C - Programming

Loops

Jayakumar P

Department of Computer Science and Applications

Iterative Control Flow

- *Iterative Control Flow* constructs (also called loops) execute a set of statements repeatedly until a certain condition is satisfied
- Three types of loops in C
  1. for loop
  2. while loop
  3. do while loop
- The number of iterations of the loops can be either bounded (for loop) or unbounded (while, do while loops)

The while loop

- Unbounded loop
- Syntax of while loop is

```
while(test-expression)
{
    statement1;
    statement2;
    .
    .
}
```

For example, the following while loop prints the counting numbers from 1 to 100.

```
int c = 1;
while (c<=100)
{
    printf("%d",c);
    c++;
}
```

- Initially the test expression is evaluated, if it is true then the body of the loop is executed.
- Then the test expression is evaluated again and if it is true then the body of the loop is executed again.
- This gets repeated as long as the test expression is evaluated to true

The for loop

- Bounded loop. Syntax of for loop,

```
for(Initialization;Test-condition;Updating)
{
    statement1;
    statement2;
    .
    .
}
```

- For example, the following for loop prints the numbers from 1 to 100

```
for(int c = 1; c <= 100 ; c++)
{
   printf("%d",c);
}
```

The do-while loop

- unbounded loop. Syntax of do-while loop,

```
do
{
    Statement 1;
    .
    .
}
while(test-condition);
```

- For example, the following do while loop prints the numbers from 1 to 100

```
int c = 1;
do
{
   printf("%d",c);
}
while(c <=100);
```

Q1

Write a program to find the sum of first n natural numbers

Q1 - Solution

Using while loop

```
int n=0,sum=0;
scanf("%d",&n);
while(n>0)
{
    sum += n;
    n--;
}
printf("%d",sum);
```

Using for loop

```
int n=0,sum=0;
scanf("%d",&n);
for(int i=1;i<=n;i++)
{
    sum += i;
}
printf("%d",sum);
```

Using do while loop

```
int n=0,sum=0;
scanf("%d",&n);
do
{
    sum += n;
    n--;
}
while(n>0);
printf("%d",sum);
```

Q2

Write a program to find the factorial of a given number n

Q1 - Solution

Using while loop

```
int n=0,f=1;
scanf("%d",&n);
while(n>0)
{
    f *= n;
    n--;
}
printf("%d",f);
```

Using for loop

```
int n=0,f=1;
scanf("%d",&n);
for(int i=1;i<=n;i++)
{
    f *= i;
}
printf("%d",f);
```

Using do while loop

```
int n=0,f=1;
scanf("%d",&n);
do
{
    f *= n;
    n--;
}
while(n>0);
printf("%d",f);
```

break statement

- break statement is a special control statement
- break is used to terminate the execution of a loop or a switch statement
- The program then executes the statement after the loop/switch.

```
7   ******************************************
8
9   #include <stdio.h>
10
11  int main()
12  {
13      int i =10;
14      while (i--)
15      {
16          printf("%d\n",i);
17          break;
18      }
19      return 0;
20  }
```

```
9

...Program finished with exit code 0
Press ENTER to exit console.
```

continue statement

- continue is also a special control statement used in loops
- continue is used to terminate the execution the current iteration of a loop. (It skips all statements after continue, in the loop)
- The program then starts the next iteration

```
7   ********************************
8
9   #include <stdio.h>
10
11  int main()
12  {
13      int i =4;
14      while (i--)
15      {
16          printf("%d\n",i);
17          continue;
18          printf("*%d\n",i);
19      }
20      return 0;
```

```
3
2
1
0
```

Predict the output

```
#include <stdio.h>
int main( )
{
    int i=1;
    for(;i<=5;i++)
    {
        if(i==3)
            break;
        printf("%d\n",i);
    }
    return 0;
}
```

Predict the output - Solution

1

2

When the value of i is 3, the if condition evaluates to true and thus the break statement is executed and the loop is terminated.

Predict the output

```
int main( )
{
    int i=1;
    while(1)
    {
        if(i==5)
            break;
        printf("%d", i);
        i++;
    }
    return 0;
}
```

Predict the output - Solution

Though the condition while(1) makes it a infinite loop, the break statement in the body helps terminate the loop at the value i ==5. So the output is 1
2
3
4

Predict the Output

```
int main()
{
    int n = 0;
    while(n<=5)
    {
        n++;
        if(n==3)
            continue;
        printf("%d\n", n);
    }
    return 0;
}
```

Predict the output - Solution

The continue statement is executed when the value of n == 3. Thus in that execution the remaining statement in the body of the loop (the printf statment) is not executed. So the program will print all numbers except 3.

1
2
4
5
6

Q4

Write a program to check whether a given number is prime or not.

Solution

```c
int main()
{
    int n,i=1,flag=0;
    scanf("%d",&n);
    while(++i<=sqrt(n))
    {
        if(n/i==0)
        {
            flag=1;
            break;
        }
    }
    if(flag) printf("Not Prime");
    else printf("Prime");
    return 0;
}
```

Nested Loops

- Nested loop refers to a loop that is included in the body of another loop.
- For example, the follwing code has a nested loop that prints characters (a to c) for every iteration of the out loop.

```
for (int i = 1;i<=5;i++)
{
    for(char ch ='a';ch<='c';ch++)
    {
        printf("%d%c",i,ch);
    }
    printf("\n");
}
```

Output

```
1a1b1c
2a2b2c
3a3b3c
4a4b4c
5a5b5c
```

Pattern 1

Write a C program to print first n rows of the following pattern.

```
*****
*****
*****
.
.
*****
```

Pattern 1-Solution

```
int main()
{
    int n;
    scanf("%d",&n);
    for (int i=1;i<=n;i++)
    {
        for(int j=1;j<=5;j++)
        {
            printf("*");
        }
      printf("\n");
    }
}
```

Pattern 2

Write a C program to print first n rows of the following pattern.

12345

12345

12345

.

.

12345

Pattern 2-Solution

```
int main()
{
    int n;
    scanf("%d",&n);
    for (int i=1;i<=n;i++)
    {
        for(int j=1;j<=5;j++)
        {
            printf("%d",j);
        }
      printf("\n");
    }
}
```

Pattern 3

Write a C program to print first n rows of the following pattern.

```
*
**
***
.
.
****...*
```

Pattern 3-Solution

```
int main()
{
    int n;
    scanf("%d",&n);
    for (int i=1;i<=n;i++)
    {
        for(int j=1;j<=i;j++)
        {
            printf("*");
        }
       printf("\n");
    }
}
```