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

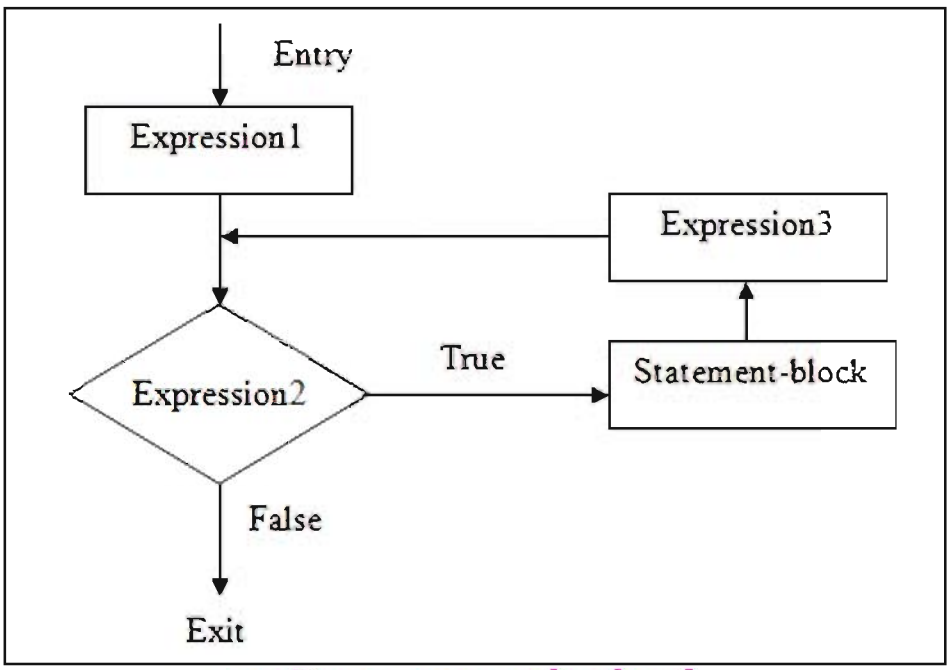
સમજૂતી

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 બ્લોકના 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

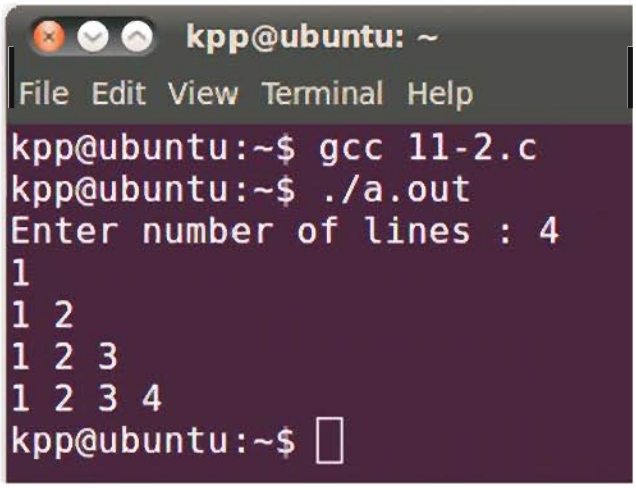
11-2.c

```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



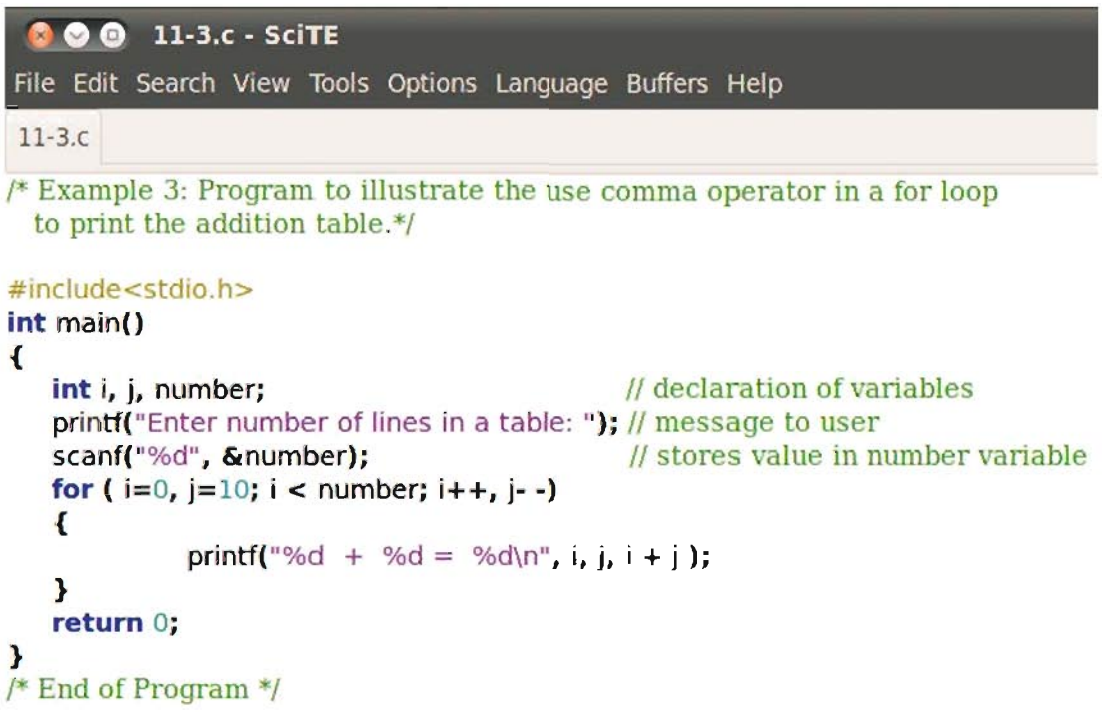
આકૃતિ 14.4 : ઉદાહરણ 14.2નું પરિણામ

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.



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)

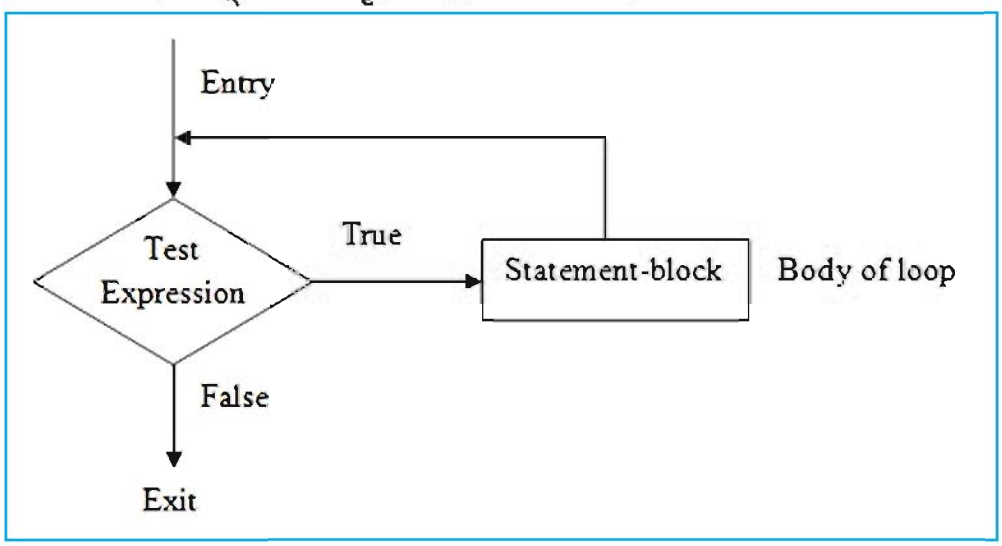
જ્યારે ફેરાની સંખ્યા પૂર્વનિશ્ચિત ન હોય અને લૂપને અટકાવવા માટેની શરત લૂપમાં પ્રવેશ કરતાં પહેલાં ચકાસવાની હોય ત્યારે while લૂપનો ઉપયોગ વધુ અનુકૂળ છે. ઉદાહરણ તરીકે,

- 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.

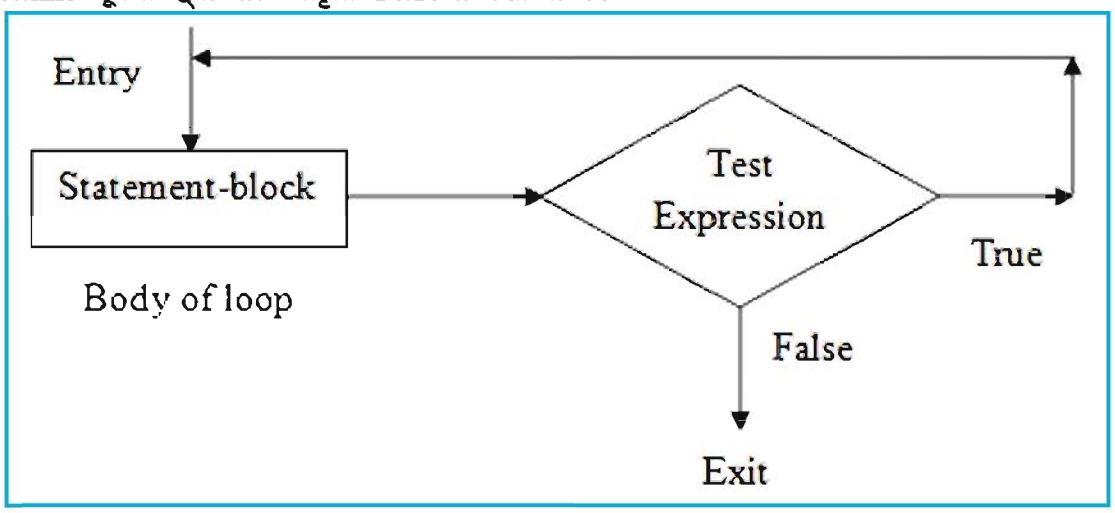
do...while લૂપ (The do...while loop)

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

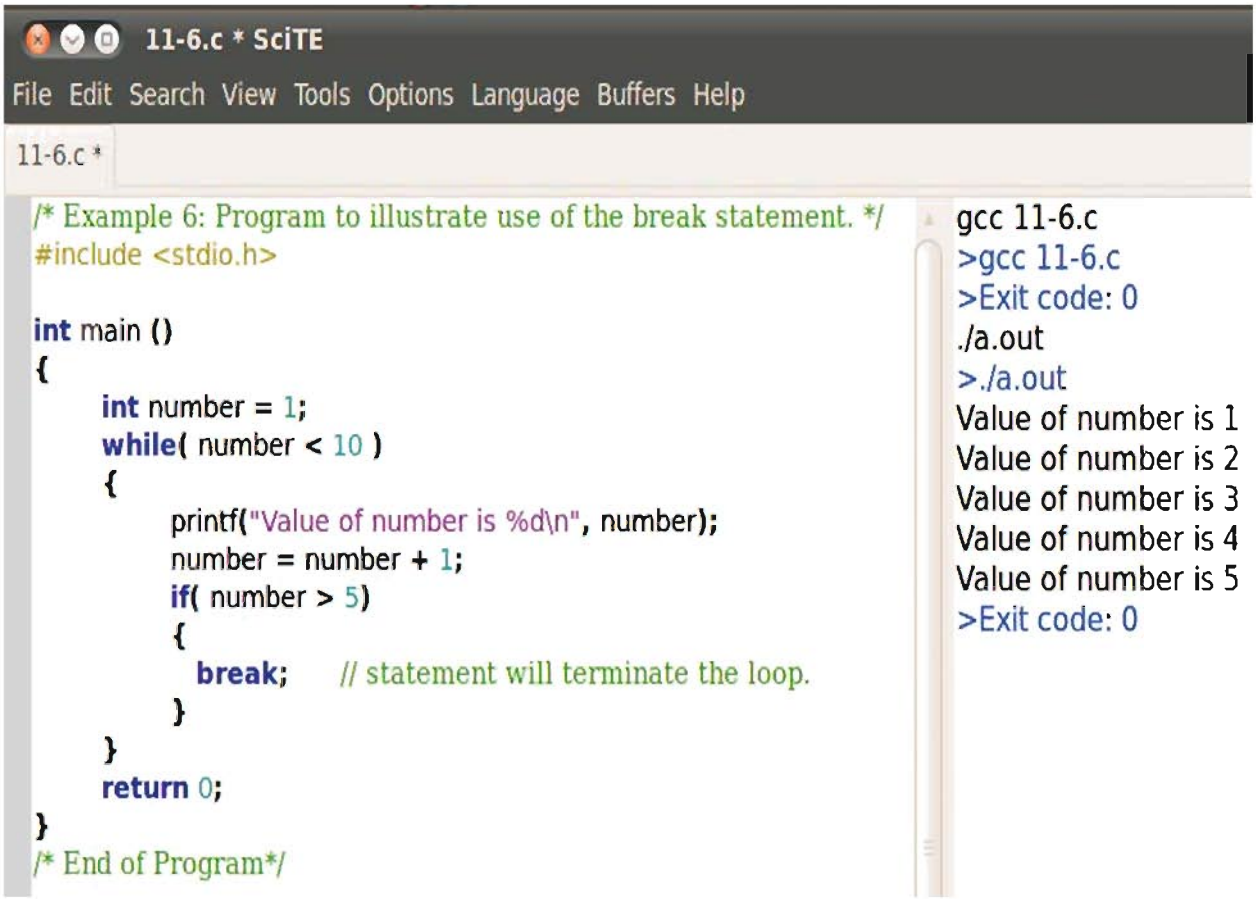
break વિધાન (The break statement)

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

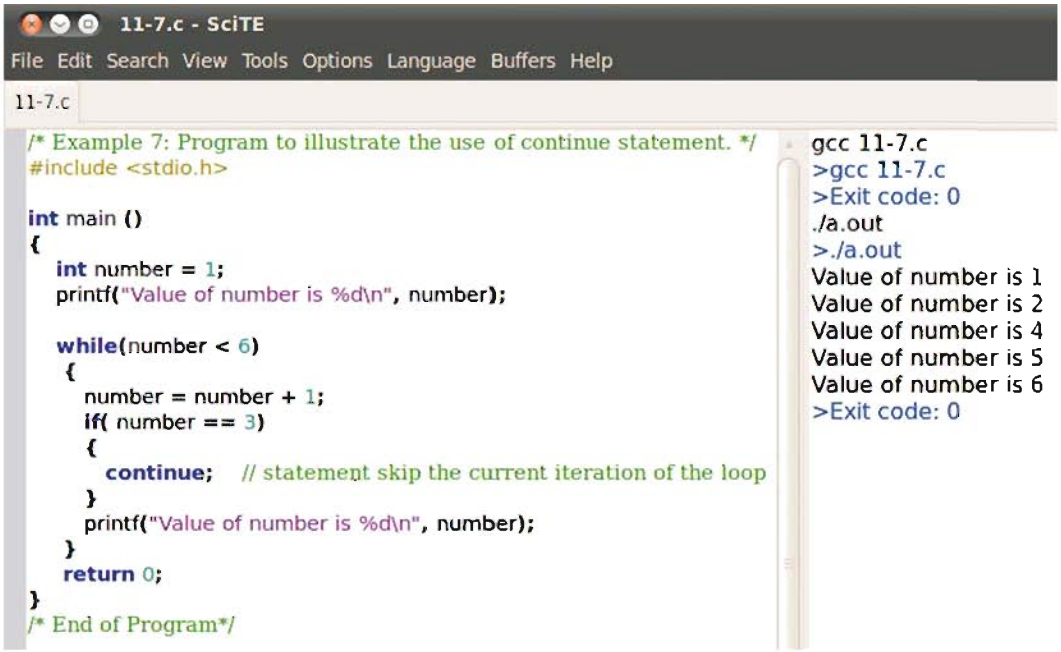
main() વિધેય પછીનું પ્રથમ વિધાન એક પૂર્ણાંક ચલની પ્રારંભિક કિંમત નિશ્ચિત કરે છે. લૂપને નવ (9) વખત ફેરવવા માટે *while* વિધાનમાં શરત લખવામાં આવી છે. લૂપની અંદર આવેલું printf વિધાન number ચલની કિંમત દર્શાવશે. number ચલની કિંમત દર્શાવ્યા પછી તેની કિંમતમાં 1નો વધારો કરવામાં આવશે. ત્યારપછી, *if* વિધાનનો અમલ 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.



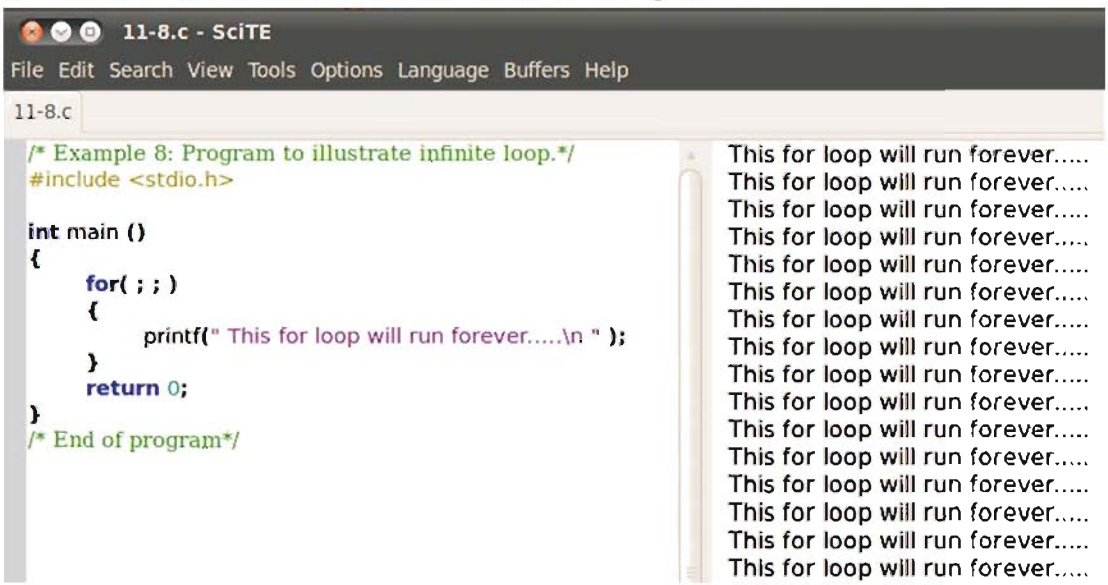
આકૃતિ 14.14 : ઉદાહરણ 14.7નું કોડ-લિસ્ટિંગ અને પરિણામ

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.



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 પૅટર્ન 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.