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 પ્રોગ્રામની શરૂઆતમાં જે-તે લાઇબ્રેરી ફાઇલ 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 ફાઇલમાં કરવામાં આવ્યો છે. #include વિધાન stdio.h ફાઇલને શોધી તેની વિગતોને પ્રોગ્રામની શરૂઆતમાં ઉમેરવાની કમ્પાઇલરને સૂચના આપે છે. ત્યારપછી આ હેડર ફાઇલની 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 ફાઇલ 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 કોઈ 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

```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 ટાઇપ 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 ફાઇલમાં 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.

9-2.c - SciTE

File Edit Search View Tools Options Language Buffers Help

9-2.c

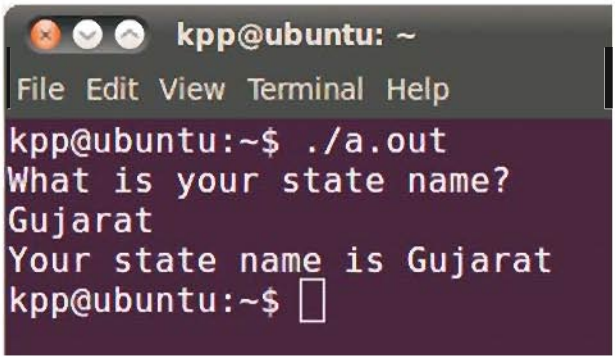
```c
/* Example 2: Program to demonstrate use of gets() function. */

#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.

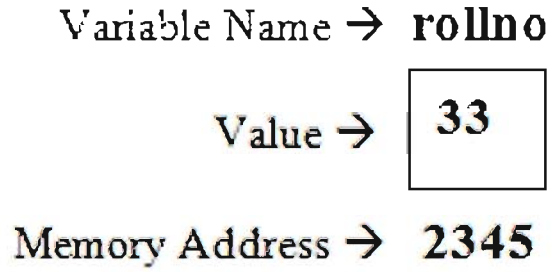
આ પ્રકારનો ડેટા મેળવવા માટે સી ભાષા સુગ્રથિત નિવેશ (formatted input) વિધેય નામની સુવિધા પૂરી પાડે છે. એક ફૉર્મેટેડ ઇનપુટ વિધેય scanf()નો ઉપયોગ આપણે ઘણા પ્રોગ્રામોમાં કર્યો છે. scanf() વિધેય દ્વારા ફૉર્મેટેડ ઇનપુટ માટે ઉપલબ્ધ વિવિધ વિકલ્પોનો અભ્યાસ કરીએ.

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 હોઈ 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 ઍરેમાં 'Gandhinagar' કિંમતનો સંગ્રહ કરવામાં આવશે. અક્ષરોની ઍરે 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

```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

```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.

9-5.c - SciTE

File Edit Search View Tools Options Language Buffers Help

9-5.c

```c
/* 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: ~

File Edit View Terminal Help

```
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.

9-6.c - SciTE

File Edit Search View Tools Options Language Buffers Help

9-6.c

```c
/* Example 6: Program to demonstrate the use of puts() function.*/

#include<stdio.h>
int main()
{
      char state[20];

      printf("Enter your state name:\n");
      gets(state);
      printf("Your state name is ");
      puts(state);
      return 0;
}
//End of Program.
```

આકૃતિ 12.12 : ઉદાહરણ 12.6નું કોડ-લિસ્ટિંગ

kpp@ubuntu: ~

File Edit View Terminal Help

```
kpp@ubuntu:~$ ./a.out
Enter your state name:
Gujarat
Your state name is Gujarat
kpp@ubuntu:~$
```

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) કંટ્રોલ 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 મૉનિટર પર દર્શાવવાના છે.
- દરેક ચલના ફૉર્મેટ સ્પેસિફાયર
- \n (નવી લીટી), \t (ટેબ), \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 :

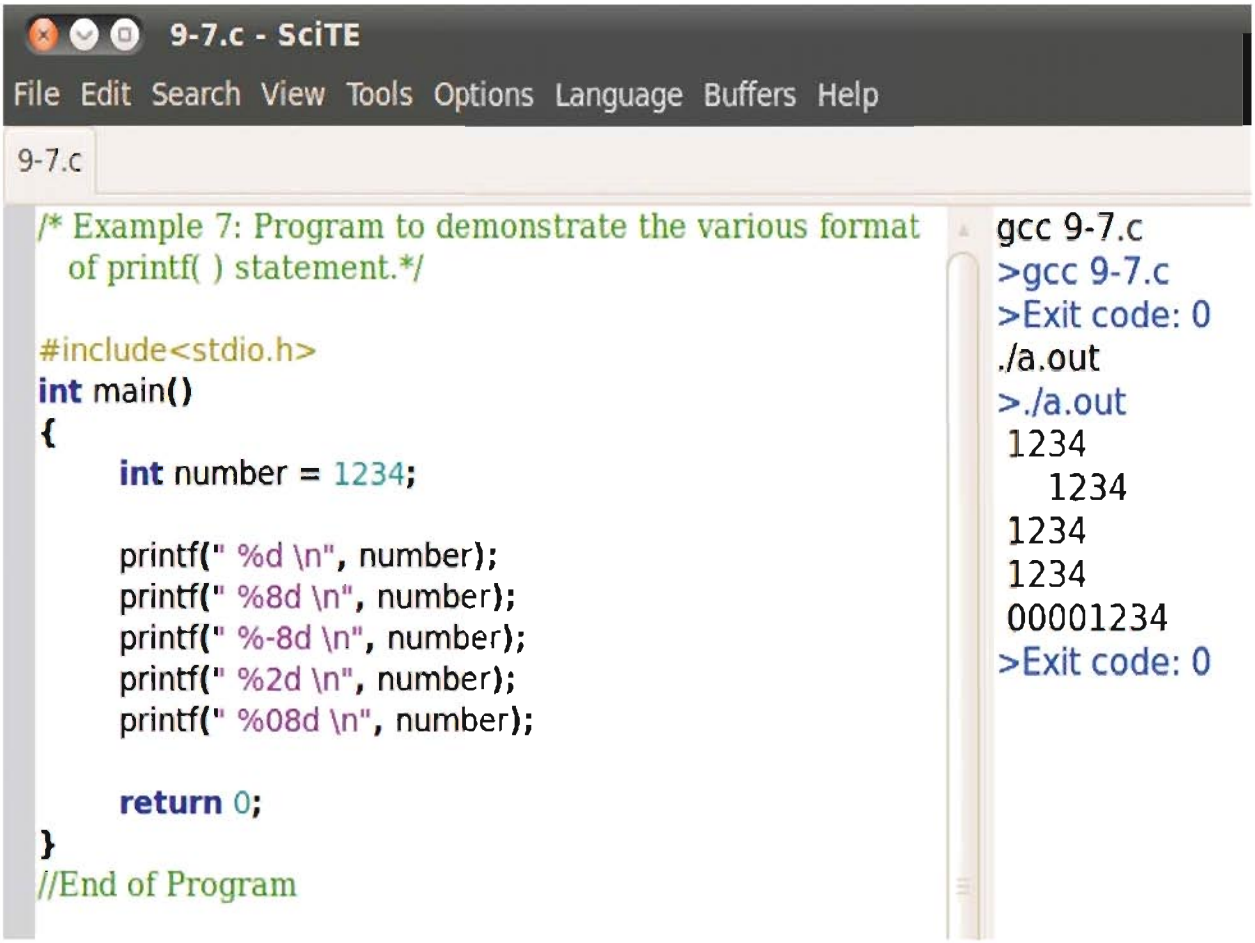
%l.mp

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 લઈ 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
લૉંગ દશાંકી પૂર્ણાંક દર્શાવવા માટે	%ld
ડબલ સંખ્યા દર્શાવવા માટે	%lf
લૉંગ 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.

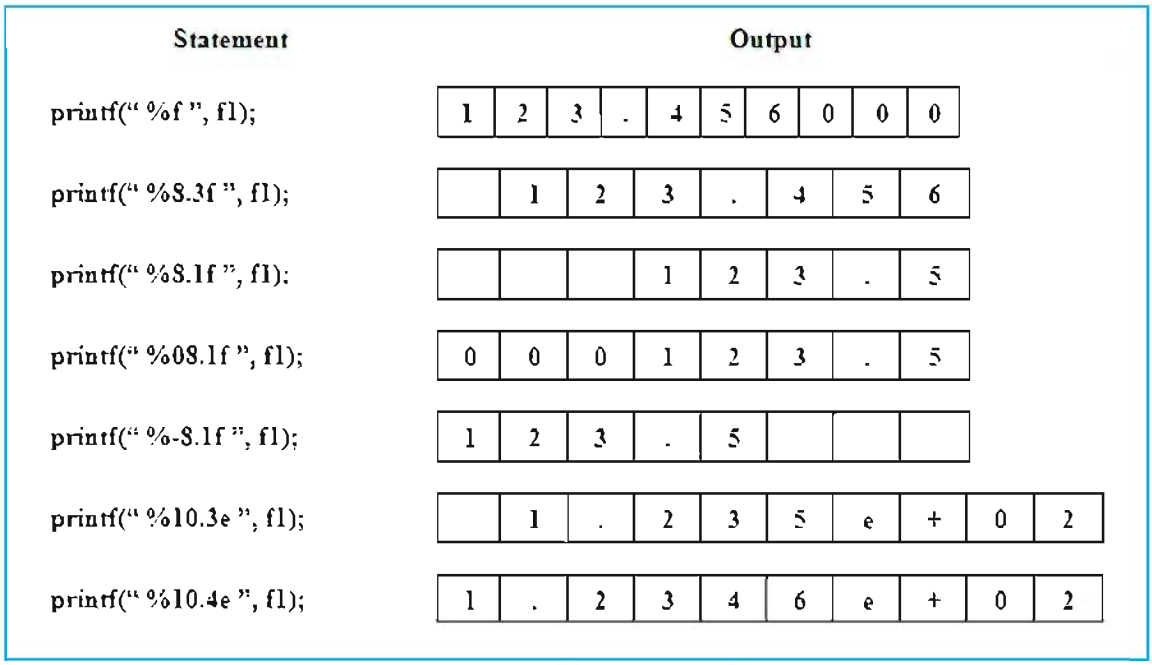


આકૃતિ 12.14 : ઉદાહરણ 12.7નું કોડ-લિસ્ટિંગ અને પરિણામ

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.

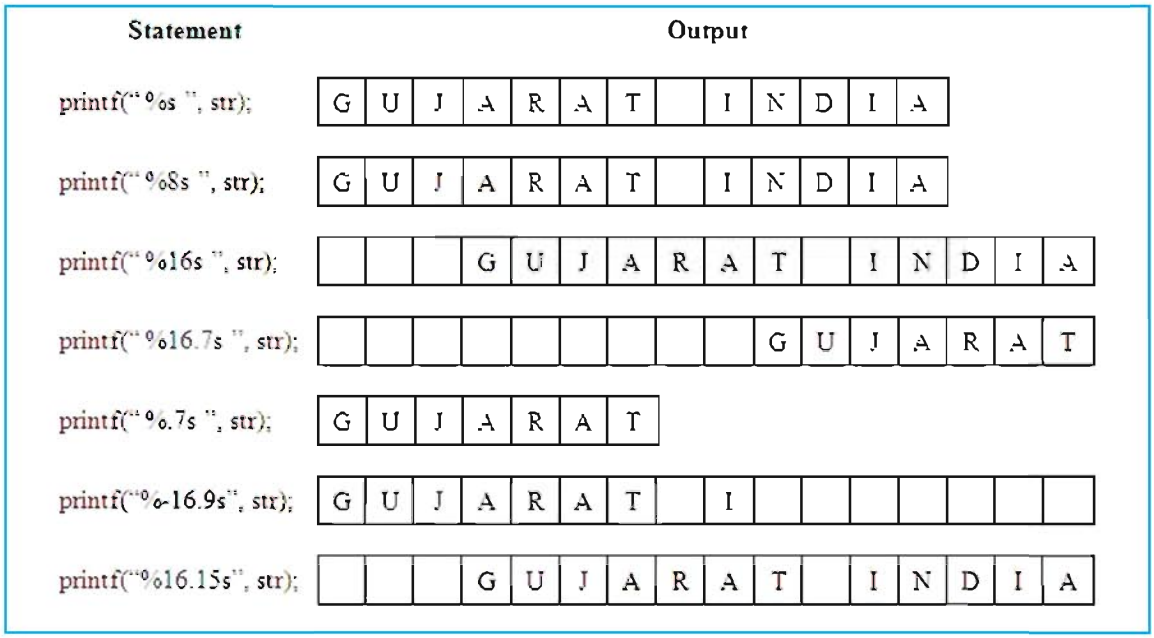
અગાઉના ઉદાહરણમાં આપણે જોયું કે પૂર્ણાંક ચલને જુદા જુદા પ્રકારની સંરચનાઓ સાથે કેવી રીતે દર્શાવી શકાય. હવે, વિવિધ સંરચનાઓ દ્વારા અપૂર્ણાંક સંખ્યાને કેવી રીતે દર્શાવી શકાય તે જોઈએ. સામાન્ય રચના %l.mp યાદ કરો જેમાં *m* એ દશાંશ પછી દર્શાવવાના અંકોનો નિર્દેશ કરે છે. જ્યારે અપૂર્ણાંક સંખ્યા દર્શાવવામાં આવે 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.



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.



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) મોનિટર (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) મોનિટર

(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)