છે.

શરતી પ્રક્રિયક (Conditional Operators)

શરતોને ચકાસવા માટે સી ભાષા ટર્નરી પ્રક્રિયક (ternary operator) અથવા શરતી પ્રક્રિયક (conditional operator) નામે ઓળખાતાં પ્રક્રિયક પૂરાં પાડે છે. આ પ્રક્રિયકને વ્યાખ્યાયિત કરવા માટે બે નિશાનીઓ- "? :" નો ઉપયોગ કરવામાં આવે છે. શરતી પ્રક્રિયકના ઉપયોગ માટેની વાક્યરચના નીચે મુજબ છે :

(condition) ? (True statement) : (False statement);

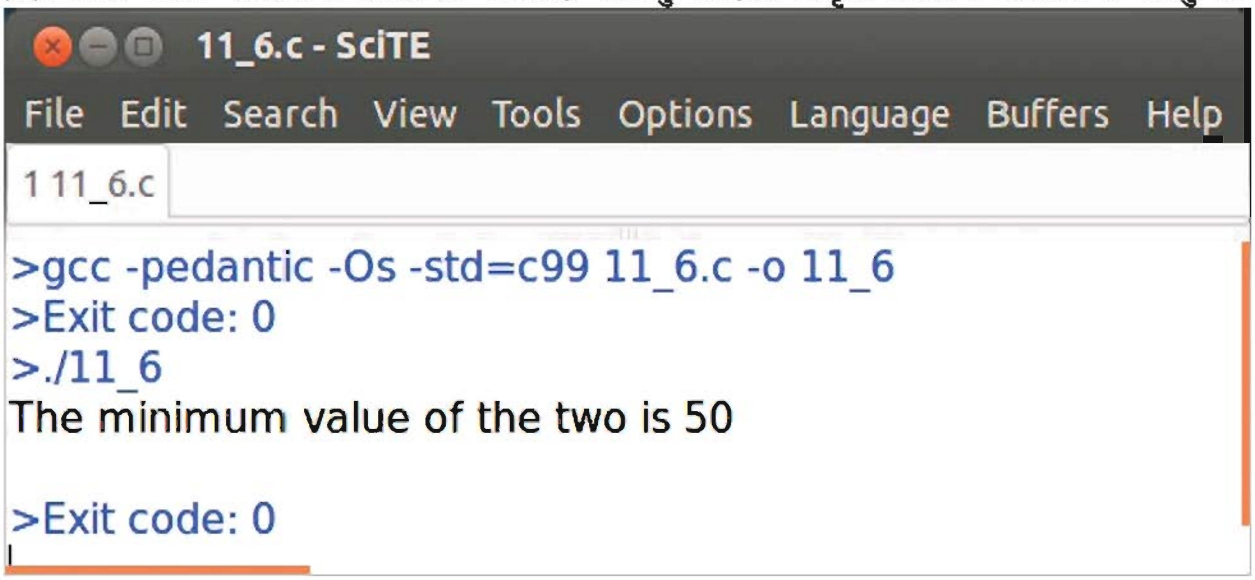
અહીં, condition એ first < second જેવી કોઈ સંબંધસૂચક પદાવલિનો નિર્દેશ કરે છે. બે સંખ્યાઓમાંથી નાની સંખ્યા શોધવા માટેના પ્રોગ્રામની મદદથી શરતી પ્રક્રિયકનો ઉપયોગ સમજીએ. આકૃતિ 11.12 આ પ્રોગ્રામ માટેનું કોડ લિસ્ટિંગ દર્શાવે છે.



આકૃતિ 11.12 : શરતી પ્રક્રિયકનો ઉપયોગ દર્શાવતો પ્રોગ્રામ

સમજૂતી (Explanation)

પ્રોગ્રામનું પ્રથમ વિધાન ત્રણ પૂર્ણાંક ચલની ઘોષણા કરી તેમાંથી બે ચલમાં કિંમત ઉમેરે છે. "first" અને "second" ચલમાં અનુક્રમે 100 અને 50 કિંમતો ઉમેરવામાં આવે છે. ત્યાર પછી શરતી પ્રક્રિયક દ્વારા ચકાસવામાં આવે છે કે "first" ચલની કિંમત "second" ચલની કિંમત કરતાં ઓછી અથવા તેના જેટલી છે કે નહીં. જો શરત સાચી (true) હોય તો "minimum" ચલને first ચલની કિંમત આપવામાં આવે છે. અન્યથા તેમાં 'second' ચલની કિંમત ઉમેરવામાં આવે છે. ત્યાર પછી યોગ્ય સંદેશ સાથે minimum ચલની કિંમત સ્ક્રીન પર દર્શાવવામાં આવે છે. અંતમાં પ્રોગ્રામમાંથી બહાર નીકળવાનો નિર્દેશ છે. ઉદાહરણ 11.6નું પરિણામ આકૃતિ 11.13માં દર્શાવવામાં આવ્યું છે.



આકૃતિ 11.13 : ઉદાહરણ 11.6નું પરિણામ

તાર્કિક પ્રક્રિયકો (Logical Operators)

કેટલીકવાર પરિણામ મેળવવા માટે એક જ સમયે એકથી વધુ શરતોની ચકાસણી કરવાની જરૂર પડે છે. જ્યારે કેટલીકવાર શરતોના ગણમાંથી કોઈ પણ એક શરત સંતોષાય તો પણ પરિણામ મેળવવું જરૂરી બને છે. આ પ્રકારના સંબંધને સામાન્ય રીતે તાર્કિક સંબંધ (logical relation) તરીકે ઓળખવામાં આવે છે. ઉદાહરણ તરીકે, બે વિષયોની પરીક્ષા આપી હોય તેવા કોઈ વિદ્યાર્થીનો વિચાર કરો. જો તે વિદ્યાર્થીએ બંને વિષયોમાં 35 કે તેથી વધુ ગુણ મેળવ્યા હશે તો જ તેને 'ઉત્તીર્ણ' જાહેર કરવામાં આવશે. અહીં આ સંબંધને 'જો (પ્રથમ વિષયના ગુણ >= 35) અને (બીજા વિષયના ગુણ >= 35) હોય તો ઉત્તીર્ણ અન્યથા અનુત્તીર્ણ'- આ પ્રકારે દર્શાવી શકાય.

આ પ્રકારના સંબંધ રજૂ કરવા માટે સી ભાષા તાર્કિક પ્રક્રિયકોની સુવિધા પૂરી પાડે છે. કોષ્ટક 11.6 આ તાર્કિક પ્રક્રિયકોની યાદી અને તેના ઉપયોગો દર્શાવે છે.

પ્રક્રિયક	ઉપયોગ
&&	તાર્કિક AND
||	તાર્કિક OR
!	તાર્કિક NOT

કોષ્ટક 11.6 : તાર્કિક પ્રક્રિયકો

આપેલ તમામ શરતો ફરજિયાત સંતોષવી હોય ત્યારે AND તાર્કિક પ્રક્રિયકનો ઉપયોગ કરવામાં આવે છે. તથા આપેલ તમામ શરતોમાંથી ઓછામાં ઓછી એક શરત સંતોષવી હોય ત્યારે OR તાર્કિક પ્રક્રિયકનો ઉપયોગ કરવામાં આવે છે. તાર્કિક પ્રક્રિયકોના પરિણામ સ્વરૂપે 0 અથવા 1 મળે છે. અહીં 'શૂન્ય' એ 'ખોટા' (false) તથા 1 એ 'સાચા' (true) પરિણામનો નિર્દેશ કરે છે. કોષ્ટક 11.7 જુદા જુદા તાર્કિક પ્રક્રિયકોનું પરિણામ દર્શાવે છે.

Expression 1	Expression 2	! (Expression 2)	Expression 1 && Expression 2	Expression 1 || Expression 2
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

કોષ્ટક 11.7 : તાર્કિક પ્રક્રિયકોનું પરિણામ

અહીં જોઈ શકાય છે કે, જો બંને પદાવલિ '1' પરત કરે તો જ AND પ્રક્રિયક 1 પરત કરે છે. જ્યારે આપેલમાંથી કોઈ પણ એક પદાવલિ 1 પરત કરે તો OR પ્રક્રિયક 1 પરત કરે છે.

હવે, નીચે આપેલ માપદંડને આધારે વિદ્યાર્થીનો વર્ગ (Grade) શોધી આપે તે માટેનો પ્રોગ્રામ બનાવીએ :

વર્ગ (Grade)	ગુણ (Marks)
A વર્ગ	જો ટકા 70 કે તેથી વધુ હોય
B વર્ગ	જો ટકા 60 થી 69ની વચ્ચે હોય
C વર્ગ	જો ટકા 35 થી 59ની વચ્ચે હોય
F વર્ગ	જો ટકા 35 થી ઓછા હોય.

આકૃતિ 11.14માં આ પ્રોગ્રામનું કોડ લિસ્ટિંગ આપવામાં આવ્યું છે તથા આકૃતિ 11.15 તેનું પરિણામ દર્શાવે છે.

11_7.c - SciTE

File Edit Search View Tools Options Language Buffers Help

1 11_7.c

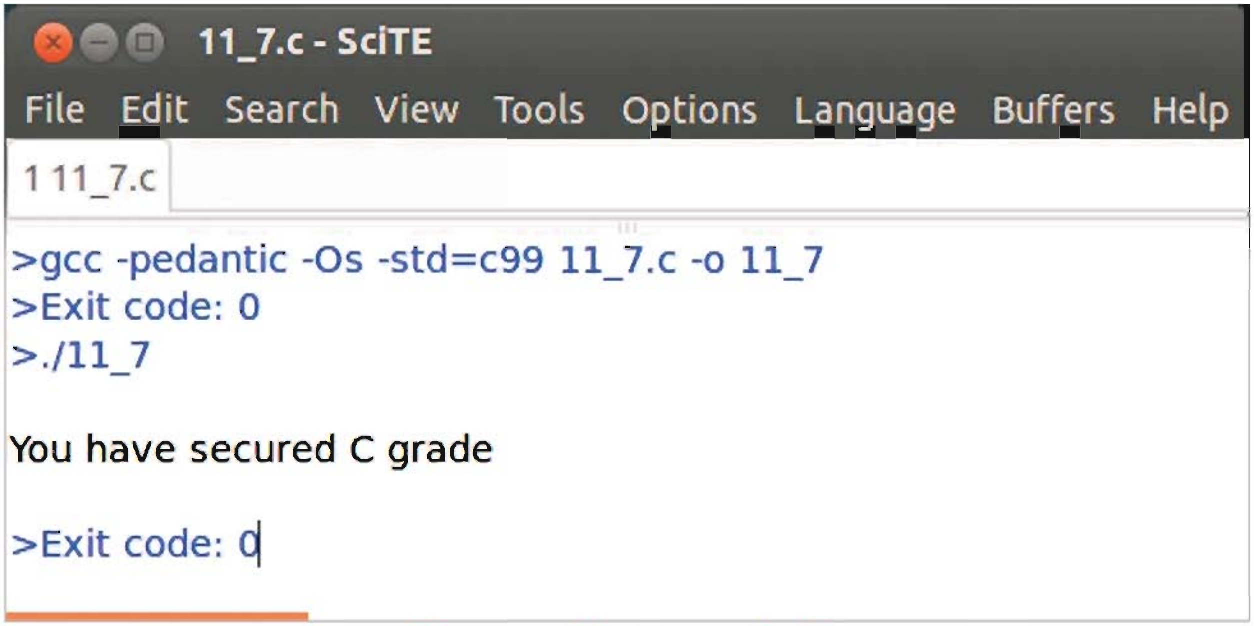
```c
/* Example 7: Program to illustrate use of logical operator */

#include <stdio.h>
int main( )
{
    int percentage = 55;

    /* Verify whether percentage is between 0 and 100 using AND */
    if (  percentage >= 0  &&  percentage <= 100  )
    {
      /* Check whether percentage greater than 70 */
      if ( percentage >= 70 )
      printf("\nYou have secured A grade\n\n");

      /* Check whether  percentage between 60 and 69 */
      if ( percentage >= 60 && percentage < 70 )
      printf("\nYou have secured B grade\n\n");

      /* Check whether  percentage between 35 and 60 */
      if ( percentage >= 35 && percentage < 60 )
      printf("\nYou have secured C grade\n\n");

      /* Check whether percentage less than 35 */
     if ( percentage < 35 )
     printf("\nYou have secured F grade\n\n");
    }
    else    /* Executed when percentage below 0 or above 100 */
    {
     printf("\nYou have entered wrong value in percentage");
     printf("\nPlease enter value between 0 and 100\n\n");
    }
    return 0;
}
/* End of Program */
```

આકૃતિ 11.14 : તાર્કિક પ્રક્રિયકનો ઉપયોગ દર્શાવતો પ્રોગ્રામ

સમજૂતી (Explanation)

અહીં પ્રથમ વિધાન દ્વારા "percentage" નામના પૂર્ણાંક ચલને ઘોષિત કરી તેમાં કિંમત આપવામાં આવે છે. શરૂઆતમાં "percentage" ચલની કિંમત 0 થી 100 વચ્ચે આપેલ છે કે નહીં તે ચકાસવામાં આવે છે. જો જવાબ 'ના' મળે તો 'else' વિભાગમાં આવેલ સંદેશ "You have entered wrong value in percentage, Please enter value between 0 and 100" દર્શાવી પ્રોગ્રામમાંથી બહાર નીકળી શકાય છે.

જો સરખામણીનું પરિણામ 'હા' (true) મળશે તો ટકાની કિંમતને if વિધાન અને તાર્કિક પ્રક્રિયકો દ્વારા ચકાસવામાં આવશે. જો if વિધાનમાં આપેલ માપદંડનું પરિણામ હકારાત્મક મળશે તો તેનો સંદેશ સ્ક્રીન પર દર્શાવવામાં આવશે. આકૃતિ 11.15 જુઓ.



આકૃતિ 11.15 : ઉદાહરણ 11.7નું પરિણામ

બિટવાઇઝ પ્રક્રિયકો (Bitwise Operators)

આપણે આ પ્રકરણમાં જોયું કે ડેટાને મેમરીસ્થાન પર બિટ (bit) સ્વરૂપે સંગ્રહવામાં આવે છે. સીધા જ બિટ સ્તરે કાર્ય કરવા માટે સી ભાષામાં બિટવાઇઝ પ્રક્રિયકોની સુવિધા આપવામાં આવી છે. સી ભાષામાં ઉપલબ્ધ બિટવાઇઝ પ્રક્રિયકોની યાદી અને તેના ઉપયોગ કોષ્ટક 11.8માં આપવામાં આવ્યા છે.

પ્રક્રિયક	ઉપયોગ
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT
^	Bitwise Exclusive OR
<<	આપેલ બિટ જેટલા અંકોને ડાબી બાજુ ખસેડવા
>>	આપેલ બિટ જેટલા અંકોને જમણી બાજુ ખસેડવા

કોષ્ટક 11.8 : બિટવાઇઝ પ્રક્રિયકો (Bitwise Operators)

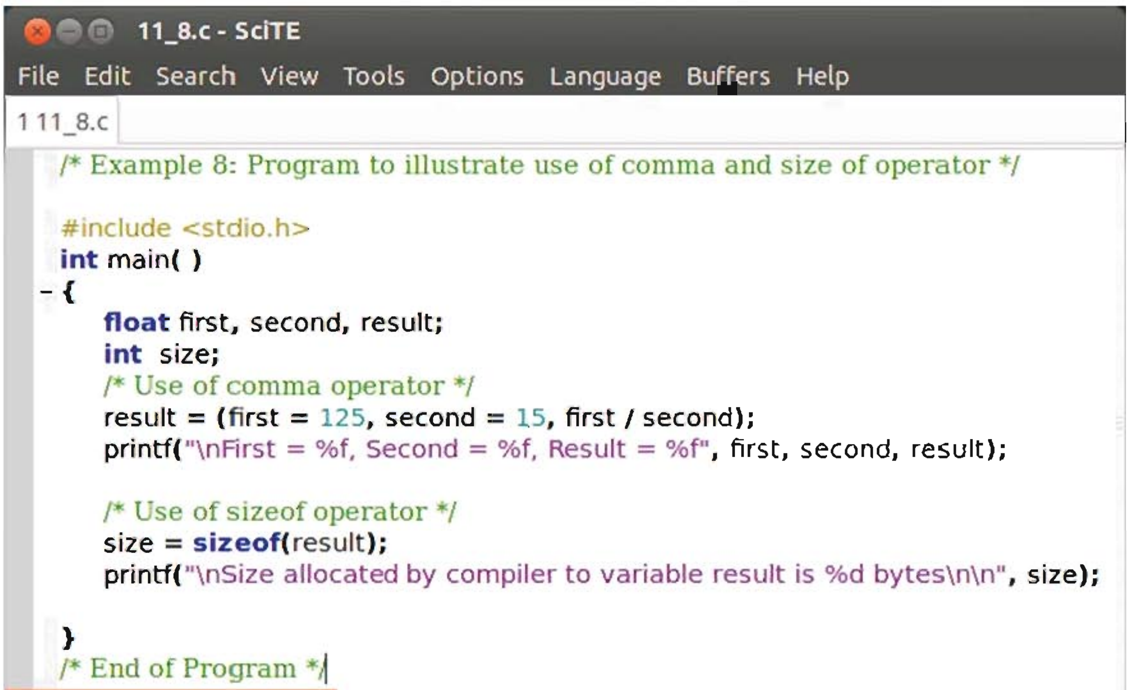
બિટવાઇઝ AND, OR, Exclusive OR (XOR), Left Shift અને Right Shift પ્રક્રિયકોનો ઉપયોગ કરવા માટે બે ઑપરેન્ડ જરૂરી છે. બીજી તરફ NOT પ્રક્રિયક માત્ર એક ઑપરેન્ડ સાથે કાર્ય કરે છે. અહીં ઑપરેન્ડ 0 અને 1 કિંમતો સ્વરૂપે હોય છે. આ પ્રક્રિયકોની વિસ્તૃત ચર્ચા પુસ્તકની મર્યાદા બહાર છે.

વિશિષ્ટ પ્રક્રિયકો (Special Operators)

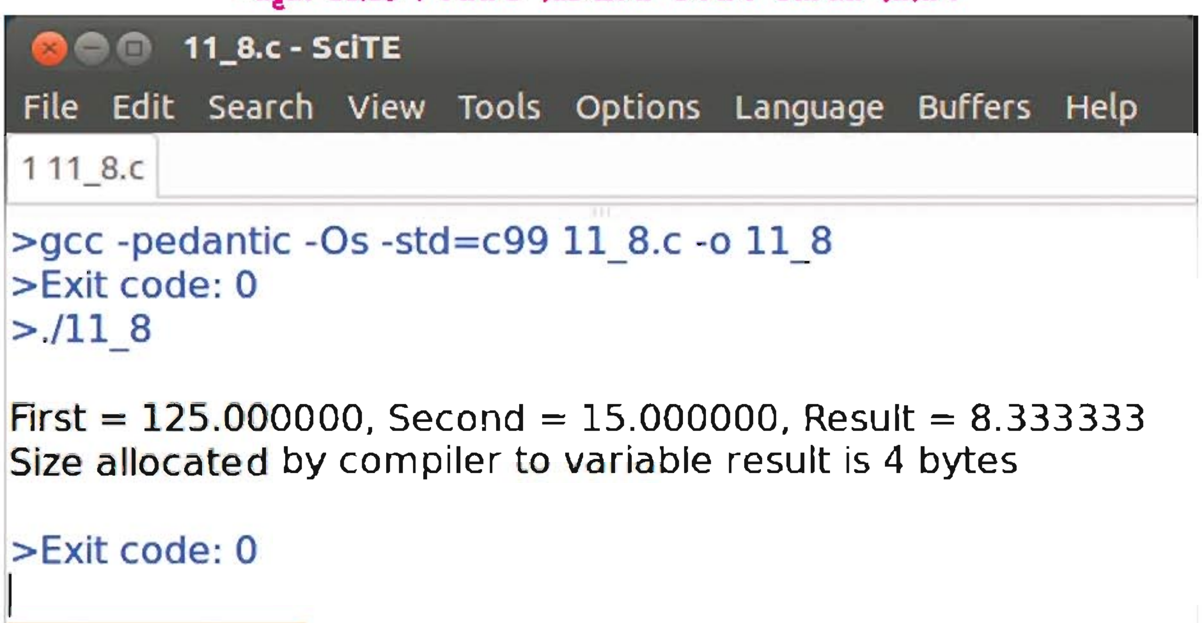
સી ભાષા sizeof( ), ",", ".", "– >", "&" અને "*" જેવાં વિશિષ્ટ પ્રક્રિયકો પૂરાં પાડે છે. આ વિભાગમાં આપણે માત્ર "," અને sizeof() પ્રક્રિયકોની ચર્ચા કરીશું. "," (અલ્પવિરામ) પ્રક્રિયકનો ઉપયોગ ઘણી જગ્યાએ કરવામાં આવે છે. અલ્પવિરામ (comma) પ્રક્રિયકનો ઉપયોગ નિર્ણય માળખાં અને નિયંત્રણ માળખાંમાં પણ કરવામાં આવે છે જેની ચર્ચા હવે પછીના પ્રકરણોમાં કરવામાં આવી છે.

ઘટકનો સંગ્રહ કરવા માટે જરૂરી બાઈટની સંખ્યા પરત કરવા માટે sizeof( ) નામના એક વિશિષ્ટ પ્રક્રિયકનો ઉપયોગ કરવામાં આવે છે. ઉદાહરણ તરીકે, size = *sizeof(int)* વિધાન "size" ચલમાં "4" કિંમતનો સંગ્રહ કરશે. કારણ કે int ડેટા પ્રકાર મેમરીની 4 બાઈટ જેટલી જગ્યાનો ઉપયોગ કરે છે.

અલ્પવિરામ (comma) પ્રક્રિયક અને sizeof( ) પ્રક્રિયકનો ઉપયોગ દર્શાવતો પ્રોગ્રામ આકૃતિ 11.16માં આપેલ છે તથા તેનું પરિણામ 11.17માં દર્શાવવામાં આવ્યું છે.



આકૃતિ 11.16 : વિશિષ્ટ પ્રક્રિયકોનો ઉપયોગ દર્શાવતો પ્રોગ્રામ



આકૃતિ 11.17 : ઉદાહરણ 11.8નું પરિણામ

સમજૂતી (Explanation)

અહીં, પ્રથમ વિધાન ત્રણ અપૂર્ણ (float) ચલની ઘોષણા કરે છે તથા બીજું વિધાન એક પૂર્ણાંક ચલને ઘોષિત કરે છે. અમલ કરી શકાય તેવા ત્રીજા વિધાનમાં અલ્પવિરામ (comma) પ્રક્રિયકનો ઉપયોગ કરી પદાવલિ લખવામાં આવી છે. આપેલ વિધાન આ મુજબ કાર્ય કરશે : પ્રથમ જમણી બાજુની પદાવલિનું મૂલ્યાંકન કરવામાં આવશે. પ્રથમ પ્રક્રિયામાં "first" ચલને 125 કિંમત આપવામાં આવશે, બીજી પ્રક્રિયામાં "second" ચલને 15 કિંમત આપવામાં આવશે. અને ત્રીજી પ્રક્રિયામાં "first" નો "second" વડે ભાગાકાર કરવામાં આવશે. અંતમાં, ભાગાકારનું પરિણામ "result" ચલમાં સંગ્રહવામાં આવશે. ત્યાર પછીનું વિધાન "first", "second" અને "result" ચલની કિંમત દર્શાવે છે. ત્યાર પછીના વિધાનમાં "result" ચલમાં આવેલ કિંમતને મેમરીમાં આપવામાં આવેલી જગ્યાનો "size" નામના ચલમાં સંગ્રહ કરવામાં આવશે.

પદાવલિનું મૂલ્યાંકન (Evaluation of Expression)

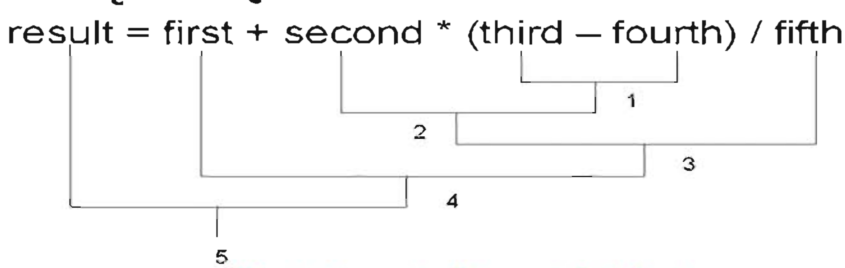
અત્યાર સુધીમાં આપણે સી ભાષામાં ઉપલબ્ધ પ્રક્રિયકોનો અભ્યાસ કરી તેનો પદાવલિમાં કેવી રીતે ઉપયોગ કરવો તે શીખ્યા. હવે, પદાવલિનું મૂલ્યાંકન કેવી રીતે કરવામાં આવે છે તે જોઈએ. નીચેના વિધાનમાં રહેલ પદાવલિ જુઓ :

*result = first + second * third – fourth;*

અહીં, પ્રથમ 'second' અને 'third' ચલની કિંમતોનો ગુણાકાર કરવામાં આવશે. આ ગુણાકારની કિંમતને ત્યાર પછી first ચલની કિંમતમાં ઉમેરવામાં આવશે અને આ મધ્યવર્તી કિંમતમાંથી fourthની કિંમત બાદ કરવામાં આવશે. જો પદાવલિમાં એક જ અગ્રતા ધરાવતા બે પ્રક્રિયકોનો ઉપયોગ કરવામાં આવ્યો હશે તો તેમનું મૂલ્યાંકન ડાબીથી જમણી તરફ કરવામાં આવશે. કૌંસનો ઉપયોગ કરી આ ક્રમને બદલી શકાય છે.

*result = first + second * (third – fourth) / fifth;*

હવે અહીં third - fourth નું મૂલ્યાંકન સૌપ્રથમ કરવામાં આવશે, મધ્યવર્તી કિંમતનો second ચલની કિંમત સાથે ગુણાકાર કરવામાં આવશે, પછી fifth ચલની કિંમત વડે ભાગાકાર કરવામાં આવશે અને અંતમાં first ચલની કિંમતમાં તેને ઉમેરવામાં આવશે. આકૃતિ 11.18 જુઓ.



આકૃતિ 11.18 : પદાવલિના મૂલ્યાંકનનો ક્રમ

પ્રક્રિયકોની અગ્રતા (Priority of Operators)

અત્યાર સુધી જોયેલા પ્રક્રિયકોને તેમની નિશ્ચિત અગ્રતા આપવામાં આવી છે. ઉદાહરણ result = first + second * (third – fourth) / fifth; વિધાનમાં જોઈ શકાય છે કે * ને +, – કે / કરતાં વધુ અગ્રતા આપવામાં આવી હતી. સી ભાષાના સર્જકોએ આ અગ્રતા પૂર્વનિશ્ચિત કરેલી છે. આમાંના કેટલાંક પ્રક્રિયકોની અગ્રતા કોષ્ટક 11.9માં દર્શાવી છે. પરિશિષ્ટ III માં આ માટેની વિસ્તૃત સમજ આપવામાં આવી છે.

Operator	Operation Used for	Associativity	Precedence
( )	Function call	Left to Right	First
[ ]	Array expression		
++	Increment	Right to Left	Second
- -	Decrement		
sizeof( )	Size of operand		
*	Multiplication	Left to Right	Third
/	Division		
%	Modulo division		
+	Binary addition	Left to Right	Fourth
-	Binary subtraction		

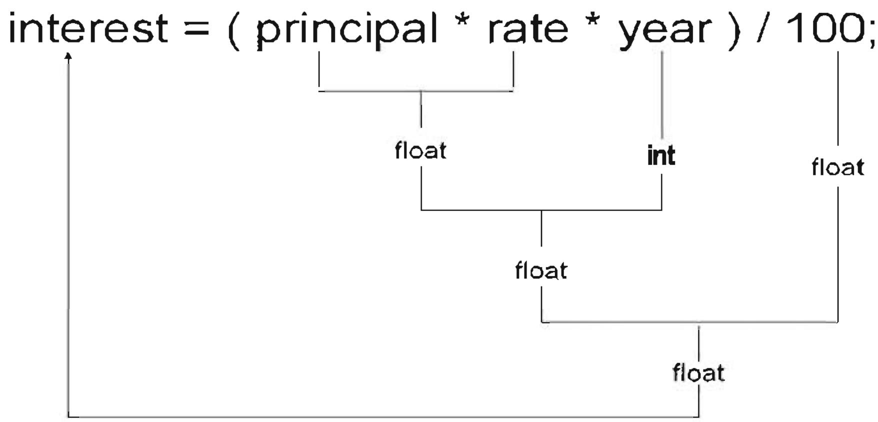
કોષ્ટક 11.9 : પ્રક્રિયકોની અગ્રતા

પ્રકાર બદલી (Type Conversion)

પદાવલિમાં આવેલ તમામ ઓપરેન્ડ એક જ પ્રકારના હોય તો જ તેનું મૂલ્યાંકન કરવામાં આવે છે. આ લાક્ષણિકતાને કારણે પદાવલિના મૂલ્યાંકન માટે તેમાં આવેલા ડેટા પ્રકારની આંતરિક બદલી જરૂરી બને છે. ઉદાહરણ તરીકે નીચેનો કોડ જુઓ :

```
int year=2;
float principal, rate, interest;
interest = (principal * rate * year) / 100;
```

આ પદાવલિનું મૂલ્યાંકન આકૃતિ 11.19માં દર્શાવ્યા મુજબ કરવામાં આવશે :

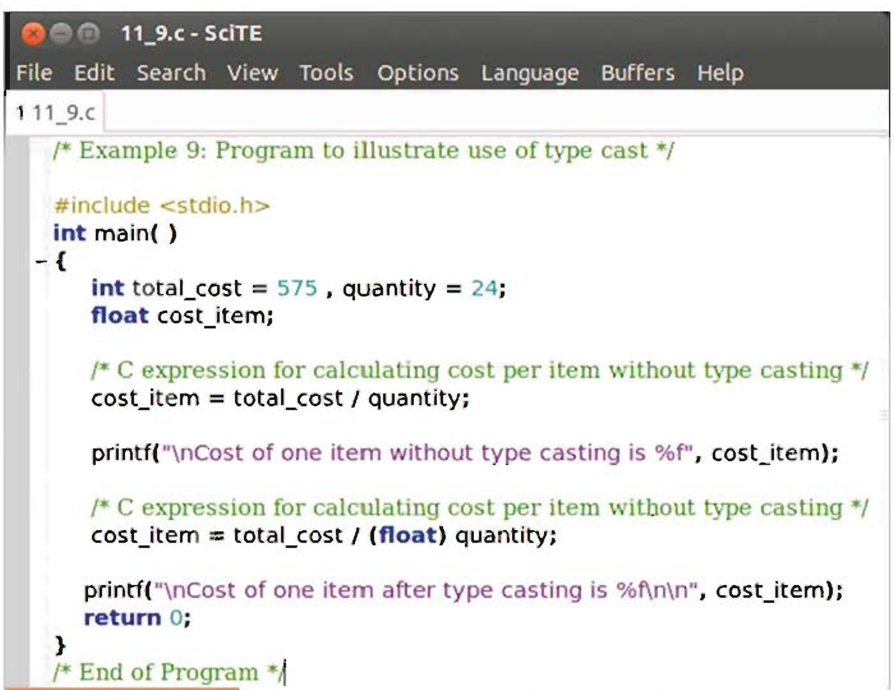


આકૃતિ 11.19 : પ્રકાર બદલી

આંતરિક ફેરબદલીને ઉપરવટ (override) થવા માટે ટાઇપ કાસ્ટિંગ (type casting) નામની પ્રક્રિયાનો ઉપયોગ કરી શકાય છે. કોઈ પણ કિંમતનો પ્રકાર બદલવા (type cast) માટે નીચે આપેલ વાક્યરચનાનો ઉપયોગ કરવામાં આવે છે.

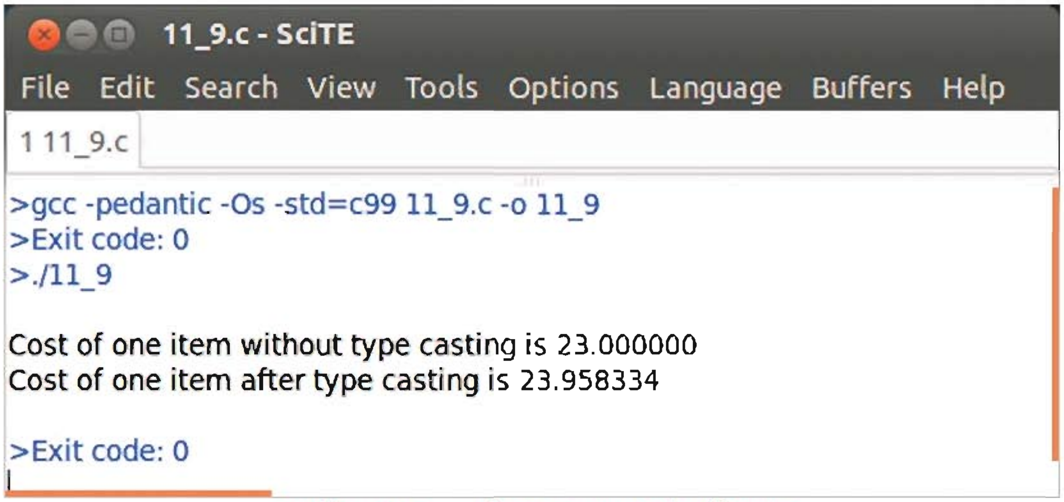
(data type) variable અથવા *(datatype) constant*

ટાઇપ કાસ્ટિંગનો ઉપયોગ કરી બનાવવામાં આવેલા પ્રોગ્રામનું કોડ લિસ્ટિંગ આકૃતિ 11.20 માં આપેલ છે. આકૃતિ 11.21 પ્રોગ્રામનું પરિણામ દર્શાવે છે.




```
/* Example 9: Program to illustrate use of type cast */

#include <stdio.h>
int main( )
{
    int total_cost = 575 , quantity = 24;
    float cost_item;

    /* C expression for calculating cost per item without type casting */
    cost_item = total_cost / quantity;

    printf("\nCost of one item without type casting is %f", cost_item);

    /* C expression for calculating cost per item without type casting */
    cost_item = total_cost / (float) quantity;

   printf("\nCost of one item after type casting is %f\n\n", cost_item);
   return 0;
}
/* End of Program */
```

આકૃતિ 11.20 : ટાઇપ કાસ્ટિંગ દર્શાવતો પ્રોગ્રામ



આકૃતિ 11.21 : ઉદાહરણ 11.9નું પરિણામ

સમજૂતી (Explanation)

આ પ્રોગ્રામમાં 24 વસ્તુઓની કિંમત આપેલી છે અને વસ્તુની એકમ કિંમત શોધવાની છે. "total_cost" અને "quantity" ચલને પૂર્ણાંક તથા "cost_item" ચલને અપૂર્ણાંક તરીકે વ્યાખ્યાયિત કરવામાં આવ્યા છે. એ જોઈ શકાય છે કે બંને ઑપરેન્ડ પૂર્ણાંક હોવાને કારણે પ્રથમ સી પદાવલિ વસ્તુની એકમ કિંમત તરીકે 23.000000 દર્શાવે છે. આ જ પદાવલિનો જ્યારે ટાઇપ કાસ્ટિંગ સાથે ઉપયોગ કરવામાં આવે છે ત્યારે તે ઇચ્છિત પરિણામ 23.958334 દર્શાવે છે.

સંગ્રાહક વર્ગો (Storage Classes)

સી પ્રોગ્રામમાં ઉપયોગમાં લેવામાં આવતા ચલનો સંગ્રહ સામાન્ય રીતે કમ્પ્યૂટરની પ્રાઇમરી મેમરીમાં કરવામાં આવે છે. કેટલાક ચલનો રજિસ્ટરમાં સંગ્રહ કરવો પણ શક્ય છે. સી ભાષા automatic, external, register, અને static નામના ચાર સંગ્રાહક વર્ગો (storage classes) પૂરાં પાડે છે. આ સંગ્રાહક વર્ગો દ્વારા ચલનો સંગ્રહ કરવા માટેનું સ્થાન નિશ્ચિત કરવામાં આવે છે. તમામ સંગ્રાહક વર્ગોની એક પછી એક ચર્ચા કરીએ.

ઑટોમેટિક ચલ (Automatic Variables)

તમામ ચલનો પૂર્વ નિર્ધારિત સંગ્રાહક વર્ગ 'ઑટોમેટિક' હોય છે. ચલને 'ઑટોમેટિક' સ્વરૂપમાં નિશ્ચિત રૂપે (explicitely) વ્યાખ્યાયિત કરવા માટે auto કી-વર્ડનો ઉપયોગ કરવામાં આવે છે. ચલ માટે ઓટોમેટિક સંગ્રાહક વર્ગ વ્યાખ્યાયિત કરવા માટેની વાક્યરચના નીચે મુજબ છે :

auto data type identifier;

ઉદાહરણ તરીકે, auto int number; વ્યાખ્યા દર્શાવે છે કે ચલ number એ પૂર્ણ પ્રકારનો છે અને તેનો સંગ્રાહક વર્ગ ઓટોમેટિક છે. જ્યારે કોઈ વિધેયનો અમલ કરવામાં આવે છે (called) ત્યારે આ પ્રકારના ચલની રચના કરવામાં આવે છે તથા જ્યારે આ વિધેય બોલાવનાર (caller) વિધેયને નિયંત્રણ પરત મોકલે છે ત્યારે આ પ્રકારના ચલનો નાશ કરવામાં આવે છે. આ પ્રકારના ચલને શરૂઆતમાં પૂર્વનિર્ધારિત રીતે અનિશ્ચિત કિંમત (garbage value - એવી કિંમત કે ઉપયોગકર્તા માટે જેનું કોઈ મહત્ત્વ નથી) આપવામાં આવે છે અને તેનો પ્રાઇમરી મેમરીમાં સંગ્રહ કરવામાં આવે છે.

એક્સ્ટર્નલ ચલ (External Variable)

કેટલીકવાર પ્રોગ્રામરને કોઈ ચલ બે વિધેય કે પ્રોગ્રામ વચ્ચે વહેંચવાની જરૂર પડે છે. બે જુદા જુદા વિધેય કે પ્રોગ્રામ વચ્ચે ચલની વહેંચણી માટે 'એક્સ્ટર્નલ' સંગ્રાહક વર્ગનો ઉપયોગ કરી શકાય છે. આ માટે એ જરૂરી છે કે ઓછામાં ઓછા એક પ્રોગ્રામમાં તે ચલ વૈશ્વિક (global) ચલ તરીકે ઘોષિત થયેલો હોવો જોઈએ. નીચેની વાક્યરચના દ્વારા ચલ માટે એક્સ્ટર્નલ સંગ્રાહક વર્ગ વ્યાખ્યાયિત કરી શકાય.

extern datatype identifier;

ઉદાહરણરૂપે, extern char choice; વ્યાખ્યા દર્શાવે છે કે choice ચલ અક્ષર પ્રકારનો છે અને તેનો સંગ્રાહક વર્ગ એક્સ્ટર્નલ છે. એક્સ્ટર્નલ ચલની પૂર્વ નિર્ધારિત કિંમત શૂન્ય છે અને તેનો પ્રાઇમરી મેમરીમાં સંગ્રહ કરવામાં આવે છે.

રજિસ્ટર ચલ (Register Variable)

ઝડપી ઉપયોગ કરવા માટે ચલને CPU રજિસ્ટરમાં સંગ્રહી શકાય છે. સી ભાષા આ માટે 'register' નામનો સંગ્રાહક વર્ગ પૂરો પાડે છે. નીચેની વાક્યરચના દ્વારા ચલ માટે રજિસ્ટર સંગ્રાહક વર્ગ વ્યાખ્યાયિત કરી શકાય છે :

register datatype identifier;

ઉદાહરણ રૂપે, register int counter; વ્યાખ્યા દર્શાવે છે કે counter ચલ પૂર્ણાંક પ્રકારનો છે અને તેનો સંગ્રાહક વર્ગ રજિસ્ટર છે. પૂર્વ નિર્ધારિત રીતે આ પ્રકારના ચલમાં અનિશ્ચિત કિંમત (garbage value) નો ઉપયોગ કરવામાં આવે છે.

સ્ટેટિક ચલ (Static Variable)

જ્યારે ચલનો 'સ્ટેટિક' સંગ્રાહક વર્ગ સાથે ઉપયોગ કરવામાં આવે છે ત્યારે નિશ્ચિત વિસ્તાર માટે તે ચલની કિંમત કાયમી બનાવી શકાય છે. સ્ટેટિક સંગ્રાહક વર્ગ ધરાવતા ચલની કિંમતને પ્રાઈમરી મેમરીમાં કાયમી ધોરણે સાચવવામાં આવે છે તથા પૂર્વનિર્ધારિત રીતે તેમાં શૂન્ય કિંમત મૂકવામાં આવે છે. ચલનો સ્ટેટિક સંગ્રાહક વર્ગ વ્યાખ્યાયિત કરવા માટે નીચે આપેલ વાક્યરચનાનો ઉપયોગ કરવામાં આવે છે :

static datatype identifier;

ઉદાહરણ રૂપે, static int max; વ્યાખ્યા દર્શાવે છે કે max ચલ પૂર્ણાંક પ્રકારનો છે અને તેનો સંગ્રાહક વર્ગ સ્ટેટિક છે.

સારાંશ

આ પ્રકરણમાં આપણે મૂળભૂત ડેટા પ્રકારો int, float, char અને void વિશે અભ્યાસ કર્યો. typedef, enum અને array જેવા ઉપયોગકર્તા દ્વારા નિર્મિત અને તારવેલા ડેટા પ્રકારો વિશે પણ માહિતી મેળવી. નિરૂપક પ્રક્રિયક દ્વારા ચલમાં કેવી રીતે કિંમત ઉમેરવી તે પણ આપણે જોયું. ત્યાર પછી આપણે ગાણિતિક, સંબંધસૂચક, વધારા-ઘટાડા સૂચક, શરતી, તાર્કિક, બિટવાઇઝ અને વિશિષ્ટ પ્રક્રિયકો વિશે અભ્યાસ કર્યો. જુદા જુદા પ્રક્રિયકોનો ઉપયોગ કરી પદાવલિની રચના કરતાં તથા તેનું મૂલ્યાંકન કરતાં પણ આપણે શીખ્યા. અંતમાં સી ભાષા દ્વારા પૂરા પાડવામાં આવતા ચાર સંગ્રાહક વર્ગો વિશે માહિતી મેળવી.

શિક્ષકોને સૂચના (Instruction for Teachers)

અહીં ચર્ચવામાં આવેલા તમામ ડેટા પ્રકારો ANSI C99 ધારાધોરણ આધારિત છે. અહીં આપવામાં આવેલાં ઉદાહરણ પ્રક્રિયકોને સમજવા માટે છે. શિક્ષક વધુ સ્વાધ્યાય બનાવી વિદ્યાર્થીઓને આપી શકે છે.

સ્વાધ્યાય

1. ડેટા પ્રકાર એટલે શું ? તેનું મહત્ત્વ જણાવો.
2. void ડેટા પ્રકારનું મહત્ત્વ જણાવો.
3. ઉપયોગકર્તા દ્વારા વ્યાખ્યાયિત ડેટા પ્રકારનું મહત્ત્વ જણાવો.
4. તારવેલા ડેટા પ્રકારનું મહત્ત્વ જણાવો.
5. નીચેના વિધાનો ખરાં છે કે ખોટાં તે જણાવો :
   (a) char ડેટા પ્રકાર માટે બે બાઇટ જગ્યા જરૂરી છે.
   (b) ચિહ્નરહિત (unsigned) પૂર્ણાંક ચલ શૂન્યથી ઓછી સંખ્યાનો સંગ્રહ કરી શકે છે.
   (c) double ડેટા પ્રકારનો ઉપયોગ પૂર્ણાંક સંખ્યાઓનો સંગ્રહ કરવા માટે છે.
   (d) int ડેટા પ્રકારનો વિસ્તાર વધારવા માટે તેની આગળ long કી-વર્ડ ઉમેરવામાં આવે છે.
   (e) અપૂર્ણાંક સંખ્યાઓનું વૈજ્ઞાનિક સ્વરૂપ exponent * 10 mantissa તરીકે રજૂ કરવામાં આવે છે.

6. આપેલા વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરો :

(1) પૂર્ણાંક ડેટા પ્રકાર માટે નીચેનામાંથી કયા કી-વર્ડનો ઉપયોગ કરવામાં આવે છે ?

(a) integer (b) Integer (c) INTEGER (d) int

(2) નીચેનામાંથી કયો વિકલ્પ ખાલી કિંમતનો નિર્દેશ કરે છે ?

(a) void (b) Void (c) char (d) float

(3) gcc કંપાઇલર દ્વારા float પ્રકારને કયા કદની મેમરી આપવામાં આવે છે ?

(a) 1 (b) 2 (c) 4 (d) 6

(4) સી ભાષામાં એક અક્ષર વ્યાખ્યાયિત કરવા માટે નીચેનામાંથી કયો કી-વર્ડ યોગ્ય છે ?

(a) char (b) character (c) CHAR (d) CHARACTER

(5) સી ભાષામાં નીચેનામાંથી કયો ડેટા પ્રકાર ઉપયોગકર્તા દ્વારા નિર્મિત છે ?

(a) int (b) enum (c) char (d) float

(6) સી ભાષામાં આવેલા ડેટા પ્રકારને ઉપનામ આપવા માટે નીચેનામાંથી કયા કી-વર્ડનો ઉપયોગ કરવામાં આવે છે ?

(a) enum (b) def (c) pointer (d) typedef

(7) સી ભાષામાં "&" નિશાની કયા પ્રકારનો પ્રક્રિયક છે ?

(a) સંબંધસૂચક (b) ગાણિતિક (c) તાર્કિક (d) બિટવાઇઝ

(8) સી ભાષામાં "!" નિશાની કયા પ્રકારનો પ્રક્રિયક છે ?

(a) સંબંધસૂચક (b) ગાણિતિક (c) તાર્કિક (d) બિટવાઇઝ

(9) સી ભાષામાં "=" નિશાનીનો ઉપયોગ કયા પ્રક્રિયક તરીકે કરવામાં આવે છે ?

(a) સમાનતા (b) નિરૂપણ (c) નોંધ (d) સમીકરણ

(10) સી ભાષામાં value++ નો અર્થ શું કરી શકાય?

(a) પોસ્ટ ઇન્ક્રિમેન્ટ (b) પોસ્ટ ડિક્રિમેન્ટ (c) પ્રી-ઇન્ક્રિમેન્ટ (d) પ્રી ડિક્રિમેન્ટ

પ્રાયોગિક સ્વાધ્યાય

1. આપેલ મીટરને મિલીમીટરમાં ફેરવવા માટેનો સી પ્રોગ્રામ લખો.
2. c = (f–32) * 5/9 સમીકરણનો ઉપયોગ કરી આપેલ ફેરનહીટને સેલ્સિયસમાં ફેરવવા માટેનો સી પ્રોગ્રામ લખો.
3. ત્રણ સંખ્યાઓની સરેરાશ શોધવા માટેનો સી પ્રોગ્રામ લખો.
4. 2000 ચોમી ના વિસ્તારમાં 50 ચો સેમીની એક એવી કુલ કેટલી લાદી જડી શકાય તે શોધવા માટેનો પ્રોગ્રામ લખો.
5. લંબચોરસનું ક્ષેત્રફળ શોધવા માટેનો સી પ્રોગ્રામ લખો.
6. આપેલ સંખ્યા અન્ય આપેલ સંખ્યા સાથે નિ:શેષ ભાજ્ય છે કે નહીં તે શોધવા માટેનો સી પ્રોગ્રામ લખો.
7. ₹ P ની લોન R% વ્યાજના દર સાથે N વર્ષો માટે લેવામાં આવી હોય તો તે પરથી ચક્રવૃદ્ધિ વ્યાજ શોધવા માટેનો સી પ્રોગ્રામ લખો. P, R અને Nની કિંમતો વિદ્યાર્થી દ્વારા આપવામાં આવશે.
8. ઘનનું ઘનફળ શોધવા માટેનો સી પ્રોગ્રામ લખો.
9. વર્તુળનું ક્ષેત્રફળ શોધવા માટેનો સી પ્રોગ્રામ લખો.
10. ₹ P ની લોન R% વ્યાજના દર સાથે N વર્ષો માટે લેવામાં આવી હોય તો તે પરથી સાદું વ્યાજ શોધવા માટેનો સી પ્રોગ્રામ લખો. P, R અને Nની કિંમતો વિદ્યાર્થી દ્વારા આપવામાં આવશે.

નિવેશ (Input) / નિર્ગમ (Output) પ્રક્રિયાઓનો ઉપયોગ

તમામ પ્રોગ્રામિંગ ભાષાઓ નિવેશ કરવામાં આવેલી વિગતો વાંચવાની સુવિધા આપે છે તેને ઇનપુટ ઑપરેશન (input operation) કહે છે તથા પરિણામ દર્શાવવાની સુવિધા આપે છે જેને આઉટપુટ ઑપરેશન (output operation) તરીકે ઓળખવામાં આવે છે. અહીં 'નિવેશ' (input) એટલે કી-બોર્ડ જેવા કોઈ ઇનપુટ સાધન તરફથી મળેલ ડેટા વાંચવો તથા આ 'નિર્ગમ' (output) એટલે મૉનિટર, પ્રિન્ટર જેવા આઉટપુટ સાધન પર ડેટાને દર્શાવવો. આ ઇનપુટ અને આઉટપુટ I/O પ્રક્રિયાઓ (I/O operations) નામથી વધુ પ્રચલિત છે. ઇનપુટ અને આઉટપુટ ક્રિયાઓના અમલ માટે સી ભાષા પાસે કોઈ આંતરસ્થાપિત વિધાન નથી. ડેટાના ઇનપુટ અને આઉટપુટ માટેની પ્રક્રિયાઓ પ્રમાણભૂત ઇનપુટ/આઉટપુટ આંતરપ્રસ્થાપિત લાઇબ્રેરી પ્રોગ્રામ દ્વારા પાર પાડવામાં આવે છે. આ પ્રકરણમાં આપણે getchar(), getch(), gets(), scanf(), printf(), putchar(), putc(), puts() વગેરે વિધેય દ્વારા I/O પ્રક્રિયાઓ કેવી રીતે પાર પાડી શકાય તે વિશે ચર્ચા કરીશું. આપણે સુગ્રથિત (formatted) ઇનપુટ અને આઉટપુટ ક્રિયાઓ વિશે પણ અભ્યાસ કરીશું.

પ્રોગ્રામમાં ડેટા પર પ્રક્રિયા કરવા માટે ચલનો ઉપયોગ કરવામાં આવે છે. ચલને ઇનપુટ આપવા માટેની મુખ્ય બે રીત છે. એક રીતમાં એસાઇનમેન્ટ ઑપરેટર(assignment operator)નો ઉપયોગ કરવામાં આવે છે. ઉદાહરણ તરીકે, number=5; વિધાન number ચલમાં 5 સંખ્યાનો સંગ્રહ કરશે. બીજી રીત મુજબ પ્રોગ્રામનો અમલ કરતી વખતે ઉપયોગકર્તાએ આપેલ ડેટા વાંચવામાં આવશે. આ માટે ઇનપુટ પ્રક્રિયા માટેના કોઈ આંતરસ્થાપિત વિધેયનો ઉપયોગ કરી શકાય છે. હવે આપણે સામાન્ય રીતે ઉપયોગમાં લેવામાં આવતા આંતરસ્થાપિત ઇનપુટ વિધેય વિશે ચર્ચા કરીએ.

આંતરસ્થાપિત ઇનપુટ વિધેય (Inbuilt input function)

કી-બોર્ડ, માઉસ વગેરે જેવાં ઇનપુટ સાધનો દ્વારા પ્રોગ્રામને ડેટા ઇનપુટ કરવો શક્ય છે. સી ભાષા ઇનપુટ સંબંધિત ઘણાં વિધેય પૂરાં પાડે છે, જેનો સી લાઇબ્રેરી દ્વારા સંગ્રહ થયેલો હોય છે. સી લાઇબ્રેરીમાં આવેલા કોઈ પણ આંતરસ્થાપિત વિધેયનો ઉપયોગ કરવા માટે, **#include** વિધાનનો ઉપયોગ કરી પ્રૉગ્રામની શરૂઆતમાં જે-તે લાઇબ્રેરી ફાઈલ ઉમેરવી જરૂરી છે. આપણે ઘણા પ્રોગ્રામમાં પ્રોગ્રામની શરૂઆતમાં નીચેના વિધાનોનો ઉપયોગ કર્યો હતો તે તમે નોંધ્યું હશે :

#include<stdio.h>

stdio.hનો અર્થ છે standard input-output header ફાઈલ. ઇનપુટ અને આઉટપુટ પ્રક્રિયાઓને સંબંધિત ઘણા વિધેયનો સમાવેશ આ હેડર ફાઈલમાં કરવામાં આવ્યો છે. #include વિધાન stdio.h ફાઈલને શોધી તેની વિગતોને પ્રોગ્રામની શરૂઆતમાં ઉમેરવાની કમ્પાઇલરને સૂચના આપે છે. ત્યારપછી આ હેડર ફાઈલની વિગતો પ્રોગ્રામનો એક ભાગ બની જાય છે. સી લાઇબ્રેરી વિધેય સંબંધિત વધુ ચર્ચા સી વિધેયના પ્રકરણમાં કરવામાં આવી છે. અહીં એ નોંધ જરૂરી છે કે, જ્યારે scanf() વિધેયનો ઉપયોગ કરવામાં આવે ત્યારે કેટલાક કમ્પ્યૂટરોમાં આ હેડર ફાઈલ ઉમેરવી જરૂરી હોતી નથી. હવે નીચેના વિભાગમાં getchar(), getch(), getc() અને gets() વિધેય વિશે ચર્ચા કરીએ.

getchar()

સી ભાષામાં પ્રોગ્રામનો અમલ કરતી વખતે એક અક્ષર વાંચવા માટેનો સૌથી સરળ માર્ગ getchar() વિધેયનો ઉપયોગ છે. getchar() વિધેયનું પ્રમાણભૂત સ્વરૂપ નીચે દર્શાવ્યું છે :

variable_name = getchar();

variable_name એ char ડેટાટાઇપ ધરાવતો સી ભાષામાં માન્ય કોઈ પણ ચલ છે. getchar() વિધેયને કોઇ પ્રાચલ (parameter) આપવામાં આવતા નથી. જ્યારે તેનો અમલ કરવામાં આવે ત્યારે તે એક અક્ષર વાંચે છે. આ વિધેય int પરત કરે છે, જે આપેલ અક્ષરનો આસ્કી કોડ હોય છે, પરંતુ આપણે ઉપયોગકર્તાએ આપેલ ઇનપુટનો char ચલમાં સંગ્રહ કરી શકીએ છીએ.

હવે આપણે ઉદાહરણ 12.1નો ઉપયોગ કરી getchar() વિધેયના ઉપયોગને સમજવાનો પ્રયત્ન કરીએ. અહીં આપણે સંદેશ દર્શાવવા માટે ઉપયોગકર્તા દ્વારા લખવામાં આવેલ ઉત્તરનો ઉપયોગ કરવા ઇચ્છીએ છીએ. ઉદાહરણ 12.1નું કોડ લિસ્ટિંગ આકૃતિ 12.1માં આપવામાં આવ્યું છે તથા આકૃતિ 12.2 તેનું પરિણામ દર્શાવે છે.

```
9-1.c - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
9-1.c

/* Example 1: Program to demonstrate use of getchar() function. */

#include<stdio.h>
int main()
{
	char answer;

	printf("Would you like to study? \n");
	printf("Enter 'Y' for yes and 'N' for no: ");

	answer = getchar();

	if(answer =='Y' || answer =='y')
		printf("Go to study room...\n");
	else
		printf("Go to play ground...\n");

	return 0;
}
// End of Program
```

આકૃતિ 12.1 : ઉદાહરણ 12.1નું કોડ લિસ્ટિંગ

```
kpp@ubuntu: ~
File  Edit  View  Terminal  Help
kpp@ubuntu:~$ gcc 9-1.c
kpp@ubuntu:~$ ./a.out
Would you like to study?
Enter 'Y' for yes and 'N' for no: Y
Go to study room...
kpp@ubuntu:~$
```

આકૃતિ 12.2 : ઉદાહરણ 12.1નું પરિણામ

સમજૂતી (Explanation)

ઉદાહરણ 12.1નું પ્રથમ વિધાન એક અક્ષરનો સંગ્રહ કરવા માટે answer નામના ચલને ઘોષિત કરે છે. getchar() વિધેયનો અમલ કરવામાં આવે ત્યારે પ્રોગ્રામ ઉપયોગકર્તા દ્વારા કોઈ કી દબાવવાની પ્રતીક્ષા કરે છે. ઉપયોગકર્તાએ ઉમેરેલો અક્ષર answer ચલમાં સંગ્રહવામાં આવે છે અને તે અક્ષર સ્ક્રીન ઉપર પણ દર્શાવવામાં આવે છે. ઉદાહરણ તરીકે, જો ઉપયોગકર્તા 'Y' કે 'y' ઉમેરે તો "Go to study room..." સંદેશ દર્શાવવામાં આવશે. જો ઉપયોગકર્તા અન્ય કોઈપણ અક્ષર ઉમેરે તો "Go to playground..." સંદેશ દર્શાવવામાં આવશે.

getch()

getch() વિધેયનો ઉપયોગ પણ એક અક્ષર મેળવવા માટે કરી શકાય છે. getchar() અને getch() વચ્ચેનો તફાવત એ છે કે, અહીં ઉમેરવામાં આવેલો અક્ષર સ્ક્રીન પર દર્શાવવામાં આવતો નથી. ઉપયોગકર્તા દ્વારા ટાઈપ કરવામાં આવેલ અક્ષર સ્ક્રીન પર દર્શાવવો ન હોય ત્યારે આ વિધેયનો ઉપયોગ કરવામાં આવે છે. ઉદાહરણ 12.1માં ઉમેરેલ getchar() વિધેયને getch() સાથે બદલી બંને વિધેય વચ્ચેના તફાવતનું અવલોકન કરો.

getc()

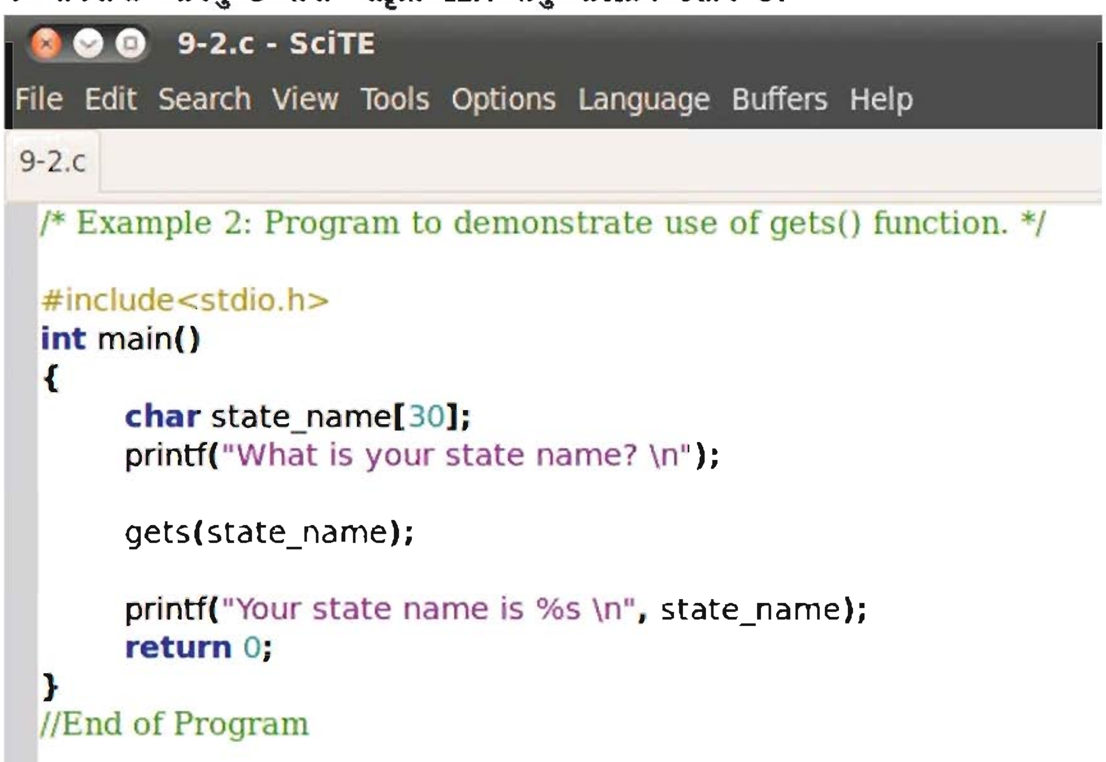
getchar() અને getc(), વિધેયની જેમ getc() વિધેયનો ઉપયોગ પણ એક અક્ષર મેળવવા માટે કરવામાં આવે છે. પરંતુ અહીં તફાવત એ છે કે getc() પ્રમાણભૂત ઇનપુટ સાધનને બદલે ફાઈલમાં રહેલ અક્ષર વાંચે છે. getc() સંબંધિત વધુ ચર્ચા આ પાઠ્યક્રમની મર્યાદા બહાર છે.

gets()

એક સમયે એક જ અક્ષર મેળવી શકાય તે માટેના getchar(), getch() અને getc() વિધેયનો આપણે અભ્યાસ કર્યો. બે અક્ષરો મેળવવા હોય તો getchar() વિધેયનો બે વખત અમલ કરી શકાય. પરંતુ અક્ષરોનું જૂથ મેળવવાની જરૂર હોય ત્યારે getchar() વિધેયનો અનેકવાર અમલ કરવો એ યોગ્ય તર્ક નથી. અક્ષરોની હારમાળા (string) મેળવવા માટે gets() વિધેયનો ઉપયોગ વધુ સારો વિકલ્પ છે. gets() વિધેયની સામાન્ય વાક્યરચના નીચે મુજબ છે :

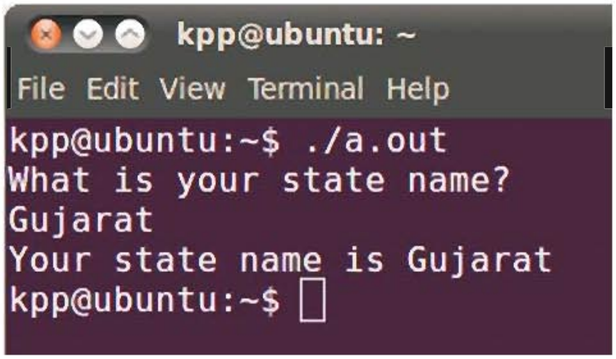
gets(variable_name);

gets() વિધેય કિંમત તરીકે એક સ્ટ્રિંગ સ્વીકારે છે. અહીં variable_name એ અક્ષરોનો એરે (character array) છે. અક્ષરોના એરે વિશેની વધુ ચર્ચા 'એરે' પ્રકરણમાં કરવામાં આવી છે. પ્રોગ્રામના અમલીકરણ દરમ્યાન gets() વિધેયનો અમલ કરવામાં આવે ત્યારે તે ઇનપુટ સાધનનો ઉપયોગ કરી ઉપયોગકર્તાએ ઉમેરેલા અક્ષરોની પ્રતીક્ષા કરે છે. ઉપયોગકર્તા દ્વારા એન્ટર (new line character) દબાવવામાં ન આવે ત્યાં સુધી અક્ષરો મેળવવામાં આવે છે અને ત્યાર પછી તેના અંત ભાગમાં 'નલ' અક્ષર ('\0') ઉમેરી સ્ટ્રિંગ પૂર્ણ કરવામાં આવે છે. આ વિધેયના અમલ બાદ તમામ અક્ષરો અને અંતમાં રહેલ નલ કિંમતનો variable_name ચલમાં સંગ્રહ કરવામાં આવે છે. ઉદાહરણ 12.2નો ઉપયોગ કરી gets() વિધેયની કાર્યપદ્ધતિ સમજવાનો પ્રયત્ન કરીએ. આકૃતિ 12.3માં ઉદાહરણ 12.2નું કોડ-લિસ્ટિંગ આપવામાં આવેલું છે તથા આકૃતિ 12.4 તેનું પરિણામ દર્શાવે છે.




```c
/* Example 2: Program to demonstrate use of gets() function. */

#include<stdio.h>
int main()
{
	char state_name[30];
	printf("What is your state name? \n");

	gets(state_name);

	printf("Your state name is %s \n", state_name);
	return 0;
}
//End of Program
```

આકૃતિ 12.3 : ઉદાહરણ 12.2નું કોડ-લિસ્ટિંગ



આકૃતિ 12.4 : ઉદાહરણ 12.2નું પરિણામ

ઉદાહરણ 12.2માં ઉપયોગકર્તાએ ઉમેરેલ ઇનપુટને state_name ચલમાં સંગ્રહવામાં આવશે. આ જ સ્ટ્રિંગને printf() વિધેય દ્વારા સ્ક્રીન પર દર્શાવવામાં આવશે. printf() વિધાન વિશે વિસ્તૃત ચર્ચા આ પ્રકરણમાં આગળ ઉપર કરવામાં આવી છે.

અહીં એ નોંધ લેવી જરૂરી છે કે, gets() વિધેય ધરાવતા કોઈ પણ પ્રોગ્રામના કમ્પાઇલેશન વખતે "the gets function is dangerous and should not be used" એ પ્રકારનો ચેતવણી-સંદેશ દર્શાવવામાં આવે છે. પરંતુ આ માત્ર ચેતવણી-સંદેશ હોવાથી પ્રોગ્રામનું અમલીકરણ ચાલુ રાખી શકાય છે.

સુગ્રથિત નિવેશ (Formatted input)

કેટલીકવાર પ્રોગ્રામમાં એવી સ્થિતિ ઉદ્‌ભવે છે જ્યારે અત્યાર સુધીમાં ચર્ચેલાં ઇનપુટ વિધેય વિશેષ ઉપયોગી બનતાં નથી. getchar(), getch() અને getc() વિધેયનો ઉપયોગ એક અક્ષર મેળવવા માટે કરવામાં આવે છે તથા ઉપયોગકર્તા એન્ટર કી ન દબાવે ત્યાં સુધી અનેક અક્ષરો મેળવવા માટે gets() વિધેયનો ઉપયોગ કરી શકાય છે. પરંતુ ઉપયોગકર્તા જો **'33 Mudra Ahmedabad'** આ પ્રકારનો ડેટા ઉમેરવા ઇચ્છતો હોય તો ?

અહીં ડેટાના પ્રથમ વિભાગમાં વિદ્યાર્થીનો અનુક્રમ તથા બીજા અને ત્રીજા વિભાગમાં અનુક્રમે વિદ્યાર્થીનું નામ અને શહેરનું નામ દર્શાવ્યું છે. અહીં એ જોઈ શકાય છે કે અનુક્રમ એ પૂર્ણાંક પ્રકારનો ડેટા છે તથા બાકીના બે ડેટામાં અક્ષરોનો સમાવેશ કરવામાં આવ્યો છે.

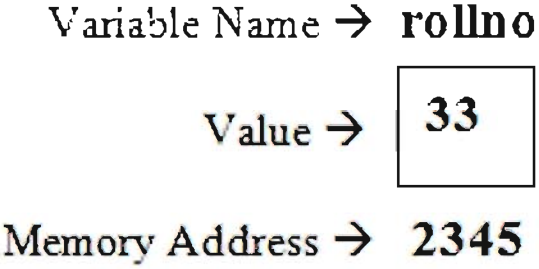
આ પ્રકારનો ડેટા મેળવવા માટે સી ભાષા સુગ્રથિત નિવેશ (formatted input) વિધેય નામની સુવિધા પૂરી પાડે છે. એક ફોર્મેટેડ ઇનપુટ વિધેય scanf()નો ઉપયોગ આપણે ઘણા પ્રોગ્રામોમાં કર્યો છે. scanf() વિધેય દ્વારા ફોર્મેટેડ ઇનપુટ માટે ઉપલબ્ધ વિવિધ વિકલ્પોનો અભ્યાસ કરીએ.

scanf()

scanf()નો અર્થ છે, **scan** formatted. આ વિધેય int, char, float વગેરે જેવા નિશ્ચિત સ્વરૂપમાં રહેલા ડેટા ઉમેરવાની સુવિધા આપે છે. scanf() વિધેય માટેની સામાન્ય વાક્યરચના નીચે આપેલી છે :

scanf("control string", &variable1, &variable2, ... &variableN);

જે ડેટા સ્વરૂપે ચલની કિંમતોનો સંગ્રહ કરવાનો હોય તેને કન્ટ્રોલ સ્ટ્રિંગ દ્વારા સ્પષ્ટ કરવામાં આવે છે. variable1, variable2, ... variableN એ ચલનાં નામ છે. ચલના નામની આગળ &(ampersand) નિશાની ઉમેરવામાં આવે છે તથા દરેક નામ અલ્પવિરામ (,) થી છૂટાં પાડવામાં આવે છે. સી ભાષામાં & નિશાનીને address of પ્રક્રિયક તરીકે ઓળખવામાં આવે છે. ઉપયોગકર્તા દ્વારા ઉમેરવામાં આવેલા ઇનપુટનો સંગ્રહ કરવા માટેનું સ્થાન તેના દ્વારા સ્પષ્ટ કરવામાં આવે છે. આ આકૃતિ 12.5 દ્વારા સમજી શકાશે.



આકૃતિ 12.5 : મેમરીની સંરચના (Memory Layout)

આકૃતિ 12.5માં દર્શાવ્યા મુજબ rollno ચલનું નામ છે તથા તેની કિંમત 33 છે. ચલની કિંમતનો વાસ્તવિક સંગ્રહ 2345 મેમરીસ્થાન પર થયો છે. (આ સંખ્યા માત્ર ઉદાહરણ છે; તે કમ્પ્યૂટરનું કોઈ પણ મેમરીસ્થાન હોઇ શકે છે.) સી ભાષામાં દરેક ચલને તેનાં નામ અને મેમરીસ્થાન દ્વારા વ્યાખ્યાયિત કરવામાં આવે છે. જ્યારે પ્રોગ્રામમાં ચલના નામનો ઉલ્લેખ કરવામાં આવે ત્યારે સી કમ્પાઇલર પ્રક્રિયા માટે ચલના મેમરીસ્થાનનો ઉપયોગ કરે છે. address of પ્રક્રિયકની વધુ ચર્ચા આ પુસ્તકની મર્યાદા બહાર છે. સી પૉઇન્ટરનો અભ્યાસ કરતી વખતે તમે આ અભિગમ શીખશો.

કન્ટ્રોલ સ્ટ્રિંગને **'ફૉર્મેટ સ્ટ્રિંગ' (format string)** તરીકે પણ ઓળખવામાં આવે છે. ઉપયોગકર્તાએ આપેલ ડેટા ઇનપુટનો અનુવાદ કરવા માટે તે કમ્પાઇલરને મદદરૂપ બને છે. ઇનપુટ દરમિયાન ચલની કિંમતોનો ક્રમ સ્પષ્ટ કરવાનું કાર્ય પણ કન્ટ્રોલ સ્ટ્રિંગનું છે. દરેક ફૉર્મેટમાં ડેટાટાઇપને સ્પષ્ટ કરતા અક્ષરની આગળ '%' નિશાની ઉમેરવામાં આવે છે. એક ઉદાહરણની મદદથી કન્ટ્રોલ સ્ટ્રિંગના ઉપયોગની સમજ મેળવીએ.

પૂર્ણાંક સંખ્યા વાંચવી (Reading Integers)

ઉપયોગકર્તા પાસેથી ત્રણ પૂર્ણાંક સંખ્યાઓ મેળવીને તેનો marks1, marks2 અને marks3 નામના ચલમાં સંગ્રહ કરવા માટે નીચે આપેલ પ્રોગ્રામખંડનો ઉપયોગ કરી શકાય.

```
int marks1, marks2, marks3;
scanf("%d %d %d", &marks1, &marks2, &marks3);
```

ઉપરના વિધાનના અમલ દરમિયાન જો ઉપયોગકર્તા 70 80 90 ઉમેરશે તો marks1 ચલમાં 70, marks2 ચલમાં 80 અને marks3 ચલમાં 90 સંખ્યાનો સંગ્રહ કરવામાં આવશે. અહીં %dની સાથે ફિલ્ડનું કદ (field width) પણ નીચે જણાવ્યા મુજબ સ્પષ્ટ કરી શકાય છે :

```
scanf("%2d %4d", &marks1, &marks2);
```

અગાઉનાં scanf() વિધાનના અમલ દરમિયાન ઉપયોગકર્તા જો 70 1234 સંખ્યાઓ ઉમેરે તો marks1માં 70 અને marks2માં 1234 સંખ્યાઓનો સંગ્રહ કરવામાં આવશે.

પરંતુ ઉપરના scanf() વિધાન માટે ઉપયોગકર્તા જો 1234 70 સંખ્યા ઉમેરશે તો marks1 ચલમાં 12 સંખ્યાનો સંગ્રહ કરવામાં આવશે. (કારણ કે, ફિલ્ડનું કદ %2d છે) તથા marks2 ચલમાં 34 સંખ્યાનો સંગ્રહ કરવામાં આવશે (જે 1234 સંખ્યાનો નહીં વંચાયેલ ભાગ છે). આ scanf() વિધાન માટે 70 કિંમત નિરર્થક બની રહેશે.

અપૂર્ણાંક સંખ્યા વાંચવી (Reading Real Numbers)

વિદ્યાર્થીએ મેળવેલ ટકા (percentage) જેવી અપૂર્ણાંક કિંમત પ્રોગ્રામમાં મેળવવા માટે નીચે આપેલ પ્રોગ્રામ-ખંડનો ઉપયોગ કરી શકાય :

```
float per1, per2;
scanf("%f %f", &per1, &per2);
```

અહીં per1 અને per2 નામના બે અપૂર્ણ ચલ વ્યાખ્યાયિત કર્યા છે. scanf() વિધાન આ ચલમાં અપૂર્ણ સંખ્યાઓનો સંગ્રહ કરે છે. આ કોડના અમલ દરમિયાન જો ઉપયોગકર્તા **85.25 90.65** સંખ્યાઓ ઉમેરે તો 85.25 સંખ્યાનો per1 ચલમાં અને 90.65 સંખ્યાનો per2 ચલમાં સંગ્રહ કરવામાં આવશે. scanf() વિધેય અપૂર્ણ સંખ્યાઓ વાંચવા માટે **"%f"**નો ઉપયોગ કરે છે. જો ઉપયોગકર્તા પાસેથી **double** ડેટાટાઇપ ધરાવતી સંખ્યા મેળવવાની હોય તો "%f"ના સ્થાને **"%lf"**નો ઉપયોગ કરવો જોઈએ.

અક્ષર અને શબ્દનું વાચન (Reading character and word)

getchar() અને gets() જેવા આંતરસ્થાપિત વિધેયોના ઉપયોગથી એક અક્ષર અને અક્ષરોનો સમૂહ (સ્ટ્રિંગ) મેળવી શકાય છે તે આપણે જોયું. અક્ષર અને શબ્દ વાંચવાનું આ જ કાર્ય scanf() વિધાન દ્વારા પણ અનુક્રમે %c અને %s સ્પેસિફાયરની મદદથી કરી શકાય છે. નીચેનો પ્રોગ્રામ-ખંડ જુઓ :

```
char c1, c2;
char city[20];
scanf("%c %c %s", &c1, &c2, city);
```

આપેલ કોડના અમલ દરમિયાન ઉપયોગકર્તા જો **A B Gandhinagar** જેવો ડેટા ઉમેરે તો, c1 ચલમાં 'A', c2 ચલમાં 'B' અને city નામના અક્ષરના એરેમાં 'Gandhinagar' કિંમતનો સંગ્રહ કરવામાં આવશે. અક્ષરોની એરે સંબંધિત વધુ ચર્ચા આ પુસ્તકના પ્રકરણ 15માં કરવામાં આવી છે.

અહીં એ નોંધ લેવી જરૂરી છે કે scanf() વિધાન દ્વારા સ્ટ્રિંગ મેળવતી વખતે ચલનાં નામ સાથે **&** (ampersand) નિશાની જરૂરી નથી. એ પણ યાદ રાખવું જરૂરી છે કે ઇનપુટમાં ખાલી જગ્યા આવે ત્યારે %s સ્પેસિફાયર ઇનપુટને અટકાવે છે. નીચેનું ઉદાહરણ જુઓ:

```
char city[20];
scanf("%s", city);
```

આ કોડના અમલ દરમિયાન ઉપયોગકર્તા જો **New Delhi** ઉમેરશે તો city નામના ચલમાં માત્ર "New" શબ્દનો સંગ્રહ કરવામાં આવશે. %s સ્પેસિફાયર New શબ્દ પછી આપેલી ખાલી જગ્યાને કારણે ઇનપુટને અટકાવશે. અને "Delhi" શબ્દને scanf() દ્વારા અવગણવામાં આવશે.

%[characters] અને %[^ characters]ની મદદથી મર્યાદિત ઇનપુટ

સી ભાષામાં આવેલ scanf() વિધેય %[characters]ના ઉપયોગ દ્વારા એક અદ્ભુત સુવિધા પૂરી પાડે છે, જેના ઉપયોગથી ઇનપુટ સ્ટ્રિંગમાં માત્ર સ્વીકાર્ય હોય તેવા જ અક્ષરો દર્શાવી શકાય છે. %[characters]માં આવેલ ન હોય તેવો કોઈ પણ અક્ષર ઉમેરવામાં આવે તો આવો અક્ષર પ્રથમ વખત ઉમેરવાની ઘટના સ્ટ્રિંગને અટકાવે છે. ઉદાહરણ 12.3નો ઉપયોગ કરી આ અભિગમ સમજવાનો પ્રયત્ન કરીએ. આકૃતિ 12.6માં ઉદાહરણ 12.3નું કોડ લિસ્ટિંગ આપવામાં આવ્યું છે તથા આકૃતિ 12.7 તેનું પરિણામ દર્શાવે છે.

```
9-3.c - SciTE
File Edit Search View Tools Options Language Buffers Help
9-3.c

/* Example 3 Program to demonstrate use of %[characters] specification. */

#include<stdio.h>
int main()
{
	char student_name[30];

	printf("Enter your name containing only alphabets and space: \n");
	scanf("%[a-z A-Z]", student_name);
	printf("Your name stored in variable is %s \n", student_name);
	return 0;
}
//End of Program
```

આકૃતિ 12.6 : ઉદાહરણ 12.3નું કોડ-લિસ્ટિંગ

```
kpp@ubuntu: ~
File Edit View Terminal Help
kpp@ubuntu:~$ gcc 9-3.c
kpp@ubuntu:~$ ./a.out
Enter your name containing only alphabets and space:
Ragi Bharat Patel
Your name stored in variable is Ragi Bharat Patel
kpp@ubuntu:~$ ./a.out
Enter your name containing only alphabets and space:
Ragi B. Patel
Your name stored in variable is Ragi B
kpp@ubuntu:~$ 
```

આકૃતિ 12.7 : ઉદાહરણ 12.3નું પરિણામ

ઉદાહરણ 12.3ના scanf() વિધાનમાં આપેલ %[a-z A-Z] દર્શાવે છે કે માત્ર aથી z, ખાલી જગ્યા અને Aથી Z અક્ષરોનો જ સ્વીકાર કરી student_name નામના ચલમાં તેને સંગૃહીત કરવામાં આવશે. માટે, પ્રથમ અમલ દરમિયાન વિદ્યાર્થીના નામનો ચલમાં સંગ્રહ કરવામાં આવ્યો છે. પરંતુ બીજા અમલ દરમિયાન ઉપયોગકર્તા પૂર્ણવિરામ (.) ઉમેરે છે, જે યાદીમાં આવેલ નથી. તેથી ચલમાં માત્ર "Ragi B"નો જ સંગ્રહ કરવામાં આવે છે.

જો ઉપયોગકર્તા યાદીમાં આપેલ અક્ષરો પૈકી કોઈ પણ અક્ષર ઉમેરશે તો scanf()માં %[^characters] સ્વરૂપનો ઉપયોગ ઇનપુટ પ્રક્રિયાને તત્કાલ અટકાવી દેશે. ઉદાહરણ 12.4નો ઉપયોગ કરી આ અભિગમ સમજવાનો પ્રયત્ન કરીએ. આકૃતિ 12.8માં ઉદાહરણ 12.4નું કોડ લિસ્ટિંગ આપવામાં આવ્યું છે તથા આકૃતિ 12.8 તેનું પરિણામ દર્શાવે છે.

```
9-4.c - SciTE
File Edit Search View Tools Options Language Buffers Help
9-4.c

/* Example 4:  Program to demonstrate use of %[^characters] specification. */

#include<stdio.h>
int main()
{
	char user_data[100];

	printf("Enter your details:\n");
	printf("Enter * to exit:\n");
	scanf(" %[^*] ", user_data);
	printf("Your details stored in variable is \n%s \n", user_data);
	return 0;
}
//End of Program
```

આકૃતિ 12.8 : ઉદાહરણ 12.4નું કોડ-લિસ્ટિંગ

```
kpp@ubuntu: ~
File Edit View Terminal Help
kpp@ubuntu:~$ gcc 9-4.c
kpp@ubuntu:~$ ./a.out
Enter your details:
Enter * to exit:
My name is Purva.
My mother's name is Krina*
Your details stored in variable is
My name is Purva.
My mother's name is Krina
kpp@ubuntu:~$
```

આકૃતિ 12.9 : ઉદાહરણ 12.4નું પરિણામ

ઉદાહરણ 12.4ના અમલ દરમિયાન કોઈ પણ અક્ષર (નવી લીટી માટેના 'એન્ટર' સહિત) ઉમેરી શકાય છે. જ્યારે * ઉમેરવામાં આવે ત્યારે ઇનપુટ સ્ટ્રિંગને સ્વીકારવાની ક્રિયા અટકાવવામાં આવે છે. અહીં એ નોંધ લેવી જરૂરી છે કે user_data ચલની લંબાઈ 100 ઘોષિત કરી હોવાથી આપણા ઉદાહરણમાં માત્ર 100 અક્ષરો ઉમેરી શકાય.

મિશ્ર ડેટાનું વાચન (Reading mixed data)

એક જ scanf() વિધાનની મદદથી જુદા જુદા પ્રકારનો ડેટા (જેમ કે પૂર્ણાંક, અપૂર્ણાંક, અક્ષર, સ્ટ્રિંગ) ઉમેરવો શક્ય છે. આ પ્રકારના કિસ્સામાં ખાતરી કરી લેવી જોઈએ કે scanf()ના કન્ટ્રોલ સ્પેસિફિકેશન સાથે ઉમેરવામાં આવેલ ડેટાનો ક્રમ અને ડેટાપ્રકાર અચૂકપણે સમાન હોય. આ અભિગમ સમજવા માટે નીચે આપેલ પ્રોગ્રામ-ખંડ જુઓ :

```
int roll_no;
char grade, name[30];
float percen;
scanf("%d %f %s %c", &roll_no, &percen, name, &grade);
```

આ કોડના અમલ દરમિયાન જો **11 90.55 Vani** A વિગતો ઉમેરવામાં આવે તો roll_no ચલમાં 11, percen ચલમાં 90.55, name ચલમાં 'Vani' અને grade ચલમાં 'A' કિંમતોનો સંગ્રહ કરવામાં આવશે. પરંતુ આ જ પ્રોગ્રામ કોડ માટે જો ઉપયોગકર્તા **11 Vani A 90.55** વિગતો ઉમેરે તો કમ્પ્યૂટર ભૂલનો સંદેશ દર્શાવશે કારણ કે કમ્પાઇલરને જ્યાં અપૂર્ણાંક સંખ્યાની અપેક્ષા હતી ત્યાં ઉપયોગકર્તાએ અક્ષર ઉમેર્યો છે. આ કિસ્સામાં scanf() વિધાન પ્રથમ કિંમત મેળવ્યા બાદ વાંચવાની પ્રક્રિયા અટકાવશે.

જુદા જુદા પ્રકારના ડેટા મેળવવા માટે scanf()ની કન્ટ્રોલ સ્ટ્રિંગમાં ઉપયોગમાં લઈ શકાય તેવા અક્ષરોની યાદી કોષ્ટક 12.1માં આપવામાં આવી છે.

ડેટાપ્રકાર	સંલગ્ન અક્ષર
દશાંકી પૂર્ણાંક વાંચવા માટે	%d
અક્ષર વાંચવા માટે	%c
અપૂર્ણાંક વાંચવા માટે	%f અથવા %e અથવા %g
સ્ટ્રિંગ વાંચવા માટે	%s
ચિહ્નરહિત (અનસાઇન્ડ) પૂર્ણાંક વાંચવા માટે	%u
ટૂંકા (શૉર્ટ) પૂર્ણાંક વાંચવા માટે	%h
લાંબા (લૉન્ગ) પૂર્ણાંક વાંચવા માટે	%ld
ડબલ સંખ્યા વાંચવા માટે	%lf
લૉન્ગ ડબલ સંખ્યા વાંચવા માટે	%L

કોષ્ટક 12.1 : scanf()ની કન્ટ્રોલ સ્ટ્રિંગમાં ઉપયોગમાં લેવામાં આવતા અક્ષરો

આંતરપ્રસ્થાપિત આઉટપુટ વિધેય (Inbuilt output function)

કમ્પ્યૂટર સિસ્ટમનાં ત્રણ મૂળભૂત કાર્યો છે : નિવેશ (input), પ્રક્રિયા (process) અને નિર્ગમ (output). ઉપયોગકર્તા પાસેથી ડેટા મેળવવા માટે ઉપલબ્ધ આંતરસ્થાપિત ઇનપુટ વિધેય વિશે આપણે જોયું. હવે, પ્રક્રિયામાંથી પસાર થયેલો ડેટા દર્શાવવા માટેના આંતરસ્થાપિત આઉટપુટ વિધેય વિશે અભ્યાસ કરીએ. પરિણામને મૉનિટર, પ્રિન્ટર અને ફાઇલ જેવાં આઉટપુટ સાધનો પર મોકલવામાં આવે છે. સી ભાષામાં પ્રમાણભૂત આઉટપુટ સાધન મૉનિટર પર પરિણામ કેવી રીતે દર્શાવી શકાય તેની ચર્ચા કરીશું. પરંતુ ફાઇલ અને પ્રિન્ટર પર પરિણામ મોકલવા અંગેની ચર્ચા આપણે નહીં કરીએ કારણ કે તે આપણા પાઠ્યક્રમમાં સમાવિષ્ટ નથી.

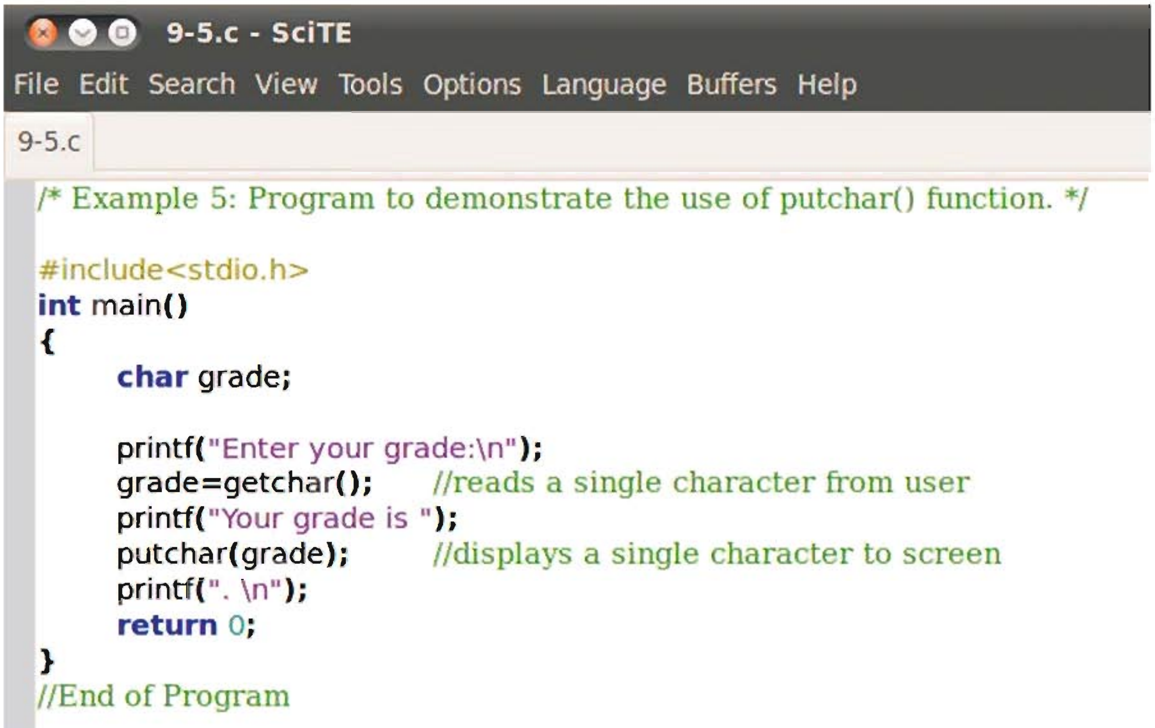
ઇનપુટની જેમ જ, સી લાઇબ્રેરી વિધેય દ્વારા આઉટપુટ સાધન પર પરિણામ મોકલવું શક્ય છે. <stdio.h> હેડર ફાઇલમાં આંતરસ્થાપિત આઉટપુટ વિધેય ઉપલબ્ધ છે. હવે putchar(), puts() અને printf() જેવા આઉટપુટ સંબંધિત આંતરસ્થાપિત વિધેય વિશે ચર્ચા કરીએ.

putchar()

પ્રમાણભૂત આઉટપુટ સાધન પર એક અક્ષર દર્શાવવા માટે putchar() વિધેયનો ઉપયોગ કરી શકાય છે. putchar()ની સામાન્ય વાક્યરચના નીચે મુજબ છે:

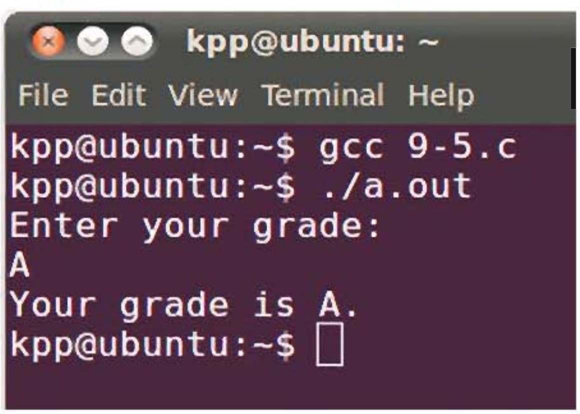
putchar(character);

અહીં character એ char પ્રકારનો ચલ અથવા તો સી ભાષામાં યોગ્ય એવો કોઈ પણ અક્ષર હોઈ શકે. જ્યારે આ વિધેયનો અમલ કરવામાં આવે છે ત્યારે મોનિટર પર અક્ષર દર્શાવવામાં આવશે. ઉદાહરણ 12.5નો ઉપયોગ કરી આ અભિગમ સમજવાનો પ્રયત્ન કરીએ. આકૃતિ 12.10માં ઉદાહરણ 12.5નું કોડ લિસ્ટિંગ આપવામાં આવ્યું છે જ્યારે આકૃતિ 12.11 તેનું પરિણામ દર્શાવે છે.




```
/* Example 5: Program to demonstrate the use of putchar() function. */

#include<stdio.h>
int main()
{
	char grade;

	printf("Enter your grade:\n");
	grade=getchar();	//reads a single character from user
	printf("Your grade is ");
	putchar(grade);	//displays a single character to screen
	printf(". \n");
	return 0;
}
//End of Program
```

આકૃતિ 12.10 : ઉદાહરણ 12.5નું કોડ લિસ્ટિંગ




```
kpp@ubuntu:~$ gcc 9-5.c
kpp@ubuntu:~$ ./a.out
Enter your grade:
A
Your grade is A.
kpp@ubuntu:~$
```

આકૃતિ 12.11 : ઉદાહરણ 12.5નું પરિણામ

આ પ્રોગ્રામ getchar() વિધેયનો ઉપયોગ કરી એક અક્ષર મેળવશે અને putchar() વિધેયનો ઉપયોગ કરી સ્ક્રીન પર એક અક્ષર દર્શાવશે.

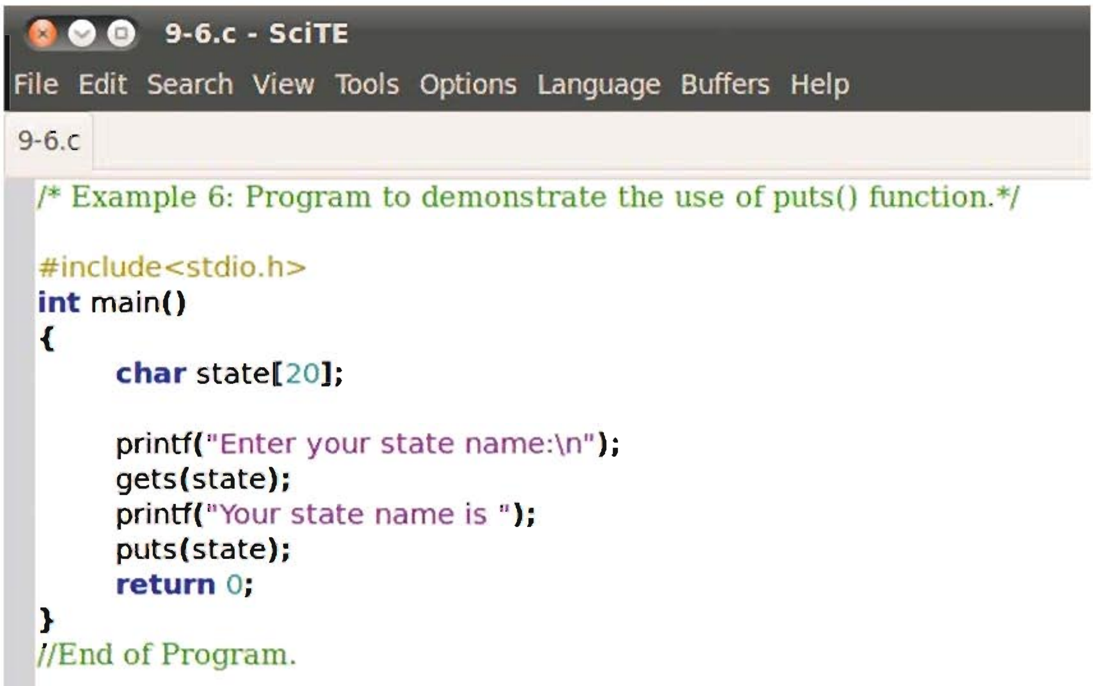
putchar() વિધેયની એક મર્યાદા એ છે કે, તે એક સમયે એક જ અક્ષર દર્શાવી શકે છે. એકથી વધુ અક્ષરો દર્શાવવા માટે સી ભાષાના લૂપ અભિગમનો ઉપયોગ કરવો પડે છે. એકથી વધુ અક્ષરો દર્શાવવા માટેનો સરળ ઉકેલ puts() વિધેયનો ઉપયોગ કરવાનો છે.

puts()

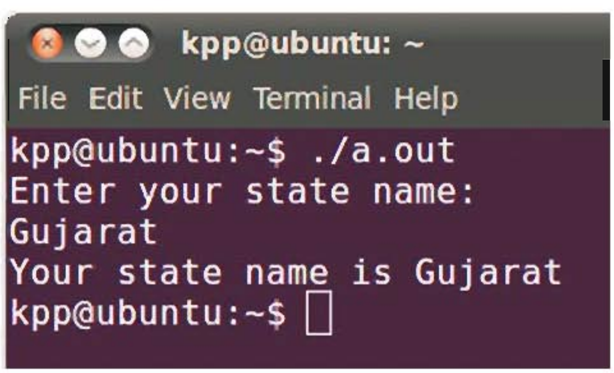
આઉટપુટ સાધન પર એકથી વધુ અક્ષરો (સ્ટ્રિંગ અથવા અક્ષરોનો એરે) દર્શાવવા માટે puts() વિધેયનો ઉપયોગ કરી શકાય છે. puts( )ની સામાન્ય વાક્યરચના નીચે મુજબ છે:

puts(variable_name);

અહીં *variable_name* એ અક્ષરનો એરે અથવા સ્ટ્રિંગ છે. 'નલ' અક્ષર ન આવે ત્યાં સુધી puts() વિધેય *variable_name* માં સંગૃહીત તમામ વિગતો મોનિટર પર દર્શાવે છે. એ નોંધ કરો કે દરેક સ્ટ્રિંગના અંતમાં 'નલ' અક્ષર આપેલ હોય છે, પરંતુ puts() વિધેય આ અક્ષરને સ્ક્રીન પર દર્શાવતું નથી. આ અભિગમ સ્પષ્ટ કરવા માટે ઉદાહરણ 12.6નો અભ્યાસ કરો. ઉદાહરણ 12.6 નું કોડ લિસ્ટિંગ આકૃતિ 12.12 માં આપેલ છે તથા આકૃતિ 12.13 તેનું પરિણામ દર્શાવે છે.



આકૃતિ 12.12 : ઉદાહરણ 12.6નું કોડ-લિસ્ટિંગ



આકૃતિ 12.13 : ઉદાહરણ 12.6નું પરિણામ

આ પ્રોગ્રામ gets() વિધેયનો ઉપયોગ કરી એકથી વધુ અક્ષરો વાંચી તેનો આપેલ ચલમાં સંગ્રહ કરશે. puts() વિધેયના ઉપયોગ દ્વારા ચલમાં આવેલી કિંમતને સ્ક્રીન પર દર્શાવવામાં આવશે.

સુગ્રથિત પરિણામ (Formatted output)

અત્યાર સુધીમાં ચર્ચવામાં આવેલા આઉટપુટ માટેના વિધેય સુગ્રથિત (formatted) ન હતા. તેના ચલમાં સંગ્રહવામાં આવેલા કિંમતને માત્ર સ્ક્રીન પર દર્શાવવામાં આવતી હતી. પરંતુ કેટલીકવાર પ્રોગ્રામ પરથી મળતાં પરિણામને એવી

રીતે તૈયાર કરવાની જરૂર પડે છે જે દેખાવમાં આકર્ષક અને સમજવામાં સરળ હોય. પરિણામને જુદી-જુદી સંરચનાઓ સાથે દર્શાવવા માટેની સુવિધા printf() વિધેય દ્વારા પૂરી પાડવામાં આવે છે.

printf()

સ્ક્રીન પર સુગ્રથિત (formatted) પરિણામ દર્શાવવા માટે printf() વિધેયનો ઉપયોગ કરવામાં આવે છે. printf() વિધેયની સામાન્ય વાક્યરચના નીચે આપેલ છે :

printf("control string", var1, var2, ..., varN);

અહીં var1, var2, ..., varN ચલ, અચળ કે પદાવલિ હોઈ શકે છે. પરિણામની ગોઠવણ માટેની સ્પષ્ટતા (output format specification) કંન્ટ્રોલ સ્ટ્રિંગમાં આપવામાં આવે છે. કન્ટ્રોલ સ્ટ્રિંગમાં નીચે દર્શાવેલ એક અથવા તમામ ઘટકોનો સમાવેશ કરવામાં આવે છે.

- અક્ષરોનો ગણ (સ્ટ્રિંગ) કે જે કન્ટ્રોલ સ્ટ્રિંગમાં છે તે જ પ્રમાણે મોનિટર પર દર્શાવવાના છે.
- દરેક ચલના ફૉર્મેટ સ્પેસિફાયર
- \n (નવી લીટી), \t (ટૅબ), \b (બૅક સ્પેસ) જેવા એસ્કેપ સિક્વન્સ અક્ષરો

કન્ટ્રોલ સ્ટ્રિંગમાં આપેલ વિગતોને scanf()ની જેમ જગ્યા આપી છૂટી પાડવામાં આવે છે અને તેની આગળ '%' નિશાની ઉમેરવામાં આવે છે. જે ચલની કિંમત દર્શાવવાની હોય તેનો સમાવેશ printf()ના પછીના ભાગમાં કરવામાં આવે છે. કન્ટ્રોલ સ્ટ્રિંગમાં આપેલ ફૉર્મેટ સ્પેસિફાયર સાથે આ ચલની સંખ્યા, ક્રમ અને પ્રકાર મેળ ખાતા હોવા જોઈએ.

પ્રોગ્રામમાં સ્ક્રીન પર પરિણામ દર્શાવવા માટે આપણે અત્યાર સુધી ઉપયોગમાં લીધેલાં કેટલાક સરળ printf( ) વિધાનનાં ઉદાહરણ નીચે દર્શાવ્યાં છે :

- printf("Hello World");
- printf("Your age is %d", age);
- printf("Your name is %s", student_name);

જુદા જુદા પ્રકારની ગોઠવણીઓ સાથે વ્યવસ્થિત પરિણામ દર્શાવવા માટે printf() વિધાનની કન્ટ્રોલ સ્ટ્રિંગમાં ફૉર્મેટ સ્પેસિફિકેશન (format specification)નો ઉપયોગ કરવામાં આવે છે. કન્ટ્રોલ સ્ટ્રિંગનું સામાન્ય સ્વરૂપ નીચે મુજબ છે :

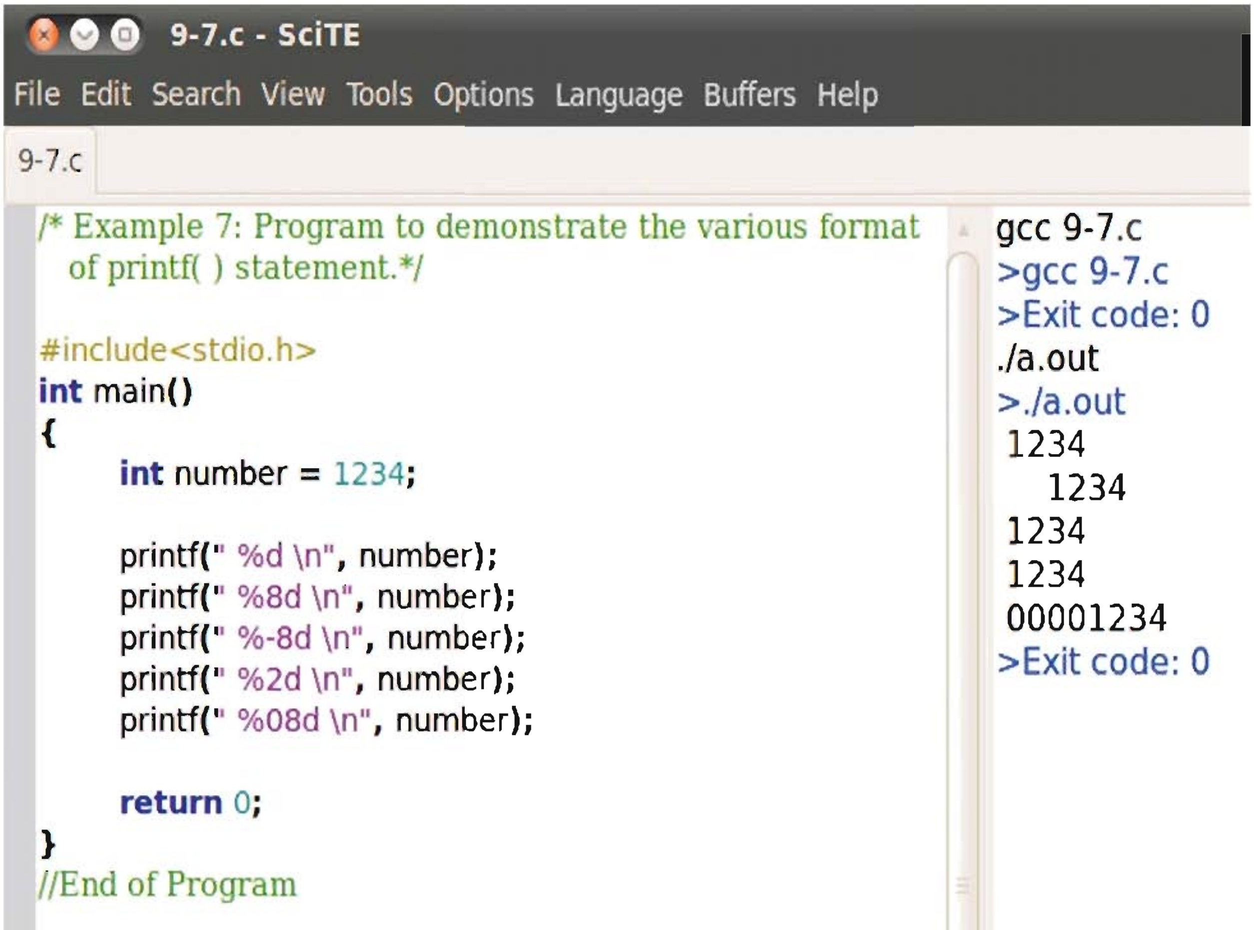
%l.mp

અહીં, 'l' એ કુલ અક્ષરોની સંખ્યાના સ્થાનનો નિર્દેશ કરતો પૂર્ણાંક છે, જેમાં ચલની કિંમત દર્શાવવાની છે. અપૂર્ણ સંખ્યામાં દશાંશ ચિહ્ન પછી દર્શાવવાના અંકો માટે અથવા સ્ટ્રિંગ ચલમાંથી દર્શાવવાના અક્ષરોની સંખ્યા માટે 'm'નો ઉપયોગ કરવામાં આવે છે. l અને mની કિંમતો આપવી વૈકલ્પિક છે. 'p' ચલના પ્રકારનો નિર્દેશ કરે છે. printf( ) વિધેય સાથે ઉપયોગમાં લઇ શકાય તેવા જુદા-જુદા ડેટાટાઇપની યાદી કોષ્ટક 12.2માં દર્શાવી છે.

ડેટાટાઇપ	સંબંધિત પ્રકાર
દશાંકી સંખ્યા દર્શાવવા માટે	%d
એક અક્ષર દર્શાવવા માટે	%c
ઘાતાંક વગર અપૂર્ણ સંખ્યા દર્શાવવા માટે	%f
ઘાતાંક સાથે અપૂર્ણ સંખ્યા દર્શાવવા માટે	%e
સ્ટ્રિંગ દર્શાવવા માટે	%s
ચિહ્નરહિત (અનસાઇન્ડ) પૂર્ણાંક દર્શાવવા માટે	%u
લોંગ દશાંકી પૂર્ણાંક દર્શાવવા માટે	%ld
ડબલ સંખ્યા દર્શાવવા માટે	%lf
લોંગ ડબલ સંખ્યા દર્શાવવા માટે	%Lf
પૂર્ણાંકને અષ્ટાંકી સ્વરૂપે દર્શાવવા માટે	%o
પૂર્ણાંકને સોળઅંકી સ્વરૂપે દર્શાવવા માટે	%x

કોષ્ટક 12.2 : printf( )ની કન્ટ્રોલ સ્ટ્રિંગમાં ઉપયોગી ડેટાટાઇપ

આકૃતિ 12.14માં આપેલ ઉદાહરણ 12.7નું કોડ લિસ્ટિંગ સમજવાનો પ્રયત્ન કરીએ, જે printf( ) વિધાનમાં વિવિધ સ્વરૂપો દર્શાવે છે.



આકૃતિ 12.14 : ઉદાહરણ 12.7નું કોડ-લિસ્ટિંગ અને પરિણામ

સમજૂતી (Explanation)

પ્રથમ પરિણામમાં દર્શાવ્યું છે તે મુજબ પૂર્વનિર્ધારિત રીતે printf( ) વિધાન પરિણામ દર્શાવવા માટે ડાબી બાજુની ગોઠવણ(left align)નો ઉપયોગ કરે છે. બીજું printf( ) વિધાન જમણી બાજુની ગોઠવણ (right justification) સાથે 8 અક્ષરોની જગ્યાનો ઉપયોગ કરે છે. ત્રીજું printf( ) વિધાન દર્શાવે છે કે % નિશાની પછી ઋણ નિશાની (–) ઉમેરવાથી પરિણામને ડાબી બાજુ ગોઠવવું શક્ય છે. ચોથા printf( ) વિધાનમાં %2d લખવાથી કોઈ અસર મેળવી શકાતી નથી. કારણ કે, આપવામાં આવેલ સંખ્યા ચલના અક્ષરોની કુલ સંખ્યા કરતાં ઓછી છે. અંતિમ printf( ) વિધાન દર્શાવે છે કે ચલની મૂળ કિંમત સાથે વધારાના અક્ષરો (શૂન્ય) દર્શાવવાનું પણ શક્ય છે.

અગાઉના ઉદાહરણમાં આપણે જોયું કે પૂર્ણાંક ચલને જુદા જુદા પ્રકારની સંરચનાઓ સાથે કેવી રીતે દર્શાવી શકાય. હવે, વિવિધ સંરચનાઓ દ્વારા અપૂર્ણાંક સંખ્યાને કેવી રીતે દર્શાવી શકાય તે જોઈએ. સામાન્ય રચના %l.mp યાદ કરો જેમાં *m* એ દશાંશ પછી દર્શાવવાના અંકોનો નિર્દેશ કરે છે. જ્યારે અપૂર્ણાંક સંખ્યા દર્શાવવામાં આવે છે ત્યારે *l* સ્તંભની પહોળાઈમાં *m* દશાંશ જગ્યાઓ સાથે જમણી બાજુ ગોઠવાયેલ અંકો રજૂ કરવામાં આવે છે. જો *m*નો ઉપયોગ કર્યો ન હોય તો દશાંશચિહ્ન પછી પૂર્વનિર્ધારિત 6 અંક દર્શાવવામાં આવે છે. f1 ચલમાં આપેલી કિંમત 123.456 વિવિધ સંરચનાઓનો ઉપયોગ કરી printf( ) વિધાનની મદદથી કેવી રીતે રજૂ કરી શકાય તે આકૃતિ 12.15માં દર્શાવ્યું છે.

Statement	Output									
printf(“ %f ”, f1);	1	2	3	.	4	5	6	0	0	0
printf(“ %8.3f ”, f1);		1	2	3	.	4	5	6		
printf(“ %8.1f ”, f1);				1	2	3	.	5		
printf(“ %08.1f ”, f1);	0	0	0	1	2	3	.	5		
printf(“ %-8.1f ”, f1);	1	2	3	.	5					
printf(“ %10.3e ”, f1);		1	.	2	3	5	e	+	0	2
printf(“ %10.4e ”, f1);	1	.	2	3	4	6	e	+	0	2

આકૃતિ 12.15 : વિવિધ સંરચનાઓ દ્વારા અપૂર્ણાંક સંખ્યાની રજૂઆત

printf( ) વિધાનનો ઉપયોગ કરી અક્ષર અને સ્ટ્રિંગની સંરચના પણ કરી શકાય છે. આ માટેનું ફૉર્મેટ સ્પેસિફિકેશન અપૂર્ણાંક સંખ્યા જેવું જ છે. *l*ની કિંમતનો ઉપયોગ ફિલ્ડની પહોળાઈ દર્શાવવા માટે કરવામાં આવે છે, જ્યારે દર્શાવવામાં આવનાર અક્ષરોની સંખ્યાનો નિર્દેશ *m*ની કિંમત દ્વારા કરવામાં આવે છે. જો *l*ની કિંમતનો ઉપયોગ કરવામાં ન આવે તો સ્ટ્રિંગને ડાબી બાજુ ગોઠવવામાં આવે છે તથા *l*નો ઉપયોગ સ્ટ્રિંગને સ્ક્રીન પર જમણી બાજુ ગોઠવે છે. પૂર્ણાંક સંખ્યાના ફૉર્મેટ સ્પેસિફિકેશન મુજબ ઋણ નિશાની ઉમેરવાથી સ્ટ્રિંગને ડાબી બાજુ ગોઠવી શકાય છે.

નીચેના **printf( )** વિધાનનો ઉપયોગ કરી સ્ટ્રિંગ માટેની સંરચનાનો ખ્યાલ વધુ સ્પષ્ટ કરીએ. અહીં **str** નામના સ્ટ્રિંગ ચલ (અક્ષરોના એરે)માં "GUJARAT INDIA" કિંમતનો સંગ્રહ કરવામાં આવ્યો છે. આકૃતિ 12.16 આ વિવિધ સંરચનાઓ દર્શાવે છે.

Statement	Output															
printf(“ %s ”, str);	G	U	J	A	R	A	T		I	N	D	I	A			
printf(“ %8s ”, str);	G	U	J	A	R	A	T		I	N	D	I	A			
printf(“ %16s ”, str);				G	U	J	A	R	A	T		I	N	D	I	A
printf(“ %16.7s ”, str);										G	U	J	A	R	A	T
printf(“ %.7s ”, str);	G	U	J	A	R	A	T									
printf(“%-16.9s”, str);	G	U	J	A	R	A	T		I							
printf(“%16.15s”, str);				G	U	J	A	R	A	T		I	N	D	I	A

આકૃતિ 12.16 : વિવિધ સંરચનાઓ દ્વારા અક્ષરોની રજૂઆત

અંતિમ **printf( )** વિધાનમાં આપેલ *m*ની કિંમત 15 **str** ચલમાં સંગ્રહવામાં આવેલા સ્ટ્રિંગની લંબાઈ કરતાં વધુ હોવાથી **str** ચલના તમામ અક્ષરો જમણી બાજુ દર્શાવવામાં આવશે.

સારાંશ

સુગ્રથિત નિવેશ અને નિર્ગમ આંતરસ્થાપિત વિધેય (formatted input and output inbuilt function)નો પ્રોગ્રામમાં કેવી રીતે ઉપયોગ થઈ શકે તે માટેનો અભ્યાસ આપણે આ પ્રકરણમાં કર્યો. પરિણામને સંરચના (formation) વગર પણ દર્શાવી શકાય છે, પરંતુ વિશિષ્ટ સંરચના દ્વારા રજૂ કરેલું પરિણામ ઉપયોગકર્તાને વધુ સુવાચ્ય અને આકર્ષક લાગે છે. પ્રોગ્રામને આકર્ષક બનાવવા આ વિધેયનો ઉપયોગ કેવી રીતે કરવો તેનો આધાર પ્રોગ્રામર પર છે.

સ્વાધ્યાય

1. scanf( ) વિધેયની કન્ટ્રોલ સ્ટ્રિંગ printf( ) વિધેયની કન્ટ્રોલ સ્ટ્રિંગ કરતા કેવી રીતે અલગ પડે છે ?
2. પ્રોગ્રામમાં સંરચના કરી શકાય તે પ્રકારના આઉટપુટ વિધેય(formatted output function)ના ઉપયોગની ચર્ચા કરો.
3. પ્રોગ્રામમાં સંરચના કરી શકાય તે પ્રકારના ઇનપુટ વિધેય(formatted input function)ના ઉપયોગની ચર્ચા કરો.
4. યોગ્ય ઉદાહરણનો ઉપયોગ કરી %[characters] અને %[^characters]ની મદદથી મર્યાદિત ઇનપુટનો અભિગમ સમજાવો.
5. નીચેનાં વિધાનો ખરાં છે કે ખોટાં તે જણાવો :
   (a) એક સમયે એકથી વધુ અક્ષરો ઉમેરવા getchar( ) વિધેયનો ઉપયોગ કરી શકાય.
   (b) gets( ) વિધેયના ઉપયોગ દ્વારા એક અક્ષર વાંચી શકાય છે.
   (c) એક જ printf( ) વિધાનની મદદથી એકથી વધુ લીટીઓમાં પરિણામ દર્શાવી શકાય છે.
   (d) %.10s ફોર્મેટ સ્પેસિફાયર સ્ટ્રિંગના પ્રથમ દસ અક્ષરો દર્શાવશે.
   (e) એકથી વધુ કિંમતો મેળવવા એક જ scanf( ) વિધાનનો ઉપયોગ કરી શકાય છે.
   (f) અપૂર્ણ સંખ્યાઓને પૂર્વનિર્ધારિત રીતે છ દશાંશ સાથે દર્શાવવામાં આવે છે.
6. માગ્યા મુજબ કરો :
   (1) નીચેનાં વિધાનોમાં કોઈ ક્ષતિ હોય તો જણાવો :
      (a) scanf("%d %c", &number, city_code);
      (b) scanf("%d %d %s" \n, &marks1, &marks2, city_name);
      (c) scanf("%d %f %c %s", &num1, &price, item_code);
   (2) નીચેનાં વિધાનોનું પરિણામ જણાવો :
      (a) printf("%d %c %f", 101, 'X', 20.20);
      (b) printf("%3d %8.2f ", 12345, 222.123);
      (c) printf("%.5s", "Hello World");
      (d) printf("%15.8s", "Hello Student");
      (e) printf("%0.5s", "Hello World");

(3) નીચેના પ્રોગ્રામનું પરિણામ લખો :

```
#include<stdio.h>
int main( )
{
	float result = 98.12345;

	printf("Your result is %f \n", result);
	printf("Your result is %.2f \n", result);
	printf("Your result is %5.3f \n", result);
	return 0;
}
```

(4) નીચેના પ્રોગ્રામમાં કોઈ ક્ષતિ હોય તો તેને દૂર કરી પરિણામ લખો :

```
#include<std.h>
int main( )
{
	int sum = 1234;
	float price = = 550.50;
	char city[20] = "Ahmedabad";

	printf("The sum is %d....", price);
	printf("The price is %s...", price);
	printf("The city name is %.3s...", city);
	return 0;
}
```

7. આપેલ વિકલ્પોમાંથી સાચો વિકલ્પ પસંદ કરો :

(1) સી ભાષામાં I/O પ્રક્રિયા માટે નીચેનામાંથી શેનો ઉપયોગ કરવામાં આવે છે ?

(a) મૉનિટર (b) કી-બોર્ડ (c) માઉસ (d) આપેલ તમામ

(2) નીચેનામાંથી કયા સાધન દ્વારા પ્રોગ્રામમાં ઇનપુટ શક્ય છે ?

(a) કી-બોર્ડ (b) સ્પીકર (c) પ્રિન્ટર (d) મૉનિટર

(3) નીચેનામાંથી કયું વિધેય ઇનપુટ માટે નથી ?

(a) getchar() (b) getch() (c) puts() (d) gets()

(4) printf("%4s", "Krusha");નું પરિણામ શું મળશે ?

(a) Krus (b) usha (c) Krusha (d) Krush

(5) સ્ટ્રિંગ સ્વીકારવા માટે નીચેનામાંથી કયું વિધેય વધુ યોગ્ય છે ?

(a) getch() (b) gets() (c) getchar() (d) getc()

પ્રાયોગિક સ્વાધ્યાય

1. ઉપયોગકર્તા પાસેથી "Hello Gujarat" સ્ટ્રિંગ મેળવી, નીચે આપેલ જુદા જુદા સ્વરૂપે દર્શાવો :

   (a) Hello (b) Gujarat (c) Hello Gujarat (d) Hello G

2. ઉપયોગકર્તાના ઇનપુટને આધારે ઘડિયો (Multiplication table) દર્શાવે તેવો સી પ્રોગ્રામ બનાવો. પરિણામને વિવિધ સંરચનાઓ સાથે દર્શાવવા આઉટપુટ ફૉર્મેટિંગના જુદા જુદા વિકલ્પોનો ઉપયોગ કરો.

3. ઉપયોગકર્તા પાસેથી માત્ર કૅપિટલ અક્ષરો સ્વીકારી શકાય તેવો પ્રોગ્રામ બનાવો. (સી ભાષામાં ઉપલબ્ધ મર્યાદિત ઇનપુટના અભિગમનો ઉપયોગ કરો.)

4. ઉપયોગકર્તા પાસેથી થોડા પૂર્ણાંકો મેળવવા માટેનો પ્રોગ્રામ બનાવો. તમામ પૂર્ણાંકોને સ્ક્રીન પર જમણી બાજુ દર્શાવો.

5. નીચેના અંકો મેળવવા માટેનો સી પ્રોગ્રામ બનાવો :

   123.45 34.56 – 7878.123 – 0.965

   યોગ્ય ડેટાપ્રકાર ધરાવતા ચલમાં આ અંકોનો સંગ્રહ કરો. તમામ અંકોને તેની પાસેના પૂર્ણાંકોમાં ફેરવીને દર્શાવો.

6. A અને Bની કિંમત મેળવવા માટેનો સી પ્રોગ્રામ લખો. ત્યાર પછી નીચેની પદાવલિઓનું પરિણામ એક જ લીટીમાં દર્શાવો :

   (a) (A + B) / (A – B) (b) (A + B) (A – B) (c) (A * A) (B + B)

નિર્ણય માળખાં

અત્યાર સુધી રજૂ કરવામાં આવેલા મોટાભાગના તમામ સી પ્રોગ્રામ તેના અમલ વખતે એક ક્રમબદ્ધ માળખાને અનુસરે છે. પ્રોગ્રામમાં લખવામાં આવેલા ક્રમ પ્રમાણે એક પછી એક તમામ સૂચનાઓનો અમલ કરવામાં આવે છે.

રોજિંદા જીવનમાં સી પ્રોગ્રામના અમલ દરમિયાન આપણને કેટલીક શરતોને આધારે અમુક જ વિધાનોનો અમલ કરવાની જરૂર પડે એમ પણ બને. આ હેતુ પાર પાડવા માટે સૂચનાઓના ક્રમનો પ્રવાહ બદલવો જરૂરી બને છે. પ્રોગ્રામમાં સૂચનાઓના ક્રમનો પ્રવાહ બદલી શકાય તે માટે સી ભાષા વિશિષ્ટ પ્રકારનાં વિધાનોની સુવિધા પૂરી પાડે છે. આ વિધાનોને નિર્ણય માળખાંનાં વિધાનો તરીકે ઓળખવામાં આવે છે.

સી ભાષામાં નિર્ણય માળખાંની જરૂરિયાત (Need for decision structure in C)

કોઈ શરતના પરિણામને આધારે પ્રોગ્રામના એક વિભાગ પરથી અન્ય વિભાગ પર જવા માટે નિર્ણય માળખાં મદદરૂપ બને છે. કેટલીકવાર નિર્ણય માળખાંનાં વિધાનોને પસંદગી માળખાંનાં વિધાનો (selective structure statements), શાખાકીય વિધાનો (branching statements) અથવા નિર્ણય લેનાર વિધાનો (decision making statements) તરીકે પણ ઓળખવામાં આવે છે. અમલીકરણના પ્રવાહને નિયંત્રિત કરતાં હોવાથી આ વિધાનોને નિયંત્રણ વિધાનો (control statements) પણ કહેવામાં આવે છે.

સી ભાષા મૂળભૂત રીતે બે પ્રકારનાં નિર્ણય માળખાં માટેનાં વિધાનો પૂરાં પાડે છે : ***if*** અને ***switch.*** પ્રોગ્રામમાં સૂચનાઓનો પ્રવાહ ***if*** અને ***switch*** વિધાનોની મદદથી કેવી રીતે બદલી શકાય તેનો આપણે આ પ્રકરણમાં અભ્યાસ કરીશું.

if વિધાન (The if statement)

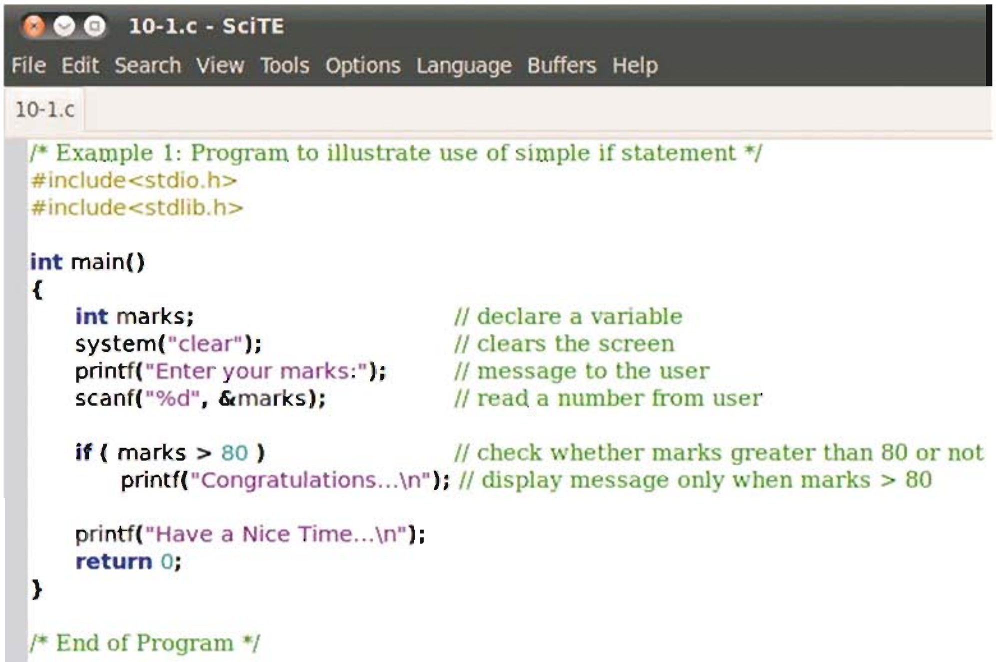
નિર્ણય માળખા માટેનું ***if*** એક એવું સક્ષમ વિધાન છે, જે સૂચનાઓના અમલને નિયંત્રિત કરવા માટે ઉપયોગમાં લઈ શકાય છે. ***if*** વિધાનનો ઉપયોગ નીચે જણાવેલ વિવિધ પ્રકારે કરી શકાય છે :

- સરળ if વિધાન (simple if statement)
- if-else વિધાન (if-else statement)
- નેસ્ટેડ if વિધાન (nested if statement)
- else-if લેડર વિધાન (else-if ladder statement)

સરળ if વિધાન (Simple if statement)

નિર્ણય માળખાનું સરળતમ સ્વરૂપ if વિધાન છે. નિર્ણયો લેવા માટે તથા પ્રોગ્રામના અમલીકરણનો પ્રવાહ બદલવા માટે આ વિધાનનો વારંવાર ઉપયોગ કરવામાં આવે છે.

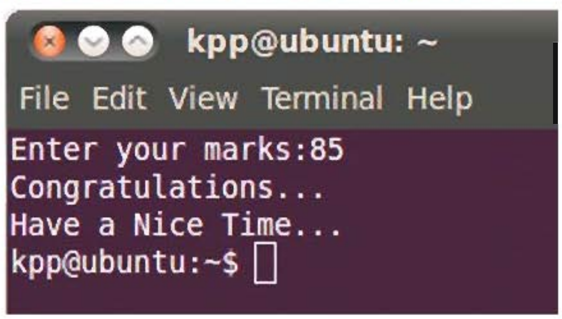
ઉદાહરણ 13.1 : આપણે એવો પ્રોગ્રામ સમજવાનો પ્રયત્ન કરીએ, જે ઉપયોગકર્તા પાસેથી કોઈ એક વિષયના ગુણ મેળવે અને જો તે 80થી વધુ હોય તો અભિનંદનનો સંદેશ દર્શાવે. ઉદાહરણ 13.1 માટેનું કોડ લિસ્ટિંગ આકૃતિ 13.1માં તથા તેનું પરિણામ આકૃતિ 13.2માં દર્શાવ્યું છે.



10-1.c - SciTE
File Edit Search View Tools Options Language Buffers Help
10-1.c

```
/* Example 1: Program to illustrate use of simple if statement */
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int marks;                      // declare a variable
    system("clear");                // clears the screen
    printf("Enter your marks:");    // message to the user
    scanf("%d", &marks);            // read a number from user

    if ( marks > 80 )               // check whether marks greater than 80 or not
        printf("Congratulations...\n"); // display message only when marks > 80

    printf("Have a Nice Time...\n");
    return 0;
}

/* End of Program */
```

આકૃતિ 13.1 : ઉદાહરણ 13.1 માટેનું કોડ-લિસ્ટિંગ



kpp@ubuntu: ~
File Edit View Terminal Help

```
Enter your marks:85
Congratulations...
Have a Nice Time...
kpp@ubuntu:~$
```

આકૃતિ 13.2 : ઉદાહરણ 13.1નું પરિણામ

સમજૂતી

main() વિધેયમાં આવેલ પ્રથમ વિધાન marks નામના ચલને ઘોષિત કરે છે. બીજું વિધાન સ્ક્રીન પરથી લખાણ દૂર કરે છે. ઉપયોગકર્તાને સંદેશ દર્શાવવા માટે printf વિધાનનો ઉપયોગ કર્યો છે. scanf વિધાનના ઉપયોગ દ્વારા ઉપયોગકર્તા પાસેથી કિંમત મેળવી તેનો marks નામના ચલમાં સંગ્રહ કરવામાં આવે છે. if વિધાન શરત ચકાસે છે : દાખલ કરવામાં આવેલ ગુણ 80થી વધુ છે? જો ટેસ્ટ એક્સ્પ્રેશન (marks > 80)નું પરિણામ True (સાચું) મળશે તો "Congratulations..." સંદેશ ધરાવતા printf વિધાનનો અમલ કરવામાં આવશે. જો શારતિક વિધાન (marks > 80)નું પરિણામ false (ખોટું) મળશે તો "Congratulations..." સંદેશ દર્શાવતા printf વિધાનનું અમલીકરણ મોકૂફ રાખવામાં આવશે. ત્યાર પછીનું printf વિધાન "Have a nice time..." સંદેશ હંમેશા દર્શાવશે.

સરળ if વિધાનની વાક્યરચના (Syntax of simple if statement)

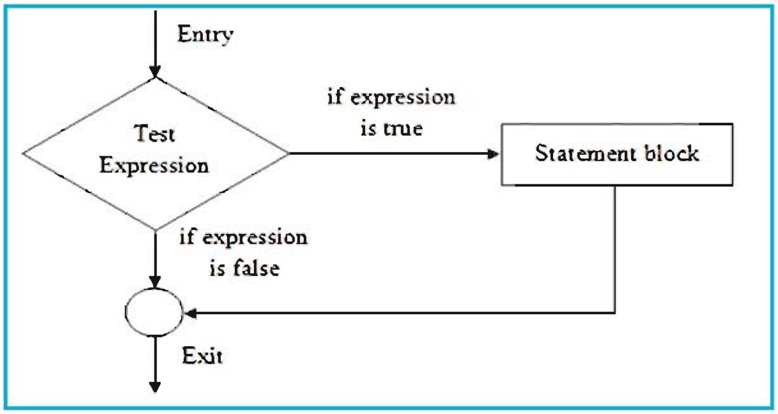
નીચેની વાક્યરચના દ્વારા સરળ if વિધાનનો ઉપયોગ કરી શકાય :

```
if (test expression)
{
    statement-block;
}
```

અહીં test expression એ સી ભાષામાં માન્ય એવી કોઈ પણ પદાવલિ છે. ટેસ્ટ ઍક્સ્પ્રેશનમાં તાર્કિક પદાવલિ (logical expression)નો ઉપયોગ કરવામાં આવે છે. statement-block તરીકે સી ભાષામાં માન્ય એવા કોઈ પણ એક કે વધુ વિધાનો હોઈ શકે છે. અહીં એ ધ્યાન આપો કે ટેસ્ટ ઍક્સ્પ્રેશનની લીટીને અર્ધવિરામ દ્વારા પૂર્ણ કરવામાં આવતી નથી.

આ વાક્યરચના દર્શાવે છે કે જો ટેસ્ટ ઍક્સ્પ્રેશન True હશે તો સ્ટેટમેન્ટ બ્લૉકનો અમલ કરવામાં આવશે અને જો ટેસ્ટ ઍક્સ્પ્રેશન False હશે તો સ્ટેટમેન્ટ બ્લૉકને અવગણીને પ્રોગ્રામમાં તેના પછી આવેલ સૂચનાઓનો અમલ કરવામાં આવશે.

સી પ્રોગ્રામિંગ ભાષા શૂન્યેતર અને ખાલી ન હોય તેવી **(non-null)** કિંમતોને True તરીકે મૂલવે છે તથા **શૂન્ય** કે ખાલી **(null)** કિંમતોને False તરીકે મૂલવે છે. આકૃતિ 13.3 સરળાંક વિધાનનો ફ્લો-ચાર્ટ દર્શાવે છે.

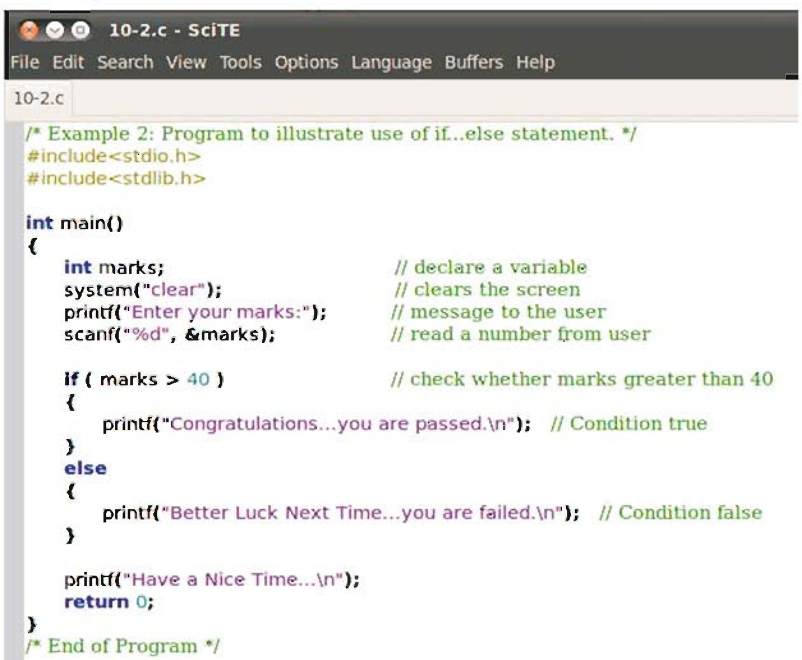


આકૃતિ 13.3 : સરળ if વિધાનનો ફ્લો-ચાર્ટ

if...else વિધાન (The if...else statement)

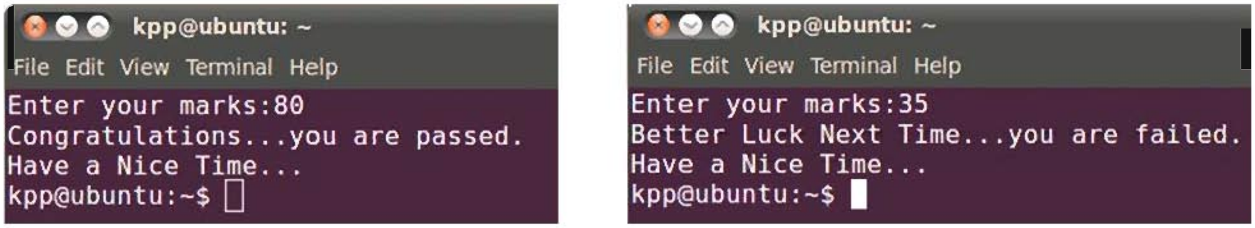
સરળ if વિધાનમાં માત્ર એક જ સ્ટેટમેન્ટ બ્લૉક હોય છે, અને જો ટેસ્ટ ઍક્સ્પ્રેશન સાચું હોય તો જ તેનો અમલ કરવામાં આવે છે. ટેસ્ટ ઍક્સ્પ્રેશન ખોટું હોય ત્યારે સરળ if વિધાન કોઈ કાર્યનો અમલ કરતું નથી. હવે, ટેસ્ટ ઍક્સ્પ્રેશનનો જવાબ False આવે ત્યારે જો કોઈ ક્રિયા અમલમાં મૂકવી હોય તો? ઉદાહરણ તરીકે, જો વિદ્યાર્થીના ગુણ 40થી વધુ હોય તો આપણે "Student Passed" સંદેશ દર્શાવવા માંગીએ છીએ અન્યથા "Student Failed" સંદેશ દર્શાવવામાં આવવો જોઈએ. આવા કિસ્સામાં if...else માળખું ઉપયોગી બને છે.

ઉદાહરણ 13.2 : આપણે એક એવો પ્રોગ્રામ સમજવાનો પ્રયત્ન કરીએ જે એક વિષયના ગુણ મેળવી, જો ગુણ 40થી વધુ હોય તો "Congratulations...You are passed" સંદેશ દર્શાવે, અન્યથા "Better Luck Next Time...You are failed" સંદેશ દર્શાવે. આકૃતિ 13.4માં ઉદાહરણ 13.2નું કોડ લિસ્ટિંગ આપેલ છે તથા તેનાં બે જુદાં જુદાં પરિણામ આકૃતિ 13.5માં દર્શાવ્યાં છે.




```c
/* Example 2: Program to illustrate use of if...else statement. */
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int marks;                          // declare a variable
    system("clear");                    // clears the screen
    printf("Enter your marks:");        // message to the user
    scanf("%d", &marks);                // read a number from user

    if ( marks > 40 )                   // check whether marks greater than 40
    {
        printf("Congratulations...you are passed.\n");   // Condition true
    }
    else
    {
        printf("Better Luck Next Time...you are failed.\n");   // Condition false
    }

    printf("Have a Nice Time...\n");
    return 0;
}
/* End of Program */
```

આકૃતિ 13.4 : ઉદાહરણ 13.2નું કોડ-લિસ્ટિંગ



આકૃતિ 13.5 : ઉદાહરણ 13.2નું પરિણામ

સમજૂતી

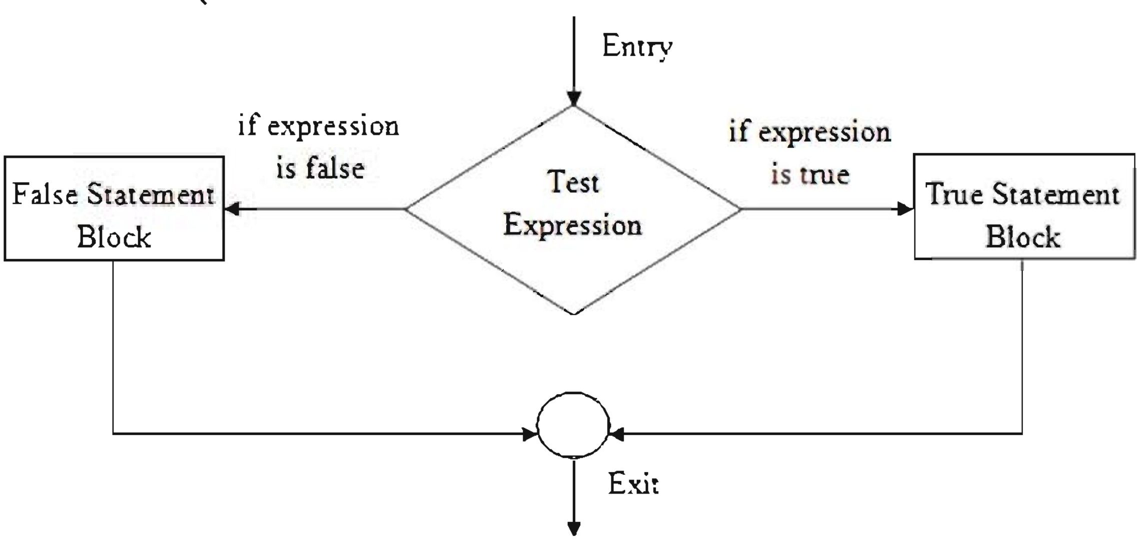
પ્રોગ્રામનું પ્રથમ વિધાન marks નામનો ચલ ઘોષિત કરે છે. બીજું વિધાન સ્ક્રીન પરથી તમામ લખાણ દૂર કરે છે. ત્રીજા અને ચોથા વિધાન દ્વારા જરૂરી સંદેશ દર્શાવી ઉપયોગકર્તા પાસેથી marks ચલની કિંમત મેળવવામાં આવે છે. આપેલ ગુણ 40થી વધુ છે કે નહીં તે ચોથા વિધાન દ્વારા ચકાસવામાં આવે છે. જો ઉપયોગકર્તા 40થી વધુ ગુણ ઉમેરશે તો "Congratulations...You are passed" સંદેશ દર્શાવવામાં આવશે, અન્યથા "Better Luck Next Time...You are failed" સંદેશ દર્શાવાશે. ત્યાર પછી આવેલું printf વિધાન હંમેશા "Have a nice time..." સંદેશ હંમેશાં રજૂ કરશે.

if...else વિધાનની વાક્યરચના (Syntax of if...else statement)

if...else વિધાનની વાક્યરચના નીચે આપેલ છે :

```
if (test expression)
{
        True statement-block;
}
else
{
        False statement-block;
}
```

અહીં test expression એ સી ભાષાની કોઈ પણ યોગ્ય પદાવલિ હોઈ શકે છે. જ્યારે test expressionનો જવાબ True મળશે ત્યારે નિયંત્રણનો પ્રવાહ True statement-block તરફ જશે. એથી વિપરીત જો ટેસ્ટ ઍક્સ્પ્રેશનનો જવાબ False મળશે તો, નિયંત્રણ પ્રવાહ False statement-block તરફ જશે. બંને કિસ્સામાં if વિધાનનો અમલ કર્યા બાદ નિયંત્રણને પ્રોગ્રામમાં આવેલા તે પછીના વિધાન તરફ મોકલવામાં આવશે. આ અભિગમ આકૃતિ 13.6માં if...else વિધાનના ફ્લોચાર્ટ દ્વારા સમજાવવામાં આવ્યો છે.



આકૃતિ 13.6 : if...else વિધાનનો ફ્લોચાર્ટ

નેસ્ટેડ if...else વિધાન (Nested if...else statement)

પ્રોગ્રામમાં ક્યારેક એકથી વધુ નિર્ણયોની શ્રેણીનો અમલ કરાવવાની પણ જરૂર પડે છે. આ હેતુ પાર પાડવા માટે if...else વિધાનની શ્રેણીનો 'નેસ્ટેડ' સ્વરૂપે ઉપયોગ કરી શકાય છે. નીચેની વાક્યરચના જુઓ. જેમાં એક if...else વિધાનનો ઉપયોગ અન્ય if...else વિધાનના બ્લૉક્ની અંદર કરવામાં આવ્યો છે.

```
if (test expression-1)
{
        if(test expression-2)
        {
              Statement block-1;
        }
        else
        {
              Statement block-2;
        }
}
else
{
        Statement block-3;
}
```

નેસ્ટેડ if વિધાનમાં અમલીકરણની તક નીચે દર્શાવેલ છે :

- જો ટેસ્ટ એક્સ્પ્રેશન-1 સાચું હશે તો ટેસ્ટ એક્સ્પ્રેશન-2 ચકાસવામાં આવશે.
- જો ટેસ્ટ એક્સ્પ્રેશન-2 પણ સાચું હશે તો સ્ટેટમેન્ટ બ્લૉક-1નો અન્યથા સ્ટેટમેન્ટ બ્લૉક-2નો અમલ કરવામાં આવશે.
- જો ટેસ્ટ એક્સ્પ્રેશન-1 ખોટું હોય તો સ્ટેટમેન્ટ બ્લૉક-3નો અમલ કરવામાં આવશે. અહીં એ નોંધવું જરૂરી છે કે સ્ટેટમેન્ટ બ્લૉક-3ના ભાગ તરીકે વધુ if...else વિધાનોનો ઉપયોગ કરી શકાય છે.

નોંધ : ઉપર જણાવેલ બંધારણ માત્ર ઉદાહરણ માટે છે. ઉપયોગકર્તા પોતાની જરૂરિયાત મુજબ તેને બદલી શકે છે. ઉદાહરણ તરીકે, નીચે જણાવેલ બંધારણ પણ યોગ્ય છે :

```
if (test expression-1)
{
        Statement block-1;
}
else
{
      if(test expression-2)
        {
              Statement block-2;
        }
        else
        {
              Statement block-3;
        }
}
```

ઉદાહરણ 13.3 : ઉપયોગકર્તા પાસેથી ત્રણ સંખ્યાઓ મેળવીને નેસ્ટેડ if વિધાનની મદદથી તેમાંથી સૌથી મોટી સંખ્યા દર્શાવવા માટેના સી પ્રોગ્રામને સમજવાનો પ્રયત્ન કરીએ. આકૃતિ 13.7માં ઉદાહરણ 13.3નું કોડ-લિસ્ટિંગ આપવામાં આવ્યું છે તથા આકૃતિ 13.8 તેનું પરિણામ દર્શાવે છે.

```
10-3.c - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
10-3.c

/* Example 3: Program to demonstrate the use of nested if statement. */
#include<stdio.h>

int main()
{
    int number1, number2, number3;        //declares three variables
    printf("Enter three numbers:");       // message to the user
    scanf("%d %d %d", &number1, &number2, &number3); //read 3 numbers

    if ( number1 > number2 )              // test condition 1
    {
        if ( number1 > number3 )          // test condition 2
           printf("The largest number is %d \n", number1);
        else
          printf("The largest number is %d \n", number3);
    }
    else
    {
        if ( number3 > number2 )          // test condition 3
           printf("The largest number is %d \n", number3);
        else
          printf("The largest number is %d \n", number2);
    }
    return 0;
}
/* End of Program.*/
```

આકૃતિ 13.7 : ઉદાહરણ 13.3નું કોડ-લિસ્ટિંગ

```
kpp@ubuntu: ~
File  Edit  View  Terminal  Help
kpp@ubuntu:~$ gcc 10-3.c
kpp@ubuntu:~$ ./a.out
Enter three numbers:4 6 8
The largest number is 8
kpp@ubuntu:~$
```

આકૃતિ 13.8 : ઉદાહરણ 13.3નું પરિણામ

સમજૂતી

main() વિધેયની અંદર આવેલું પ્રથમ વિધાન ત્રણ ચલ ઘોષિત કરશે. બીજું વિધાન ઉપયોગકર્તા સમક્ષ સંદેશ દર્શાવશે. ત્રીજું વિધાન ઉપયોગકર્તાને ત્રણ કિંમતો ઉમેરવાની પરવાનગી આપશે, જેને number1, number2 અને number3 નામના ચલમાં સંગૃહીત કરવામાં આવશે. ચોથા વિધાન (ટેસ્ટ કન્ડિશન-1) દ્વારા ચકાસણી કરવામાં આવશે કે number1ની કિંમત number2થી વધુ છે કે નહીં. જો ટેસ્ટ કન્ડિશન-1 સાચી હશે તો અંદરના if વિધાન(ટેસ્ટ કન્ડિશન-2)નો અમલ કરી number1 > number3 છે કે નહીં તે ચકાસવામાં આવશે. ટેસ્ટ કન્ડિશન-2ના સાચા કે ખોટા પરિણામને આધારે અંદરનું if વિધાન ઉપયોગકર્તા સમક્ષ સંદેશ રજૂ કરશે. પરંતુ જો ચોથું વિધાન (ટેસ્ટ કન્ડિશન-1) ખોટું પડશે તો બહારનાં if વિધાનનો else વિભાગ અમલમાં મૂકાશે. અહીં, ટેસ્ટ કન્ડિશન-3 ચકાસશે કે number3 > number2 છે કે નહીં, અને તેના સાચા કે ખોટા પરિણામને આધારે ઉપયોગકર્તા સમક્ષ સંદેશ દર્શાવવામાં આવશે.

else...if લેડર વિધાન (The else...if ladder statement)

એકથી વધારે નિર્ણયોનું મૂલ્યાંકન કરવા માટે એક કરતાં વધુ if વિધાનોની જરૂર પડે છે. આ માટે એક અન્ય બંધારણ પણ ઉપલબ્ધ છે. જ્યારે અન્ય if વિધાનોનો નેસ્ટેડ ifના else (false) બ્લૉકમાં ઉપયોગ કરવામાં આવે ત્યારે તે else...if લેડર વિધાન રજૂ કરે છે.

આપેલ વિષયના ગુણ અનુસાર વિદ્યાર્થીને વર્ગ / શ્રેણી આપવા માટેના પ્રોગ્રામમાં else...if લેડરનો ઉપયોગ કરીએ. શ્રેણી આપવા માટેના પ્રોગ્રામમાં આપણે કોષ્ટક 13.1માં આવેલ નિયમોનો ઉપયોગ કરીશું.

ગુણ-વિસ્તાર	વર્ગ / શ્રેણી
70 – 100	First Class with Distinction
60 – 69	First Class
50 – 59	Second Class
40 – 49	Pass Class
0 – 39	Fail

કોષ્ટક 13.1 : નમૂનારૂપ વર્ગના નિયમો

ઉદાહરણ 13.4 : ઉપયોગકર્તા પાસેથી ગુણ મેળવી તેના પરથી વર્ગ / શ્રેણી શોધી આપે તેવા એક C પ્રોગ્રામને સમજવાનો પ્રયત્ન કરીએ. આકૃતિ 13.9 ઉદાહરણ 13.4નું કોડ લિસ્ટિંગ દર્શાવે છે તથા આકૃતિ 13.10માં તેનાં બે પરિણામ દર્શાવ્યાં છે.

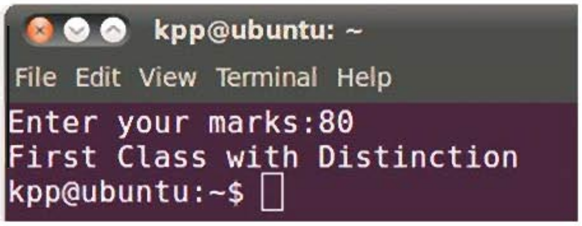
10-4.c - SciTE

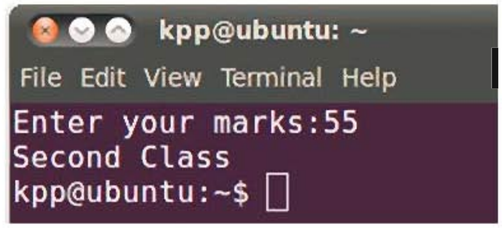
File Edit Search View Tools Options Language Buffers Help

10-4.c

```c
/* Example 4: Program to illustrate use of else...if ladder statement. */
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int marks;                                  // declare a variable
    system("clear");                            //clears the screen
    printf("Enter your marks:");                // message to the user
    scanf("%d", &marks);                        // read a number from user

    if ( marks > 69 )                           // check condition 1
       printf("First Class with Distinction \n");
    else if ( marks > 59 )                      // check condition 2
           printf("First Class \n");
        else if ( marks > 49 )                  // check condition 3
               printf("Second Class \n");
            else if ( marks > 39 )              // check condition 4
                   printf("Pass Class \n");
                else
                   printf("Fail \n");              // default else statement
     return 0;
}
/* End of Program */
```

આકૃતિ 13.9 : ઉદાહરણ 13.4નું કોડ લિસ્ટિંગ





આકૃતિ 13.10 : ઉદાહરણ 13.4નું પરિણામ

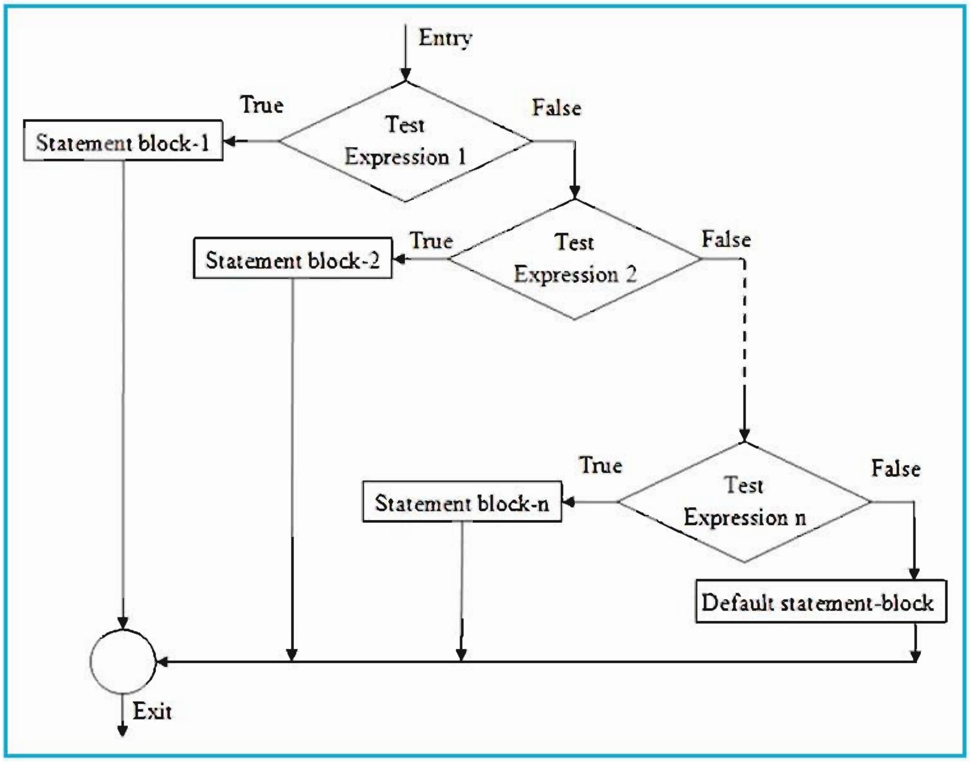
સમજૂતી

main() વિધેય પછીનું પ્રથમ, બીજું, ત્રીજું અને ચોથું વિધાન અનુક્રમે ચલ ઘોષિત કરે છે, સ્ક્રીન ચોખ્ખી કરે છે, સંદેશ દર્શાવે છે અને ગુણ સ્વીકારે છે. પાંચમું વિધાન પ્રોગ્રામની પ્રથમ શરત marks >69 તપાસે છે. જો શરત સાચી હોય તો "First Class with Distinction" સંદેશ દર્શાવી પ્રોગ્રામ પૂર્ણ કરવામાં આવે છે. જો પ્રથમ શરતનું પરિણામ False મળશે તો બીજી શરત ચકાસવામાં આવશે. જો બીજી શરત ખોટી હશે તો ત્રીજી ને એમ ક્રમાનુસાર પ્રવાહ આગળ વધશે. જો આપેલ ચારેય શરતોનું પરિણામ False મળશે તો તે કિસ્સામાં પૂર્વનિર્ધારિત else બ્લૉક્નો અમલ કરી "Fail" સંદેશ દર્શાવવામાં આવશે.

else-if લેડર વિધાનની વાક્યરચના (Syntax of else...if ladder statement)

```
if (test expression-1)
        Statement block-1;

else if (test expression-2)
               Statement block-2;

     else if (test expression-3)
                    Statement block-3;
             .....
             .....
            else if (test expression-n)
       Statement block-n;
                     else
                           Default-statement-block;
program-statement-x;
```

આ બંધારણને else-if લેડર તરીકે ઓળખવામાં આવે છે. લેડરમાં ટેસ્ટ એક્સ્પ્રેશનનું ઉપરથી નીચેના ક્રમમાં મૂલ્યાંકન કરવામાં આવે છે. જ્યારે કોઈ પણ ટેસ્ટ એક્સ્પ્રેશન True પરિણમે ત્યારે તેની સાથે સંકળાયેલાં વિધાનોના બ્લૉક્નો અમલ કરવામાં આવે છે. ત્યાર પછી લેડરના બાકીના વિભાગને કુદાવીને નિયંત્રણ program-statement-x તરફ વાળવામાં આવે છે. જો તમામ ટેસ્ટ-એક્સ્પ્રેશનનું મૂલ્યાંકન Falseમાં પરિણમે તો નિયંત્રણને અંતિમ else વિભાગના Default-statement-block તરફ લઈ જવામાં આવશે. else-if લેડર વિધાનનો ફ્લોચાર્ટ આકૃતિ 13.11માં દર્શાવ્યો છે.



આકૃતિ 13.11 : else-if લેડર વિધાનનો ફ્લો-ચાર્ટ

સ્વિચ વિધાન (The switch statement)

પ્રોગ્રામના અમલીકરણ દરમિયાન વિધાનોના જુદા-જુદા બ્લૉકને પસંદ કરવા માટે આપણે if-else વિધાનનો ઉપયોગ કર્યો. જો કે, જ્યારે અનેક if-else વિધાનોનો ઉપયોગ કરવામાં આવે ત્યારે પ્રોગ્રામ જટિલ બની જાય છે. પ્રોગ્રામ વાંચવામાં અને તેનો તર્ક ઉકેલવામાં મુશ્કેલી પડે છે. સી ભાષા પ્રોગ્રામને સરળ બનાવવા માટે switch નામના એક આંતરપ્રસ્થાપિત બહુમાર્ગીય (multiway) નિર્ણય માટેનું વિધાન પૂરું પાડે છે. જ્યારે આપેલ અનેક પસંદગીઓમાંથી કોઈ એક ક્રિયા પસંદ કરવાની હોય ત્યારે switch વિધાન ઘણું ઉપયોગી બને છે.

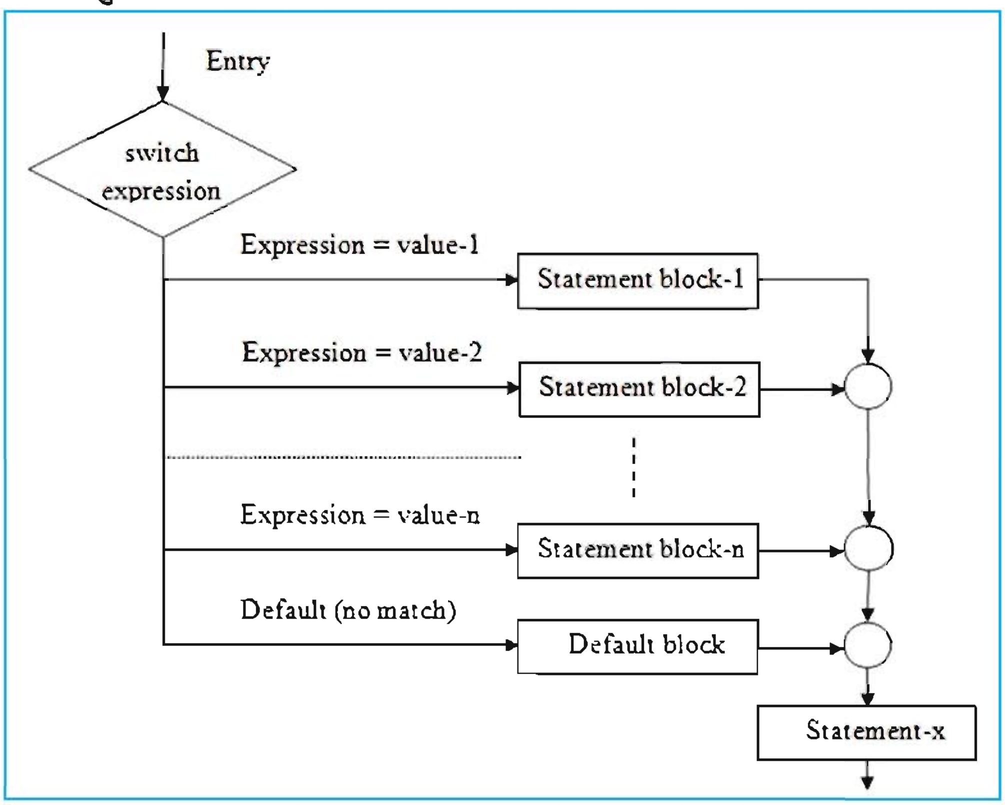
switch વિધાનની વાક્યરચના (Syntax of the switch statement)

```
switch (expression)
{
        case value-1:
                statement block-1;
                break;
        case value-2:
                statement block-2;
                break;
        ....
        ....
        case value-n:
                statement block-n;
                break;
        default:
                default statement block;
                break;
}
statement-x;
```

switch વિધાનને સંબંધિત એવા નીચેના અગત્યના મુદ્દા નોંધી લો :

- switch વિધાન એક આર્ગ્યુમેન્ટ(પદાવલિ અથવા ચલના નામ)નો ઉપયોગ કરે છે. આ આર્ગ્યુમેન્ટને switch વિધાનની અંદર આવેલા અનેક કેસ વિકલ્પો સાથે સરખામણી માટે ચકાસવામાં આવે છે. આર્ગ્યુમેન્ટ એ પૂર્ણાંક કે અક્ષર ધરાવતો ચલ કે પદાવલિ હોઈ શકે.
- દરેક case વિકલ્પ અચળ કે અચળ પદાવલિ ધરાવે છે. આ અચળને કેસ લેબલ તરીકે ઓળખવામાં આવે છે તથા તેના અંતમાં વિસર્ગ (:) નિશાની મૂકવામાં આવે છે. દરેક કેસ લેબલ અનન્ય હોવા જરૂરી છે. કોઈ પણ બે કેસ લેબલની કિંમતો સમાન હોઈ શકે નહીં.
- Statement block-1, statement block-2 વગેરે વિધાનોની યાદી છે જેમાં શૂન્ય કે વધુ વિધાનોનો સમાવેશ થયેલો હોઈ શકે. આ વિધાનોના બ્લૉકને છગડિયા કૌંસમાં મૂકવાની જરૂર નથી.
- જ્યારે switch વિધાનનો અમલ કરવામાં આવે ત્યારે તે સૌપ્રથમ પદાવલિનું મૂલ્યાંકન કરે છે અને પછી ઉપરથી નીચેના ક્રમમાં કેસ અચળ સાથે તેની સરખામણી કરે છે. જ્યારે કિંમતની સરખામણી આપેલ પદાવલિ સાથે મળે ત્યારે તે કેસને સંબંધિત વિધાનોના વિભાગનો અમલ કરવામાં આવશે.
- switch વિધાનમાં દરેક કેસ વિભાગના વિધાનો પછી આપવામાં આવેલ break વિધાન જે-તે કેસનો અંત સૂચવે છે અને નિયંત્રણને switch વિધાનની બહાર લઈ જાય છે. ત્યાર પછી નિયંત્રણનો પ્રવાહ પ્રોગ્રામ પછીના વિધાન (statement-x) તરફ જાય છે.
- default એ મરજિયાત કેસ છે. અમલીકરણ દરમિયાન જ્યારે કોઈ પણ કેસની સરખામણી મેળવી શકાતી નથી, ત્યારે default વિધાનોના વિભાગનો અમલ કરવામાં આવે છે. default વિધાનનો ઉપયોગ એક જ વખત કરી શકાય છે તથા તેને switch વિધાનમાં કોઈ પણ સ્થાને લખી શકાય છે. પરંતુ સામાન્ય રીતે આપણે તેને switch વિધાનના અંત ભાગમાં ઉમેરતા હોઈએ છીએ.

switch વિધાનનું અમલીકરણ આકૃતિ 13.12માં દર્શાવેલ ફ્લૉ-ચાર્ટ દ્વારા સમજાવવામાં આવ્યું છે. આકૃતિમાં એવું માની લેવામાં આવ્યું છે કે દરેક કેસ વિભાગમાં અંતિમ વિભાગ તરીકે break વિધાન છે.



આકૃતિ 13.12 : switch વિધાનનો ફ્લૉ-ચાર્ટ

ઉદાહરણ 13.5 : હવે આપણે આપેલ એક અંક (0થી 9) પરથી તેને સુસંગત એવા શબ્દની રજૂઆત કરે તેવા સી પ્રોગ્રામને સમજીએ. આકૃતિ 13.13માં ઉદાહરણ 13.5નું કોડ-લિસ્ટિંગ આપેલું છે તથા આકૃતિ 13.14 તેનું પરિણામ દર્શાવે છે.

```c
/* Example 5: Program to illustrate use of switch statement.*/
#include<stdio.h>
int main()
{
    int number;
    printf("Enter a single digit: ");          // message to the user
    scanf("%d", &number);                      // read a number from user
    switch (number)
     {
        case 0:
               printf("0 = Zero \n");          break;
        case 1:
               printf("1 = One \n");           break;
        case 2:
               printf("2 = Two \n");           break;
        case 3:
               printf("3 = Three \n");         break;
        case 4:
               printf("4 = Four \n");          break;
        case 5:
               printf("5 = Five \n");          break;
        case 6:
               printf("6 = Six \n");           break;
        case 7:
               printf("7 = Seven \n");         break;
        case 8:
               printf("8 = Eight \n");         break;
        case 9:
               printf("9 = Nine \n");          break;
        default:
               printf("Out of range number entered. \n");
     }
     return 0;
} /* End of Program */
```

આકૃતિ 13.13 : ઉદાહરણ 13.5નું કોડ-લિસ્ટિંગ

```
kpp@ubuntu:~$ gcc 10-5.c
kpp@ubuntu:~$ ./a.out
Enter a single digit: 5
5 = Five
kpp@ubuntu:~$ ./a.out
Enter a single digit: 105
Out of range number entered.
kpp@ubuntu:~$
```

આકૃતિ 13.14 : ઉદાહરણ 13.5નું પરિણામ

સમજૂતી

main() વિધેયમાં પ્રથમ, બીજું અને ત્રીજું વિધાન અનુક્રમે ચલ ઘોષિત કરે છે, સંદેશ દર્શાવે છે તથા ઉપયોગકર્તા પાસેથી એક અંક મેળવે છે. ચોથા વિધાનમાં switch વિધાનની આર્ગ્યુમેન્ટ તરીકે number ચલમાં સંગૃહીત કિંમત અનુસાર સુસંગત કેસના વિભાગનો અમલ કરવામાં આવશે. જો કોઈ પણ caseમાં સુસંગતતા નહીં મળે તો default વિભાગ "Out of range number entered" સંદેશ દર્શાવશે અને પ્રોગ્રામ પૂરો થશે.

હવે, switch વિધાનમાં અક્ષર પ્રકારના અચળનો ઉપયોગ કેસ લેબલ તરીકે કેવી રીતે કરી શકાય તે ઉદાહરણ 13.6નો ઉપયોગ કરીને સમજીએ.

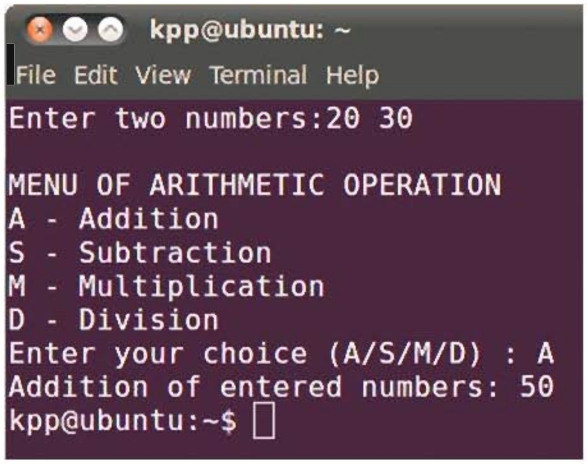
ઉદાહરણ 13.6 : ઉપયોગકર્તા પાસેથી બે અંકો મેળવી મેનૂની મદદથી ગાણિતિક પ્રક્રિયક(arithmetic operator)ની પસંદગી માંગો. આપેલ પસંદગી પ્રમાણે આપેલ અંકો પર ગાણિતિક પ્રક્રિયા કરી પરિણામ દર્શાવો. આકૃતિ 13.15માં ઉદાહરણ 13.6નું કોડ લિસ્ટિંગ આપવામાં આવ્યું છે તથા આકૃતિ 13.16 તેનું પરિણામ દર્શાવે છે.

10-6.c - SciTE

File Edit Search View Tools Options Language Buffers Help

```c
/* Example 6: Program to illustrate use of switch statement using character case label.*/
#include<stdio.h>
#include<stdlib.h>
int main() {
      char choice;
      int number1, number2;
      system("clear");
      printf("Enter two numbers:");                  // message to the user
      scanf("%d %d", &number1, &number2);    // read two numbers from user
      printf("\nMENU OF ARITHMETIC OPERATION \n");
      printf("A - Addition \n");
      printf("S - Subtraction \n");
      printf("M - Multiplication \n");
      printf("D - Division \n");
      printf("Enter your choice (A/S/M/D) : ");      // message to the user
      scanf(" %c", &choice);                          // read choice from user
      switch (choice)
      {
        case 'A':
            printf("Addition of entered numbers: %d \n", number1 + number2);
            break;
        case 'S':
            printf("Subtraction of entered number: %d \n", number1 - number2);
            break;
        case 'M':
            printf("Multiplication of entered number: %d \n", number1 * number2);
            break;
        case 'D':
            printf("Division of entered number: %d \n", number1 / number2);
            break;
         default:
            printf("Invalid menu choice. \n");
            break;
       }
      return 0;
} /* End of Program */
```

આકૃતિ 13.15 : ઉદાહરણ 13.6નું કોડ-લિસ્ટિંગ



આકૃતિ 13.16 : ઉદાહરણ 13.6નું પરિણામ

સમજૂતી

main( ) વિધેયનું ચોથું અને પાંચમું વિધાન અનુક્રમે સંદેશ દર્શાવશે અને ઉપયોગકર્તા પાસેથી બે કિંમતો મેળવશે. વિધાન છ થી દસ દ્વારા સ્ક્રીન પર મેનૂ વિકલ્પો દર્શાવવામાં આવશે. અગિયારમા અને બારમા વિધાનમાં અનુક્રમે સંદેશ દર્શાવશે અને ઉપયોગકર્તા પાસેથી મેનૂની પસંદગી મેળવવામાં આવશે. ઉપયોગકર્તાએ કરેલ પસંદગીને આધારે તેને સુસંગત કેસ વિભાગનો અમલ કરવામાં આવશે. જો કોઈ પણ કેસમાં સરખામણી નહીં મળે તો પૂર્વનિર્ધારિત (default) વિભાગ "Invalid menu choice" સંદેશ દર્શાવશે અને પ્રોગ્રામ પૂર્ણ થશે. આ પ્રોગ્રામમાં 'શૂન્ય વડે ભાગાકાર' જેવી ભૂલની સ્થિતિ તપાસવા માટે યોગ્ય જગ્યાએ વધારાનાં વિધાનો પણ ઉમેરી શકાય.

સંયોજિત વિધાનો (Compound statement)

સી ભાષામાં સંયોજન સંબંધિત ચકાસણીઓ (compound relational tests) અથવા એકથી વધુ પદાવલિઓની ચકાસણી (multiple test expression) પાર પાડવા માટેની જરૂરી સુવિધા પણ આપવામાં આવી છે. સંયોજન સંબંધિત ચકાસણીઓ એટલે એક કે વધુ સંબંધિત ચકાસણીઓને તાર્કિક AND અથવા તાર્કિક OR પ્રક્રિયક દ્વારા જોડવી. આ તાર્કિક પ્રક્રિયકો અનુક્રમે && અને || અક્ષર-યુગ્મ દ્વારા રજૂ કરવામાં આવે છે. સંયોજન સંબંધિત ચકાસણીઓ પ્રોગ્રામમાં આવેલ if...else વિધાનોની સંખ્યાને ઘટાડવામાં મદદરૂપ બને છે.

સારાંશ

આ પ્રકરણમાં આપણે સી ભાષાની વિશિષ્ટ પ્રકારની સુવિધાઓ વિશે અભ્યાસ કર્યો, કે જેના દ્વારા પ્રોગ્રામમાં સૂચનાઓના ક્રમનો પ્રવાહ બદલી શકાય છે. પ્રોગ્રામમાં આવેલ સૂચનાઓના ક્રમનો પ્રવાહ બદલવા માટે બે નિર્ણય માળખાં : if અને switchનો ઉપયોગ કેવી રીતે કરવો તે આપણે શીખ્યા.

સ્વાધ્યાય

1. if...else વિધાનની સરખામણીમાં if વિધાનની મર્યાદા જણાવો.
2. નેસ્ટેડ if (nested if) એટલે શું ?
3. switch વિધાનનો ઉપયોગ સલાહભર્યો ગણાય તેવું એક યોગ્ય ઉદાહરણ આપો.
4. switch બંધારણમાં breakનું મહત્ત્વ જણાવો.
5. યોગ્ય ઉદાહરણ સાથે else...if લેડર વિધાન સમજાવો.

6. નીચેનાં વિધાનો ખરાં છે કે ખોટાં તે જણાવો :

(a) if વિધાનને switch વિધાનની અંદર લખી શકાય છે.

(b) જ્યારે if વિધાનના ટેસ્ટ-એક્સપ્રેશનનું પરિણામ Flase આવે છે ત્યારે ifની અંદર આવેલાં વિધાનોને અવગણવામાં આવે છે.

(c) switch વિધાનમાં default વિધાન ફરજિયાત છે.

(d) switch વિધાનમાં default વિધાન હંમેશા અંતિમ વિધાન હોવું જોઈએ.

(e) break વિધાન પ્રોગ્રામના અમલીકરણને અટકાવે છે.

(f) if વિધાનની અંદર switch વિધાનને ઉમેરી શકાય છે.

7. આપેલ વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરો :

(1) નીચેના if વિધાનના અમલ બાદ flag ચલની કિંમત કઈ હશે ?

```
int flag=0;
if(5 < 8) {flag=1;}
```

(a) 0 (b) 1 (c) 5 (d) 8

(2) નીચેના switch વિધાનના અમલ પછી ચલ 's' ની કિંમત કઈ હશે ?

```
x = 3;
switch (x) {
   case 1 : s = 'A'; break;
   case 2 : s = 'B'; break;
   case 3 : s = 'C'; break;
   default : s = 'D'; break;
}
```

(a) A (b) B (c) C (d) D

(3) નીચેનાં વિધાનોના અમલ બાદ ચલ sumની કિંમત કઈ હશે ?

```
int number1=5, number2=10, sum=0;
if (number1 > number2)
        sum = sum + number1;
else
        sum = sum + number2;
```

(a) 0 (b) 5 (c) 10 (d) 12

(4) નીચે આપેલા પ્રોગ્રામ-ખંડનો અમલ કરવામાં આવે તો શું પરિણામ મળશે ?

```
int number1=10, number2=20;
if ((number1+number2) > 35 || (number1>number2))
        printf("%d", number1);
else
        printf("%d", number2);
```

(a) 10 (b) 20 (c) 35 (d) ભૂલનો સંદેશ

(5) નીચે આપેલા પ્રોગ્રામ-ખંડનો અમલ કરવામાં આવે તો શું પરિણામ મળશે ?

```
char chr='A';
switch (chr)
{
        case 'A' : printf("A"); break;
        case 'B' : printf("B"); break;
        case 'C' : printf("C"); break;
}
```

(a) A (b) B (c) C (d) ભૂલનો સંદેશ

પ્રાયોગિક સ્વાધ્યાય

નીચેનાં કાર્યો માટે સી પ્રોગ્રામ લખો :

1. આપેલ સંખ્યા ધન છે કે ઋણ તે તપાસો.
2. વ્યક્તિની ઉંમર મેળવી તે મતાધિકારને પાત્ર છે કે નહીં તે દર્શાવો.
3. ઉપયોગકર્તા પાસેથી ત્રણ સંખ્યાઓ મેળવી નેસ્ટેડ ifની મદદથી તેમાંથી નાનામાં નાની સંખ્યા શોધો.
4. આપેલ સંખ્યા એકી છે કે બેકી તે તપાસો.
5. switch વિધાનની મદદથી આપેલ અક્ષર સ્વર છે કે નહીં તે ચકાસો.

લૂપ નિયંત્રણ માળખાં

સી પ્રોગ્રામમાં ક્યારેક એવું બની શકે કે જેમાં એકસરખાં વિધાનોના ખંડને એકથી વધુ વખત અમલમાં મૂકવાનો હોય. એક વિધાન કે વિધાનોના સમૂહને એકથી વધુ વાર અમલમાં મૂકવા માટે પ્રોગ્રામરોને તમામ પ્રોગ્રામિંગ ભાષાઓ દ્વારા લૂપ નિયંત્રણ માળખાં (જે **લૂપિંગ / looping** તરીકે પણ ઓળખાય છે તે) પૂરાં પાડવામાં આવે છે. આ પ્રકરણમાં આપણે સી પ્રોગ્રામિંગ ભાષા દ્વારા પૂરા પાડવામાં આવતા લૂપ (પુનરાવર્તિત) નિયંત્રણ માળખાં વિશે ચર્ચા કરીશું. અહીં નિયંત્રણ માળખાં(control structure)નો અર્થ એ થાય છે કે અમલીકરણનો પ્રવાહ ક્રમબદ્ધ હોવો જરૂરી નથી તથા પ્રોગ્રામમાં આપવામાં આવેલી શરત અનુસાર નિયંત્રણ કોઈપણ વિધાન તરફ મોકલી શકાય છે.

લૂપિંગમાં કોઈ નિર્ગમ શરત (exit condition) ન સંતોષાય ત્યાં સુધી વિધાનોની શ્રેણીનો અમલ કરવામાં આવે છે. લૂપિંગ માળખાં બે વિભાગોમાં રચવામાં આવે છે : લૂપનો મુખ્ય ભાગ (body of loop) અને નિયંત્રણ વિધાનો (control statement). નિયંત્રણ વિધાનના સ્થાનને આધારે લૂપને **પ્રવેશ નિયંત્રણ (entry controlled)** લૂપ અને **નિર્ગમ નિયંત્રણ (exit controlled)** લૂપ એમ બે પ્રકારોમાં વર્ગીકૃત કરી શકાય. નિવેશ નિયંત્રણ લૂપમાં લૂપના મુખ્ય ભાગમાં આવેલાં વિધાનોના અમલ પહેલાં નિર્ગમ / નિયંત્રણ શરત ચકાસવામાં આવે છે. જ્યારે, નિર્ગમ નિયંત્રણ લૂપમાં લૂપના મુખ્ય ભાગમાં આવેલાં વિધાનોના અમલ બાદ નિર્ગમ/નિયંત્રણ શરતને ચકાસવામાં આવે છે. આનો અર્થ એ થાય કે નિર્ગમ નિયંત્રણ લૂપમાંથી બહાર નીકળતાં પહેલાં લૂપનાં વિધાનોનો અમલ ઓછામાં ઓછી એક વાર તો કરવામાં આવશે જ. આ માટેનું વધુ વિસ્તૃત વિવરણ જુદા જુદા પ્રકારના લૂપની સમજૂતી વખતે આપવામાં આવ્યું છે.

હવે, સી ભાષામાં ઉપલબ્ધ નીચેનાં ત્રણ લૂપ નિયંત્રણ માળખાં સંબંધિત વાક્યરચના અને સામાન્ય નિયમો વિશે ચર્ચા કરીએ :

- for
- while
- do…while

for લૂપ (The for loop)

વિધાનોના સમૂહનો નિશ્ચિત સમય સુધી અમલ કરવા માટે સામાન્ય રીતે *for* લૂપનો ઉપયોગ કરવામાં આવે છે. *for* લૂપને વધુ ક્રિયાશીલ (dynamic) બનાવવા માટે તેમાં નિર્ગમ શરતનો ઉપયોગ કરી શકાય છે. ઉદાહરણ 14.1 દ્વારા સરળ *for* લૂપ વિધાનનો ઉપયોગ સમજવાનો પ્રયત્ન કરીએ. આ ઉદાહરણ સરળ *for* લૂપ વિધાનનો ઉપયોગ કરી પાંચ " * " દર્શાવે છે. આકૃતિ 14.1માં ઉદાહરણ 14.1નું કોડ લિસ્ટિંગ અને પરિણામ આપવામાં આવ્યું છે.

11-1.c - SciTE

File Edit Search View Tools Options Language Buffers Help

11-1.c

```c
/* Example 1: Program to illustrate simple for loop statement.*/
#include<stdio.h>

int main()
{
    int count;                                  // declaration of variables
    for ( count = 0; count < 5 ; count++)  // for loop body repeats 5 times
        printf(" * ");
    return 0;
}
/*End of Program */
```

```
gcc 11-1.c
>gcc 11-1.c
>Exit code: 0
./a.out
>./a.out
 * * * * * >Exit code: 0
```

આકૃતિ 14.1 : ઉદાહરણ 14.1નું કોડ-લિસ્ટિંગ અને પરિણામ

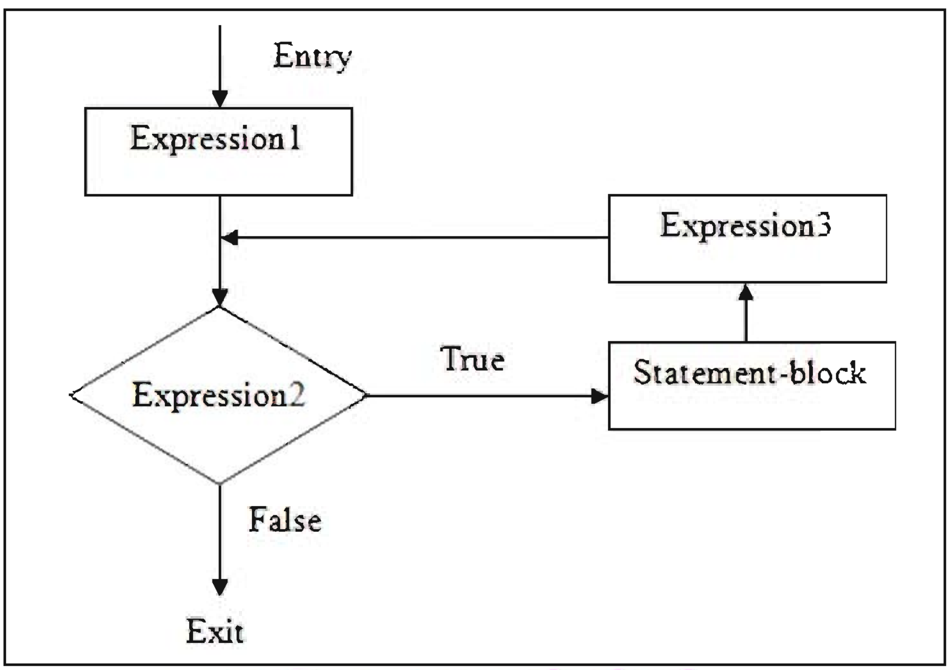
સમજૂતી

પ્રોગ્રામની શરૂઆતમાં count નામનો એક પૂર્ણાંક ચલ ઘોષિત કરવામાં આવ્યો છે. for લૂપ માટે count ચલનો ઉપયોગ ગણતરી માટેના ચલ (counter variable) તરીકે કરવામાં આવ્યો છે. શરૂઆતમાં count ચલની કિંમત શૂન્ય આપવામાં આવી છે. ત્યારબાદ બીજી પદાવલિ (count <5)ને ચકાસવામાં આવે છે. જો તેનું મૂલ્યાંકન સાચું (true) થાય તો for લૂપનાં વિધાનોનો અમલ કરવામાં આવે છે. આ ઉદાહરણમાં અહીં printf વિધાનની મદદથી સ્ક્રીન પર એક " * " દર્શાવવામાં આવશે. ત્યારપછી for લૂપની ત્રીજી પદાવલિ (count++)નો અમલ કરી countની કિંમતમાં 1નો વધારો કરવામાં આવશે. ફરી બીજી પદાવલિ (count <5)નું મૂલ્યાંકન કરવામાં આવશે અને તેના પરિણામને આધારે for લૂપનાં વિધાનોનો અમલ કરવામાં આવશે. આ પ્રોગ્રામમાં printf વિધાનનો અમલ પાંચ વખત કરવામાં આવશે અને સ્ક્રીન પર પાંચ વાર ફૂદડીની નિશાની " * * * * * " દર્શાવવામાં આવશે.

for લૂપની વાક્યરચના (Syntax of for loop)

```
for (expression1; expression2; expression3)
{
        statement-block;
}
```

for લૂપના મથાળામાં અર્ધવિરામથી જુદી પાડેલી ત્રણ પદાવલિઓ કૌંસમાં મૂકવામાં આવે છે. આ તમામ પદાવલિઓ મરજિયાત છે. સ્ટેટમેન્ટ બ્લૉકના નામે ઓળખાતા *for* લૂપના મુખ્ય ભાગમાં એક વિધાન કે એકથી વધુ સંમિશ્રિત (compound) વિધાનો હોઈ શકે છે. નીચે આપેલ આકૃતિ 14.2માં *for* લૂપનો ફ્લો-ચાર્ટ દર્શાવ્યો છે :



આકૃતિ 14.2 : for લૂપનો ફ્લો ચાર્ટ

આકૃતિ 14.2માં દર્શાવ્યા મુજબ for લૂપના અમલીકરણનાં સોપાન નીચે પ્રમાણે છે :

સોપાન 1 : પદાવલિ-1(expreson1)નું મૂલ્યાંકન કરો. for લૂપ શરૂ કરવામાં આવે ત્યારે પદાવલિ-1નો અમલ એક જ વાર કરવામાં આવે છે.

સોપાન 2 : પદાવલિ-2નું મૂલ્યાંકન કરો. જો પરિણામ ખોટું (false) મળે તો લૂપને અટકાવો.

સોપાન 3 : જો પદાવલિ-2નું પરિણામ સાચું (true) મળે તો લૂપનાં વિધાનોનો અમલ કરો.

સોપાન 4 : પદાવલિ-3નું મૂલ્યાંકન કરો.

સોપાન 5 : સોપાન 2 પર જાઓ.

for લૂપને સંબંધિત નીચેના મહત્ત્વના મુદ્દાઓની નોંધ કરો :

- કાઉન્ટર ચલની શરૂઆતની કિંમત આપવા માટે પદાવલિ-1નો ઉપયોગ કરવામાં આવે છે. કાઉન્ટર ચલનો ઉપયોગ લૂપના અમલીકરણની કુલ સંખ્યાનું નિયંત્રણ કરે છે. આ ચલને નિયંત્રણ ચલ (counter variable) તરીકે ઓળખવામાં આવે છે.
- પદાવલિ-2 ચકાસણીની શરત (ટેસ્ટ કન્ડિશન) તરીકે કાર્ય કરે છે. લૂપને અટકાવવા માટેના માપદંડને તપાસવા માટે અહીં નિયંત્રણ ચલનો ઉપયોગ કરવામાં આવે છે.
- નિયંત્રણ ચલની કિંમત વધારવા કે ઘટાડવા માટે પદાવલિ-3નો ઉપયોગ કરવામાં આવે છે.

નેસ્ટેડ for લૂપ (Nested for loop)

એક for લૂપની અંદર અન્ય for લૂપના ઉપયોગને નેસ્ટેડ for (nested for) કહે છે. આ અભિગમ સમજવા માટે ઉદાહરણ 14.2માં આવેલ પ્રોગ્રામનો ઉપયોગ કરીએ. આ ઉદાહરણ આપેલ લીટીની સંખ્યા પ્રમાણે નીચે દર્શાવેલ ભાત (pattern) દર્શાવશે. ઉદાહરણ તરીકે, જો લીટીની સંખ્યા 4 આપવામાં આવે તો પ્રોગ્રામ નીચે દર્શાવેલ પરિણામ આપશે :

```
1
1 2
1 2 3
1 2 3 4
```

આકૃતિ 14.3માં ઉદાહરણ 14.2નું કોડ લિસ્ટિંગ આપવામાં આવ્યું છે તથા આકૃતિ 14.4 તેનું પરિણામ દર્શાવે છે.

11-2.c - SciTE

File Edit Search View Tools Options Language Buffers Help

11-2.c

```c
/* Example 2: Program to illustrate the use nested for loop.*/
#include<stdio.h>

int main()
{
   int i, j, lines;                          // declaration of variables
   printf("Enter number of lines : ");       // message to user
   scanf("%d", &lines);                      // stores value in lines variable

   for ( i = 1; i <= lines ; i++)            // Outer loop, repeat number of lines time
   {
      for( j = 1 ; j <= i ; j++)     // inner loop for printing numbers from 1 to value of i
      {
            printf("%d ", j);                // print value of j
       }
      printf("\n");                          // move cursor to next line
   }
   return 0;
}
/* End of Program */
```

આકૃતિ 14.3 : ઉદાહરણ 14.2નું કોડ-લિસ્ટિંગ

```
kpp@ubuntu: ~
File  Edit  View  Terminal  Help
kpp@ubuntu:~$ gcc 11-2.c
kpp@ubuntu:~$ ./a.out
Enter number of lines : 4
1
1 2
1 2 3
1 2 3 4
kpp@ubuntu:~$ 
```

આકૃતિ 14.4 : ઉદાહરણ 14.2નું પરિણામ

સમજૂતી

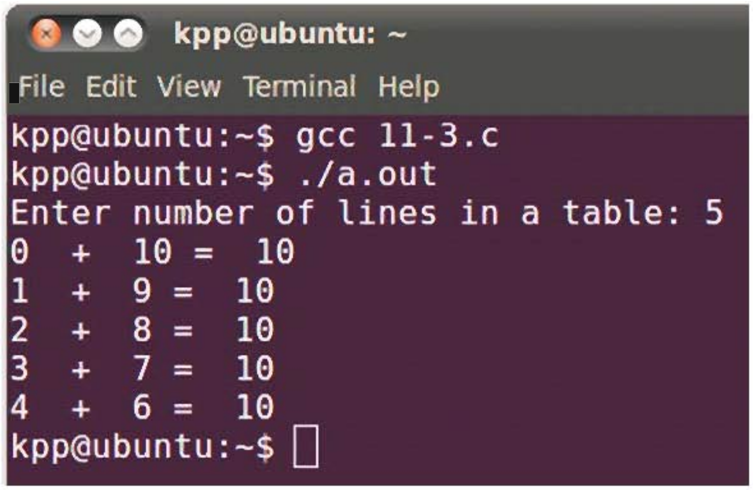
પ્રોગ્રામમાં બહારનું લૂપ નિયંત્રણ ચલ i ધરાવે છે, જેની કિંમત એક પછી એક વધારતા જઈ 'lines' ચલની કિંમત જેટલી કરવામાં આવે છે. બહારના લૂપના iની દરેક કિંમત માટે અંદરના લૂપમાં નિયંત્રણ ચલ jનો i વખત અમલ કરવામાં આવશે. અને jની કિંમત 1થી શરૂ કરી એક પછી એક i સુધી વધારો કરી દર્શાવવામાં આવશે. i જેટલા અંકોને i જેટલી લીટીમાં દર્શાવ્યા પછી બહારના લૂપમાં આવેલું printf("\n") વિધાન કર્સરને નીચેની લીટીમાં લાવશે. તેથી ત્યાર પછીના અંકો નવી લીટીમાં દર્શાવશે.

for લૂપમાં કૉમા પ્રક્રિયકનો ઉપયોગ (Use of comma operator in *for* loop)

for લૂપમાં કૉમા પ્રક્રિયકનો ઉપયોગ કરી આપણે એકથી વધુ પ્રાચલનો આરંભ કરી શકીએ છીએ. આ જ રીતે for લૂપમાં એકથી વધુ ચલોની કિંમત વધારવા કે ઘટાડવા માટે પણ કૉમા પ્રક્રિયકનો ઉપયોગ કરી શકાય છે. ઉદાહરણ 14.3ની મદદથી કૉમા પ્રક્રિયકના ઉપયોગને સમજવાનો પ્રયત્ન કરીએ. આકૃતિ 14.5માં ઉદાહરણ 14.3નું કોડ લિસ્ટિંગ આપવામાં આવ્યું છે તથા આકૃતિ 14.6 તેનું પરિણામ દર્શાવે છે.

```
11-3.c - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
11-3.c

/* Example 3: Program to illustrate the use comma operator in a for loop
   to print the addition table.*/

#include<stdio.h>
int main()
{
    int i, j, number;                          // declaration of variables
    printf("Enter number of lines in a table: "); // message to user
    scanf("%d", &number);                      // stores value in number variable
    for ( i=0, j=10; i < number; i++, j- -)
    {
            printf("%d  +  %d =  %d\n", i, j, i + j );
    }
    return 0;
}
/* End of Program */
```

આકૃતિ 14.5 : ઉદાહરણ 14.3નું કોડ-લિસ્ટિંગ



```
kpp@ubuntu: ~
File  Edit  View  Terminal  Help
kpp@ubuntu:~$ gcc 11-3.c
kpp@ubuntu:~$ ./a.out
Enter number of lines in a table: 5
0  +  10 =  10
1  +  9 =  10
2  +  8 =  10
3  +  7 =  10
4  +  6 =  10
kpp@ubuntu:~$
```

આકૃતિ 14.6 : ઉદાહરણ 14.3નું પરિણામ

ઉદાહરણ 14.3માં i=0 અને j=10 કિંમતો સાથે બે ચલનો ઉપયોગ કરવામાં આવ્યો છે. વળી, એક જ *for* લૂપમાં iની કિંમત વધારવામાં અને jની કિંમત ઘટાડવામાં આવી છે. આપવામાં આવેલ ઇનપુટને આધારે ઉદાહરણમાં દર્શાવ્યા મુજબ પરિણામ દર્શાવવામાં આવે છે.

while લૂપ (The while loop)

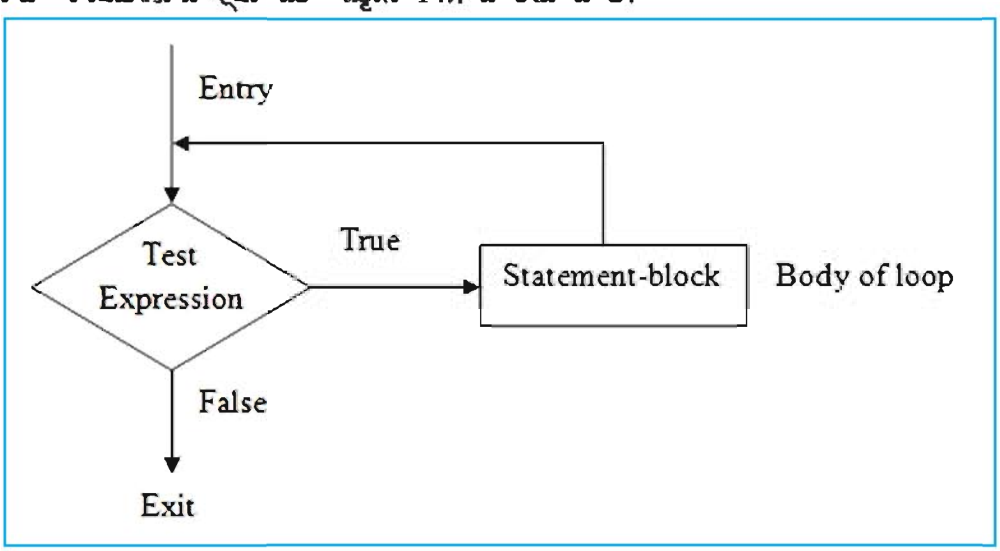
જ્યારે ફેરાની સંખ્યા પૂર્વનિશ્ચિત ન હોય અને લૂપને અટકાવવા માટેની શરત લૂપમાં પ્રવેશ કરતાં પહેલાં ચકાસવાની હોય ત્યારે while લૂપનો ઉપયોગ વધુ અનુકૂળ છે. ઉદાહરણ તરીકે,

- ઉપયોગકર્તા શૂન્ય ન ઉમેરે ત્યાં સુધી ઉમેરવામાં આવેલી તમામ સંખ્યાઓનો સરવાળો કરવો.
- મેનૂ વિકલ્પો દર્શાવવા અને ઉપયોગકર્તા બહાર નીકળવાનો (exit) વિકલ્પ પસંદ ન કરે ત્યાં સુધી યોગ્ય ક્રિયાઓનો અમલ કરવો.

while લૂપની વાક્યરચના (Syntax of while loop)

```
while (test expression)
{
        statement-block;    /* while લૂપનાં વિધાનો */
}
```

while લૂપના અમલીકરણનો ફ્લો-ચાર્ટ આકૃતિ 14.7માં દર્શાવ્યો છે.



આકૃતિ 14.7 : while લૂપનો ફ્લો-ચાર્ટ

આકૃતિ 14.7માં દર્શાવ્યા મુજબ સૌ પ્રથમ શરત(ટેસ્ટ એક્સ્પ્રેશન)નું મૂલ્યાંકન કરવામાં આવશે. જો શરત true પરત કરશે તો લૂપનાં વિધાનોનો વિભાગ અમલી બનશે. પ્રોગ્રામ ફરી શરતનું મૂલ્યાંકન કરશે. શરતનું પરિણામ false ન મળે ત્યાં સુધી આ પ્રક્રિયાનું પુનરાવર્તન કરવામાં આવશે. જ્યારે શરતનું મૂલ્યાંકન false મળશે ત્યારે લૂપ અટકાવવામાં આવશે અને નિયંત્રણને પ્રોગ્રામમાં લૂપ પછી આપેલા વિધાન તરફ મોકલવામાં આવશે. લૂપનાં વિધાનો એક અથવા સંમિશ્રિત (compound) હોઈ શકે છે.

અહીં એ નોંધ લેવી જરૂરી છે કે, લૂપના પ્રવેશ ઉપર શરતની ચકાસણી થતી હોવાથી તેને પ્રવેશ નિયંત્રણ *(entry controlled)* લૂપ તરીકે ઓળખવામાં આવે છે. જો શરતનું પરિણામ પ્રથમ વખત જ false મળશે તો લૂપનાં વિધાનો એકપણ વાર અમલમાં મુકાશે નહીં.

હવે, ઉદાહરણ 14.4ની મદદથી *while* લૂપનો ઉપયોગ સમજવાનો પ્રયત્ન કરીએ. ઉપયોગકર્તા શૂન્ય ન ઉમેરે ત્યાં સુધી આપવામાં આવેલ તમામ સંખ્યાઓનો સરવાળો શોધવા માટેનો પ્રોગ્રામ ઉદાહરણ 14.4માં આપવામાં આવ્યો છે. આકૃતિ 14.8માં ઉદાહરણ 14.4નું કોડ-લિસ્ટિંગ આપવામાં આવ્યું છે જ્યારે આકૃતિ 14.9 તેનું પરિણામ દર્શાવે છે.

11-4.c - SciTE

File Edit Search View Tools Options Language Buffers Help

11-4.c

```c
/* Example 4: Program to illustrate use of while loop.*/
#include<stdio.h>

int main()
{
    int number, sum=0;                          // declaration of variables
    printf("Enter 0 to exit from program.\n");
    printf("Enter your number : ");             // message to user
    scanf("%d", &number);                       // stores value in number variable
    sum = number;

    while(number != 0)
    {
        printf("Enter your number : ");         // message to user
        scanf("%d", &number);                   // stores value in number variable
        sum = sum + number;
    }
    printf("The sum of entered numbers : %d \n", sum);
    return 0;
}
/*End of Program */
```

આકૃતિ 14.8 : ઉદાહરણ 14.4નું કોડ-લિસ્ટિંગ

kpp@ubuntu: ~

File Edit View Terminal Help

```
kpp@ubuntu:~$ gcc 11-4.c
kpp@ubuntu:~$ ./a.out
Enter 0 to exit from program.
Enter your number : 5
Enter your number : 10
Enter your number : 0
The sum of entered numbers : 15
kpp@ubuntu:~$
```

આકૃતિ 14.9 : ઉદાહરણ 14.4નું પરિણામ

સમજૂતી

પ્રોગ્રામમાં જરૂરી એવા બે ચલની ઘોષણા main() વિધેય પછીના પ્રથમ વિધાન દ્વારા કરવામાં આવે છે. બીજા વિધાન દ્વારા પ્રોગ્રામમાંથી બહાર નીકળવાની રીતનો નિર્દેશ કરવામાં આવે છે. ત્રીજા અને ચોથા વિધાન દ્વારા અનુક્રમે સંદેશ દર્શાવી ઉપયોગકર્તા પાસેથી સંખ્યા મેળવવામાં આવે છે. પાંચમું વિધાન number ચલની કિંમત sum ચલને આપે છે. છઠ્ઠા વિધાનમાં શરત દ્વારા ચકાસવામાં આવે છે કે number ચલની કિંમત શૂન્ય છે કે નહીં. શરત સાચી હોય તો સાતમા અને આઠમા વિધાન દ્વારા અનુક્રમે સંદેશ દર્શાવી ઉપયોગકર્તા પાસેથી એક સંખ્યા મેળવવામાં આવે છે. દસમું વિધાન આપેલ સંખ્યાને sum ચલમાં ઉમેરે છે. ત્યાર પછી ફરી *while* લૂપની શરતનું મૂલ્યાંકન કરવામાં આવે છે. શરતનું પરિણામ નકારાત્મક (false) મળે ત્યારે ઉપયોગકર્તા દ્વારા ઉમેરવામાં આવેલી તમામ સંખ્યાઓનો સરવાળો દસમા વિધાનના અમલ દ્વારા દર્શાવવામાં આવે છે.

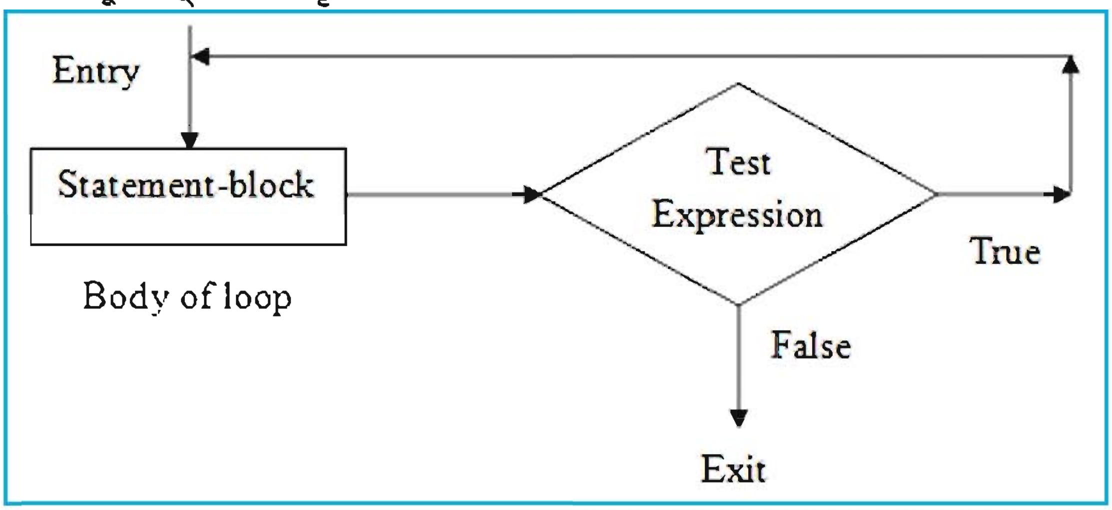
do...while લૂપ (The do...while loop)

આપણે જોયું કે, *while* લૂપમાં લૂપનાં વિધાનોનો અમલ કરતાં પહેલાં શરત ચકાસવામાં આવે છે. કેટલીકવાર આપણે શરતને ચકાસતાં પહેલાં લૂપનાં વિધાનોનો અમલ કરવા ઇચ્છતા હોઈએ છીએ. લૂપનાં વિધાનોનો અમલ કર્યા બાદ શરતને ચકાસવાની હોય ત્યારે do...*while* લૂપનો ઉપયોગ કરવો જોઈએ. લૂપના અંતમાં શરતની ચકાસણી કરવામાં આવતી હોવાથી do...*while* લૂપને નિર્ગમ-નિયંત્રણ *(exit controlled)* પ્રકારનું લૂપ કહે છે. અહીં એ નોંધ લેવી જરૂરી છે કે do...*while* લૂપમાં લૂપનાં વિધાનોનો અમલ એકવાર તો કરવામાં આવશે જ.

do...while લૂપની વાક્યરચના (Syntax of do...while loop)

```
do
{
        statement-block;        /* લૂપનાં વિધાનો */
}
while (tests expression);
```

do...while લૂપનો ફ્લો-ચાર્ટ આકૃતિ 14.10માં દર્શાવ્યો છે.



આકૃતિ 14.10 : do...while લૂપનો ફ્લો-ચાર્ટ

આકૃતિ 14.10માં દર્શાવ્યા મુજબ લૂપમાં આવેલ સ્ટેટમેન્ટ બ્લૉકનો પ્રથમ અમલ કરવામાં આવે છે. ત્યાર પછી શરતનું મૂલ્યાંકન કરવામાં આવે છે. જો શરતનું પરિણામ true મળે તો સ્ટેટમેન્ટ બ્લૉક ધરાવતા લૂપનાં વિધાનોનો ફરી અમલ કરવામાં આવે છે. સ્ટેટમેન્ટ-બ્લૉકના અમલ પછી ફરી એકવાર શરતનું મૂલ્યાંકન કરવામાં આવે છે. શરતનું પરિણામ false ન મળે ત્યાં સુધી આ ક્રિયાનું પુનરાવર્તન કરવામાં આવે છે. જ્યારે શરતનું પરિણામ નકારાત્મક આવે ત્યારે લૂપને અટકાવવામાં આવે છે અને નિયંત્રણને લૂપ પછીના વિધાન પર લઈ જવામાં આવે છે. લૂપમાં એક અથવા સંમિશ્રિત (compound) વિધાનો હોઈ શકે.

આકૃતિ 14.5નો ઉપયોગ કરી do...*while* લૂપની કાર્યપદ્ધતિ સમજવાનો પ્રયત્ન કરીએ. ઉપયોગકર્તા શૂન્ય ન ઉમેરે ત્યાં સુધી ઉમેરવામાં આવેલ તમામ સંખ્યાઓનો સરવાળો કરવા માટે do...*while* લૂપનો ઉપયોગ ઉદાહરણ 14.5માં કરવામાં આવ્યો છે. ઉદાહરણ 14.5નું કોડ લિસ્ટિંગ આકૃતિ 14.11માં આપવામાં આવ્યું છે તથા આકૃતિ 14.12 તેનું પરિણામ દર્શાવે છે.

11-5.c - SciTE

File Edit Search View Tools Options Language Buffers Help

11-5.c

```
/* Example 5: Program to illustrate use of do...while loop.*/
#include<stdio.h>

int main()
{
    int number, sum=0;                          // declaration of variables
    printf("Enter 0 to exit from program.\n");
    do
    {
        printf("Enter your number : ");         // message to user
        scanf("%d", &number);                   // stores value in number variable
        sum = sum + number;
     } while(number != 0);

    printf("The sum of entered numbers : %d \n", sum);
    return 0;
}
/*End of Program */
```

આકૃતિ 14.11 : ઉદાહરણ 14.5નું કોડ-લિસ્ટિંગ

kpp@ubuntu: ~

File Edit View Terminal Help

```
kpp@ubuntu:~$ gcc 11-5.c
kpp@ubuntu:~$ ./a.out
Enter 0 to exit from program.
Enter your number : 3
Enter your number : 4
Enter your number : 0
The sum of entered numbers : 7
kpp@ubuntu:~$ 
```

આકૃતિ 14.12 : ઉદાહરણ 14.5નું પરિણામ

સમજૂતી

main() વિધેય પછીનું પ્રથમ વિધાન પ્રોગ્રામમાં જરૂરી એવા બે ચલ ઘોષિત કરી તેમાંના એક ચલમાં કિંમત ઉમેરે છે. બીજું વિધાન પ્રોગ્રામમાંથી બહાર નીકળવાની રીતનો નિર્દેશ કરે છે. ત્રીજા વિધાનથી *do...while* લૂપની શરૂઆત થાય છે. ચોથું અને પાંચમું વિધાન અનુક્રમે સંદેશ દર્શાવે છે અને ઉપયોગકર્તા પાસેથી કિંમત મેળવે છે. છઠ્ઠું વિધાન ઉપયોગકર્તાએ આપેલ સંખ્યા sum ચલમાં ઉમેરે છે. સાતમું વિધાન શરતનું મૂલ્યાંકન કરે છે. જ્યાં સુધી શરત સાચી હોય ત્યાં સુધી ચોથાથી છઠ્ઠા વિધાનોનું પુનરાવર્તન કરવામાં આવે છે. જ્યારે શરતનું પરિણામ false મળે ત્યારે આઠમા વિધાનનો અમલ કરી ઉપયોગકર્તાએ ઉમેરેલી તમામ સંખ્યાઓનો સરવાળો દર્શાવવામાં આવે છે.

નેસ્ટેડ લૂપ (Nested loop)

લૂપ નિયંત્રણના પ્રકારને અનપેક્ષ એક લૂપ નિયંત્રણનો અન્ય લૂપની અંદર ઉપયોગ કરી જુદા જુદા પ્રકારનું નેસ્ટિંગ શક્ય છે. ઉદાહરણ તરીકે *for* લૂપની અંદર *while* કે *do...while* લૂપનો ઉપયોગ કરી શકાય છે. જેમ કે, 1થી 100 વચ્ચેની અવિભાજ્ય સંખ્યાઓ દર્શાવવા માટે *for* લૂપનો ઉપયોગ કરી સંખ્યાઓનું લૂપ લખી શકાય છે અને આ લૂપની અંદર *while* લૂપનો ઉપયોગ કરી આપેલ સંખ્યા અવિભાજ્ય છે કે નહીં તે શોધી શકાય છે.

યોગ્ય લૂપની પસંદગી (Selecting a loop)

અત્યાર સુધીમાં આપણે આ પ્રકરણમાં સી ભાષાના ત્રણ લૂપિંગ બંધારણો – *for, while* અને *do...while* ની ચર્ચા કરી. પ્રોગ્રામમાં વધુ યોગ્ય લૂપની પસંદગી કરવા માટે નીચે આપેલા મુદ્દાઓનો ઉપયોગ કરી શકાય :

- આપેલ પ્રશ્ન માટે પ્રવેશ-નિયંત્રિત (entry controlled) કે નિર્ગમ-નિયંત્રિત (exit controlled) લૂપની જરૂર પડશે તે નક્કી કરો.
- પ્રવેશ-નિયંત્રિત લૂપ માટે *for* કે *while* લૂપનો ઉપયોગ કરવો જોઈએ. ગણતરી આધારિત નિર્ગમન શરત (counter based exit condition) માટે for લૂપનો ઉપયોગ સલાહભર્યો છે.
- નિર્ગમ-નિયંત્રિત લૂપ માટે *do...while* બંધારણનો ઉપયોગ કરવો જોઈએ.

આ ઉપરાંત, પ્રોગ્રામમાં કોઈ પણ લૂપનો ઉપયોગ કરતી વખતે નીચેના ત્રણ મુદ્દાઓનું ધ્યાન રાખવાથી લૂપનું અનિચ્છનીય વર્તન અટકાવી શકાય છે :

- લૂપમાં ગણતરીનો આરંભ કરવો.
- લૂપમાંથી બહાર આવવા માટેની નિર્ગમન શરત (exit condition) ચકાસવી.
- લૂપમાં ગણતરી માટેના કાઉન્ટરની કિંમત વધારવી અથવા ઘટાડવી.

લૂપનો કોઈ ભાગ છોડી દેવો (Skipping a part of loop)

પ્રોગ્રામના અમલીકરણ વખતે ઘણીવાર આપણે કેટલાંક વિધાનો અથવા ફેરા (iterations)ને છોડી દેવા માંગતા હોઈએ છીએ. સી ભાષામાં નીચેનાં વિધાનોનો ઉપયોગ કરવાથી આ શક્ય છે :

(i) break વિધાન

(ii) continue વિધાન

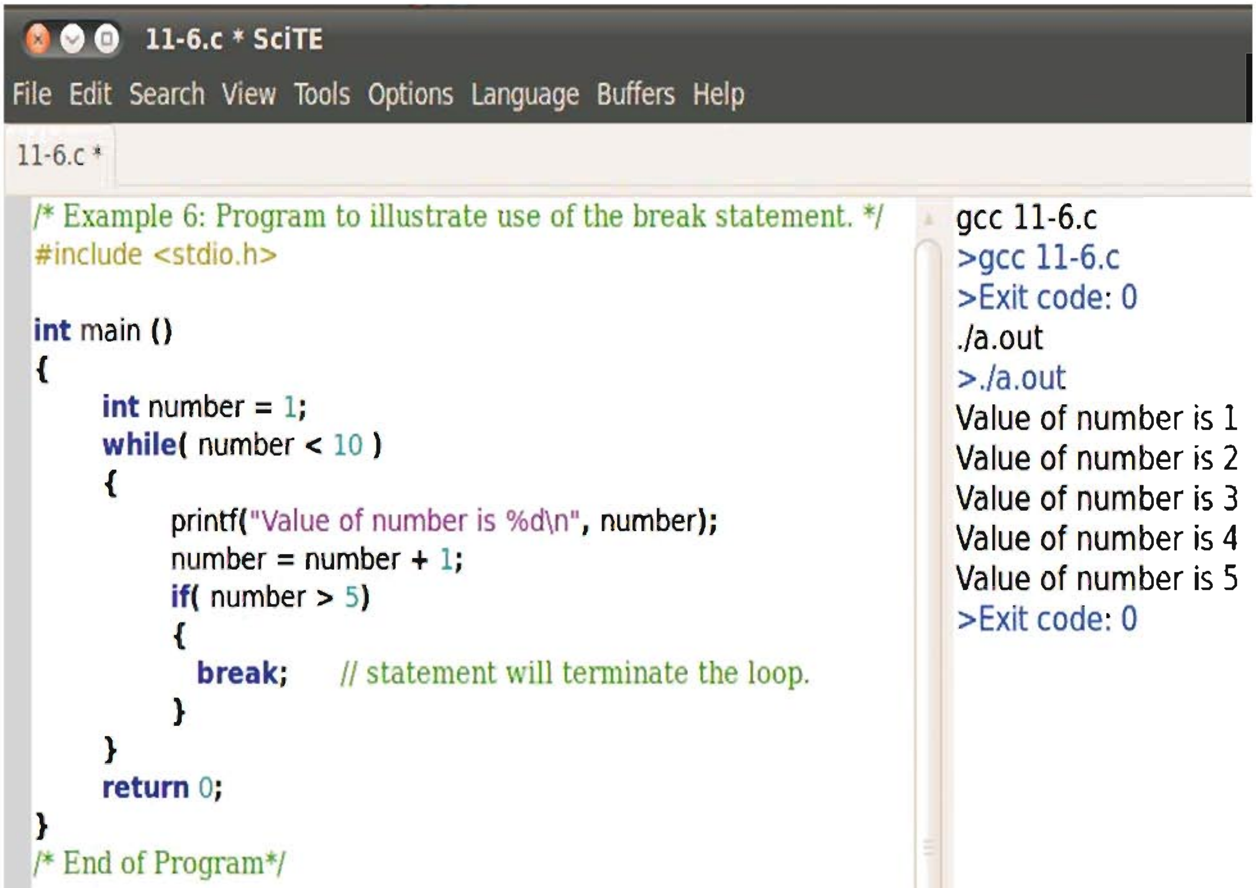
break વિધાન (The break statement)

લૂપના અમલ દરમિયાન કેટલીકવાર લૂપનાં અન્ય વિધાનોનો અમલ કર્યા વિના લૂપને અચાનક જ અટકાવી દેવાની જરૂર પડતી હોય છે. break વિધાનના ઉપયોગથી આ શક્ય છે. સી પ્રોગ્રામિંગ ભાષામાં break વિધાનના ઉપયોગની માહિતી નીચે જણાવેલ છે :

- લૂપમાં break વિધાન આવે ત્યારે લૂપને તત્કાલ અટકાવવામાં આવે છે તથા પ્રોગ્રામ નિયંત્રણ દ્વારા લૂપ પછીના વિધાનનો અમલ કરવામાં આવે છે.
- *switch* વિધાનમાં caseનો અમલ અટકાવવા માટે પણ break વિધાનનો ઉપયોગ કરવામાં આવે છે.

નેસ્ટેડ લૂપના કિસ્સામાં *break* વિધાન તત્કાલ અમલમાં હોય તે લૂપનું અમલીકરણ અટકાવે છે અને કોડના જૂથ પછી આવેલા વિધાનનો અમલ શરૂ કરવામાં આવે છે.

પ્રોગ્રામમાં *break* વિધાનનો ઉપયોગ ઉદાહરણ 14.6 દ્વારા દર્શાવ્યો છે. આકૃતિ 14.13માં ઉદાહરણ 14.6નું કોડ-લિસ્ટિંગ તથા પરિણામ દર્શાવ્યું છે.



આકૃતિ 14.13 : ઉદાહરણ 14.6નું કોડ-લિસ્ટિંગ અને પરિણામ

સમજૂતી

main() વિધેય પછીનું પ્રથમ વિધાન એક પૂર્ણાંક ચલની પ્રારંભિક કિંમત નિશ્ચિત કરે છે. લૂપને નવ (9) વખત ફેરવવા માટે *while* વિધાનમાં શરત લખવામાં આવી છે. લૂપની અંદર આવેલું printf વિધાન number ચલની કિંમત દર્શાવશે. number ચલની કિંમત દર્શાવ્યા પછી તેની કિંમતમાં 1નો વધારો કરવામાં આવશે. ત્યારપછી, *if* વિધાનનો અમલ કરવામાં આવશે. પરંતુ પ્રોગ્રામના અમલ દરમિયાન જ્યારે *if* વિધાનનું મૂલ્યાંકન true મળશે ત્યારે *break* વિધાનનો અમલ કરવામાં આવશે, જે *while* લૂપનો અમલ અટકાવશે. આમ, આ પ્રોગ્રામમાં *while* લૂપમાં આવેલું printf વિધાન માત્ર પાંચ વખત અમલમાં મુકાશે.

continue વિધાન (The continue statement)

continue વિધાન કેટલેક અંશે break વિધાન જેવું કાર્ય કરે છે. જો કે, લૂપને અટકાવવાને બદલે continue વિધાન લૂપના અન્ય કોડનું અમલીકરણ સ્થગિત કરી લૂપના ત્યાર પછીના ફેરાનો અમલ શરૂ કરે છે.

ઉદાહરણ તરીકે, for લૂપમાં continue વિધાન દ્વારા શરતની ચકાસણી અને ચલમાં કરવામાં આવતા વધારાના વિભાગનો અમલ કરવામાં આવે છે. while અને do...while લૂપ માટે continue વિધાન પ્રોગ્રામના નિયંત્રણને શરતની ચકાસણી તરફ મોકલે છે.

પ્રોગ્રામમાં continue વિધાનનો ઉપયોગ ઉદાહરણ 14.7માં દર્શાવ્યો છે. આકૃતિ 14.14માં ઉદાહરણ 14.7નું કોડ-લિસ્ટિંગ અને પરિણામ દર્શાવ્યાં છે.

```
11-7.c - SciTE
File Edit Search View Tools Options Language Buffers Help
11-7.c

/* Example 7: Program to illustrate the use of continue statement. */
#include <stdio.h>

int main ()
{
   int number = 1;
   printf("Value of number is %d\n", number);

   while(number < 6)
    {
      number = number + 1;
      if( number == 3)
      {
        continue;    // statement skip the current iteration of the loop
      }
      printf("Value of number is %d\n", number);
    }
    return 0;
}
/* End of Program*/
```

```
gcc 11-7.c
>gcc 11-7.c
>Exit code: 0
./a.out
>./a.out
Value of number is 1
Value of number is 2
Value of number is 4
Value of number is 5
Value of number is 6
>Exit code: 0
```

આકૃતિ 14.14 : ઉદાહરણ 14.7નું કોડ-લિસ્ટિંગ અને પરિણામ

સમજૂતી

પ્રથમ વિધાન દ્વારા પ્રોગ્રામમાં જરૂરી એવા number ચલની ઘોષણા કરી પ્રારંભની કિંમત આપવામાં આવે છે. બીજું વિધાન number ચલની કિંમત દર્શાવે છે. લૂપનો પાંચ વખત અમલ કરવા માટે while વિધાનમાં શરત લખવામાં આવી છે. while લૂપની અંદર number ચલની કિંમત 1 વડે વધારવામાં આવે છે. ત્યાર પછી if વિધાન શરતની ચકાસણી કરે છે. જ્યારે તેનું મૂલ્યાંકન true મળે ત્યારે continue વિધાનનો અમલ કરવામાં આવે છે, જે વર્તમાન ફેરાને અટકાવે છે. તેથી number ચલની કિંમત 3 દર્શાવતું નથી અને ત્યારપછીના ફેરાનો અમલ કરે છે. તેથી આ પ્રોગ્રામ number ચલની 4, 5 અને 6 સંખ્યા દર્શાવે છે.

અનંત લૂપ (The Infinite loop)

જો કોઈ લૂપ સતત ચાલ્યા જ કરે અને પ્રોગ્રામનું નિયંત્રણ ક્યારેય તેની બહાર આવી ન શકે તો તેવા લૂપને અનંત લૂપ કહે છે. લૂપના તર્કમાં નિર્ગમન શરત(exit condition)ની અનુપલબ્ધતાને કારણે લૂપ અનંત લૂપ બને છે. આકૃતિ 14.15માં આપવામાં આવેલા ઉદાહરણ 14.8ના પ્રોગ્રામનો તર્ક જુઓ.

```
11-8.c - SciTE
File Edit Search View Tools Options Language Buffers Help
11-8.c

/* Example 8: Program to illustrate infinite loop.*/
#include <stdio.h>

int main ()
{
        for( ; ; )
        {
                printf(" This for loop will run forever.....\n " );
        }
        return 0;
}
/* End of program*/
```

```
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
This for loop will run forever.....
```

આકૃતિ 14.15 : ઉદાહરણ 14.8નું કોડ-લિસ્ટિંગ અને પરિણામ

સમજૂતી

જ્યારે આ કોડને કમ્પાઈલ કરી અમલ કરવામાં આવશે ત્યારે તેનું અમલીકરણ સતત અને અવિરત ચાલ્યા કરશે. સી પ્રોગ્રામના *for* લૂપમાં તમામ ત્રણ પદાવલિઓ મરજિયાત છે. જો *for* લૂપમાં શરત માટેની પદાવલિ લખવામાં ન આવે તો તેને true સમજી લેવામાં આવે છે. પ્રારંભિક કિંમત આપવા માટેની પદાવલિ તથા ચલની કિંમતમાં વધારો કરવા માટેની પદાવલિ *for* લૂપની અંદર હોઈ શકે છે, પરંતુ *for* (; ;) બંધારણનો ઉપયોગ કરવાથી સી પ્રોગ્રામનું અમલીકરણ અવિરત ચાલ્યા કરે છે. અનંત લૂપને અટકાવવા માટે Ctrl + C કીનો ઉપયોગ કરી શકાય છે.

સારાંશ

પ્રોગ્રામરને વિધાન કે વિધાનોના સમૂહનું અમલીકરણ એકથી વધુ વખત કરવા માટેની અનુમતિ આપતા સી ભાષાના **લૂપ નિયંત્રણ બંધારણ** વિશે આપણે આ પ્રકરણમાં અભ્યાસ કર્યો. સી ભાષા દ્વારા પૂરા પાડવામાં આવતા ત્રણ લૂપ નિયંત્રણ બંધારણ(for, while અને do...while)નો અભ્યાસ કર્યો. આપણા પ્રોગ્રામમાં પ્રવેશ નિયંત્રણ લૂપ અને નિર્ગમ નિયંત્રણ લૂપનો ઉપયોગ કેવી રીતે કરી શકાય તેનો પણ આપણે અભ્યાસ કર્યો. પ્રોગ્રામના અમલ દરમિયાન કેટલાંક વિધાનો કે ફેરાનું અમલીકરણ મોકૂફ રાખવા break અને continue વિધાનનો ઉપયોગ કેવી રીતે કરી શકાય તે આપણે આ પ્રકરણના અંતમાં શીખ્યા.

સ્વાધ્યાય

1. જરૂરી ઉદાહરણો સાથે for લૂપ બંધારણની વાક્યરચના સમજાવો.
2. લૂપિંગ વિધાનો એટલે શું ? સી પ્રોગ્રામમાં લૂપિંગ વિધાનોનો ઉપયોગ જણાવો.
3. while અને do...while લૂપ વચ્ચેનો તફાવત જણાવો.
4. break અને continue વિધાન દ્વારા લૂપના કોઈ ભાગનું અમલીકરણ કેવી રીતે મોકૂફ રાખી શકાય તે સમજાવો.
5. પ્રવેશ નિયંત્રણ અને નિર્ગમ નિયંત્રણ લૂપ એટલે શું ? યોગ્ય ઉદાહરણ આપો.
6. ખાલી જગ્યા પૂરો :
   - (a) લૂપમાં આવેલા નિયંત્રણ વિધાનના સ્થાનને આધારે લૂપને __________ અને __________ એમ બે વર્ગોમાં વર્ગીકૃત કરી શકાય.
   - (b) for લૂપ __________ નિયંત્રિત લૂપ છે.
   - (c) do...while લૂપ __________ નિયંત્રિત લૂપ છે.
   - (d) નિર્ગમ નિયંત્રિત લૂપમાં લૂપનાં વિધાનોનો ઓછામાં ઓછા __________ વખત અમલ કરવામાં આવે છે.
   - (e) લૂપ નિયંત્રણ બંધારણમાંથી તત્કાલ બહાર આવવા માટે __________ વિધાનનો ઉપયોગ કરી શકાય.
   - (f) લૂપમાં આવેલાં પછીનાં વિધાનોના અમલને અટકાવી ત્યાર પછીનો ફેરો શરૂ કરવા માટે __________ વિધાન જરૂરી છે.
7. નીચેનાં વિધાનો ખરાં છે કે ખોટાં તે જણાવો :
   - (a) for લૂપની અંદર do...while લૂપનો ઉપયોગ કરી શકાતો નથી.
   - (b) લૂપમાં આવેલ break વિધાનનું અમલીકરણ પ્રોગ્રામનો અમલ તત્કાલ અટકાવે છે.

(c) for લૂપના મથાળામાં પ્રારંભિક કિંમત ઉમેરવી, શરત લખવી અને ચલમાં વધારો કરવા માટેનો વિભાગ ઉમેરવો - આ તમામ વૈકલ્પિક છે.

(d) switch વિધાનના caseને અટકાવવા માટે break વિધાનનો ઉપયોગ કરી શકાતો નથી.

(e) પ્રારંભિક કિંમત ઉમેરવી, શરત અને ચલમાં વધારો કરવા માટેના વિભાગને અલ્પવિરામ દ્વારા છૂટા પાડવામાં આવે છે.

(f) દરેક while વાક્યાંશ સાથે do હોવું ફરજિયાત છે.

(g) દરેક do વાક્યાંશ સાથે તેને સંબંધિત while હોવું ફરજિયાત છે.

8. આપેલ વિકલ્પોમાંથી સાચો વિકલ્પ પસંદ કરો :

(1) નીચેના પ્રોગ્રામ-ખંડમાં printf વિધાનનો અમલ કેટલીવાર કરવામાં આવશે ?

```
int num1 = 3, num2 = 6;
while(num1 <num2) {
    printf("Hello students...");
    num1 = num1 + 1;
}
```

(a) 1 (b) 2 (c) 3 (d) 6

(2) નીચેના પ્રોગ્રામ-ખંડનું પરિણામ શું મળશે ?

```
int a = 5, b = 10;
do
{
    a = a + 1;
}while(a <= b);
printf("%d", a);
```

(a) 10 (b) 9 (c) 11 (d) 15

(3) નીચેના પ્રોગ્રામ-ખંડનું પરિણામ શું મળશે ?

```
int main( ){
int i;
for(i = 0; i <10; i++);
printf("%d", i);
}
```

(a) 0123456789 (b) 9 (c) 10 (d) ભૂલનો સંદેશ

(4) નીચેના કોડના અમલ પછી number ચલની કિંમત કઈ હશે ?

```
int number = 1;
while(number < 5){
    number = number + 1;
    if( number = = 3) {
        continue;
    }
}
```

(a) 1 (b) 3 (c) 5 (d) એક પણ નહીં

પ્રાયોગિક સ્વાધ્યાય

1. નીચેના for લૂપને તેના સમકક્ષ while લૂપમાં ફેરવો :

(a)
```
for(number=0; number<5; number++) {
printf("%d", number);
}
```

(b)
```
for(number=5; number >0; number —) {
    printf("%d", number);
}
```

2. while લૂપમાં સંમિશ્રિત (compound) વિધાનોનો ઉપયોગ દર્શાવો.
3. for લૂપનો ઉપયોગ કરી પ્રથમ 100 પૂર્ણાંકોનો સરવાળો શોધવા માટેનો પ્રોગ્રામ લખો.
4. while લૂપનો ઉપયોગ કરી પ્રથમ 100 બેકી પૂર્ણાંક સંખ્યાઓનો સરવાળો કરવા માટેનો પ્રોગ્રામ બનાવો.
5. આપેલ લીટીઓની સંખ્યા માટે નીચેની પેટર્ન દર્શાવવા માટેનો પ્રોગ્રામ લખો. ઉદાહરણ તરીકે જો લીટીની સંખ્યા 4 આપવામાં આવે તો નીચે દર્શાવેલ પરિણામ દર્શાવવામાં આવે.

```
*
* *
* * *
* * * *
* * *
* *
*
```

6. સ્ક્રીન પર નીચે દર્શાવેલ શ્રેણી દર્શાવવા માટેનો for લૂપ પ્રોગ્રામ લખો :

(a) 1, 3, 5, 7, 9, 11 (b) 1, 2, 4, 8, 16, 32, 64

(c) 10, 8, 6, 4, 2, 0 (d) –10, –8, –6, –4, –2, 0

7. અપર કેસ અને લોઅર કેસ મૂળાક્ષરોની આસ્કી (ASCII) કિંમતો દર્શાવવા માટેનો પ્રોગ્રામ લખો.
8. ઉપયોગકર્તા 'Z' દાખલ ન કરે ત્યાં સુધી આપવામાં આવેલા તમામ અક્ષરોની સંખ્યા ગણવા તથા દર્શાવવા માટેનો પ્રોગ્રામ લખો.

ઍરે

'સી' પ્રોગ્રામિંગમાં અત્યાર સુધી આપણે મૂળભૂત ડેટા પ્રકાર જેવા કે int, float, char, double, વગેરેનો ઉપયોગ કર્યો. આ બધા ચલના પ્રકારો એક સમયે માત્ર એક જ કિંમત સાચવી શકે છે. એ કિંમત ઉપર પ્રક્રિયા કરવા માટે આપણને એક ચલની જરૂર પડે છે. તેથી જો આપણે પાંચ કિંમતો પર પ્રક્રિયા કરવી હોય તો એ માટે પાંચ જુદા જુદા ચલની જરૂર પડે. તો હવે એક એવા પ્રોગ્રામની કલ્પના કરો કે, જેમાં આપણને અનેક ડેટા પર પ્રક્રિયા કરવી પડતી હોય. આવી પરિસ્થિતિમાં, C ભાષા એક ખાસ પ્રકારની ઍરે ડેટા ટાઇપ આપે છે. આ પ્રકરણમાં આપણે એ શીખીશું કે C ભાષામાં ઉપલબ્ધ જુદા જુદા પ્રકારના ઍરેનો કેવી રીતે ઉપયોગ કરવો.

ઍરેની જરૂરિયાત (Need of arrays)

કલ્પના કરો કે, નીચે મુજબની જીવનની કેટલીક વાસ્તવિક પરિસ્થિતિ માટે આપણે C પ્રોગ્રામ લખવા છે :

- વર્ગના બધા વિદ્યાર્થીઓના ગુણ સાચવવા
- સુપર માર્કેટની બધી ચીજવસ્તુઓની કિંમત સાચવવી
- કર્મચારીઓ અને તેમના સંપર્ક સૂત્રની યાદી
- વર્ગના દરેક વિદ્યાર્થીની ઉંચાઈ અને વજનની માહિતી સાચવવી

દરેક વિદ્યાર્થીઓના ગુણની માહિતીનું વ્યવસ્થાપન કરવા marks0, marks1, marks2, marks3, marks4 .... અને marks59, એમ 60 વિદ્યાર્થીઓના ગુણ માટે 60 જુદા જુદા ચલ બનાવવાને બદલે આપણે એક જ ઍરે ચલ જેમ કે, marks[60] વ્યાખ્યાયિત કરવી શકાય. હવે આ ઍરેમાંથી અલગ અલગ વિદ્યાર્થીઓના ગુણ જાણવા માટે આપણે marks[0], marks[1], marks[2] ... એ પ્રમાણે ચલનો ઉપયોગ કરી શકીએ. કમ્પ્યૂટરની સ્મૃતિ(મેમરી)માં marks[60] નામનો ઍરે આકૃતિ 15.1માં દર્શાવ્યા મુજબ દેખાશે.

First element	Second element	Third element	....	....	Last element
marks[0]	marks[1]	marks[2]	marks[3]	.....	marks[59]

આકૃતિ 15.1 : સ્મૃતિમાં ઍરેના માળખાનો નમૂનો

ઍરેને સંબંધિત એવા નીચે મુજબના અગત્યના મુદ્દા નોંધો :

- તે એક જ ડેટાપ્રકારના ઘટકોનો સમૂહ છે.
- તે નિયત કદ ધરાવતા ઘટકોનો ક્રમિક સમૂહ છે. ઍરે સળંગ સ્મૃતિસ્થાનો રોકે છે.
- ઍરેનો કોઈ પણ ઘટક તેના અનુક્રમ દ્વારા ઉપયોગમાં લઈ શકાય છે. ઍરેના નામની પાછળ ચોરસ કૌંસમાં દર્શાવાતા અનુક્રમ(Index number)ને 'સબસ્ક્રિપ્ટ' (subscript) તરીકે પણ ઓળખવામાં આવે છે.
- આ સબસ્ક્રિપ્ટ તરીકે કોઈ પણ પૂર્ણાંક અથવા પૂર્ણાંક પદાવલિ જ હોવી જોઈએ.
- સબસ્ક્રિપ્ટનો ક્રમ શૂન્યથી શરૂ થાય છે.

ઍરેના પ્રકારો (Types of arrays)

'સી' ભાષામાં ઍરેને નીચે મુજબ બે પ્રકારોમાં વિભાજિત કરી શકાય :

(i) એક પરિમાણીય ઍરે (Single or One dimensional array)

(ii) બહુપરિમાણીય ઍરે (Multi dimensional array)

એક પરિમાણીય ઍરેમાં એક ઊભી હરોળ અથવા એક આડી હરોળ હોય છે જ્યારે બહુપરિમાણીય ઍરેમાં એક કે વધુ આડી હરોળ અથવા એક કે વધુ બહુ ઊભી હરોળ હોય છે. ઊભી અને આડી હરોળની સંખ્યા પ્રોગ્રામની જરૂરિયાત અનુસાર ઉપયોગકર્તા દ્વારા નક્કી કરવામાં આવે છે. આપણે માત્ર C ભાષા દ્વારા આપવામાં આવતા એક પરિમાણીય અને દ્વિપરિમાણીય ઍરેની જ ચર્ચા કરીશું.

એકપરિમાણીય ઍરે વ્યાખ્યાયિત કરવા (Declaration of single dimensional array)

C ભાષામાં ઍરેને વ્યાખ્યાયિત કરવા સ્મૃતિ ઘટકોમાં આપણે જે પ્રકારનો ડેટા સાચવવા માંગતા હોઈએ તે અને ઍરેમાં કેટલા ઘટકો રાખવા છે તેને ધ્યાને રાખી આપણે નીચે મુજબ ઍરે વ્યાખ્યાયિત કરીએ છીએ :

datatype arrayname[size];

- *datatype:* ઍરે ક્યા પ્રકારના ઘટકો સાચવે તે નક્કી કરે છે. જો આપણે ઍરેમાં શાબ્દિક માહિતી (characters) સાચવવા ઇચ્છતા હોઈએ તો ઍરેનો પ્રકાર 'char' હોય. એ જ રીતે, પૂર્ણાંક (integers) સંખ્યા સાચવવી હોય તો ઍરેનો પ્રકાર 'int' હોય. ટૂંકમાં, datatype તરીકે C ભાષાનો કોઈ પણ માન્ય ડેટાપ્રકાર હોઈ શકે.
- *arrayname:* આ પ્રોગ્રામર દ્વારા ઍરેને આપવામાં આવતું નામ છે. આ નામ તરીકે કોઈ પણ અક્ષરોનો સમૂહ ચાલી શકે. પરંતુ એવી સલાહ આપવામાં આવે છે કે, ઍરેને એક અર્થસભર નામ આપવું જોઈએ. ઍરેનું નામ બને ત્યાં સુધી પ્રોગ્રામ દ્વારા ઍરેમાં શું સાચવવામાં આવે છે તેના સંદર્ભમાં હોવું જોઈએ.
- *size:* size તરીકે આપવામાં આવતી કિંમત દ્વારા જ ઍરેમાં કેટલા ઘટકો સાચવી શકાશે તે નક્કી થાય છે. આ size તરીકે શૂન્યથી મોટો કોઈ પણ પૂર્ણાંક અચલ હોવો આવશ્યક છે.

ઉદાહરણ તરીકે, 60 વિદ્યાર્થીઓના ગુણની વિગતો સાચવવા આપણે નીચે મુજબનું વિધાન લખી શકીએ :

```
int marks[60];
```

અહીં એ નોંધ લો કે, વ્યાખ્યાયિત કરાયેલો marks નામનો ઍરે માત્ર 60 પૂર્ણાંક સંખ્યાઓ સંગ્રહવા માટે જ સક્ષમ છે. શાબ્દિક માહિતી, અપૂર્ણાંક સંખ્યાઓ અને વધુ મોટી કિંમતો (double) સાચવી શકે તેવા ઍરેના ઉદાહરણ નીચે વ્યાખ્યાયિત કરેલ છે :

```
char string [20];
float percentages [20];
double numbers [20];
```

એકપરિમાણીય ઍરેને કિંમતો આપવી (Assigning values to single dimensional arrays)

કોઈ પણ ઍરેને પ્રારંભિક કિંમતો આપવા માટે C ભાષામાં બે રીતો છે :

(i) કંપાઈલેશન સમયે ઍરેને પ્રારંભિક કિંમતો આપવી.

(ii) પ્રોગ્રામ ચાલે ત્યારે ઍરેને પ્રારંભિક કિંમતો આપવી.

કંપાઈલેશન સમયે ઍરેને પ્રારંભિક કિંમતો આપવી (Compile time array initialization)

ઍરેને વ્યાખ્યાયિત કરતી વખતે કોઈ સામાન્ય ચલની જેમ જ કિંમતો આપી શકાય છે. વિવિધ ઍરે ઘટકોને કિંમતો આપવા માટેની સામાન્ય વાક્યરચના નીચે મુજબ છે :

datatype arrayname[size] = {value1, value2, ... , valueN};

અહીં, *value1, value2, ... , valueN* એ ઍરેના એક પછી એક આગળના ઘટકો માટે પ્રારંભિક કિંમત પૂરી પાડે છે. નીચે કેટલાંક ઉદાહરણ આપ્યાં છે જેમાં આપણે ઍરેના તમામ ઘટકોનું કદ નક્કી કરી તેને કિંમતો આપી શકીએ છીએ.

```
int marks[5] = {60, 65, 70, 75, 80};
```

આ ઉદાહરણ marks નામનો ઍરે ચલ વ્યાખ્યાયિત કરશે, જે પાંચ ઘટકો સાચવી શકશે. આ ઘટકોની કિંમત

marks[0] = 60, marks[1] = 65, marks[2] = 70, marks[3] = 75 અને marks[4] = 80 ગોઠવાશે. આવી રીતે ઍરે વ્યાખ્યાયિત કરવાના કિસ્સામાં જો ઍરેના કદ કરતાં કિંમતોની યાદી નાની હશે, તો શરૂઆતના એક પછી એક ઘટકોને કિંમત આપવામાં આવશે અને પાછળના ઘટકોની કિંમત શૂન્ય રાખવામાં આવશે. ઉદાહરણ તરીકે,

```
int marks[5] = {60, 65, 70};
```

પ્રથમ ત્રણ ઍરે ઘટકોની પ્રારંભિક કિંમત marks[0] = 60, marks[1] = 65 અને marks[2] = 70 થશે. જ્યારે એ પછીના marks[3] અને marks[4] ઘટકની પ્રારંભિક કિંમત 0 અપાશે.

જો ઍરેને વ્યાખ્યાયિત કરતી વખતે જ પ્રારંભિક કિંમત આપવામાં આવે તો ઍરેનું કદ દર્શાવવું મરજિયાત છે. જો આપણે ઍરેનું કદ નહીં દર્શાવીએ તો કંપાઇલર આપમેળે છગડિયા કૌંસ({})ની અંદર આપેલી કિંમતોની સંખ્યા ગણીને નક્કી કરી લેશે. ઉદાહરણ તરીકે, નીચેનું વિધાન

```
int marks[ ] = {60, 65, 70, 75, 80};
```

પાંચ ઘટકો ધરાવતો marks નામનો ઍરે વ્યાખ્યાયિત કરશે જેને છગડિયા કૌંસમાં આપેલી કિંમતો દ્વારા પ્રારંભિક કિંમતો આપવામાં આવશે.

તો ચાલો, ઉદાહરણ 15.1 સાથે એક પરિમાણીય ઍરેને કંપાઇલેશન સમયે પ્રારંભિક કિંમત આપવાના ખ્યાલનો અભ્યાસ કરીએ. આકૃતિ 15.2 ઉદાહરણ 15.1નું કોડ-લિસ્ટિંગ અને પરિણામ રજૂ કરે છે.

```
12-1.c - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
12-1.c

/* Example 1: Program to illustrate compile time initialization of
   one dimensional array.*/

#include<stdio.h>
int main()
{
    int number[5] = {10,20,30,40,50}; // Compile time initialization
    int i;
    for( i = 0; i < 5 ; i++)           // Loop to display array elements
    {
        printf(" %d  \n", number[i] );
    }
   return 0;
}
/* End of Program*/
```

```
gcc 12-1.c
>gcc 12-1.c
>Exit code: 0
./a.out
>./a.out
 10
 20
 30
 40
 50
>Exit code: 0
```

આકૃતિ 15.2 : ઉદાહરણ 15.1નું કોડ-લિસ્ટિંગ અને પરિણામ

સમજૂતી

main વિધેય પછીની પ્રથમ લીટીમાં number નામનો ઍરે વ્યાખ્યાયિત કરવામાં આવ્યો છે અને કંપાઇલેશન સમયે જ તમામ ઍરે ઘટકોને છગડિયા કૌંસમાં આપેલી કિંમતો વડે પ્રારંભિક કિંમત આપવામાં આવે છે. for લૂપ number નામના ઍરેના અલગ-અલગ ઘટકને દર્શાવે છે.

અમલ દરમ્યાન ઍરેને પ્રારંભિક કિંમતો આપવી

જ્યારે આપણને પ્રોગ્રામનો અમલ કરતી વખતે ઉપયોગકર્તા પાસેથી કિંમતો મેળવવાની જરૂર પડતી હોય, તો આપણે અમલ દરમ્યાન ઍરેની પ્રારંભિક કિંમત આપવાની પદ્ધતિનો ઉપયોગ કરી શકીએ. ઍરેમાં સંગૃહિત કિંમતોનો કેવી રીતે ઉપયોગ કરવો તે પણ આ પ્રોગ્રામ દર્શાવે છે. આકૃતિ 15.3, ઉદાહરણ 15.2નું કોડ-લિસ્ટિંગ આપે છે, જ્યારે આકૃતિ 15.4 તેનું પરિણામ દર્શાવે છે.

```
12-2.c * SciTE
File Edit Search View Tools Options Language Buffers Help
12-2.c *

/* Example 2: Program to illustrate runtime array initialization and accessing values
   from one dimensional array. Program will read marks of five students and display the
   average marks.*/

#include<stdio.h>
int main()
{
      int marks[5], i, total=0;
      for( i = 0; i < 5; i++)                    // loop to read marks of five students
      {
            printf("Enter marks[%d] :", i);
            scanf("%d", &marks[i]);
      }
      for( i = 0; i < 5; i++)                    // loop to find total marks
      {
            total = total + marks[i];
      }
      printf("\nThe average marks of students are %d. \n", total/5);
      return 0;
}
/* End of Program*/
```

આકૃતિ 15.3 : ઉદાહરણ 15.2નું કોડ-લિસ્ટિંગ

```
kpp@ubuntu: ~
File Edit View Terminal Help
kpp@ubuntu:~$ gcc 12-2.c
kpp@ubuntu:~$ ./a.out
Enter marks[0] :5
Enter marks[1] :10
Enter marks[2] :15
Enter marks[3] :20
Enter marks[4] :25

The average marks of students are 15.
kpp@ubuntu:~$
```

આકૃતિ 15.4 : ઉદાહરણ 15.2નું પરિણામ

સમજૂતી

પ્રોગ્રામની પ્રથમ *for* લૂપ પાંચ વિદ્યાર્થીઓના ગુણ વાંચશે. બીજી *for* લૂપ marks નામના ઍરેના અલગ-અલગ ઘટક વાંચીને કુલ ગુણ ગણશે. છેલ્લું printf વિધેય વિદ્યાર્થીઓના સરેરાશ ગુણ દર્શાવશે.

કૅરેક્ટર ઍરે તરીકે શબ્દમાળા

શબ્દમાળા એ અક્ષરોની હારમાળા છે, જે સામાન્ય રીતે એક ઘટક તરીકે લેવામાં આવે છે. C ભાષા ડેટાના એક પ્રકાર તરીકે શબ્દમાળા માટે string ડેટા પ્રકારને સમર્થન આપતી નથી. જો કે, તેમ છતાં તે આપણને કૅરેક્ટર ઍરે

તરીકે કોઈ પણ શાબ્દિક માહિતીને રજૂ કરવાની છૂટ આપે છે. શાબ્દિક ચલ એ C ભાષાનું કોઈ પણ માન્ય ચલનું નામ છે, અને તે હંમેશાં કૅરેક્ટર ઍરે તરીકે જ વ્યાખ્યાયિત કરવામાં આવે છે. શાબ્દિક ચલને વ્યાખ્યાયિત કરવા માટેનું સામાન્ય સ્વરૂપ નીચે મુજબ છે:

```
char stringname [size];
```

અહીં, size એ stringname નામના ચલમાં અક્ષરોની સંખ્યા દર્શાવે છે. નીચે કેટલાક માન્ય શાબ્દિક ચલને વ્યાખ્યાયિત કરવાનાં ઉદાહરણ આપેલ છે :

```
char student_name [20];
char city [50];
char state [20];
```

અહીં એ નોંધવું જરૂરી છે કે C ભાષામાં શાબ્દિક માહિતીનો અંત ખાલી અક્ષર (null character) તરીકે ઓળખાતા '\0'; ખાસ ચિહ્નથી આવે છે. આ ખાસ ચિહ્ન પ્રોગ્રામને શાબ્દિક માહિતીનો અંત ઓળખવામાં મદદરૂપ બને છે. આમ, કોઈ પણ શાબ્દિક માહિતીના અંતે '\0' (null character) ચિહ્ન મૂકવું જરૂરી હોવાથી, આપણી શાબ્દિક માહિતીના બધા શબ્દોને સમાવી શકાય તે માટે કૅરેક્ટર ઍરેમાં શબ્દોની કુલ સંખ્યા ઉપરાંત એક વધારાના સંગ્રહસ્થાનની જરૂર પડે છે.

સાંખ્યિક ઍરેની જેમ જ, શાબ્દિક ઍરેને પણ કંપાઇલ કરતી વખતે અથવા પ્રોગ્રામના અમલ દરમ્યાન એમ બે રીતે પ્રારંભિક કિંમતો આપી શકાય છે. કૅરેક્ટર ઍરેને નીચે મુજબ જુદી જુદી રીતે પ્રારંભિક કિંમતો આપી શકાય :

```
char student_name[6] = "PURVA";
char student_name[6] = {'P', 'U', 'R', 'V', 'A', '\0'};
```

student_name નામના ઍરે કદ 6 ઘટકોનું રાખવા પાછળનું કારણ એ છે કે, PURVA શબ્દમાં પાંચ અક્ષર હોવાથી તેના 5 અને છેલ્લે null character ('\0') મૂકવા એક વધારાનો ઘટક એમ કુલ મળી 6 ઘટક. અહીં એ નોંધવું જોઈએ કે, જ્યારે તેના ઘટકોની યાદી રજૂ કરીને આપણે કૅરેક્ટર ઍરેને પ્રારંભિક કિંમતો આપીએ ત્યારે શાબ્દિક માહિતીના અંતે null character '\0' દર્શાવવું જરૂરી છે. આપણે કદ દર્શાવ્યા વગર પણ નીચે મુજબ કૅરેક્ટર ઍરે વ્યાખ્યાયિત કરી શકીએ :

```
char state[ ] = {'G', 'U', 'J', 'A', 'R', 'A', 'T', '\0'};
```

આ કિસ્સામાં state નામનો કૅરેક્ટર ઍરે વ્યાખ્યાયિત થઈ જશે અને તેને 8 પ્રારંભિક કિંમતો અપાઈ જશે. નીચેનાં વિધેયોનો ઉપયોગ કરીને આપણે અગાઉ ચર્ચા કરેલ કૅરેક્ટર ઍરેની માહિતી જોઈ શકીએ :

```
printf("The name of student is %s", student_name);
printf("The name of state is %s", state);
printf("The first character of your name is %c", student_name[0]);
```

બહુપરિમાણીય ઍરે (Multi dimensional array)

એકપરિમાણીય ઍરે માહિતીને રૈખિક ક્રમમાં જાળવે છે. જો કે, કેટલીક ઉપયોગી પદ્ધતિઓ (જેવી કે, સાંખ્યિક છબી, ચેસ રમતનું બોર્ડ વગેરે) સાથે સંકળાયેલ માહિતી દ્વિપરિમાણીય હોય છે. પ્રોગ્રામમાં આ પ્રકારની પદ્ધતિનો વિચાર કરતી વખતે આપણને બહુપરિમાણીય ઍરે બનાવવાની છૂટ આપે છે. અગાઉ જણાવ્યા મુજબ, બહુપરિમાણીય ઍરેમાં એક કરતાં વધુ ઊભી અને આડી હરોળ હોય છે. C ભાષામાં આપણે દ્વિપરિમાણીય, ત્રિપરિમાણીય અથવા N-પરિમાણીય ઍરે બનાવી શકીએ. બહુપરિમાણીય ઍરેનું સૌથી સાદું સ્વરૂપ દ્વિપરિમાણીય ઍરે છે. દ્વિપરિમાણીય ઍરેમાં આડી હરોળ માટે એક અને ઊભી હરોળ માટે એક એમ બે સબસ્ક્રિપ્ટ હોય છે. આ પ્રકરણમાં આપણે માત્ર દ્વિપરિમાણીય ઍરેની જ ચર્ચા કરીશું.

દ્વિપરિમાણીય ઍરેને વ્યાખ્યાયિત કરવો

અત્યાર સુધી આપણે એકપરિમાણીય ઍરેની ચર્ચા કરી, જે તેની અંદર ઘણી કિંમતો સાચવી શકે. જીવનમાં એવી ઘણી પરિસ્થિતિ હોય છે કે જેમાં આપણને ડેટાનું કોષ્ટક સાચવવાની જરૂર પડે. કોષ્ટક 15.1માં આપેલ કિસ્સા અંગે વિચારો. તે સોમવારથી શુક્રવાર દરમિયાન જુદા જુદા ત્રણ સેલ્સમૅન દ્વારા વેચવામાં આવેલ વસ્તુની વિગત દર્શાવે છે.

	Monday	Tuesday	Wednesday	Thursday	Friday
Salesman1	100	150	200	250	200
Salesman2	200	250	300	350	300
Salesman3	150	200	250	300	250

કોષ્ટક 15.1 : વેચાણની વિગતો

આ કોષ્ટક જુદી જુદી 15 કિંમતો ધરાવે છે. અહીં એકપરિમાણીય ઍરે વ્યાખ્યાયિત કરવો સલાહભર્યો નથી. દ્વિપરિમાણીય ઍરેનો ઉપયોગ કરવો એ વધુ સારો વિકલ્પ છે. C ભાષાનો દ્વિપરિમાણીય ઍરે વ્યાખ્યાયિત કરી કોષ્ટક 15.1ને સરળતાથી રજૂ કરી શકાય. દ્વિપરિમાણીય ઍરે નીચે મુજબ વ્યાખ્યાયિત કરી શકાય :

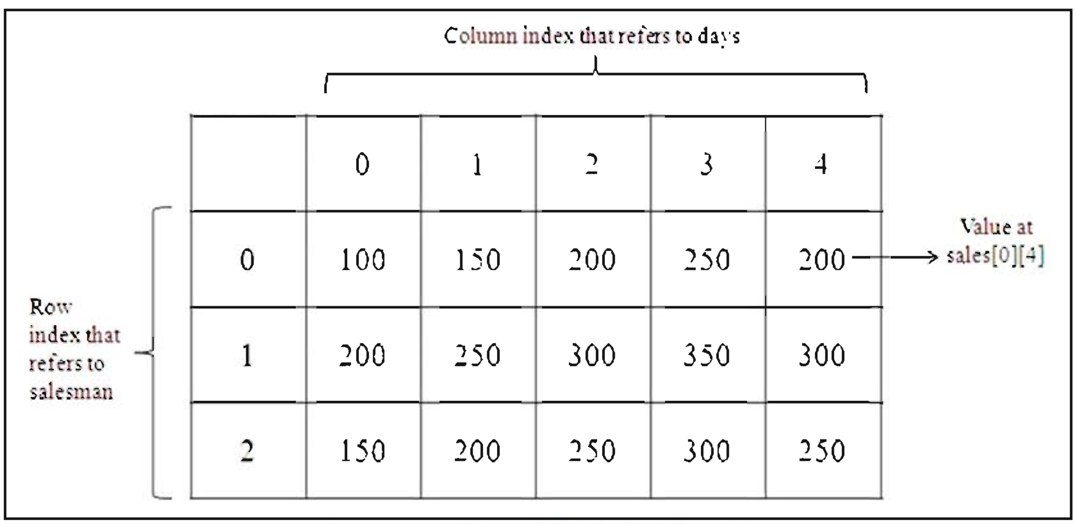
datatype arrayname [row size][column size];

row size અને ***column size*** શૂન્ય કરતાં મોટી પૂર્ણાંક અચલ સંખ્યા જ હોવી જોઈએ અને datatype તરીકે C ભાષાનો કોઈ પણ માન્ય ડેટા પ્રકાર હોઈ શકે. અહીં એ નોંધ કરો કે, C ભાષામાં બહુપરિમાણીય ઍરે એ આડી હાર અગ્રિમતા (row major) ધરાવે છે. બીજા શબ્દોમાં કહીએ તો, ઍરેને વ્યાખ્યાયિત કરતી વખતે દર્શાવવામાં આવતા બે કૌંસમાંથી પ્રથમ કૌંસ આડી હાર(rows)ની સંખ્યા નક્કી કરે છે.

કોષ્ટક 15.1માં દર્શાવેલ વિગતોને સાચવવા આપણે નીચે મુજબ દ્વિપરિમાણીય ઍરે વ્યાખ્યાયિત કરી શકીએ છીએ:

int sales[3][5];

અહીં, આડી હારની સંખ્યા 3 છે, જે સેલ્સમૅનને સંબોધે છે. (Salesman1 = 0, Salesman2 = 1 અને Salesman3 = 2) જ્યારે ઊભી હરોળની સંખ્યા 5 છે, જે અઠવાડિયાના જુદા જુદા વારને સંબોધવા ઉપયોગમાં લેવાશે. (Monday = 0, Tuesday = 1, Wednesday = 2, Thursday = 3 અને Friday = 4) આપણે એવું અનુમાન કર્યું છે કે, વેચાણના એકમોની સંખ્યા પૂર્ણાંક જ હશે. આકૃતિ 15.5 વેચાણની વિગતોની સ્મૃતિરચનાનો નમૂનો દર્શાવે છે.



આકૃતિ 15.5 : વેચાણની વિગતોની સ્મૃતિરચના

હવે, Salesman1 દ્વારા Mondayના રોજ કરવામાં આવેલ ચીજવસ્તુના વેચાણની વિગત ચકાસવા, આપણે ઍરે ઘટક sales[0][0]નો ઉપયોગ કરી શકીએ. sales[0][0] ખાતે સાચવેલ કિંમત 100 છે. એ જ રીતે, Mondayના રોજ Salesman2 અને Salesman3 દ્વારા કરવામાં આવેલ વેચાણની વિગત ચકાસવા આપણે અનુક્રમે ઍરે ઘટક sales[1][0] અને sales[2][0]નો ઉપયોગ કરી શકીએ. આકૃતિ 15.5માં દર્શાવ્યા મુજબ sales[0][4] સ્થાન પર સાચવેલી કિંમત 200 છે અને તે Fridayના રોજ Salesman1 દ્વારા કરવામાં આવેલ વસ્તુનું વેચાણ દર્શાવે છે.

દ્વિપરિમાણીય ઍરેને પ્રારંભિક કિંમતો આપવી

એકપરિમાણીય ઍરેની જેમ જ, દ્વિપરિમાણીય ઍરેને પણ આપણે કંપાઈલ કરતી વખતે અથવા પ્રોગ્રામના અમલ દરમિયાન પ્રારંભિક કિંમતો આપી શકીએ.

કંપાઈલ કરતી વખતે દ્વિપરિમાણીય ઍરેને પ્રારંભિક કિંમતો આપવી

કોષ્ટકના પ્રોગ્રામને કંપાઇલ કરતી વખતે કોષ્ટક 15.1 માં દર્શાવેલ ઍરેને પ્રારંભિક કિંમતો આપવાનું કાર્ય નીચે મુજબ કરી શકાય :

```
int sales[3][5]={
                {100, 150, 200, 250, 200},
                {200, 250, 300, 350, 300},
                {150, 200, 250, 300, 250}
        };
```

બહારના છગડિયા કૌંસની અંદર રહેલા છગડિયા કૌંસ, જે નિશ્ચિત સેલ્સમૅન માટેની આડી હરોળ દર્શાવે છે, તે મરજિયાત છે. નીચે મુજબ પ્રારંભિક કિંમતો આપવાની પદ્ધતિ અગાઉની પદ્ધતિ જેવું જ સમાન કાર્ય કરે છે.

```
int sales[3][5]={100, 150, 200, 250, 200, 200, 250, 300, 350, 300, 150, 200, 250, 300, 250};
```

પ્રોગ્રામના અમલ વખતે દ્વિપરિમાણીય ઍરેને પ્રારંભિક કિંમતો આપવી

પ્રોગ્રામનો અમલ કરતી વખતે કોષ્ટક 15.1માં દર્શાવેલ કોષ્ટકના ઍરેને પ્રારંભિક કિંમતો આપવાનું કાર્ય કોડ-લિસ્ટિંગ 15.1માં આપેલ પ્રોગ્રામ દ્વારા કરી શકાય.

```
int sales[3][5], row, column;
for( row = 0; row < 3; row++) {
        for (column = 0; column < 5; column++){
                printf("Enter sales item [%d][%d]:", row, column);
                scanf("%d", &sales[row][column]);
        }
}
```

કોડ-લિસ્ટિંગ 15.1 : અમલ દરમિયાન ડેટા ઉમેરવાનો પ્રોગ્રામ

આ પ્રોગ્રામ ત્રણ આડી હરોળ અને પાંચ ઊભી હરોળ સાથેનું sales નામનું દ્વિપરિમાણીય ઍરે વ્યાખ્યાયિત કરે છે. for લૂપ દ્વારા આપેલ તર્ક, sales નામના ઍરે માટે વિવિધ કુલ પંદર કિંમતો વાંચશે. sales ઍરેના અલગ-અલગ ઘટકોને કોડ-લિસ્ટિંગ 15.2 માં આપેલ પ્રોગ્રામ દ્વારા સ્ક્રીન પર દર્શાવી શકાય.

```
for( row = 0; row < 3; row++) {
        for (column = 0; column < 5; column++){
                printf("[%d][%d] : = %d \n", row, column, sales[row][column]);
        }
}
```

કોડ-લિસ્ટિંગ 15.2 : ઍરેની માહિતી દર્શાવવા માટેનો પ્રોગ્રામ

દ્વિપરિમાણીય ઍરેનું ઉદાહરણ

હવે આપણે પ્રોગ્રામના અમલ દરમિયાન એકપરિમાણીય અને દ્વિપરિમાણીય ઍરે વ્યાખ્યાયિત કરવા અને તેની કિંમતોનો ઉપયોગ દર્શાવતો પ્રોગ્રામ સમજીએ. આ પ્રોગ્રામ વિદ્યાર્થીઓના ગુણને સાચવવા માટે દ્વિપરિમાણીય ઍરે વ્યાખ્યાયિત કરશે. આ પ્રોગ્રામની સૂચનાઓ તેના પરિણામસહિત કોડ-લિસ્ટિંગ 15.3માં આપેલ છે.

```
/* Example 3 : Program to illustrate use of one અને two dimensional array. The program
will maintain marks of two students in three quizzes using two dimensional arrays. The
total marks of each student will be displayed.*/

#include<stdio.h>
#define STD 2    /*Number of rows for student data*/
#define QUIZ 3   /*Number of columns for quiz data*/

int main(){
        int marks[STD][QUIZ];              /* Marks array for 2 students અને 3 quizzes */
        int i, j, total_quiz[STD] = {0};

        for( i = 0; i < STD; i++)           /* Loop for student row */
        {
            for( j = 0; j < QUIZ;  j++)     /* Loop for quiz column */
                    {
               printf("Enter marks of student %d in quiz %d: ", i+1, j+1);
               scanf("%d", &marks[ i ][ j ]);          /*  Reading marks of quiz */
               total_quiz[ i] = total_quiz[ i] + marks[ i][ j];
            }
        }

          /* Logic to print student no., quiz marks અને total... */

        for ( i = 0 ; i < QUIZ; i++)        /* Loop for printing quiz number... */
        {
          printf(" \tQuiz %d" , i+1);
        }

        printf(" Total \n");
        for( i = 0; i < STD; i++)           /*  Loop for student row */
        {
          printf("Student %d:", i+1);
          for( j = 0; j < QUIZ;  j++)       /* Loop for quiz column */
          {
                 printf("%d \t", marks[ i ][ j ]);
           }
          printf("%d \n", total_quiz[ i]);
        }
        return 0;
}
/* End of Program*/
```

```
/******************************************************************************/
Output:

Enter marks of student 1 in quiz 1 : 20
Enter marks of student 1 in quiz 2 : 30
Enter marks of student 1 in quiz 3 : 40
Enter marks of student 2 in quiz 1 : 50
Enter marks of student 2 in quiz 2 : 40
Enter marks of student 2 in quiz 3 : 20

              Quiz1     Quiz2     Quiz3     Total
Student 1 :    20        30        40        90
Student 2 :    50        40        20        110
/******************************************************************************/
```

કોડ-લિસ્ટિંગ 15.3 : ઉદાહરણ 15.3ની સૂચનાઓ અને પરિણામ

સમજૂતી

અહીં, આપણે આપણા પ્રોગ્રામમાં બે પ્રતીકાત્મક અચલ STD અને QUIZનો ઉપયોગ કરેલ છે. પ્રોગ્રામ દ્વારા કેટલા વિદ્યાર્થીઓની સંખ્યાને સંભાળવાની છે તે STD દ્વારા નિયંત્રિત થાય છે. એ જ રીતે, પ્રોગ્રામ દ્વારા સંભાળવામાં આવનાર પ્રશ્નોની સંખ્યા QUIZ દ્વારા નક્કી થશે. આપણે, વિદ્યાર્થીઓના પ્રશ્નોના ગુણ સાચવવા marks નામનો દ્વિપરિમાણીય ઍરે વ્યાખ્યાયિત કર્યો છે. કોઈ એક વિદ્યાર્થી દ્વારા મેળવાયેલ તમામ પ્રશ્નોના કુલ ગુણ સાચવવા માટે total_quiz નામનો એકપરિમાણીય ઍરે વ્યાખ્યાયિત કરવામાં આવ્યો છે. total_quizને વ્યાખ્યાયિત કરતી વખતે જ તેના બધા ઘટકોને પ્રારંભિક કિંમત તરીકે શૂન્ય આપવામાં આવેલ છે. આપણી જરૂરિયાત અનુસાર for લૂપના વિધાનોને પુનરાવર્તિત કરવા માટે i અને j નામના બે ગણતરી દર્શક (counter) ચલ વ્યાખ્યાયિત કરવામાં આવ્યા છે.

વિદ્યાર્થીઓના પ્રશ્નોના ગુણ વાંચવા એક for લૂપની અંદર વળી બીજી for લૂપ(નેસ્ટેડ લૂપ)નો ઉપયોગ કરવામાં આવ્યો છે. લૂપની અંદર આપણે એક વિદ્યાર્થી દ્વારા તમામ પ્રશ્નોના કુલ ગુણની ગણતરી કરી છે. એ પછીની for લૂપનો ઉપયોગ પરિણામને સ્ક્રીન પર યોગ્ય રીતે રજૂ કરવા માટે કરવામાં આવ્યો છે. આપેલ પ્રોગ્રામ માત્ર બે વિદ્યાર્થીઓના ત્રણ પ્રશ્નોના ગુણને સાચવવા માટે જ સક્ષમ છે. પરંતુ, પ્રોગ્રામમાં STD અને QUIZની કિંમતો બદલીને આપણે આપણા પ્રોગ્રામને વધુ ક્રિયાશીલ બનાવી શકીએ.

ઍરેની સીમાની ચકાસણી

ઍરે એ C ભાષાની એક સક્ષમતા છે, પરંતુ અહીં એ નોંધવું અગત્યનું છે કે, C ભાષામાં ઍરેની સીમાની ચકાસણીની કોઈ વ્યવસ્થા નથી. ઍરેની સીમાની ચકાસણી એટલે કે પ્રોગ્રામમાં વ્યાખ્યાયિત કરેલ ઍરેની સીમાની ચકાસણી કરવી. ઍરે વ્યાખ્યાયિત કરતું નીચે આપેલું વિધાન ચકાસો.

```
int number[5];
```

આ વિધાન પાંચ પૂર્ણાંક કિંમતો સાચવતા number નામના ઍરેને વ્યાખ્યાયિત કરશે. આપણે number[0], number[1] ..... number[4]નો ઉપયોગ કરીને અલગ-અલગ ઍરે ઘટકોની કિંમતોનો ઉપયોગ કરી શકીશું. C ભાષા, ઍરેના [ ]માં કોઈ પણ ક્રમસંખ્યા લખવા માટે પ્રોગ્રામરને સત્તા આપે છે. એનો મતલબ, આપણે number [-1] અને number [6] અથવા અન્ય કોઈ પણ અમાન્ય અનુક્રમનો ઉપયોગ કરી શકીએ. ઍરેના અનુક્રમની સીમા ચકાસવા માટેની કોઈ સુવિધા ન હોવાને કારણે આપણને પ્રોગ્રામમાં કોઈ ભૂલ દર્શાવશે નહીં. પરંતુ, કદાચ પ્રોગ્રામ આપણને કિંમતરૂપે કોઈ ખોટી કિંમત આપશે અથવા પ્રોગ્રામ અધવચ્ચેથી બંધ (crash) થઈ જશે.

સારાંશ

આ પ્રકરણમાં આપણે C ભાષામાં એકપરિમાણીય અને દ્વિપરિમાણીય ઍરેના ઉપયોગ સંબંધિત અગત્યની લાક્ષણિકતાઓ શીખ્યા. આપણે એ યાદ કરવું જોઈએ કે, ઍરેનો ડેટા પ્રકાર C ભાષાનો કોઈ પણ માન્ય ડેટા પ્રકાર હોઈ શકે છે. ઍરેને વ્યાખ્યાયિત કરવામાં સબસ્ક્રિપ્ટ તરીકે કોઈ પણ પૂર્ણાંક કે પૂર્ણાંક પદાવલિ જ હોવી જરૂરી છે. ઍરેનો સબસ્ક્રીપ્ટ ક્રમ (index) શૂન્યથી શરૂ થાય છે. કૅરેક્ટર ઍરેનો અંત ખાસ ચિહ્ન (null character) '\0' થી જ આવે છે. ઍરેની સીમાની ચકાસણીની કોઈ વ્યવસ્થા C ભાષામાં નથી. પ્રોગ્રામમાં માન્ય સબસ્ક્રિપ્ટ ક્રમનો ઉપયોગ કરવાની જવાબદારી પ્રોગ્રામરની છે.

સ્વાધ્યાય

1. પ્રોગ્રામમાં એકપરિમાણીય ઍરેનો ઉપયોગ કરવાના ફાયદા જણાવો.
2. એકપરિમાણીય અને દ્વિપરિમાણીય ઍરે વચ્ચેનો તફાવત સ્પષ્ટ કરો.
3. કંપાઇલ સમયે ઍરેના ઘટકોને પ્રારંભિક કિંમત આપવી અને પ્રોગ્રામના અમલ દરમિયાન ઍરેના ઘટકોને પ્રારંભિક કિંમત આપવી એટલે શું?
4. નીચેના પ્રોગ્રામ કોડમાં ભૂલો શોધો, જો ભૂલ જણાય તો તેને સુધારો અને તેનું પરિણામ જણાવો :

(a)
```
#include <stdio.h>
int main( ) {
    int num[3][3] = {{1, 2}, {3, 4}};
    printf("%d \n", number[1][1]);
    return 0;
}
```

(b)
```
#include<stdio.h>
void main()
{
    char str[ ] = 'INDIA'
    printf("%c %c %c %c  c", str[0], str[1], str[2], str[0], str[4]);
}
```

5. ખાલી જગ્યા પૂરો :

(a) દ્વિપરિમાણીય ઍરેને આડી હરોળ અને ____________ હોય છે.

(b) ઍરે ઘટક ____________ સ્મૃતિસ્થાનો રોકે છે.

(c) ઍરેની પાછળ ચોરસ કૌંસમાં લખવામાં આવતા અનુક્રમ (Index)ને ____________ તરીકે પણ ઓળખવામાં આવે છે.

(d) સબસ્ક્રિપ્ટ પૂર્ણાંક હોવો જોઈએ અથવા ____________ પદાવલિ હોઈ શકે.

(e) C ભાષાના કૅરેક્ટર ઍરેમાં શાબ્દિક માહિતીનો ખાસ ચિહ્ન ____________ દ્વારા અંત લાવવામાં આવે છે.

6. નીચેનાં વિધાનો ખરાં છે કે ખોટાં તે જણાવો :

(a) ઍરેના ઘટકોને પ્રારંભિક કિંમતો માત્ર પ્રોગ્રામનો અમલ કરતી વખતે જ આપી શકાય છે.

(b) C ભાષામાં ઍરેના સબસ્ક્રિપ્ટની કિંમત 1થી શરૂ થાય છે.

(c) બહુપરિમાણીય ઍરેનું સાદું સ્વરૂપ એકપરિમાણીય ઍરે છે.

(d) C ભાષા string ડેટા ટાઇપનું સમર્થન કરે છે.

(e) દ્વિપરિમાણીય ઍરેને વ્યાખ્યાયિત કરતી વખતે પ્રથમ કૌંસ કુલ આડી હારની સંખ્યાનો નિર્દેશ કરે છે.

(f) int temperature[6] = {35, 32, 42}; એ ઍરેને પ્રારંભિક કિંમતો આપવાની માન્ય પદ્ધતિ છે.

(g) int num[ ] = {{1, 1}, {2, 2}}; એ પ્રારંભિક કિંમતો આપવાની માન્ય પદ્ધતિ છે.

7. નીચેના પ્રશ્નોના આપેલા વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરી જવાબ આપો :

(1) C ભાષામાં જો કોઈ ઍરેના કુલ ઘટકની સંખ્યા કરતાં મોટી સંખ્યાના સબસ્ક્રિપ્ટ દ્વારા માહિતી મેળવવાનો પ્રયત્ન કરવામાં આવે તો શું થશે?

(a) ઘટકને શૂન્ય બનાવી દેવાશે.

(b) કંપાઇલર ભૂલ દર્શાવશે.

(c) પ્રોગ્રામ અધવચ્ચેથી અટકી પડે અથવા ખોટી કિંમત આપશે.

(d) ઍરેનું કદ આપમેળે વધારી આપવામાં આવશે.

(2) નીચેના પ્રોગ્રામનું પરિણામ શું આવશે?

```
int num[5] = {1, 2, 3, 4, 5};
int i, j;
i = num[1];
j = num[2];
printf("%d, %d, %d", i, j, num[0]);
```

(a) 1, 2, 3 (b) 2, 3, 1 (c) 1, 2, 0 (d) 3, 4, 5

(3) નીચેના પ્રોગ્રામનું પરિણામ શું આવશે?

```
int num[5] = {1, 2, 3};
printf("%d, %d", num[0], num[3]);
```

(a) 1, 2 (b) 1, 0 (c) 1, 3 (d) 2, 3

પ્રાયોગિક સ્વાધ્યાય

નીચેનાં કાર્યો પાર પાડવા માટેનો C પ્રોગ્રામ લખો :

1. ઉપયોગકર્તા પાસેથી 5 × 5નાં બે કોષ્ટક વાંચો. આ બે કોષ્ટકોનો સરવાળો કરી ત્રીજું કોષ્ટક બનાવી તેનું પરિણામ સ્ક્રીન પર દર્શાવો.

2. નીચે દર્શાવ્યા મુજબ પ્રોગ્રામમાં ઍરેનો ઉપયોગ કરી 20 વિદ્યાર્થીઓના ગુણ વાંચો :

Roll No.	Marks
1	60
2	70
...	....
20	75

- બધા પૈકી સૌથી વધુ ગુણ ધરાવતા વિદ્યાર્થીનો રોલ નંબર દર્શાવો.
- 50 કરતાં વધુ ગુણ ધરાવતા કુલ વિદ્યાર્થીઓની સંખ્યા દર્શાવો.
- 50 કરતાં ઓછા ગુણ ધરાવતા કુલ વિદ્યાર્થીઓની સંખ્યા દર્શાવો.

3. એકપરિમાણીય ઍરેમાં 10 પૂર્ણાંક સંખ્યા વાંચો. ઍરેના ઘટકોને અવળા ક્રમમાં દર્શાવો. ઉદાહરણ તરીકે, જો ઉપયોગકર્તાનું ઇનપુટ 10, 20, 30, 40 હોય તો પરિણામ 40, 30, 20, 10 એમ આવવું જોઈએ.

4. ઍરેના N ઘટકોમાંથી મહત્તમ (maximum), લઘુતમ (minimum) અને સરેરાશ શોધો.

5. ઉપયોગકર્તા પાસેથી 3 × 3ના કોષ્ટકની પૂર્ણાંક સંખ્યા વાંચો. કોષ્ટકના દરેક ઘટકનો સરવાળો ગણીને દર્શાવો.

વિધેય

C ભાષાની અનેક ક્ષમતાઓ પૈકીની એક ક્ષમતા છે વિધેય (funcion) બનાવવા અને તેનો ઉપયોગ કરવો. વિધેય એ C ભાષાના કેટલાંક માન્ય વિધાનોનો સમૂહ છે, જે કોઈ ચોક્કસ કાર્ય પાર પાડે છે. વિધેય (funcion) ને પદ્ધતિ (method) સબરૂટિન (sub-routine) અથવા પ્રક્રિયા (procedure) તરીકે પણ ઓળખવામાં આવે છે. અત્યાર સુધી આપણે આપણા પ્રોગ્રામમાં C ભાષાના અસંખ્ય વિધેયોનો ઉપયોગ કર્યો છે. આપણે main( ), printf( ) અને scanf( ) જેવા વિધેયોનો ઉપયોગ કરવાથી પરિચિત છીએ. તે C ભાષાના વિધેયોનું જ ઉદાહરણ છે. ચલાવી શકાય તેવા (executable) કોઈ પણ C પ્રોગ્રામમાં ઓછામાં ઓછું એક વિધેય main( ) તો હોય જ છે.

કોઈ પણ પ્રોગ્રામમાં વિધેયનો ઉપયોગ તે પ્રોગ્રામને મૉડ્યુલર (modular) બનાવે છે. મૉડ્યુલારિટી (Modularity) એટલે, કોઈ પણ જટિલ સમસ્યાને નાની નાની પેટા-સમસ્યાઓમાં વિભાજિત કરવી, જે સમજવામાં અને જાળવણીમાં સરળ બને. એકવાર કોઈ સમસ્યાને નાની નાની પેટા સમસ્યાઓમાં વિભાજિત કરી દેવાય એ પછી આપણે એ દરેક પેટા સમસ્યા માટે વિધેય લખી શકીએ. એ પછી, પ્રોગ્રામમાં બધા વિધેયોનો એક સાથે ઉપયોગ કરવાથી સમસ્યાનો ઉકેલ મળી જશે. આ પ્રકરણમાં આપણે C ભાષાના વિવિધ પ્રકારના વિધેયોની ચર્ચા કરીશું. પ્રોગ્રામમાં વિધેયને કેવી રીતે વ્યાખ્યાયિત કરવા અને કેવી રીતે ઉપયોગમાં લેવા તે પણ શીખીશું.

C ભાષામાં વિધેયોના બે પ્રકાર છે : (1) લાઈબ્રેરી વિધેય અથવા સિસ્ટમ ડિફાઈન્ડ વિધેય (2) યુઝર ડિફાઈન્ડ વિધેય

લાઈબ્રેરી વિધેય

C ભાષાની સ્ટાન્ડર્ડ લાઇબ્રેરીમાં એવા ઘણા આંતરપ્રસ્થાપિત વિધેયો ઉપલબ્ધ છે, જેને આપણે આપણા પ્રોગ્રામમાં ઉપયોગમાં લઈ શકીએ તેને સિસ્ટમ ડિફાઈન્ડ ફંક્શન તરીકે પણ ઓળખવામાં આવે છે. ઉદાહરણ તરીકે scanf( ), printf( ), sqrt( ) અને cos( ) એ લાઇબ્રેરી વિધેયનાં ઉદાહરણ છે. સામાન્ય ઉપયોગકર્તા દ્વારા લાઇબ્રેરી વિધેયો લખવાની જરૂર નથી. આ વિધેયો ઉપયોગમાં લેવા C ભાષાની લાયબ્રેરીમાં કંપાઈલ (compile) કરેલા તૈયાર આપવામાં આવેલાં હોય છે. ઉદાહરણ તરીકે, sqrt( ) અને cos( ) એ ગણિત સંબંધી છે અને તેથી તેને <math.h> નામની હેડર (header) ફાઈલમાં મૂકવામાં આવેલ છે. કોઈ પણ લાઇબ્રેરી વિધેયનો ઉપયોગ કરવા માટે, આપણે તેની સંબંધિત હેડર ફાઈલને આપણા પ્રોગ્રામમાં સામેલ કરવી પડે. કેટલીક જાણીતી હેડર ફાઈલની યાદી અને તે ફાઈલમાંના વિધેયો આ પુસ્તકના પરિશિષ્ટ-IVમાં આપેલ છે.

તો ચાલો, ઉદાહરણ 16.1 મારફત એકાદ લાઇબ્રેરી વિધેયને સમજીએ. આ પ્રોગ્રામ આંતરપ્રસ્થાપિત લાઇબ્રેરી વિધેય pow( )નો ઉપયોગ કરે છે, જે xનો ઘાતાંક y પરત આપે છે. આકૃતિ 16.1 કોડ-લિસ્ટિંગ અને ઉદાહરણ 16.1નું પરિણામ આપે છે. ઉદાહરણ 16.1માં આપણે printf() વિધેયનો ઉપયોગ કરેલ છે, જે <stdio.h> હેડર ફાઈલમાં વ્યાખ્યાયિત કરેલ છે. આપણે pow() વિધેયનો ઉપયોગ કરેલ છે, જે 8 નો ઘાતાંક 2 કિંમત પરિણામ તરીકે આપે છે. pow() વિધેય <math.h> ફાઈલમાં વ્યાખ્યાયિત કરેલ છે.

```
13-1.c - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 13-1.c

/* Example 1: Program to illustrate use of pow() library
funtion */

#include<stdio.h>
#include<math.h>
int main()
{
        float answer;

        answer = pow(8,2);
        printf("The value of 8 raised to 2 is %f\n",answer);

        return 0;
}
```

```
>gcc -pedantic -Os -std=c99 13-1.c -o 13-1
>Exit code: 0
>./13-1
The value of 8 raised to 2 is 64.000000
>Exit code: 0
```

આકૃતિ 16.1 : ઉદાહરણ 16.1નું કોડ-લિસ્ટિંગ અને પરિણામ

ઉપયોગકર્તા દ્વારા વ્યાખ્યાયિત વિધેય (User defined function)

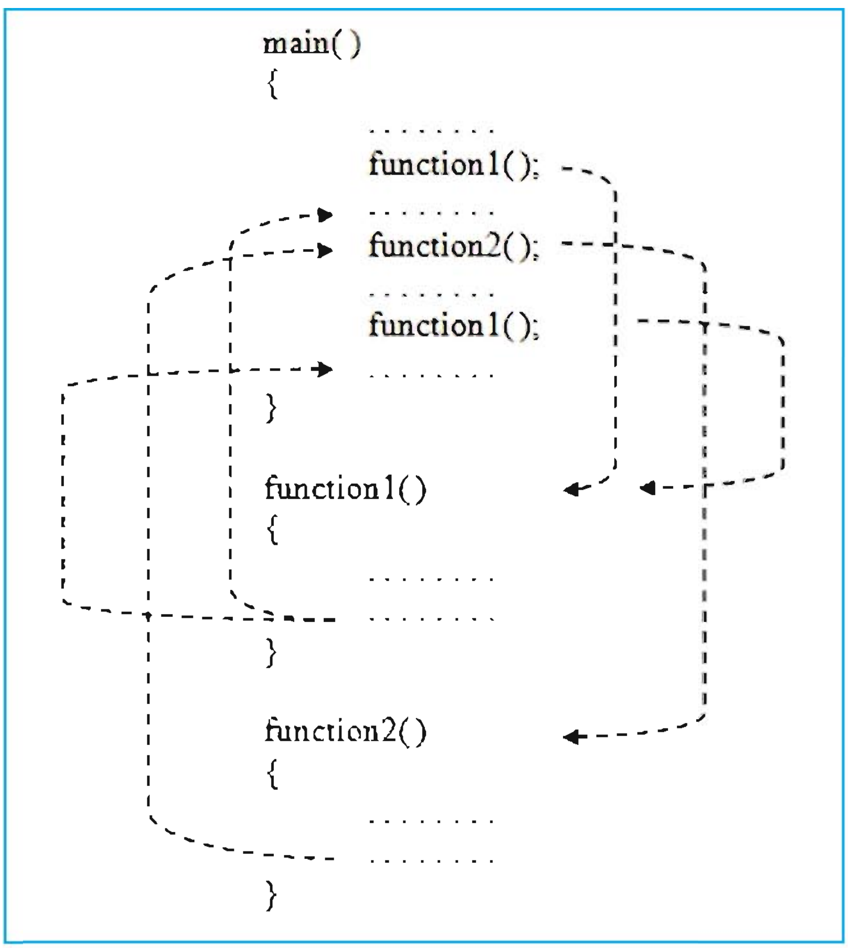
ઉપયોગકર્તા દ્વારા તૈયાર કરવામાં આવેલા વિધેયોને યુઝર ડિફાઈન્ડ (user defined) વિધેય તરીકે ઓળખવામાં આવે છે. main( ) એ C ભાષામાં ખાસ પ્રકારનું યુઝર ડિફાઈન્ડ વિધેય છે. પ્રોગ્રામનો અમલ main( ) વિધેયમાંથી શરૂ થાય છે. જો આપણે આપણો પ્રોગ્રામ માત્ર main( ) વિધેયનો ઉપયોગ કરીને લખીએ, તો તે કદાચ મોટો બની જાય. આવા પ્રોગ્રામની ચકાસણી કરવી અને સુધારા-વધારા કરવા કદાચ થોડા મુશ્કેલ બની જાય. જો પ્રોગ્રામને નાના નાના સબપ્રોગ્રામ(વિધેય)માં વહેંચી દેવામાં આવે તો તેને સમજવામાં, ચકાસવામાં અને જરૂરી સુધારા-વધારા કરવાનું ઘણું સરળ બની જાય.

વિધેયના ફાયદા

આપણા પ્રોગ્રામમાં વિધેયોનો ઉપયોગ કરવાના ફાયદા નીચે મુજબ છે :

- પ્રોગ્રામને વિધેયોમાં વિભાજિત કરવાથી પ્રોગ્રામ સમજવામાં અને સંભાળવામાં સરળ બને છે.
- એકવાર લખેલા વિધેયને પ્રોગ્રામ દ્વારા જરૂરિયાત અનુસાર અનેકવાર ઉપયોગમાં લઈ શકાય છે. આને કારણે પ્રોગ્રામ ટૂંકો બને છે અને એ રીતે મેમરીમાં બચત થાય છે.
- વિધેયને લીધે એક જ પ્રકારનું પ્રોગ્રામિંગ કાર્ય બેવડાતું અટકે છે.
- એક પ્રોગ્રામ માટે લખેલ વિધેય અન્ય કોઈ પણ પ્રોગ્રામ દ્વારા ઉપયોગમાં લઈ શકાય છે.
- જુદી જુદી કિંમતો આપીને એ જ વિધેયનો વારંવાર ઉપયોગ કરી શકાય છે.

તો ચાલો, હવે આપણા પ્રોગ્રામમાં જુદા જુદા વિધેયોનો કેવી રીતે ઉપયોગ કરવો તે સમજીએ. આકૃતિ 16.2 અનેક વિધેયોનો ઉપયોગ કરતાં પ્રોગ્રામના પ્રવાહનું નિયમન દર્શાવે છે.



આકૃતિ 16.2 : અનેક વિધેયવાળા પ્રોગ્રામના પ્રવાહનું નિયમન

આકૃતિ 16.2 એવું દર્શાવે છે કે, main( ) વિધેયમાં function1 બે વાર બોલાવવામાં આવે છે, અને function2 એક વાર બોલાવવામાં આવે છે. main( )માં વિધાનોના અમલ દરમ્યાન જ્યારે નિયંત્રણ function1( ) નામના વિધાન પર પહોંચે છે ત્યારે, નિયંત્રણ function1( ) માં પરિવર્તિત થાય છે. function1( )નાં બધાં વિધાનોનો અમલ કર્યા પછી નિયંત્રણ ફરી main( ) વિધેયની અંદરના function1( ) વિધાન પછીના વિધાન પર પહોંચી જશે. એ જ રીતે, main( ) વિધેયમાંથી ફરી function2( ) અને એ પછી function1( ) ને બોલાવવામાં આવશે.

વિધેયને વ્યાખ્યાયિત કરવો

જો આપણે આપણા પ્રોગ્રામમાં ઉપયોગકર્તા દ્વારા વ્યાખ્યાયિત વિધેયનો ઉપયોગ કરવા ઇચ્છતા હોઈએ તો આપણે આપણા કંપાઇલર (compiler)ને તેની જાણ કરવી પડે. આ માટે આપણે વિધેયને વ્યાખ્યાયિત કરતું વિધાન લખીને કંપાઇલરને જાણ કરી શકીએ. અગાઉથી વ્યાખ્યાયિત કર્યા વગર વિધેયનો ઉપયોગ કરી શકાય નહીં. C ભાષામાં નીચે મુજબની વાક્યરચનાનો ઉપયોગ કરીને કોઈ પણ વિધેયને વ્યાખ્યાયિત કરી શકાય છે.

```
return_data_type    function name (arguments)
{
        function statements;
}
```

અહીં return_data_type એ વિધેય દ્વારા પરત કરવામાં આવતી માહિતીનો પ્રકાર નક્કી કરે છે. જો વિધેય દ્વારા પરત આપવામાં આવતી કિંમત પૂર્ણાંક હોય તો return_data_type તરીકે int હોય. જો વિધેય કોઈ પણ કિંમત પરત ન કરે તેમ હોય તો return_data_type તરીકે void હોઈ શકે. functionname યુઝર ડિફાઇન્ડ વિધેયનું નામ નક્કી કરે છે. વિધેયને નામ આપવાના નિયમો, ચલનું નામ આપવાના નિયમો જેવા જ છે. વિધેયનાં નામ બને ત્યાં સુધી અર્થસભર આપવાં જોઈએ. ઉદાહરણ તરીકે સરવાળા કરવા માટે add ( ), સરેરાશ ગણવા માટે avg( ) વગેરે. arguments વિધેયને આપવામાં આવતી ડેટા પ્રકાર સહિતની ઇનપુટ કિંમતો દર્શાવે છે. એક કરતા વધુ વિધેયને વ્યાખ્યાયિત કરતી વખતે એક કરતા વધુ આર્ગ્યુમેન્ટ (સંલગ્ન મૂલ્ય) માટે બધાં આર્ગ્યુમેન્ટ અલ્પવિરામ દ્વારા અલગ પાડવામાં આવે છે. બે છગડિયા કૌંસની વચ્ચેનાં વિધાનોને 'ફંક્શન બૉડી' (function body) કહે છે. પ્રોગ્રામની જરૂરિયાતને આધારે ફંક્શન બૉડીમાં સ્થાનિક ચલની ઘોષણા અને return વિધાન હોઈ શકે. વિધેયમાં ચલની ઘોષણા અને return વિધાન આપણે આ પ્રકરણમાં હવે પછી ચર્ચીશું.

અહીં એ નોંધો કે, લાઇબ્રેરી વિધેયને વ્યાખ્યાયિત કરવાની કોઈ જરૂર હોતી નથી, કારણ કે તે અગાઉ વ્યાખ્યાયિત થયેલ જ હોય છે. કોઈ પ્રોગ્રામ દ્વારા ઉપયોગમાં લેતા પહેલાં, બધા યુઝર ડિફાઇન્ડ વિધેય, વિધેય પ્રતિકૃતિ (function prototype) તરીકે વ્યાખ્યાયિત થવા જરૂરી છે. વિધેય પ્રોટોટાઇપ એટલે, main ( ) વિધેયમાં ઉપયોગ પૂર્વે વિધેયને વ્યાખ્યાયિત કરવો.

વિધેયને બોલાવવા

અગાઉ જણાવ્યા પ્રમાણે, દરેક C પ્રોગ્રામ main ( ) વિધેય સાથે ચાલુ થાય છે. main ( ) વિધેયમાંથી આપણે બીજા યુઝર ડિફાઇન્ડ વિધેય અથવા લાઇબ્રેરી ફંક્શન બોલાવીએ છીએ. કોઈ પણ વિધેયને ઉપયોગ માટે બોલાવવા જરૂરિયાત મુજબના પેરામીટર સાથે તે વિધેયના નામનો ઉપયોગ કરો. તો ચાલો, ઉદાહરણ 16.2 જુઓ, જે દર્શાવે છે કે main( )માંથી કોઈ વિધેયને કેવી રીતે બોલવવા. આકૃતિ 16.3, ઉદાહરણ 16.2નું કોડ-લિસ્ટિંગ આપે છે, તેનું પરિણામ આકૃતિ 16.4માં દર્શાવાય છે.

```
/* Example 2: Program to illustrate use of function which returns no value.*/
#include<stdio.h>

void print_line( );          // Function Prototype Declaration

void main( )
{
      print_line( );          // Function call
      printf("...Wel Come to the World of C Function...\n");
      print_line( );          // Function call
}
// End of main() function

void print_line( )
{
      printf("= = = = = = = = = = = = = = = = = = = = = = = = = \n");
}
// End of print_line() function
```

આકૃતિ 16.3 : ઉદાહરણ 16.2નું કોડ-લિસ્ટિંગ

```
gcc 13-2.c
>gcc 13-2.c
>Exit code: 0
./a.out
>./a.out
= = = = = = = = = = = = = = = = = = = = = = = = =
...Wel Come to the World of C Function...
= = = = = = = = = = = = = = = = = = = = = = = = =
>Exit code: 45
```

આકૃતિ 16.4 : ઉદાહરણ 16.2નું પરિણામ

સમજૂતી

પ્રોગ્રામમાં print_line( ) નામના એક યુઝર ડિફાઇન્ડ વિધેયનો ઉપયોગ કરવામાં આવ્યો છે. main( ) વિધેયમાં લખેલું print_line( ) વિધાન આ વિધેયને બોલાવે છે. પ્રોગ્રામનું નિયંત્રણ main( ) વિધેય પાસેથી print_line( ) વિધેય પાસે જાય છે, જ્યાં તેના દ્વારા વિવિધ શૈલીની લીટીઓ છાપીને પ્રોગ્રામનું નિયંત્રણ main( ) વિધેયમાં પાછું આવી જાય છે. main( ) વિધેયમાં એ પછીનું printf વિધાન એક સંદેશ છાપે છે. main( ) વિધેયનું એના પછીનું વિધાન ફરીથી print_line( ) વિધેયને બોલાવે છે. આમ આ ઉદાહરણમાં print_line( ) વિધેયને બે વાર બોલાવવામાં આવે છે. આ વિધેય main( ) વિધેયને કોઈ કિંમત પરત આપતું ન હોવાથી તેનો પરત પ્રકાર (return type) void છે.

Return વિધાન

યુઝર ડિફાઇન્ડ વિધેય તેને બોલાવનાર વિધેયને કદાચ કોઈ કિંમત પરત કરે કે ન પણ કરે. બોલાવનાર વિધેયને કોઈ પણ કિંમત પરત કરવા માટે ફંકશન બૉડીની અંદર આપણે return વિધાનનો ઉપયોગ કરી શકીએ. return વિધાનનું સ્વરૂપ નીચે મુજબ છે :

return;   અથવા   return (expression);

return વિધાનનું પ્રથમ સ્વરૂપ કોઈ પણ કિંમત પરત આપશે નહીં. તે માત્ર તેને બોલાવનાર પ્રોગ્રામને પ્રોગ્રામનું નિયંત્રણ પાછું સોંપે છે. જ્યારે કોઈ વિધેયની 'રિટર્ન ટાઇપ' (return type) void હોય તો તે વિધેયની ફંક્શન બૉડીમાં return વિધાન લખવાની કોઈ જરૂર રહેતી નથી.

કોઈ વિધેય તેને બોલાવનાર વિધેયને કેવી રીતે કિંમત પરત કરે છે તે સમજવા આકૃતિ 16.5 પર એક નજર નાંખો, જે ઉદાહરણ 16.3નું કોડ-લિસ્ટિંગ દર્શાવે છે.

સમજૂતી

આકૃતિ 16.5માં આપણે add( ) નામના યુઝર ડિફાઈન્ડ વિધેયનો ઉપયોગ કર્યો છે, main( ) વિધેયમાં આપણે total નામના એક ચલને વ્યાખ્યાયિત કર્યો છે. આપણે જ્યારે main( )માંથી add( ) વિધેયને બોલાવીએ છીએ ત્યારે પ્રોગ્રામનું નિયંત્રણ add( ) વિધેય પાસે જાય છે. add વિધેયમાં num1 અને num2 નામના બે સ્થાનિક ચલ ઘોષિત કરવામાં આવેલ છે. add વિધેયનું એ પછીનું વિધાન સંદેશ દર્શાવે છે અને અનુક્રમે printf અને scanf વિધાનનો ઉપયોગ કરીને બે સંખ્યા વાંચે છે. છેલ્લું વિધાન બે સંખ્યાઓનો સરવાળો પરત આપે છે, જે main( ) વિધેયના total નામના ચલમાં સાચવવામાં આવે છે. એ પછી totalને સ્ક્રીન પર દર્શાવવામાં આવે છે.

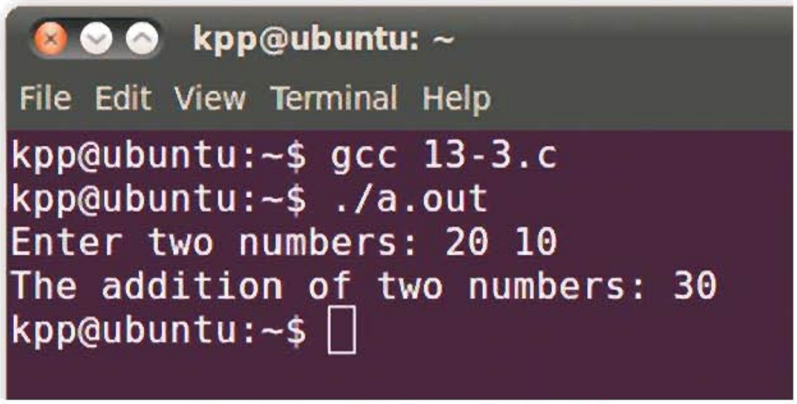
13-3.c - SciTE

File Edit Search View Tools Options Language Buffers Help

13-3.c

```
/* Example 3: Program to illustrate use of function which returns a value.*/

#include<stdio.h>
int add( );                    // Function Prototype Declaration

void main( )
{
        int total;
        total = add( );        // Function call
        printf("The addition of two numbers: %d \n", total);
}
// End of main() function...

int add( )
{
        int num1, num2;
        printf("Enter two numbers: ");
        scanf("%d %d", &num1, &num2);
        return(num1 + num2);
}
// End of add() function...
```

આકૃતિ 16.5 : ઉદાહરણ 16.3નું કોડ-લિસ્ટિંગ



આકૃતિ 16.6 : ઉદાહરણ 16.3નું પરિણામ

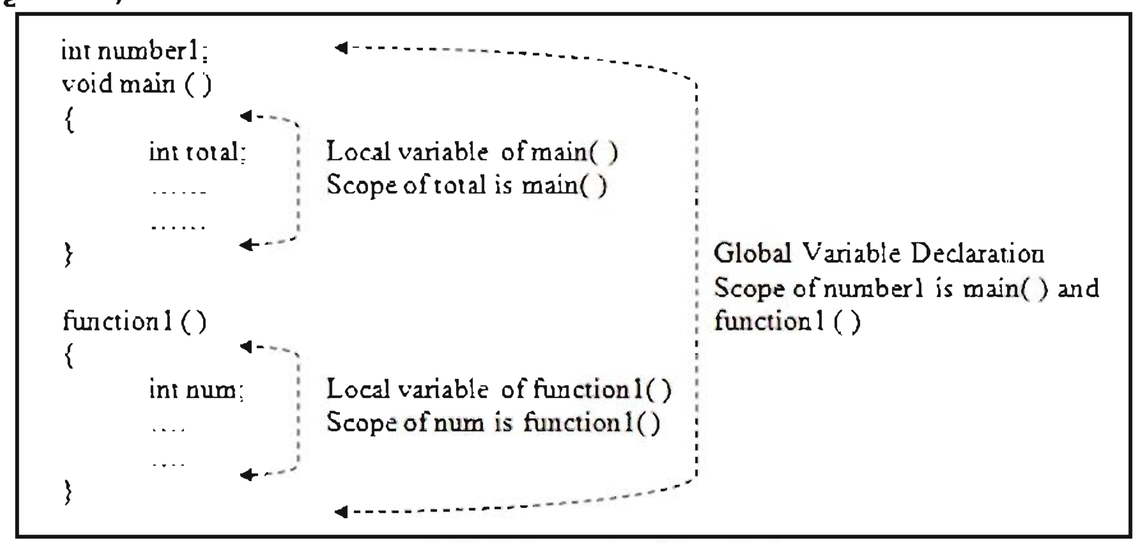
ચલનો અવકાશ (Scope of Variables)

ચલના અવકાશ(scope)નો અર્થ છે પ્રોગ્રામના ક્યા ભાગમાં ચલનો ઉપયોગ કરી શકાય છે. C ભાષામાં ચલના અવકાશના બે પ્રકાર છે:

(i) સાર્વત્રિક ચલ (Global Variable)

(ii) સ્થાનિક ચલ (Local Variable)

આકૃતિ 16.7, C ભાષાના પ્રોગ્રામમાં ચલના અવકાશને સમજાવે છે.



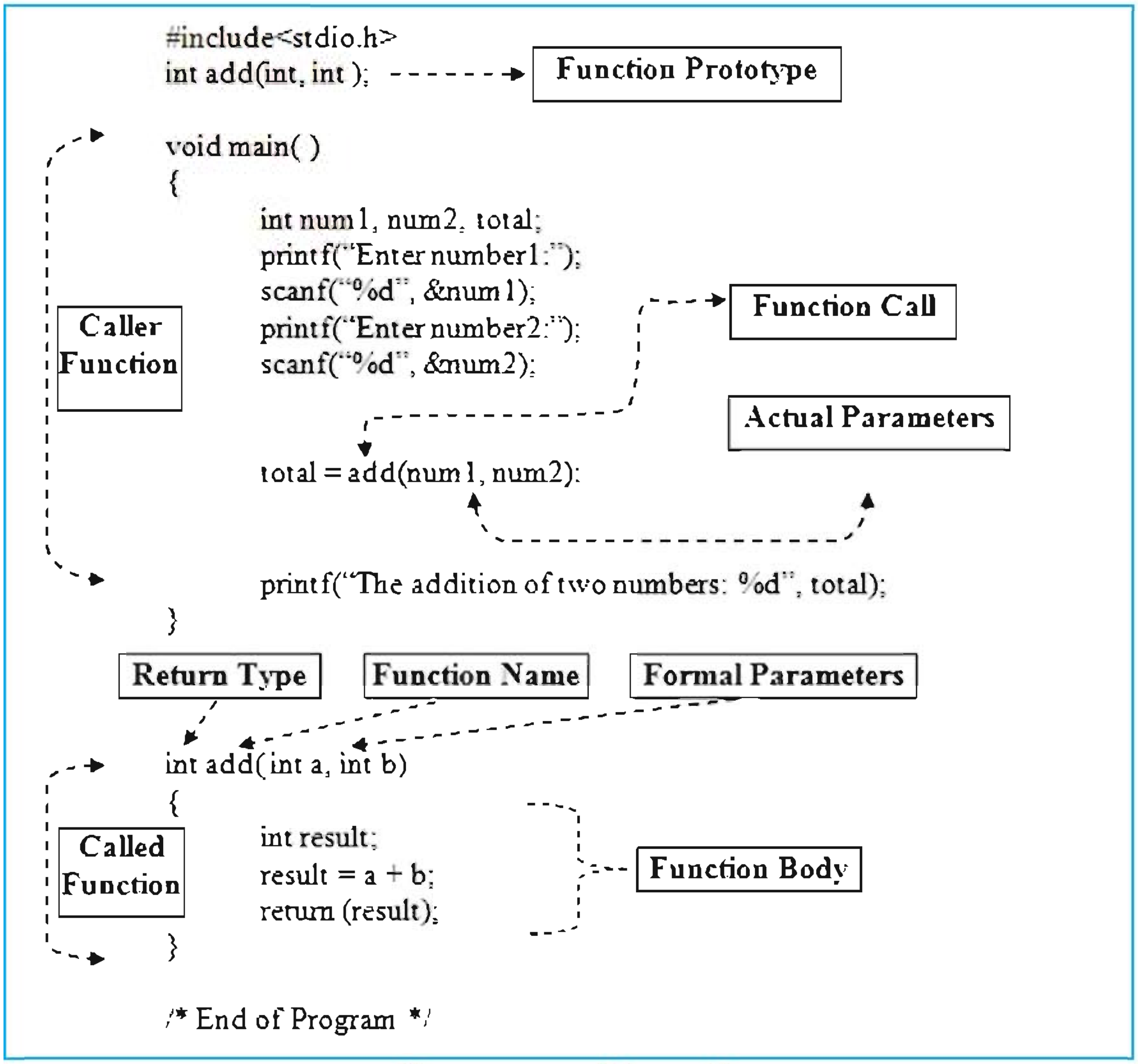
આકૃતિ 16.7 : ચલનો અવકાશ

અહીં number1 નામનો ચલ એ સાર્વત્રિક ચલ ગણાશે, કારણ કે પ્રોગ્રામમાં તેને main( ) વિધેયની ઉપર વ્યાખ્યાયિત કરવામાં આવ્યો છે. આ સાર્વત્રિક ચલ number1 ને main( ) ફંકશનમાં અને function1( ) માં એમ બંને જગ્યાએ ઉપયોગમાં લઈ શકાશે. total નામનો સ્થાનિક ચલ માત્ર main( ) વિધેયમાં જ ઉપયોગમાં લઈ શકાશે, જ્યારે function1( ) વિધેયમાં તેને ઉપયોગમાં લઈ શકાશે નહીં. એ જ રીતે, સ્થાનિક ચલ num માત્ર function1( ) નામના વિધેય દ્વારા ઉપયોગમાં લઈ શકાશે પણ main( ) વિધેયમાં ઉપયોગમાં લઈ શકાશે નહીં.

C ભાષાના વિધેયોની વિસ્તૃત લાક્ષણિકતાઓની ચર્ચા કરીએ તે પહેલાં ચાલો આપણે આકૃતિ 16.8 જોઈએ, જે વિધેયના વિવિધ વિભાગ દર્શાવે છે.

આકૃતિ 16.8માં દર્શાવ્યા પ્રમાણે આપણે int add(int, int); એ પ્રમાણે વિધેયની પ્રતિકૃતિ વ્યાખ્યાયિત કરેલ છે. આ દર્શાવે છે કે અહીં પૂર્ણાંક (integer) પ્રકારના બે આર્ગ્યુમેન્ટ છે. વિધેયની પ્રતિકૃતિ વિશે વિસ્તૃત ચર્ચા પછી આ પ્રકરણમાં કરવામાં આવી છે. void main( ) { .... } એ બોલાવનાર વિધેય (caller function) છે જે બે ખરા પેરામીટર num1 અને num2 સાથે add( ) વિધેયને બોલાવે છે. અહીં int add( ) નામનું યુઝર ડિફાઈન્ડ વિધેય એવું દર્શાવે છે કે, તે તેને બોલાવનાર વિધેય void main( ) ને પૂર્ણાંક પરત કરશે. add વિધેયમાં બે

ઔપચારિક પૂર્ણાંક પેરામીટર "a" અને "b" વ્યાખ્યાયિત કરવામાં આવ્યા છે. આ ઔપચારિક ચલ, main() માંથી વિધેયને બોલાવતી વખતે મોકલવામાં આવતી કિંમતોનો સ્વીકાર કરે છે. result નામનો પૂર્ણાંક ચલ, બોલાવનાર વિધેયને બે સંખ્યાનો સરવાળો પરત આપે છે, જે main( )ના total નામના ચલમાં સાચવવામાં આવે છે.



આકૃતિ 16.8 : C વિધેયના વિભાગ

વિધેયની પ્રતિકૃતિ

કોઈ પણ પ્રોગ્રામમાં જ્યારે કોઈ વિધેયને main( ) પછી વ્યાખ્યાયિત કરવામાં આવે ત્યારે વિધેયની પ્રતિકૃતિ (function prototype)ની આવશ્યકતા ઊભી થાય છે. આપણે જ્યારે આપણા પ્રોગ્રામમાં કોઈ પણ વિધેયને બોલાવીએ છીએ, ત્યારે કંપાઈલર આવા સંબંધિત વિધેયની વ્યાખ્યાને શોધે છે. આવું એ ચકાસવા માટે થાય છે કે વિધેયને બોલાવવાની ક્રિયા ખરી છે કે નહીં. જો બોલાવવામાં આવેલ વિધેય આર્ગ્યુમેન્ટની સંખ્યા અને તેના પ્રકારની બાબતે વિધેયના મથાળાની લીટી (header of the function) સાથે બંધબેસતી હશે, તો તે વિધેય માન્ય ગણાશે અન્યથા કંપાઈલર દ્વારા ભૂલસંદેશ દર્શાવાશે. ઉદાહરણ તરીકે,

***int add(int, int);* જે *int add(int a, int b)* વિધેયની પ્રતિકૃતિ (prototype) છે.**

અહીં પ્રતિકૃતિ એવું દર્શાવે છે કે, add( ) એ બે પૂર્ણાંક આર્ગ્યુમેન્ટ ધરાવતું એક વિધેય છે, અને તે પૂર્ણાંક કિંમત પરત આપે છે. વિધેયની પ્રતિકૃતિ દરેક ઉપયોગકર્તા નિર્મિત વિધેય માટે પ્રોગ્રામની શરૂઆતમાં જ લખવામાં આવે છે.

વિધેયનું સંલગ્ન મૂલ્ય અથવા પેરામીટર

વિધેયના સંલગ્ન મૂલ્ય અથવા પેરામીટરનો ઉપયોગ કરવા પાછળનું મૂળ પ્રયોજન બોલાવવામાં આવેલ અને બોલાવનાર વિધેય વચ્ચે ડેટાનું આદાન-પ્રદાન કરવાનું છે. જ્યારે બોલાવનાર વિધેય (calling function) બોલાવવામાં આવેલ વિધેય (called function) ને ડેટા મોકલે છે ત્યારે તેને વિધેયનું સંલગ્ન મૂલ્ય (arguments) અથવા પેરામીટરની તબદીલી (parameter passing) કહે છે.

વિધેયને વ્યાખ્યાયિત કરતી વખતે ઉપયોગમાં લેવાતા સંલગ્ન મૂલ્ય(arguments)ને ''ઔપચારિક સંલગ્ન મૂલ્ય'' (*formal* arguments) તરીકે ઓળખવામાં આવે છે. આવા વિધેયને બોલાવવાની જરૂર પડે ત્યારે બોલાવતી વખતે તેની સાથે એટલી જ સંખ્યામાં સંલગ્ન મૂલ્યોને તબદીલ કરવાની જરૂર પડે છે. આ તબદીલ કરવામાં આવતા સંલગ્ન મૂલ્યોને વાસ્તવિક સંલગ્ન મૂલ્યો (*actual* arguments) તરીકે ઓળખવામાં આવે છે. ''ઔપચારિક સંલગ્ન મૂલ્ય'' અને ''વાસ્તવિક સંલગ્ન મૂલ્ય''ની સંખ્યા, પ્રકાર અને ક્રમ બાબતે એકબીજા સાથે મળતા હોવા જોઈએ. ''વાસ્તવિક સંલગ્ન મૂલ્યો'' જો ''ઔપચારિક સંલગ્ન મૂલ્યો'' કરતાં વધુ હશે તો વધારાના વાસ્તવિક સંલગ્ન મૂલ્યને છોડી દેવામાં આવશે. તેનાથી ઊલટું ''વાસ્તવિક સંલગ્ન મૂલ્યો'' જો ''ઔપચારિક મૂલ્યો'' કરતાં ઓછાં હશે તો બંધબેસતાં નહીં હોય તેવા ''ઔપચારિક-સંલગ્ન મૂલ્યોને ખોટી કિંમત વડે પ્રારંભિક કિંમત અપાશે. એ જ રીતે, કોઈ પણ સંલગ્ન મૂલ્યોના ડેટા પ્રકારમાં કોઈ પણ જાતની વિસંગતતા ખોટી પ્રારંભિક કિંમત આપવા માટે કારણભૂત નીવડશે.

વિધેયના પ્રકાર

વિધેયમાં સંલગ્ન મૂલ્યોની ઉપસ્થિતિ અને પરત મળતી કિંમતને આધારે વિધેયોના નીચે પ્રમાણે ત્રણ પ્રકાર પાડી શકાય :

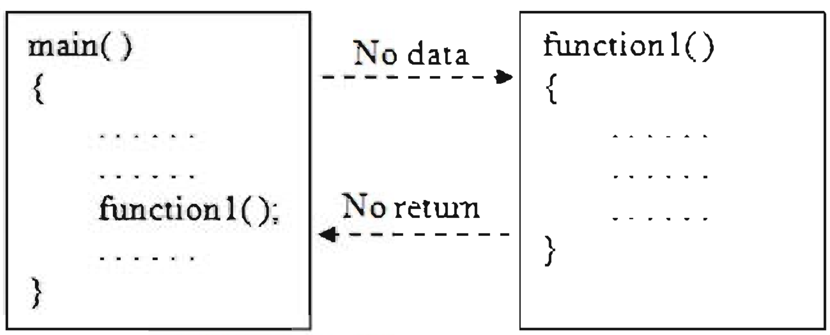
(i) કોઈ પણ સંલગ્ન મૂલ્ય કે પરત કિંમત ન આપતા વિધેય

(ii) સંલગ્ન મૂલ્ય ધરાવતા પરંતુ પરત કિંમત વિનાના વિધેય

(iii) સંલગ્ન મૂલ્ય અને પરત કિંમત ધરાવતા વિધેય

કોઈ પણ સંલગ્ન મૂલ્ય કે કોઈ પરત કિંમત ન આપતા વિધેય

જ્યારે કોઈ વિધેય કોઈ સંલગ્ન મૂલ્ય ન ધરાવતા હોય અને કોઈ પણ કિંમત પરત ન આપતા હોય તો તે આ પ્રકારના વિધેયમાં આવે છે. બોલાવવામાં આવેલ વિધેય(called function)ને બોલાવનાર વિધેય (calling function) તરફથી કોઈ પણ ડેટા કિંમતો પ્રાપ્ત થતી નથી. એ જ રીતે, કોઈ પરત કિંમત ન હોવાને લીધે બોલાવનાર વિધેય (calling function) ને પણ બોલાવવામાં આવેલ વિધેય (called function) તરફથી કોઈ પણ ડેટા પરત મળતો નથી. ટૂંકમાં, બોલાવનાર વિધેય અને બોલાવવામાં આવેલ વિધેય વચ્ચે કોઈ પણ જાતના ડેટાની આપ-લે થતી નથી. આ બાબત આકૃતિ 16.9માં દર્શાવેલ છે. આકૃતિમાં બે પ્રતીકાત્મક વિધેયની વચ્ચે દેખાતી ટપકાંવાળી રેખા કોઈ ડેટાની તબદીલી દર્શાવતા નથી પરંતુ માત્ર પ્રોગ્રામના નિયંત્રણની તબદીલી દર્શાવે છે.

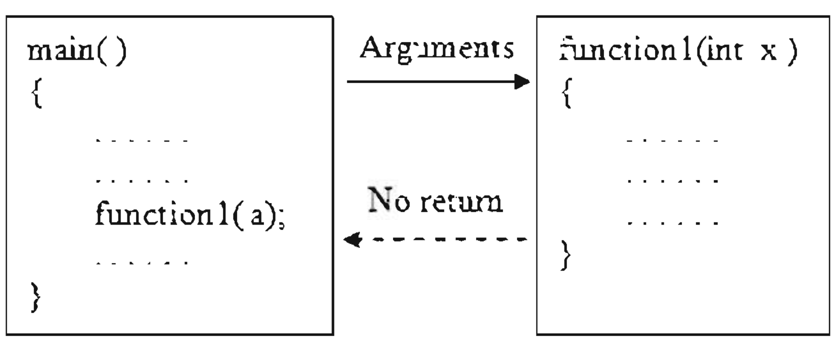


આકૃતિ 16.9 : કોઈ પણ ડેટાસંચાર વિનાનું વિધેય

આ પ્રકરણમાં અગાઉ સમજાવેલ ઉદાહરણ 16.2 એ આ પ્રકારના વિધેયનું ઉદાહરણ છે.

સંલગ્ન મૂલ્ય સહિતનું પરંતુ પરતમૂલ્ય વિનાનું વિધેય

બોલાવનાર વિધેય અને બોલાવવામાં આવેલ વિધેય વચ્ચે જરૂરી સંલગ્ન મૂલ્ય સાથે પરંતુ કોઈ પણ પરતમૂલ્ય વગરનું પ્રત્યાયન (communication) આકૃતિ 16.10માં દર્શાવેલ છે.



આકૃતિ 16.10 : એકતરફી ડેટાસંચાર ધરાવતું વિધેય

હવે, આપણે ઉદાહરણ 16.4 સમજીએ, જે ઉપયોગકર્તા પાસેથી એક સંખ્યા મેળવશે અને વિધેયનો ઉપયોગ કરી તેનું ઘનફળ (cube) શોધશે. આ માટે આપણે main( ) (બોલાવનાર) વિધેયમાંથી સંલગ્ન મૂલ્ય તરીકે એક સંખ્યાને cube( ) (બોલાવવામાં આવેલ) વિધેયને મોકલીશું. બોલાવવામાં આવેલ (called) વિધેય ઘનફળ શોધશે અને તેને સ્ક્રીન પર દર્શાવશે. આકૃતિ 16.11, ઉદાહરણ 16.4નું કોડ-લિસ્ટિંગ આપે છે, જ્યારે આકૃતિ 16.12 પરિણામ દર્શાવે છે.

```
13-4.c - SciTE
File Edit Search View Tools Options Language Buffers Help
13-4.c

/* Example 4: Program to illustrate use of function with arguments and no return value.*/
#include<stdio.h>

void cube ( int );                // Function Prototype Declaration

void main( )
{
      int number;
      printf("Enter your number: ", number);
      scanf("%d", &number);
      cube( number );             // Function call
}
//End of main () function

void cube(int num1)          // Function to find and display cube of a given number.
{
      printf("The cube of entered number is : %d \n", num1*num1*num1);
}

// End of cube() function
```

આકૃતિ 16.11 : ઉદાહરણ 16.4નું કોડ-લિસ્ટિંગ

```
kpp@ubuntu: ~
File Edit View Terminal Help
kpp@ubuntu:~$ gcc 13-4.c
kpp@ubuntu:~$ ./a.out
Enter your number: 10
The cube of entered number is : 1000
kpp@ubuntu:~$
```

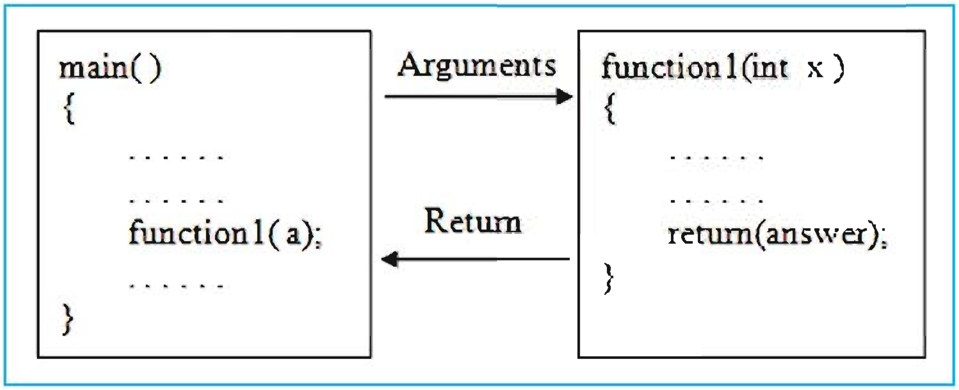
આકૃતિ 16.12 : ઉદાહરણ 16.4નું પરિણામ

સમજૂતી

અહીં main( ) વિધેય ઉપયોગકર્તા પાસેથી એક સંખ્યા સ્વીકારે છે, જે number( ) નામના ચલમાં સાચવવામાં આવે છે. આ number નામનો ચલ cube( ) નામના વિધેયને ખરેખરા સંલગ્ન મૂલ્ય તરીકે પહોંચાડવામાં આવે છે. cube( ) વિધેયમાં num1 નામનું ઔપચારિક સંલગ્ન મૂલ્ય number નામના ચલની કિંમત મેળવે છે. એ પછી, cube( ) વિધેય ગણતરી કરે છે અને printf વિધાનનો ઉપયોગ કરી આપેલ સંખ્યાનું ઘનફળ દર્શાવે છે.

સંલગ્નમૂલ્ય અને પરતમૂલ્ય સહિતના વિધેય

તો ચાલો બોલાવનાર વિધેય અને બોલાવવામાં આવેલ વિધેય વચ્ચે દ્વિમાર્ગીય સંચાર સમજીએ. સંલગ્ન મૂલ્ય અને પરત મૂલ્ય ધરાવતું વિધેય આકૃતિ 16.13માં દર્શાવેલ છે.



આકૃતિ 16.13 : દ્વિતરફી ડેટાસંચાર સાથેનું વિધેય

આપણે ઉદાહરણ 16.5 સમજવા પ્રયત્ન કરીએ, જે ઉપયોગકર્તા પાસેથી એક સંખ્યા મેળવશે અને sqr( ) વિધેયનો ઉપયોગ કરી તેનો વર્ગ (square) શોધશે. આપણે main( ) નામના બોલાવનાર વિધેયમાંથી બોલાવવામાં આવનાર વિધેય sqr( )ને સંલગ્ન મૂલ્ય તરીકે એક સંખ્યા મોકલીશું. બોલાવેલ વિધેય સંખ્યાનો વર્ગ શોધીને તેને બોલાવનાર વિધેયને પરત મોકલે છે. આકૃતિ 16.14 ઉદાહરણ 16.5નું કોડ-લિસ્ટિંગ આપે છે, જ્યારે આકૃતિ 16.15 તેનું પરિણામ દર્શાવે છે.

```
13-5.c - SciTE
File Edit Search View Tools Options Language Buffers Help
13-5.c

/* Example 5: Program to illustrate use of function with arguments and return value.*/
#include<stdio.h>

int sqr ( int );          // Function Prototype Declaration

void main( )
{
        int number, result;
        printf("Enter your number: ", number);
        scanf("%d", &number);
        result = sqr( number );          // Function call with argument
        printf("The square of %d is %d. \n", number, result);
}
//End of main() function

int sqr(int num1)  // Function to find square of a number and return the answer
{
        int answer;
        answer = num1 * num1;
        return (answer);
}
//End of sqr() function
```

આકૃતિ 16.14 : ઉદાહરણ 16.5નું કોડ-લિસ્ટિંગ

```
kpp@ubuntu: ~
File Edit View Terminal Help
kpp@ubuntu:~$ gcc 13-5.c
kpp@ubuntu:~$ ./a.out
Enter your number: 10
The square of 10 is 100.
kpp@ubuntu:~$
```

આકૃતિ 16.15 : ઉદાહરણ 16.5નું પરિણામ

સમજૂતી

main( ) વિધેય ઉપયોગકર્તા પાસેથી એક સંખ્યા મેળવે છે, જેને number નામના ચલમાં સાચવવામાં આવે છે. એ પછી, sqr( ) નામના વિધેયને number નામનો ચલ સંલગ્ન મૂલ્ય તરીકે મોકલવામાં આવે છે. sqr( ) વિધેયમાં ઔપચારિક સંલગ્ન મૂલ્ય num1, number નામના ચલની કિંમત મેળવે છે. એ પછી sqr( ) વિધેય વર્ગફળ ગણે છે અને તેને answer નામના ચલમાં સાચવે છે. sqr( ) વિધેયનું છેલ્લું વિધાન answer નામના ચલમાં સાચવેલી કિંમત બોલાવનાર વિધેયને પરત આપે છે. બોલાવનાર વિધેય પરત મળતી કિંમતને સ્વીકારે છે અને તેને result નામના ચલમાં સાચવે છે. main( ) વિધેયનું છેલ્લું વિધાન સ્ક્રીન પર આપેલ સંખ્યાનું વર્ગફળ દર્શાવે છે.

સારાંશ

આ પ્રકરણમાં આપણે C ભાષાના વિધેય સાથે સંબંધિત મૂળભૂત ખ્યાલોની ચર્ચા કરી. આપણે એ જોયું કે, પ્રોગ્રામમાં વિધેયનો ઉપયોગ તેને મૉડ્યુલર (modular) બનાવે છે. લાઇબ્રેરી વિધેય અને ઉપયોગકર્તા નિર્મિત વિધેયોને આપણા પ્રોગ્રામમાં કેવી રીતે ઉપયોગમાં લેવા તે પણ આપણે શીખ્યા. એ જ રીતે આપણે એ પણ શીખ્યા કે, આપણા પ્રોગ્રામમાં સ્થાનિક ચલ અને સાર્વત્રિક ચલોનો ઉપયોગ કેવી રીતે કરવો ? આપણે એ જોયું કે કોઈ એક વિધેય કેવી રીતે, બોલાવનાર વિધેયને એક કિંમત પરત આપી શકે. વિધેયમાં સંલગ્ન મૂલ્ય કે પેરામીટરના ઉપયોગ પાછળના મૂળભૂત ઉદ્દેશની ચર્ચા કરી.

સ્વાધ્યાય

1. વિધેય એટલે શું? પ્રોગ્રામમાં વિધેયનો ઉપયોગ કરવાના ફાયદા જણાવો.
2. પ્રોગ્રામમાં લાઇબ્રેરી વિધેયનો ઉપયોગ કરવાના ફાયદા જણાવો.
3. ચલનો અવકાશ (scope of a variable) એટલે શું?
4. કોઈ પણ વિધેયને ડેટા / માહિતી કેવી રીતે પહોંચાડી શકાય?
5. વિધેયની પ્રતિકૃતિ (function prototype) એટલે શું?
6. નીચેના વચ્ચે તફાવત સ્પષ્ટ કરો :
   (a) સ્થાનિક અને સાર્વત્રિક ચલ
   (b) ઔપચારિક અને વાસ્તવિક પેરામીટર
   (c) લાઇબ્રેરી વિધેય અને ઉપયોગકર્તા નિર્મિત વિધેય (User defined function)
7. નીચેના પ્રોગ્રામમાં જો કોઈ ભૂલ હોય તો તે શોધી કાઢી, તેને ફરીથી લખી પ્રોગ્રામનું પરિણામ શું આવશે તે જણાવો :
   (a)

```
#include<stdio.h>
void draw_line( );
void main( )
{
   draw_line;
               printf("...Wel Come to the C Function...\n");
   draw_line;
}
void print_line( )
{
   printf("= = = = = = = = = = = = = = = = = = = = = = \n");
}
```

(b)

```
#include<stdio.h>
int sqr ( int );
void main( )
{
    int number = 5, result;
    result = sqr( );
    printf("The square of 5 is %d.", result);
}
int sqr(int num1)
{
    int answer;
    answer = num1 * num1;
    print (answer);
}
```

8. ખાલી જગ્યા પૂરો :

(a) વિધેયમાંથી __________ વિધાન માહિતીને પરત આપે છે.

(b) જો કોઈ વિધેય કોઈ પણ કિંમત પરત ન કરતું હોય તો તે વિધેયની "રિટર્ન ડેટા ટાઇપ" __________ હોય.

(c) sqrt( ) નામનું લાઇબ્રેરી વિધેય __________ હેડર ફાઈલમાં વ્યાખ્યાયિત કરવામાં આવેલું છે.

(d) main( ) વિધેયની ઉપર ઘોષિત કરેલ ચલને __________ ચલ તરીકે ઓળખવામાં આવે છે.

9. નીચેનાં વિધાનો ખરાં છે કે ખોટાં તે જણાવો:

(a) C ભાષાનો પ્રોગ્રામ માત્ર એક કિંમત મોકલી (pass કરી) શકે છે.

(b) વિધેય એક સાથે અનેક સંલગ્ન મૂલ્યો (arguments) મોકલી શકે નહીં.

(c) main( ) વિધેયમાંથી એકનું એક ઉપયોગકર્તા નિર્મિત વિધેય અનેકવાર બોલાવી શકાય નહીં.

(d) C ભાષાના પ્રોગ્રામમાં main( ) પોતે પણ એક પ્રકારનું વિધેય જ છે.

(e) લાઇબ્રેરી વિધેય માટે પ્રતિકૃતિ (function prototype)ની ઘોષણા પ્રોગ્રામની શરૂઆતમાં કરવી જરૂરી નથી.

10. નીચેના પ્રશ્નો માટે આપેલા વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરી જવાબ લખો :

(1) C ભાષામાં main() વિધેય એ ....

(a) ઉપયોગકર્તાનિર્મિત વિધેય છે.

(b) લાઇબ્રેરી વિધેય છે.

(c) ચાવીરૂપ શબ્દ છે.

(d) અનામત વિધેય છે.

(2) સામાન્ય રીતે main() વિધેય શું પરત કરે છે?

(a) Char કિંમત (b) Float કિંમત

(c) Int કિંમત (d) Double કિંમત

(3) જ્યારે વિધેય કોઈ પણ કિંમત પરત ન કરતું હોય ત્યારે return type તરીકે ક્યા ચાવીરૂપ શબ્દનો ઉપયોગ કરાય છે?

(a) int (b) main (c) void (d) auto

(4) નીચેનામાંથી કયું આંતરપ્રસ્થાપિત લાઇબ્રેરી વિધેય નથી?

(a) pow() (b) printf() (c) sum() (d) sqrt()

(5) નીચેના પ્રોગ્રામના અમલ પછી number2 નામના ચલની કિંમત શું મળશે?

```
int main(){
    int number1 = 2, number2;
    number2 = sqr(number1);
    printf("number2 = %d",number2);
    return 0;
}
int sqr(int n){
    return(n*n);
}//End of program
```

(a) number2 = 4 (b) number2 = 6

(c) number2 = 8 (d) number2 = 18

પ્રાયોગિક સ્વાધ્યાય

નીચેનાં કાર્યો પાર પાડવા માટે C ભાષામાં પ્રોગ્રામ તૈયાર કરો :

1. main( ) વિધેયમાં N નામનો પૂર્ણાંક મેળવો. N નામના ચલની કિંમતને સંલગ્ન મૂલ્ય (argument) તરીકે વિધેયને મોકલો. વિધેયનો ઉપયોગ કરી આપેલ N સંખ્યા માટે નીચે પ્રમાણેની હારમાળા(series)નો સરવાળો દર્શાવો.

   1 + 2 + 3 + 4 +..... + N
2. એક એવું ઉપયોગકર્તાનિર્મિત વિધેય તૈયાર કરો જે કોઈ સ્વાગત સંદેશ (welcome message) દસ વાર દર્શાવે.
3. એક એવું વિધેય તૈયાર કરો જે xની n ઘાતની ગણતરી કરે.
4. એક એવું ઉપયોગકર્તાનિર્મિત વિધેય તૈયાર કરો જે આપેલી બે સંખ્યા પૈકી મોટી સંખ્યા દર્શાવે.
5. main( ) વિધેયમાંથી એક સંખ્યા વાંચો. તેને ઉપયોગકર્તાનિર્મિત વિધેય તરફ સંલગ્ન મૂલ્ય તરીકે તબદીલ કરો અને દર્શાવો કે આપેલ સંખ્યા ધન સંખ્યા છે કે ઋણ સંખ્યા છે?
6. તમારા પ્રોગ્રામમાં એક સંખ્યા વાંચો, તેના ઉપર લાઇબ્રેરી વિધેય abs( ), sin( ), cos( ) અને log( )નો અમલ કરો અને તેનું પરિણામ દર્શાવો.
7. ઉપયોગકર્તા પાસેથી કેરેક્ટર ઍરે (string) મેળવો અને મળેલ અક્ષરમાળા(string)ના અક્ષરોને ઊંધા ક્રમમાં દર્શાવો.

પરિશિષ્ટ I

દશાંકી સંખ્યાની તેને સંલગ્ન દ્વિઅંકી, અષ્ટાંકી અને સોળઅંકી સંખ્યાઓ સાથેની યાદી.

દશાંકી સંખ્યા (0,1,2,…,9)	દ્વિઅંકી સંખ્યા (0,1)	અષ્ટાંકી સંખ્યા (0,1,2,…,7)	સોળઅંકી સંખ્યા (0,1,2,…,9,A,B,C,D,E,F)
1	00001	1	1
2	00010	2	2
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
9	01001	11	9
10	01010	12	A
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

પરિશિષ્ટ II

અક્ષરોની આસ્કી (ASCII) કિંમતો

ASCII		ASCII		ASCII		ASCII	
કિંમત	અક્ષર	કિંમત	અક્ષર	કિંમત	અક્ષર	કિંમત	અક્ષર
000	NUL	032	blank	064	@	096	
001	SOH	033	!	065	A	097	a
002	STX	034	"	066	B	098	b
003	ETX	035	#	067	C	099	c
004	EOT	036	$	068	D	100	d
005	ENQ	037	%	069	E	101	e
006	ACK	038	&	070	F	102	f
007	BEL	039	'	071	G	103	g
008	BS	040	(	072	H	104	h
009	HT	041	)	073	I	105	i
010	LF	042	*	074	J	106	j
011	VT	043	+	075	K	107	k
012	FF	044	,	076	L	108	l
013	CR	045	-	077	M	109	m
014	SO	046	.	078	N	110	n
015	SI	047	/	079	O	111	o
016	DLE	048	0	080	P	112	p
017	DC1	049	1	081	Q	113	q
018	DC2	050	2	082	R	114	r
019	DC3	051	3	083	S	115	s
020	DC4	052	4	084	T	116	t
021	NAK	053	5	085	U	117	u
022	SYN	054	6	086	V	118	v
023	ETB	055	7	087	W	119	w
024	CAN	056	8	088	X	120	x
025	EM	057	9	089	Y	121	y
026	SUB	058	:	090	Z	122	z
027	ESC	059	;	091	[	123	{
028	FS	060	<	092	\	124	|
029	GF	061	=	093	]	125	}
030	RS	062	>	094	?	126	~
031	US	063	?	095	-	127	DEL

33 થી 126 અક્ષરો મુદ્રણ થઈ શકે તે પ્રકારના (printable) છે. અન્ય અક્ષરો નિયંત્રણ અક્ષરો છે, જે મુદ્રિત કરી શકતા નથી.

પરિશિષ્ટ III

સી પ્રક્રિયકોનો સારાંશ

પ્રક્રિયક	પ્રક્રિયાનો ઉપયોગ	સંબંધિતતા	અગ્રતા
( )	Function call	Left to Right	First
[ ]	Array expression		
→	Structure operator		
.	Structure operator		
+	Unary plus	Right to Left	Second
-	Unary minus		
++	Increment		
- -	Decrement		
!	Logical NOT		
~	Bitwise NOT		
*	Pointer operator		
&	Address operator		
sizeof( )	Size of operand		
*	Multiplication	Left to Right	Third
/	Division		
%	Modulo division		
+	Binary addition	Left to Right	Fourth
-	Binary subtraction		
<<	Left shift	Left to Right	Fifth
>>	Right shift		
<	Less than	Left to Right	Sixth
<=	Less than equal to		
>	Greater than		
>=	Greater than equal to		
= =	Equal to	Left to Right	Seventh
!=	Not equals to		
&	Bitwise AND	Left to Right	Eighth
^	Bitwise XOR	Left to Right	Ninth
|	Bitwise OR	Left to Right	Tenth
&&	Logical AND	Left to Right	Eleventh
||	Logical OR	Left to Right	Twelfth
? :	Conditional operator	Right to Left	Thirteenth
=, *=, - =, &=, +=,			
^=, |=, <<=, >>=	Assignment Operator	Right to Left	Fourteenth
,	Comma operator	Left to Right	Fifteenth

પરિશિષ્ટ IV

નિયમિત ઉપયોગમાં લેવાતી કેટલીક હેડર ફાઈલ

આપણે જાણીએ છીએ કે, સી ભાષામાં વિધેયો ઘણા જ મહત્ત્વના છે. વિવિધ પ્રક્રિયાઓના અમલ માટે સી ભાષા આંતરસ્થાપિત (inbuilt) વિધેયોનો સંગ્રહ ધરાવે છે. આ વિધેયોનો સંગ્રહ હેડર ફાઈલમાં જૂથ બનાવીને કરવામાં આવ્યો છે. આવી હેડર ફાઈલોના સંગ્રહને સી લાઈબ્રેરી કહે છે. પ્રોગ્રામર ઉપયોગમાં લઈ શકે તે પ્રકારની કેટલીક હેડર ફાઈલોની યાદી નીચે આપવામાં આવી છે.

ફાઈલનું નામ	ઉદ્દેશ
<stdio.h>	પ્રમાણભૂત નિવેશ/નિર્ગમ માટેના વિધેયો
<ctype.h>	અક્ષરની ચકાસણી અને રૂપાંતરણ માટેના વિધેયો
<math.h>	ગાણિતિક વિધેયો
<stdlib.h>	અક્ષરોની હારમાળા (સ્ટ્રિંગ)નું રૂપાંતરણ, મેમરીની ફાળવણી અને યાદચ્છિક અંકોના નિર્માણ માટે ઉપયોગમાં લેવામાં આવતા વિવિધ પ્રકારના વિધેયો
<string.h>	અક્ષરોની હારમાળા (સ્ટ્રિંગ)ને લગતા વિધેયો

ઉપરોક્ત હેડર ફાઈલમાં સમાવવામાં આવેલા કેટલાક વિધેયો નીચે દર્શાવવામાં આવ્યા છે. વિધેયના ઉપયોગ માટે નિવેશ (input) જરૂરી છે. ઈનપુટ એ વિધેયની વ્યાખ્યાનો એક ભાગ છે અને તેને આર્ગ્યુમેન્ટ (argument) તરીકે ઓળખવામાં આવે છે. આર્ગ્યુમેન્ટના ડેટા પ્રકાર જુદા જુદા હોઈ શકે છે. આર્ગ્યુમેન્ટને ઓળખવા માટે નીચે આપેલ અક્ષરોના કોડનો ઉપયોગ કરવામાં આવે છે.

c	- char પ્રકારની આર્ગ્યુમેન્ટ
d	- double પ્રકારની આર્ગ્યુમેન્ટ
f	- file પ્રકારની આર્ગ્યુમેન્ટ
i	- int પ્રકારની આર્ગ્યુમેન્ટ
l	- long પ્રકારની આર્ગ્યુમેન્ટ
p or *	- pointer પ્રકારની આર્ગ્યુમેન્ટ
s	- string પ્રકારની આર્ગ્યુમેન્ટ
u	- unsigned પ્રકારની આર્ગ્યુમેન્ટ

કેટલાક વિધેયો જુદા-જુદા પ્રકારની કિંમતો પરત કરવાની ક્ષમતા પણ ધરાવે છે. જુદા જુદા વિધેયો, તેની આર્ગ્યુમેન્ટ અને પરત કિંમતના પ્રકાર વિશે નીચે મુજબ વિગતો આપવામાં આવી છે.

Functions in <stdio.h> file

વિધેય	પરત કિંમત	સમજૂતી
getc(f)	int	ફાઈલ f માંથી એક અક્ષર ઉમેરવા માટે
getchar(void)	int	પ્રમાણભૂત ઈનપુટ સાધનની મદદથી એક અક્ષર ઉમેરવા માટે
gets(s)	char*	પ્રમાણભૂત ઈનપુટ સાધનની મદદથી અક્ષરોની હારમાળા (સ્ટ્રિંગ) ઉમેરવા માટે
printf(...)	int	પ્રમાણભૂત આઉટપુટ સાધન તરફ વિગતો મોકલવા માટે.
putc(c,f)	int	ફાઈલ f માં એક અક્ષર મોકલવા માટે.
putchar(c)	int	પ્રમાણભૂત આઉટપુટ સાધન તરફ એક અક્ષર મોકલવા માટે

puts(s)	int	પ્રમાણભૂત આઉટપુટ સાધન તરફ અક્ષરોની હારમાળા (સ્ટ્રિંગ) મોકલવા માટે
scanf(...)	int	પ્રમાણભૂત ઈનપુટ સાધનની મદદથી વિગતો ઉમેરવા માટે.

<ctype.h> ફાઈલમાં આવેલ વિધેયો

વિધેય	પરત કિંમત	સમજૂતી
isalnum(c)	int	આર્ગ્યુમેન્ટ અક્ષર કે અંક પ્રકારની (alphanumeric) છે કે નહીં તે ચકાસવા માટે. True માટે શૂન્યેતર અને False માટે શૂન્ય પરત કરશે
isaplpha(c)	int	આર્ગ્યુમેન્ટ અક્ષર (alphabet) છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
isacii(c)	int	આર્ગ્યુમેન્ટ ASCII અક્ષર છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
iscntrl(c)	int	આર્ગ્યુમેન્ટ ASCII નિયંત્રણ અક્ષર છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
isdigit(c)	int	આર્ગ્યુમેન્ટ દશાંકી પ્રકારનો અંક છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
islower(c)	int	આર્ગ્યુમેન્ટ નાના (small/lower) અક્ષરમાં છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
isodigit(c)	int	આર્ગ્યુમેન્ટ અષ્ટાંકી પ્રકારનો અંક છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
isprint(c)	int	આર્ગ્યુમેન્ટ મુદ્રણક્ષમ (printable) ASCII અક્ષર છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
ispunct(c)	int	આર્ગ્યુમેન્ટ વિરામચિહ્ન (punctuation) પ્રકારનો અક્ષર છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
isspace(c)	int	આર્ગ્યુમેન્ટ વ્હાઈટ-સ્પેસ પ્રકારનો અક્ષર છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
isupper(c)	int	આર્ગ્યુમેન્ટ મોટા (capital/upper) અક્ષર છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
isxdigit(c)	int	આર્ગ્યુમેન્ટ સોળઅંકી પ્રકારનો અંક છે કે નહીં તે ચકાસશે. True માટે શૂન્યેતર સંખ્યા અને False માટે શૂન્ય પરત કરશે.
toascii(c)	int	આર્ગ્યુમેન્ટની કિંમતને ASCIIમાં ફેરવવા માટે.
tolower(c)	int	અક્ષરને લોઅર કેસમાં ફેરવવા માટે.
toupper(c)	int	અક્ષરને અપર કેસમાં ફેરવવા માટે.

<math.h> ફાઈલમાં આવેલ વિધેયો

વિધેય	પરત કિંમત	સમજૂતી
acos(d)	double	d ની આર્ક કોસાઈન કિંમત પરત કરવા માટે.
asin(d)	double	dની આર્ક સાઈન કિંમત પરત કરવા માટે.
atan(d)	double	d ની આર્ક ટેન્જન્ટ કિંમત પરત કરવા માટે.
atan2(d1, d2)	double	d1/d2 ની આર્ક ટેન્જન્ટ કિંમત પરત કરવા માટે.
ceil(d)	double	આપેલ સંખ્યાથી વધુ હોય તેવી સૌથી નજીકની પૂર્ણાંક સંખ્યા પરત કરવા માટે
cos(d)	double	d ની કોસાઈન કિંમત પરત કરવા માટે
cosh(d)	double	d ની અતિપરવલય (hyperbolic) કોસાઈન કિંમત પરત કરવા માટે
exp(d)	double	d નો e ઘાતાંક પરત કરવા માટે

fabs(d)	double	d ની નિરપેક્ષ કિંમત પરત કરવા માટે
floor(d)	double	આપેલ સંખ્યાથી ઓછી હોય તેવી સૌથી નજીકની પૂર્ણાંક સંખ્યા પરત કરવા માટે
fmod(d1,d2)	double	d1/d2ની શેષ પરત કરવા માટે (d1ની નિશાની સાથે).
labs(1)	long int	1 ની નિરપેક્ષ (absolute) કિંમત પરત કરવા માટે
log(d)	double	d નો કુદરતી લઘુગુણક (natural logarithm) પરત કરવા માટે
log10(d)	double	d નો (10 આધારિત) લઘુગુણક પરત કરવા માટે.
pow(d1,d2)	double	d1 ના d2 ઘાતાંક જેટલી કિંમત પરત કરવા માટે.
sin(d)	double	d ની સાઈન કિંમત પરત કરવા માટે.
sinh(d)	double	d ની અતિપરવલય (hyperbolic) સાઈન કિંમત પરત કરવા માટે.
sqrt(d)	double	d નું વર્ગમૂળ પરત કરવા માટે.
tan(d)	double	d ની ટેન્જન્ટ કિંમત પરત કરવા માટે.
tanh(d)	double	d ની અતિપરવલય (hyperbolic) ટેન્જન્ટ કિંમત પરત કરવા માટે

<stdlib.h> ફાઈલમાં આવેલ વિધેયો

વિધેય	પરત કિંમત	સમજૂતી
abs(i)	int	i ની નિરપેક્ષ (absolute) કિંમત પરત કરવા માટે.
atof(s)	double	સ્ટ્રિંગને બમણી ચોક્સાઈ ધરાવતી (double-precision) કિંમતમાં ફેરવવા માટે
atoi(s)	int	સ્ટ્રિંગને પૂર્ણાંક સંખ્યામાં ફેરવવા માટે.
atol(s)	long	સ્ટ્રિંગને લૉંગ પૂર્ણાંક સંખ્યામાં ફેરવવા માટે
calloc(u1,u2)	void*	u1 ઘટકોને u2 બાઈટ લંબાઈ ધરાવતા મેમરીના ખંડ ફાળવે છે. ફાળવણીની શરૂઆતના ખંડનું પોઈન્ટર પરત કરશે.
exit(u)	void	તમામ ફાઈલો અને બફર બંધ કરી પ્રોગ્રામ બંધ કરવા માટે. (પ્રોગ્રામના અંતની સ્થિતિ દર્શાવવા વિધેય દ્વારા u ની કિંમત ઉમેરવામાં આવશે.)
free(p)	void	p થી શરૂઆત થતી હોય તેવા મેમરીના ફાળવેલા ખંડને મુક્ત કરવા માટે.
malloc(u)	void*	u બાઈટ જેટલી મેમરી ફાળવવા માટે ફાળવવામાં આવેલ જગ્યાની શરૂઆતનું પોઈન્ટર પરત કરશે.
rand(void)	int	યાદૃચ્છિક (random) ધન પૂર્ણાંક પરત કરશે.
realloc(p,u)	void*	પોઈન્ટર ચલ p ને u બાઈટ જેટલી નવી મેમરી ફાળવશે. નવી મેમરી જગ્યાની શરૂઆતનું પોઈન્ટર પરત કરશે.

<string.h> ફાઈલમાં આવેલ વિધેયો

વિધેય	પરત કિંમત	સમજૂતી
strcmp(s1, s2)	int	બે સ્ટ્રિંગને શબ્દકોશ રચના પ્રમાણે (lexicographically) સરખાવશે. જો s1 < s2 હોય તો ઋણ સંખ્યા, s1 અને s2 સરખા હોય તો 0 અને s1 > s2 હોય તો ધન સંખ્યા પરત કરશે.
strcmpi(s1,s2)	int	બે સ્ટ્રિંગના કેસને અવગણી શબ્દકોશ રચના પ્રમાણે (lexicographically) સરખાવશે. જો s1 < s2 હોય તો ઋણ સંખ્યા, s1 અને s2 સરખા હોય તો 0 અને s1 > s2 હોય તો ધન સંખ્યા પરત કરશે.
strcpy(s1, s2)	char*	s2 સ્ટ્રિંગની નકલ s1માં કરશે.
strlen(s)	int	s સ્ટ્રિંગમાં આવેલ અક્ષરોની સંખ્યા પરત કરશે.