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.

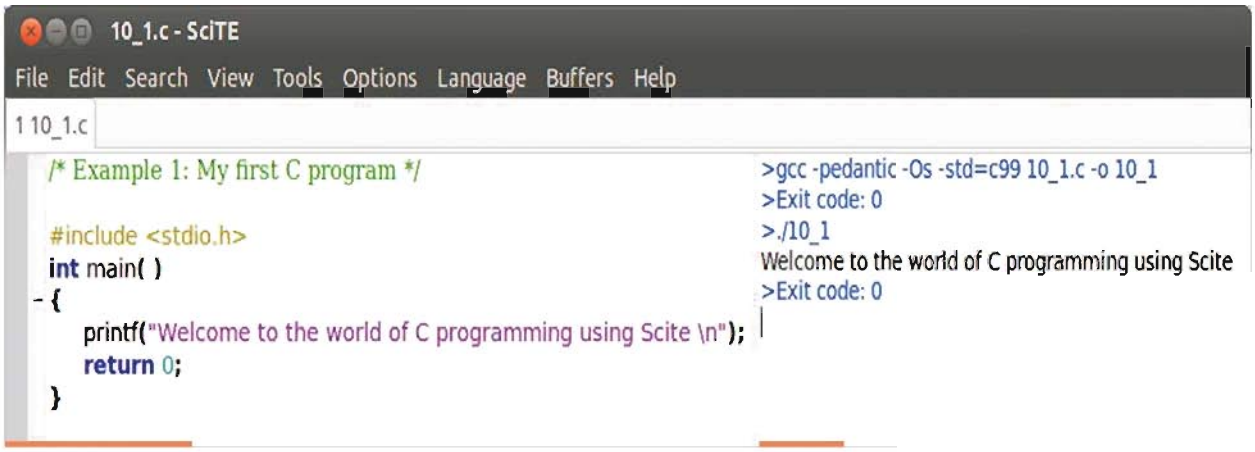
એક સારો પ્રોગ્રામ નીચે જણાવેલ લાક્ષણિકતાઓ ધરાવતો હોવો જોઈએ :

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.

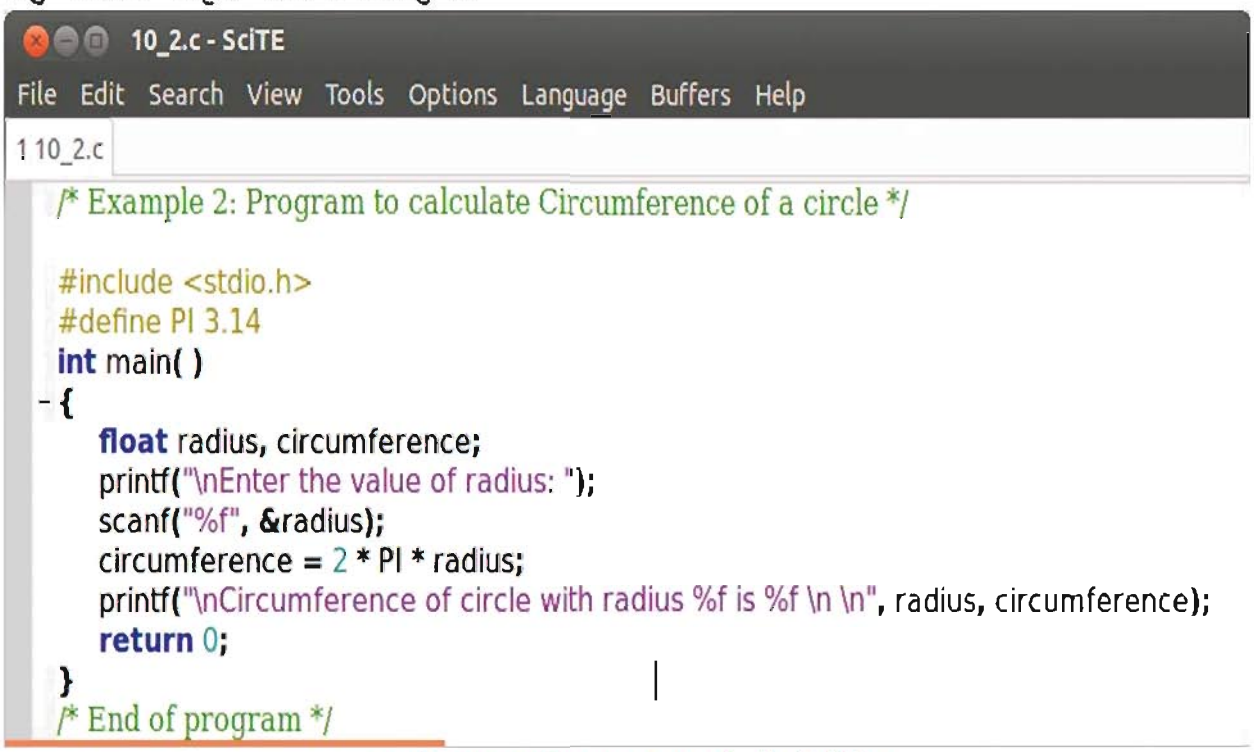
તમે જોઈ શકો છો કે, આ લાક્ષણિકતાઓ અલગોરિધમની લાક્ષણિકતાઓને મળતી આવે છે. ઉદાહરણ 10.1 એક નમૂનારૂપ સી પ્રોગ્રામ દર્શાવે છે. ઉદાહરણ 10.1 માટેનું કોડ લિસ્ટિંગ અને પરિણામ આકૃતિ 10.1માં દર્શાવ્યા છે. SciTE એડિટરના ઉપયોગની કાર્ય પદ્ધતિ આપણે આ પ્રકરણના અંતમાં શીખીશું.



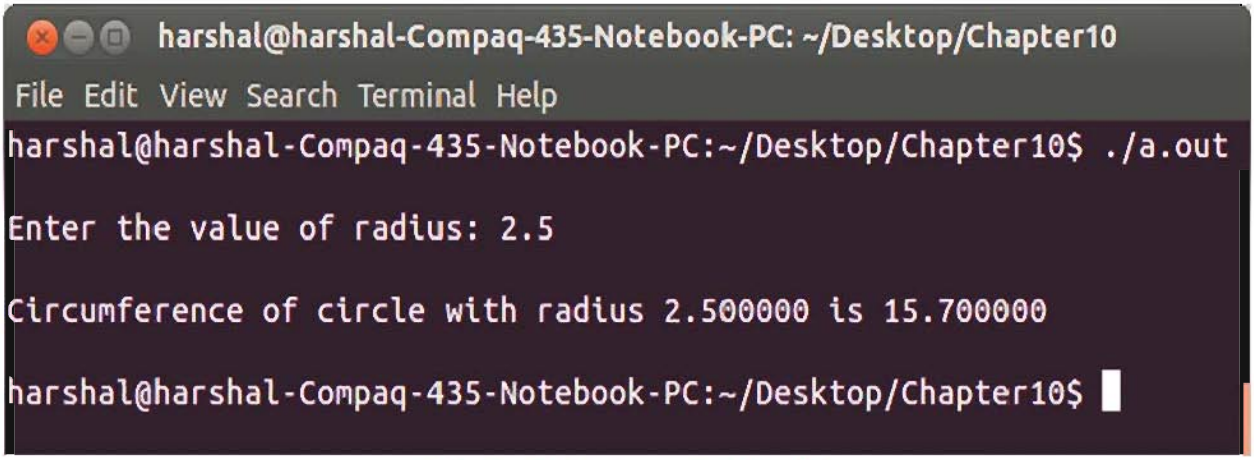
આકૃતિ 10.1 : પ્રથમ 'સી' પ્રોગ્રામ

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.



આકૃતિ 10.2 ઉદાહરણ 10.2નું કોડ-લિસ્ટિંગ



આકૃતિ 10.3 : ઉદાહરણ 10.2નું પરિણામ

સમજૂતી

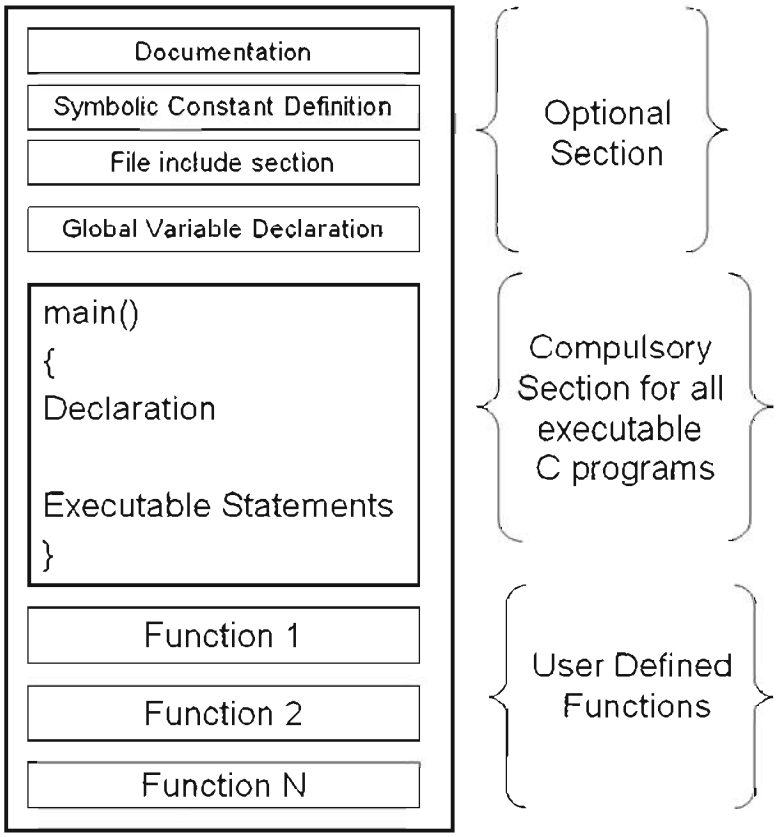
અહીં આપેલ પ્રોગ્રામનો ઉપયોગ કરી આપેલ ત્રિજ્યાને આધારે વર્તુળનો પરિઘ શોધી શકાય છે.

- 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 ડાઇરેક્ટિવ (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 એક્સ્પ્રેશન (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 કમ્પાઇલર 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 ઉદ્દેશો 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

```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.

```
harshal@harshal-Compaq-435-Notebook-PC: ~/Desktop/Chapter10
File  Edit  View  Search  Terminal  Help
harshal@harshal-Compaq-435-Notebook-PC:~/Desktop/Chapter10$ ./a.out

Enter the value of first: 55

Enter the value of second: 24

The value of 55 - 24 is 31

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

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

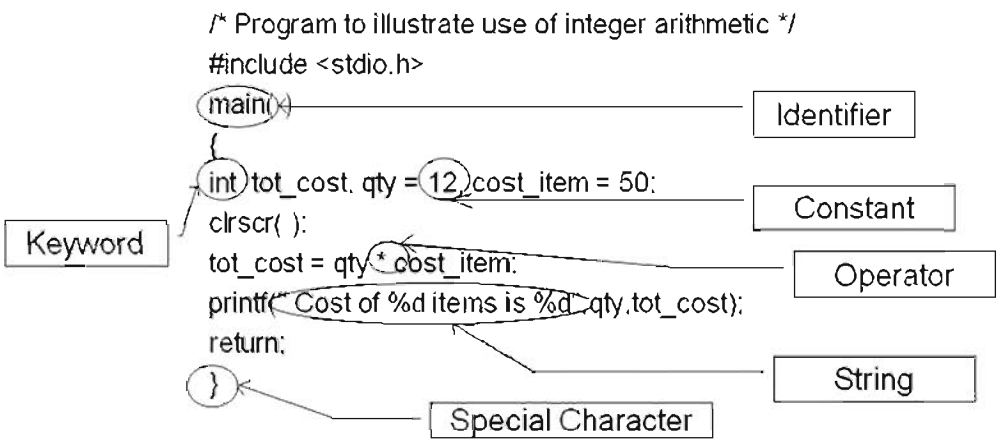
Special Characters			
Operator	**Use**	**Operator**	**Use**
&	Ampersand	>	Greater than
'	Apostrophe	#	Hash sign
*	Asterisk	<	Less than
@	At the rate	-	Minus
\	Backward slash	( )	Parenthesis left / right
{ }	Braces left / right	%	Per cent
[ ]	Bracket left / right	.	Period
^	Caret	+	Plus
:	Colon	?	Question 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 જોઈએ.

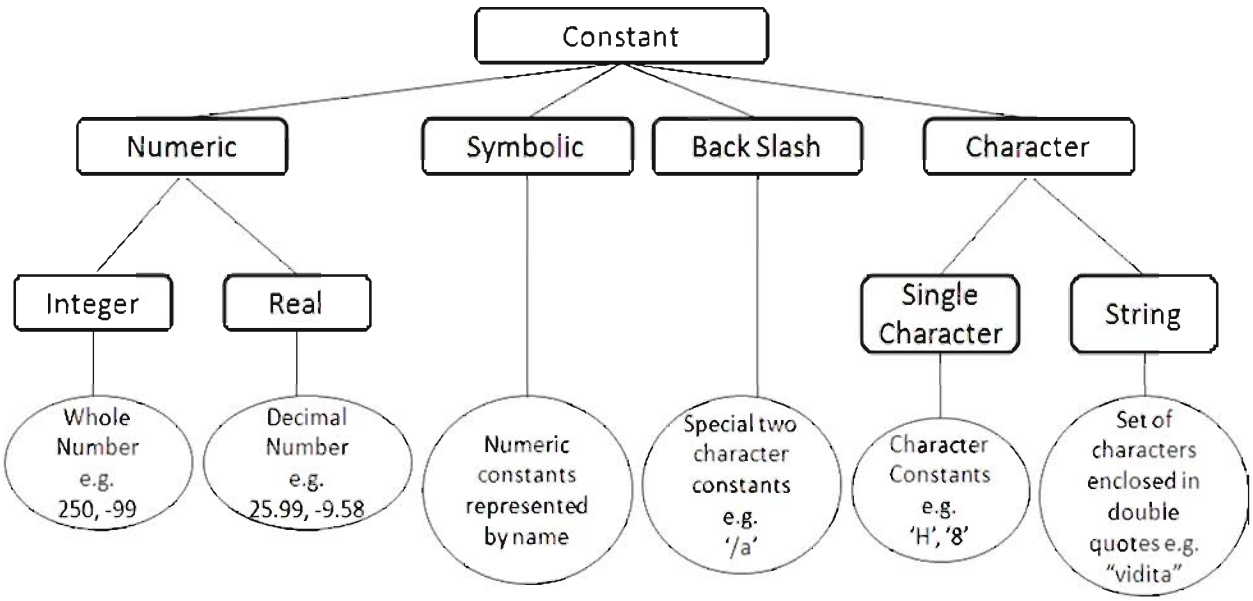
કોષ્ટક 10.3 : સી ચલનાં ઉદાહરણ

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 યુનરી 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 જોઈએ.
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 સીકવન્સ' 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 રીટર્ન 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 સ્લેશ 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.

```
10_5.c - SciTE
File Edit Search View Tools Options Language Buffers Help
1 10_5.c
/* 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
File Edit View Search Terminal Help
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)

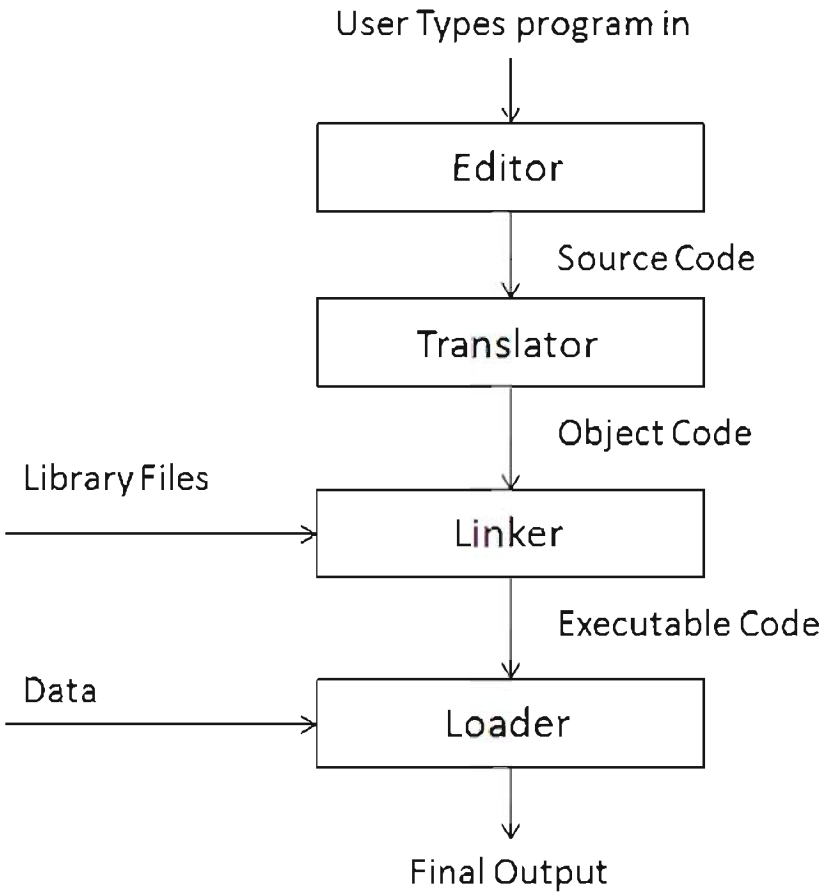
સી ભાષામાં પ્રોગ્રામિંગ સરળ હોવા છતાં, તેનો મહાવરો જરૂરી છે. તમામ સી પ્રોગ્રામરોએ નીચેના વિભાગમાં આપવામાં આવેલ મુદ્દા યાદ રાખવા જોઈએ :

- 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 વિષેય 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.

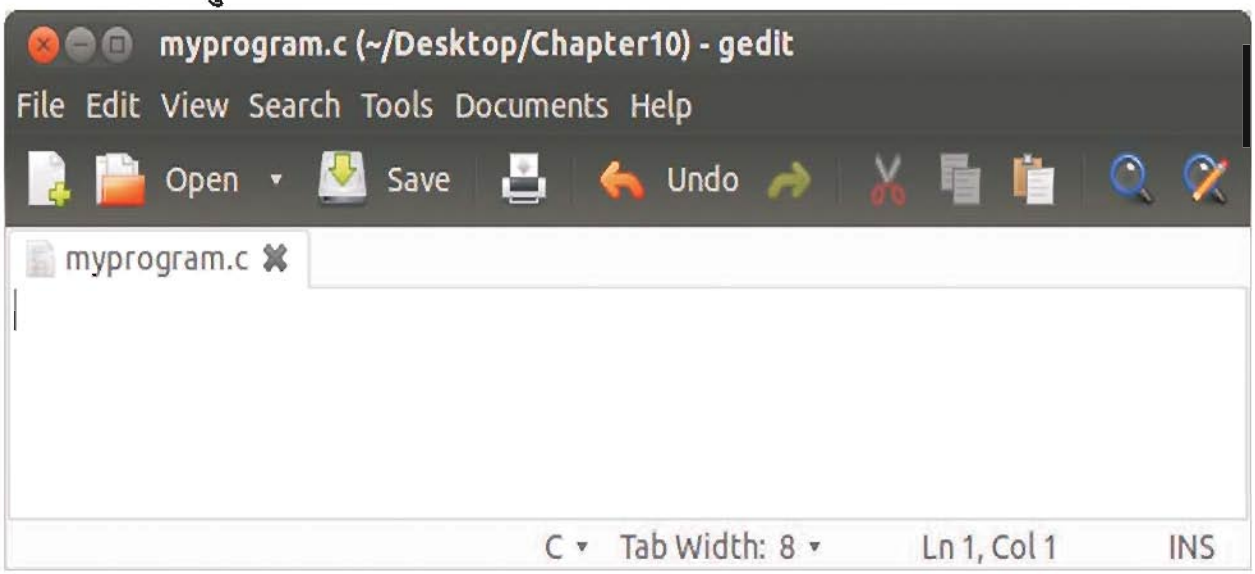


આકૃતિ 10.9 (a) : સી પ્રોગ્રામનાં અમલીકરણનાં પગલાં

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 કંપાઇલરની મદદથી પ્રોગ્રામને સંકલિત (compile) કરવામાં આવે છે. કંપાઇલર 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 લાઇબ્રેરી 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 વગતો 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

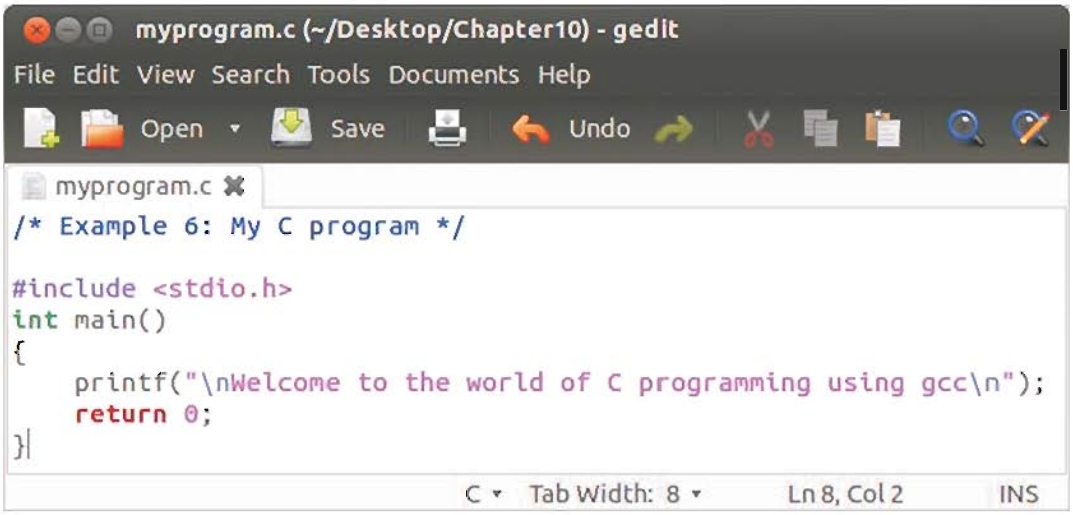
આ ખાલી એડિટર સ્ક્રીનમાં હવે કોડ લિસ્ટિંગ 10.2માં આપવામાં આવેલ વિગતો ઉમેરો.

```
/* 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

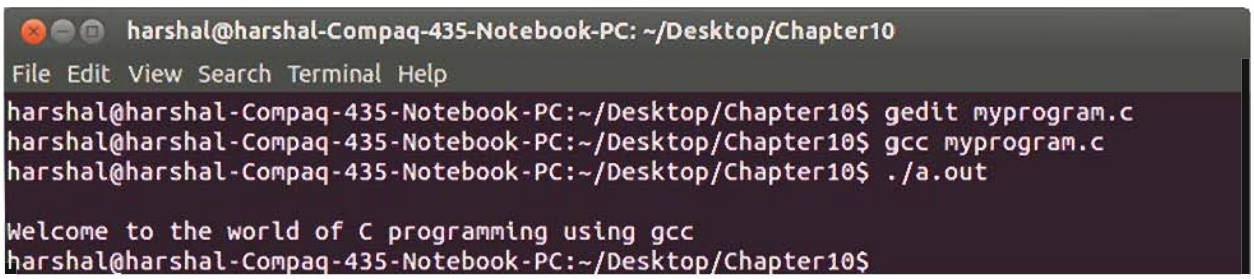
હવે આપણે તેનું કમ્પાઇલેશન કરવા તૈયાર છીએ. ટર્મિનલમાં પાછા ફરી નીચે જણાવેલ કમાન્ડ ટાઇપ કરો :

```
$ 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 :

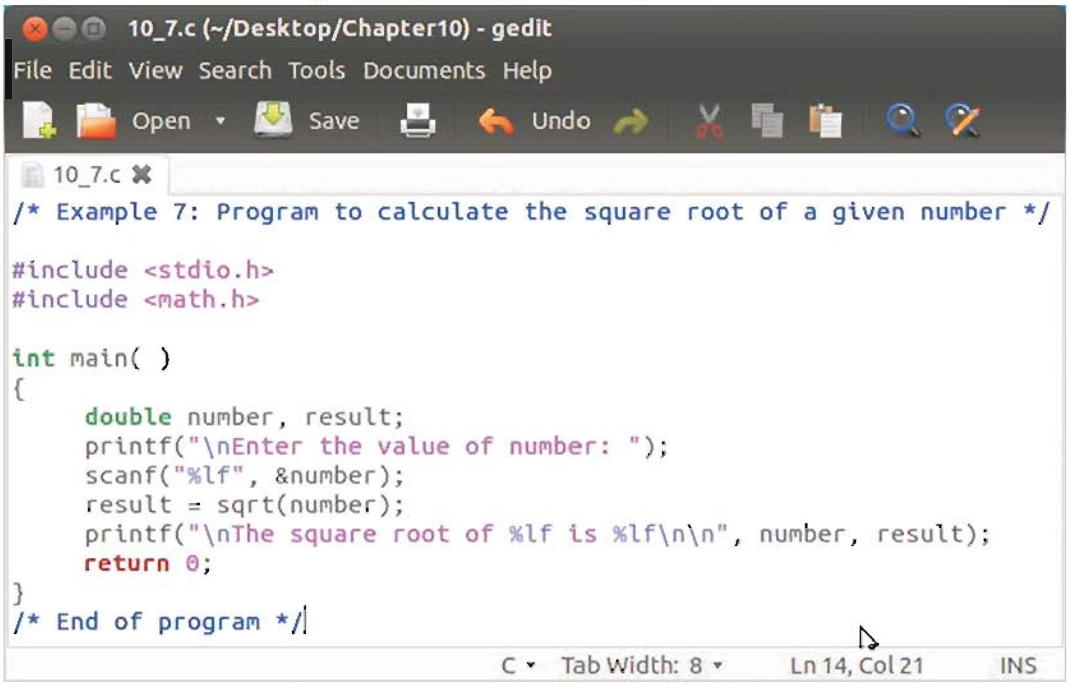
```
$ ./myprogram.o
```

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.

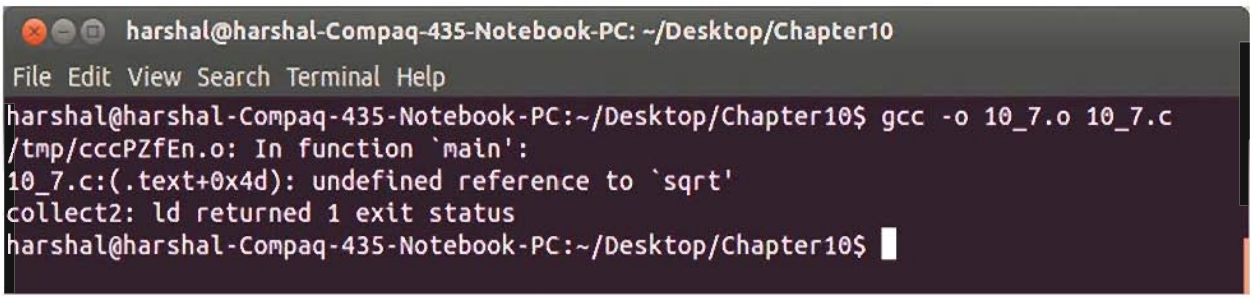


આકૃતિ 10.13 : ઉદાહરણ 10.7નું કોડ લિસ્ટિંગ

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

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

આ gcc કમાન્ડનું પરિણામ આકૃતિ 10.14માં દર્શાવ્યું છે.

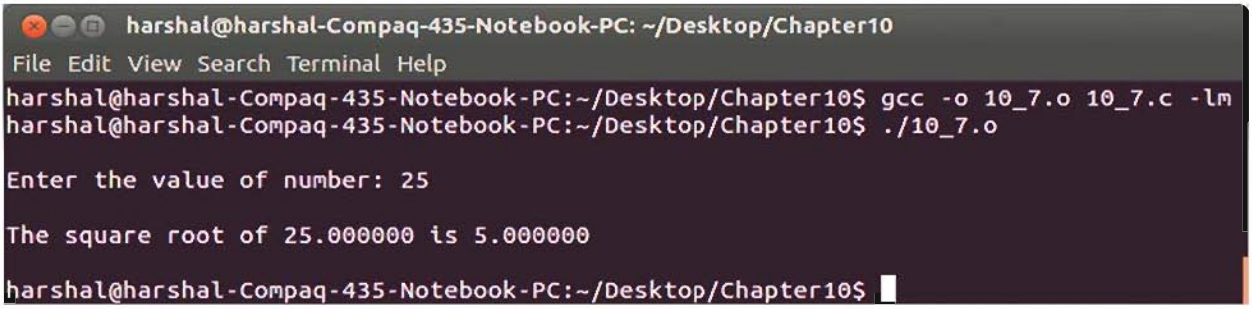


આકૃતિ 10.14 : ઉદાહરણ 10.7નું પરિણામ

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 કમ્પાઇલેશન સમયે તેની સાથે ગાણિતિક લાઇબ્રેરી જોડવી જરૂરી છે. નીચે આપેલ કમાન્ડ દ્વારા ગાણિતિક લાઇબ્રેરી જોડી શકાશે.

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

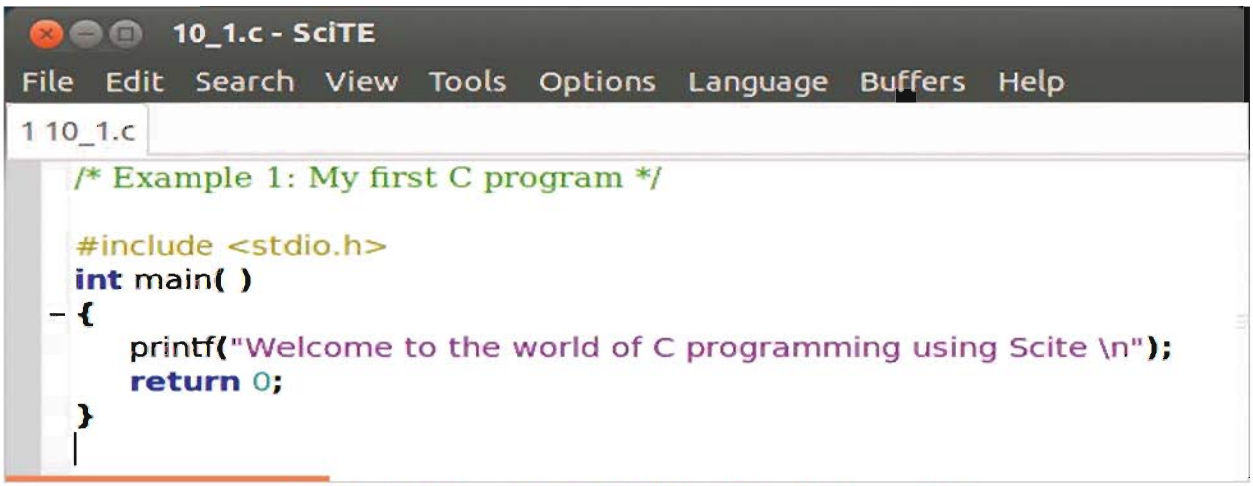
આ કમાન્ડ આપવાથી પ્રોગ્રામ કોઈ પણ ભૂલના સંદેશ વગર કમ્પાઇલ થઈ જશે. હવે પ્રોમ્પ્ટ પર ./10_7.o ટાઇપ કરી પરિણામ જુઓ. આકૃતિ 10.15માં સાચું પરિણામ દર્શાવ્યું છે.



આકૃતિ 10.15 : ઉદાહરણ 10.7નું સાચું પરિણામ

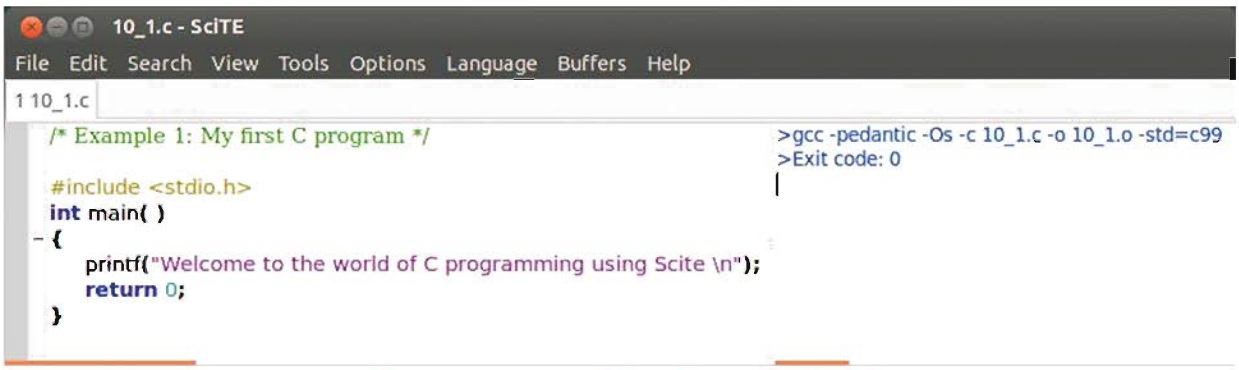
અહીં દર્શાવેલી પદ્ધતિ લિનક્સ ઑપરેટિંગ સિસ્ટમમાં સી પ્રોગ્રામ લખવા અને કમ્પાઇલ કરવા માટેની સૌથી પાયારૂપ પદ્ધતિ છે. અન્ય કોઈ પણ સાદા ટેક્સ્ટ એડિટરનો ઉપયોગ કરવાને બદલે આ પુસ્તકમાં આપણે પ્રોગ્રામ લખવા અને અમલ કરવા માટે SciTE નામના ટેક્સ્ટ એડિટરનો ઉપયોગ કરીશું. SciTE એક જ વિન્ડોમાં પ્રોગ્રામને કંપાઇલ અને 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. ફાઇલને 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 : કમ્પાઇલેશનની 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 કમ્પાઈલ 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

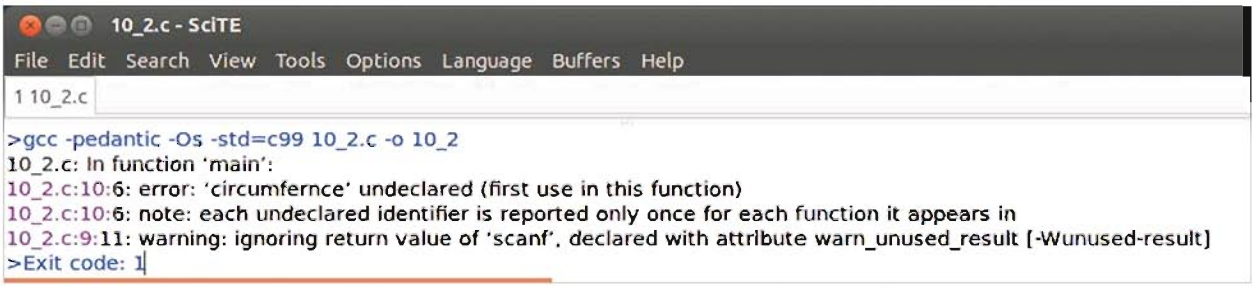
>Exit code : 0

અહીં ફાઈલ પર બે તબક્કામાં પ્રક્રિયા કરવામાં આવે છે. પ્રથમ તબક્કામાં ફાઈલને કંપાઈલ 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 કમ્પાઈલર 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 કમ્પાઈલેશનની 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 કમ્પાઈલેશન

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 ફાઇલના 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 લેબોરેટરી (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 પોર્ટેબલ (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 ફાઇલમાં નીચે જણાવેલ લીટી ઉમેરવી જરૂરી બનશે :

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

નીચે દર્શાવેલ પગલાં દ્વારા પ્રોપર્ટી ફાઇલ 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 લાક્ષણિકતાની 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 ડાઇરેક્ટિવનો 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 ઇચ્છા હો તો "Wishing you a Happy and Prosperous Diwali" સંદેશ દર્શાવો.