CSE 06131223 ♦ CSE 06131224

Structured Programming

Lecture 10

Decision Making and Looping in C

Prepared by

Md. Mijanur Rahman, Prof. Dr.
Dept. of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University, Bangladesh
www.mijanrahman.com

Contents

DECISION MAKING AND LOOPING IN C

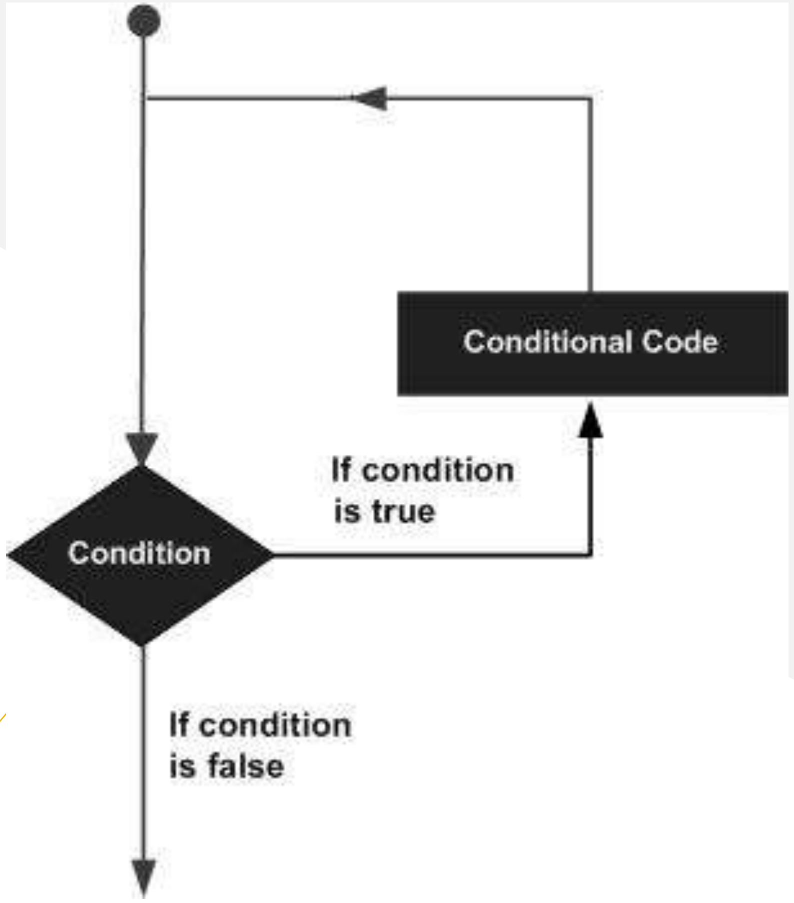
- Conditional Control Structures
- Decision making and looping (Iteration) statement
    - For loop
    - While loop
    - Do…While loop

Conditional Control Structures

- **Control Structures** are statements that change the flow of a program to a different code segment based on certain conditions.
- The control structures are categorized into three major Conditional types; they are:
  - Decision making and branching statements
    - Selection statements
    - Jump statements
  - **Decision making and looping (Iteration) statements**
    - **For loop**
    - **While loop**
    - **Do…While loop**

Looping (Iteration) Statements

- **A loop statement allows us to execute a statement or group of statements multiple times.**
- Given below is the general form of a loop statement in most of the programming languages.

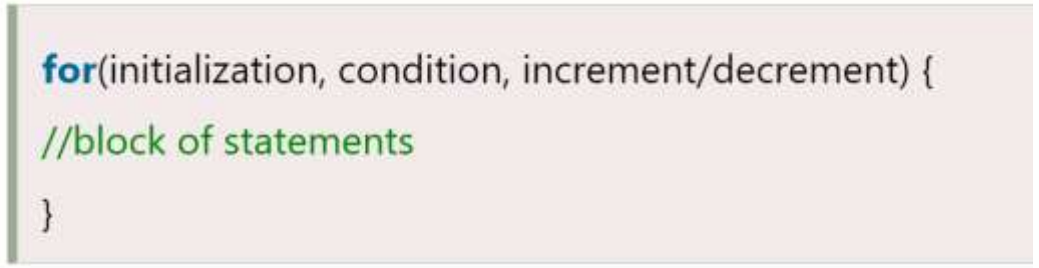


Looping (Iteration) Statements

- C programming language provides the following types of loops to handle looping requirements.
  - **For loop:** Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
  - **While loop:** Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
  - **Do…While loop:** It is more like a while statement, except that it tests the condition at the end of the loop body.
  - **Nested loops:** You can use one or more loops inside any other while, for, or do..while loop.

For Loop

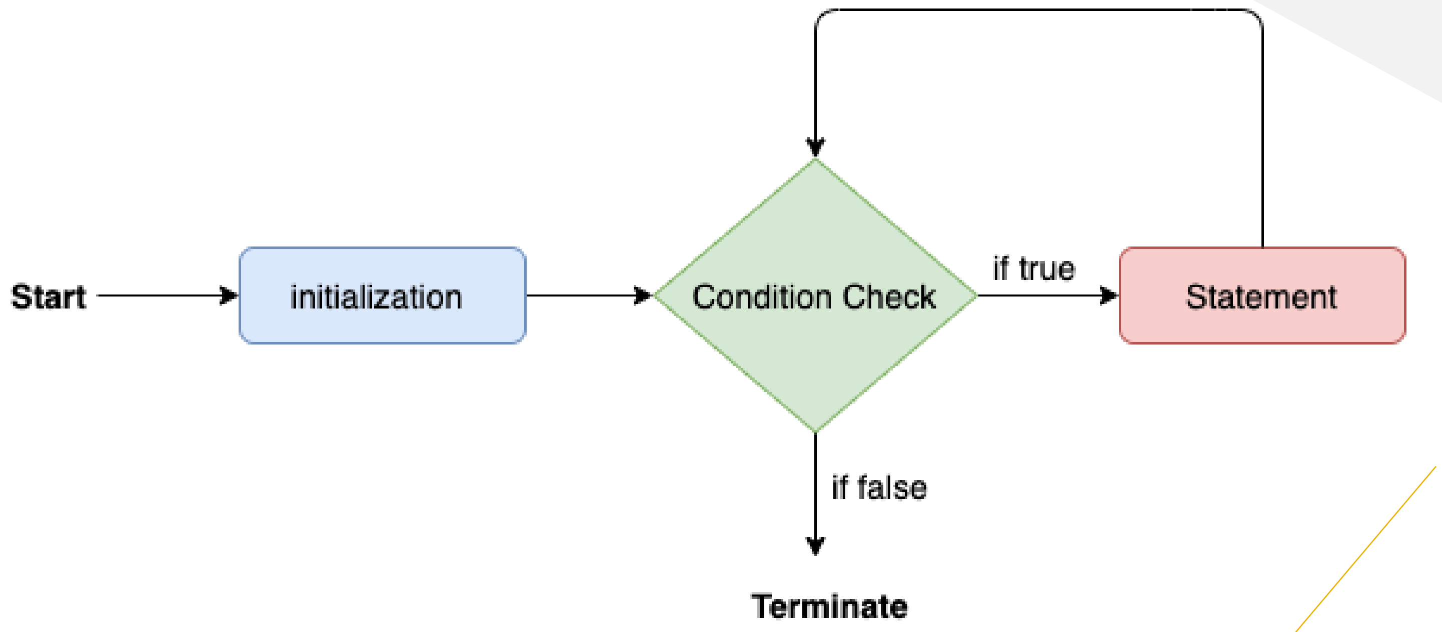
- The **For Loop Statement** is an iteration statement that executes a set of code repeatedly, given the initial value, the condition, increment value.
- It enables us to initialize the loop variable, check the condition, and increment/decrement in a single line of code. We use the for loop only when we exactly know the number of times, we want to execute the block of code.
- **Syntax:**



```
for(initialization, condition, increment/decrement) {
//block of statements
}
```

For Loop

- The flow chart for the for-loop is given below.



For Loop

- **Example:**

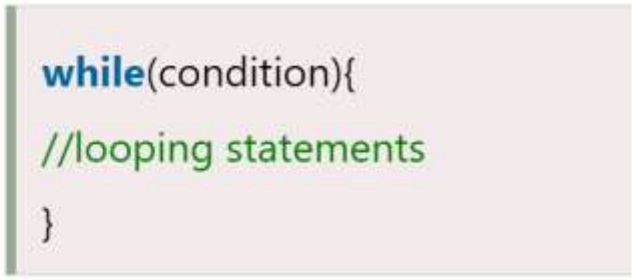
```
int main()
{
    int sum = 0;
    for(int j = 1; j<=10; j++) {
            sum = sum + j;
    }
printf("The sum of first 10 natural numbers is %d“, sum);
return 0;
}
```

- **Output:**

```
    The sum of first 10 natural numbers is 55
```

While Loop

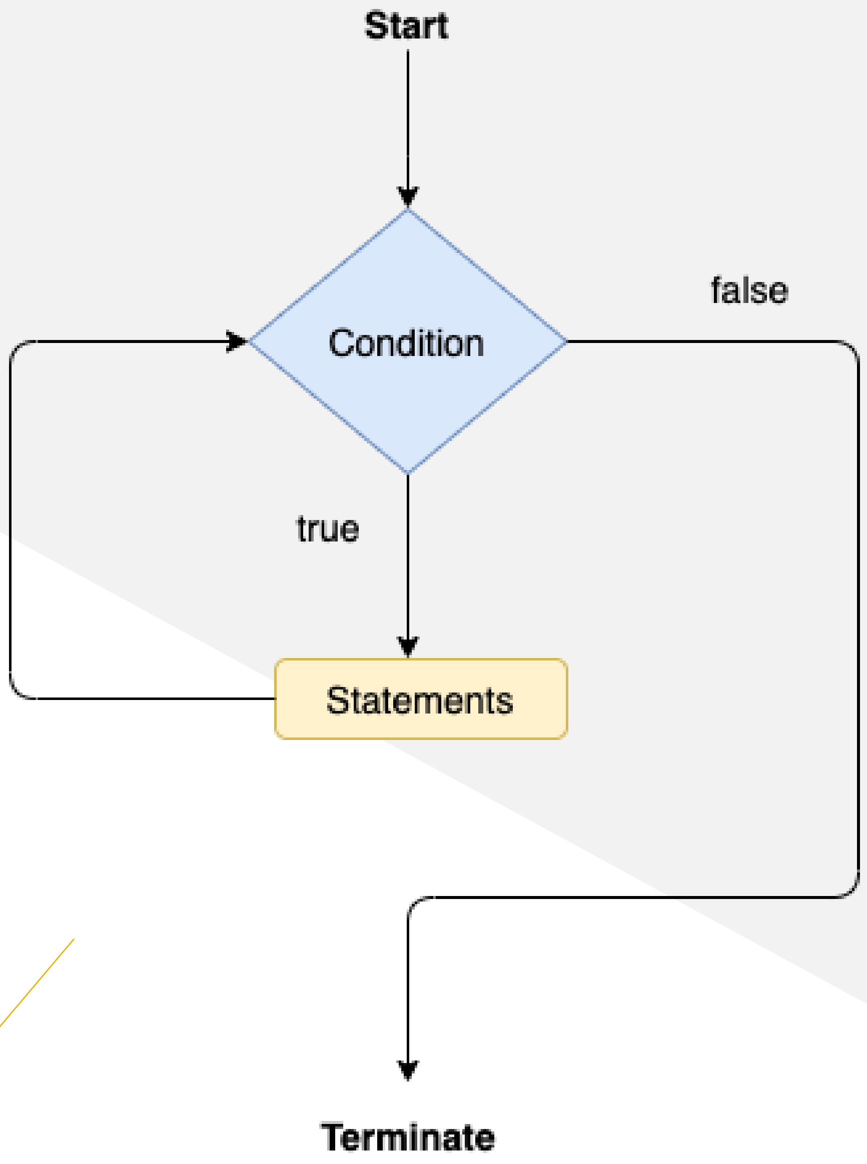
- **while** statement is an iteration structure that executes a set of code only when the condition is true. It is also known as "Entry controlled loop" since it is executed only inside the loop if the condition is satisfied. It ends with a ";" after the closing braces.
- **Syntax:** The syntax of the while loop is given below.



```
while(condition){
//looping statements
}
```

While Loop

- The flow chart for the while loop:



While Loop

- Example:

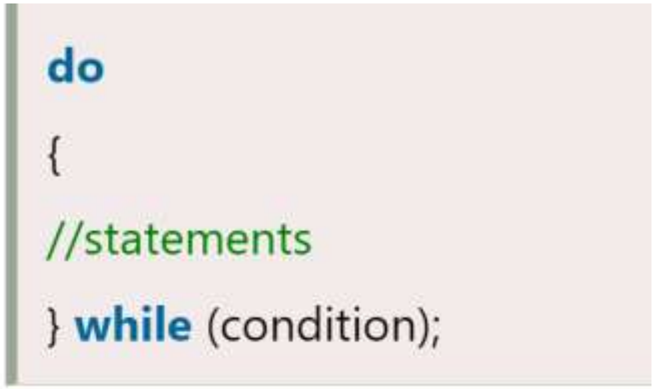
```
int i = 0;
printf("Printing the list of even numbers (<=10): ");
while(i<=10) {
        printf("%4d", i);
        i = i + 2;
}
```

- **Output:**

```
Printing the list of first 10 even numbers: 0 2 4 6 8 10
```

Do...While Loop

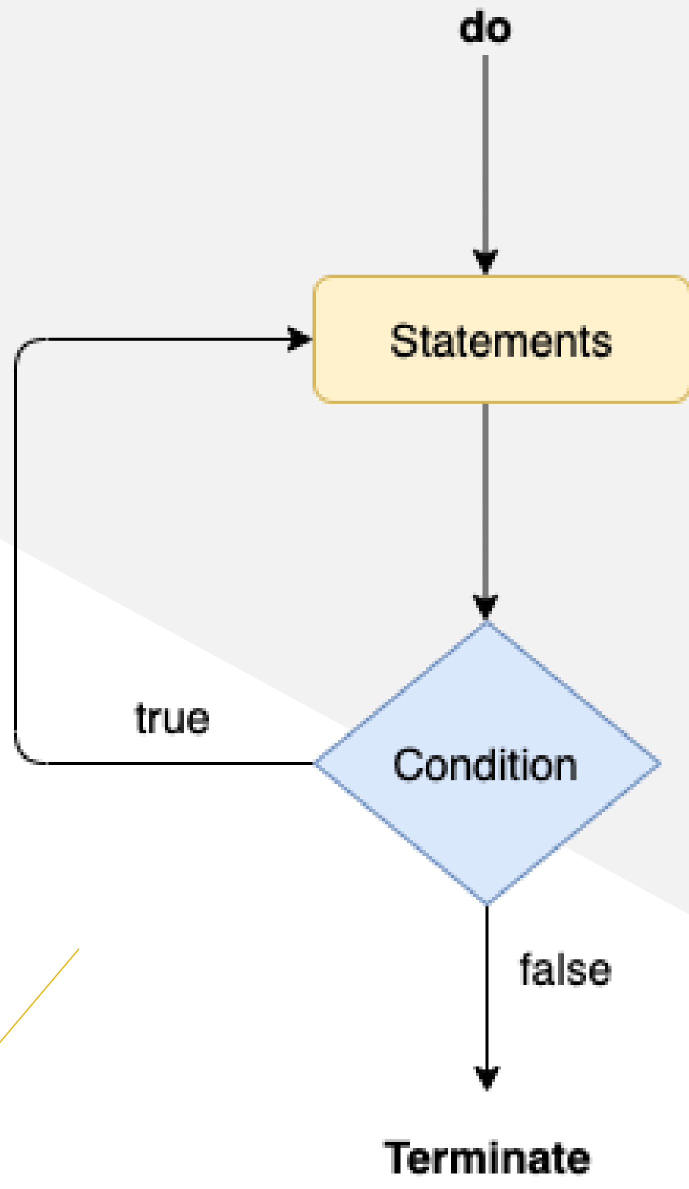
- The **Do While** statement is an iteration statement that executes a set of code first, then it checks the condition. This statement is also known as **"Exit controlled loop"** since the condition is checked only after executing the loop at least once.
- **Syntax:** The syntax of the do...while loop is given below.



```
do
{
//statements
} while (condition);
```

Do…While Loop

- The flow chart of the do-while loop:



Do…While Loop

- **Example:**

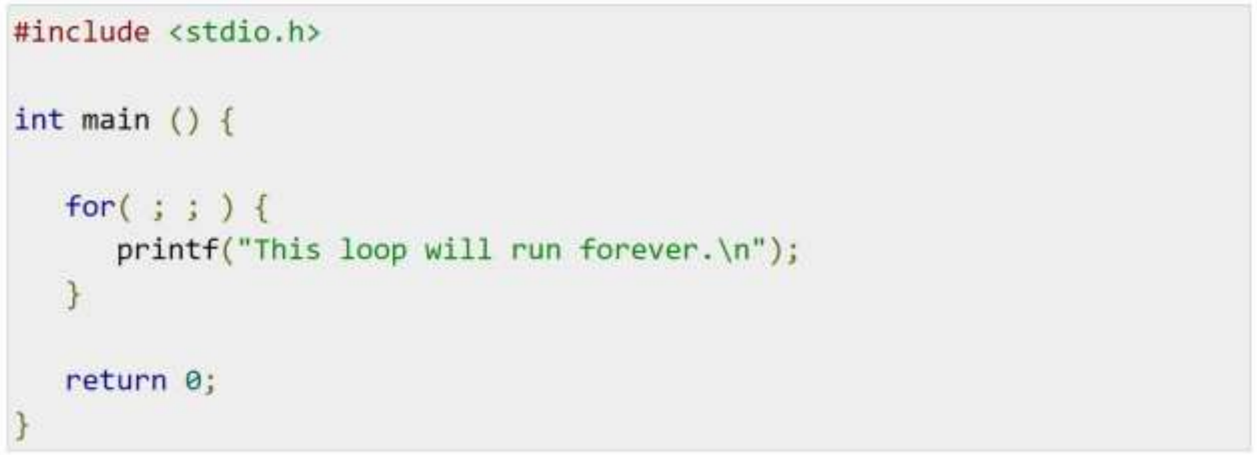
```
int i = 0;
printf("Printing the list of even numbers (<=10): ");
do {
        printf(“%4d”, i);
        i = i + 2;
}  while(i<=10);
```

- **Output:**

```
        Printing the list of first 10 even numbers: 0 2 4 6 8 10
```

The Infinite Loop

- A loop becomes an infinite loop if a condition never becomes false. The **for** loop is traditionally used for this purpose. Since none of the three expressions that form the 'for' loop are required, you can make an endless loop by leaving the conditional expression empty.

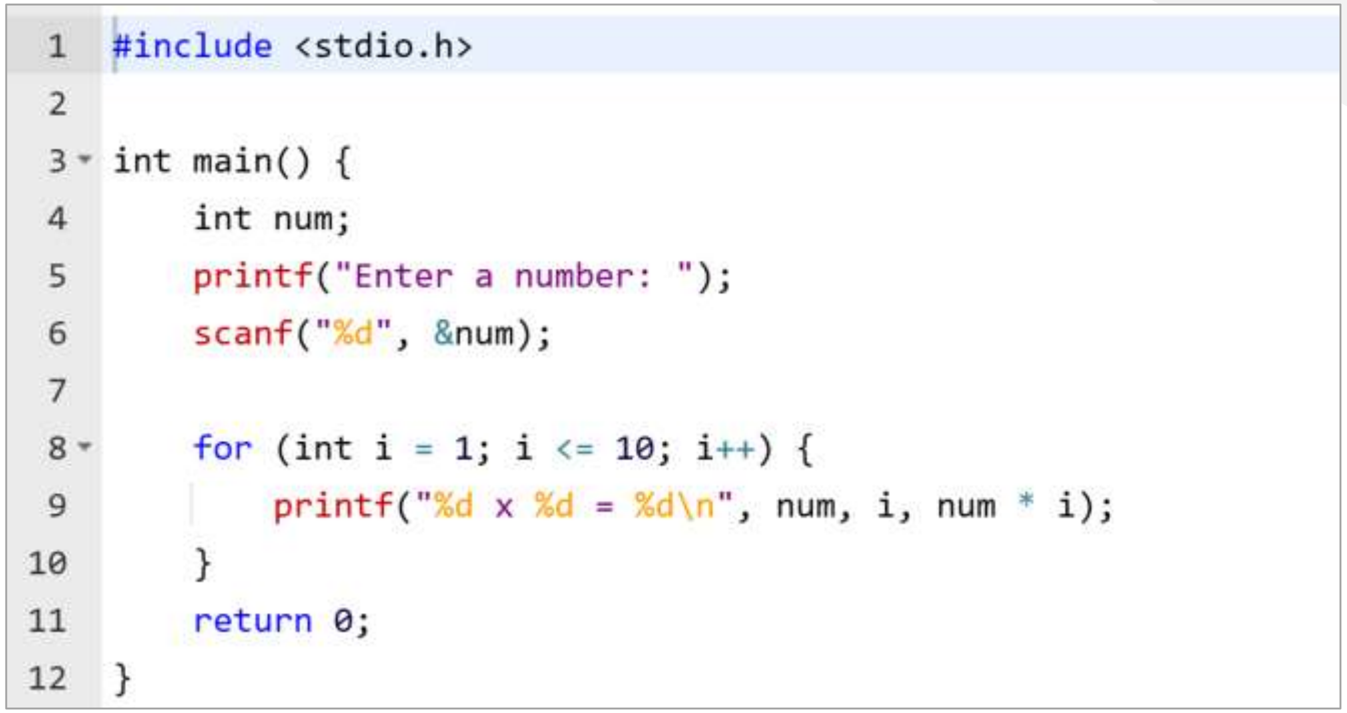


```
#include <stdio.h>

int main () {

   for( ; ; ) {
      printf("This loop will run forever.\n");
   }

   return 0;
}
```

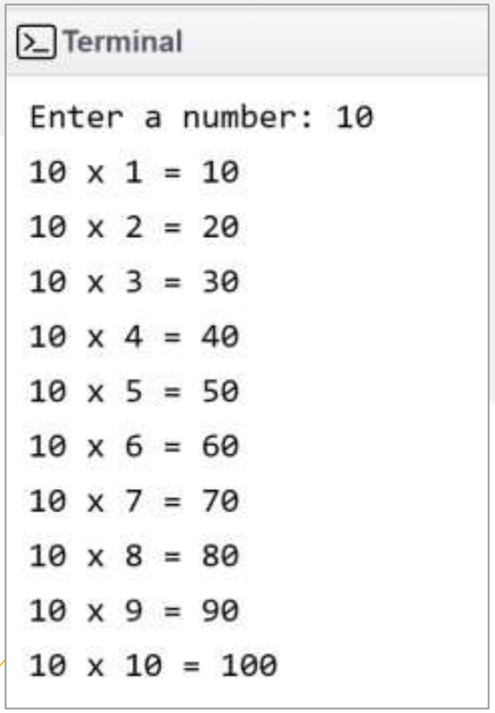
- When the conditional expression is absent, it is assumed to be true. You may have an initialization and increment expression, but C programmers more commonly use the for(;;) construct to signify an infinite loop.
- **NOTE** – You can terminate an infinite loop by pressing Ctrl + C keys.

C Programs

- **Printing the multiplication table of a given number:**



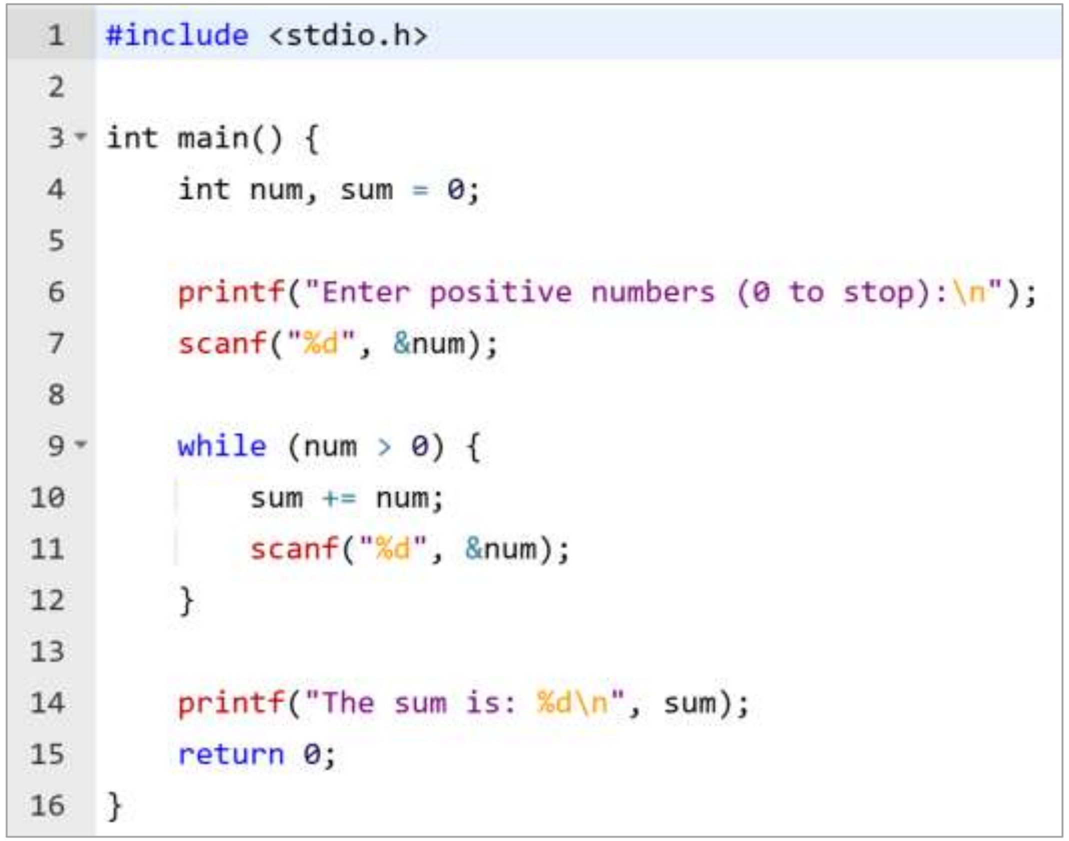
```c
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    for (int i = 1; i <= 10; i++) {
        printf("%d x %d = %d\n", num, i, num * i);
    }
    return 0;
}
```



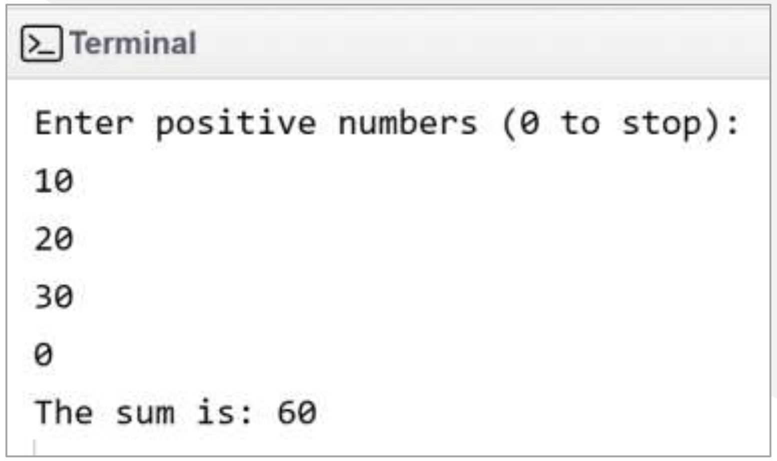
```
Enter a number: 10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

C Programs

- **Reading and summing positive numbers:**



```c
#include <stdio.h>

int main() {
    int num, sum = 0;

    printf("Enter positive numbers (0 to stop):\n");
    scanf("%d", &num);

    while (num > 0) {
        sum += num;
        scanf("%d", &num);
    }

    printf("The sum is: %d\n", sum);
    return 0;
}
```

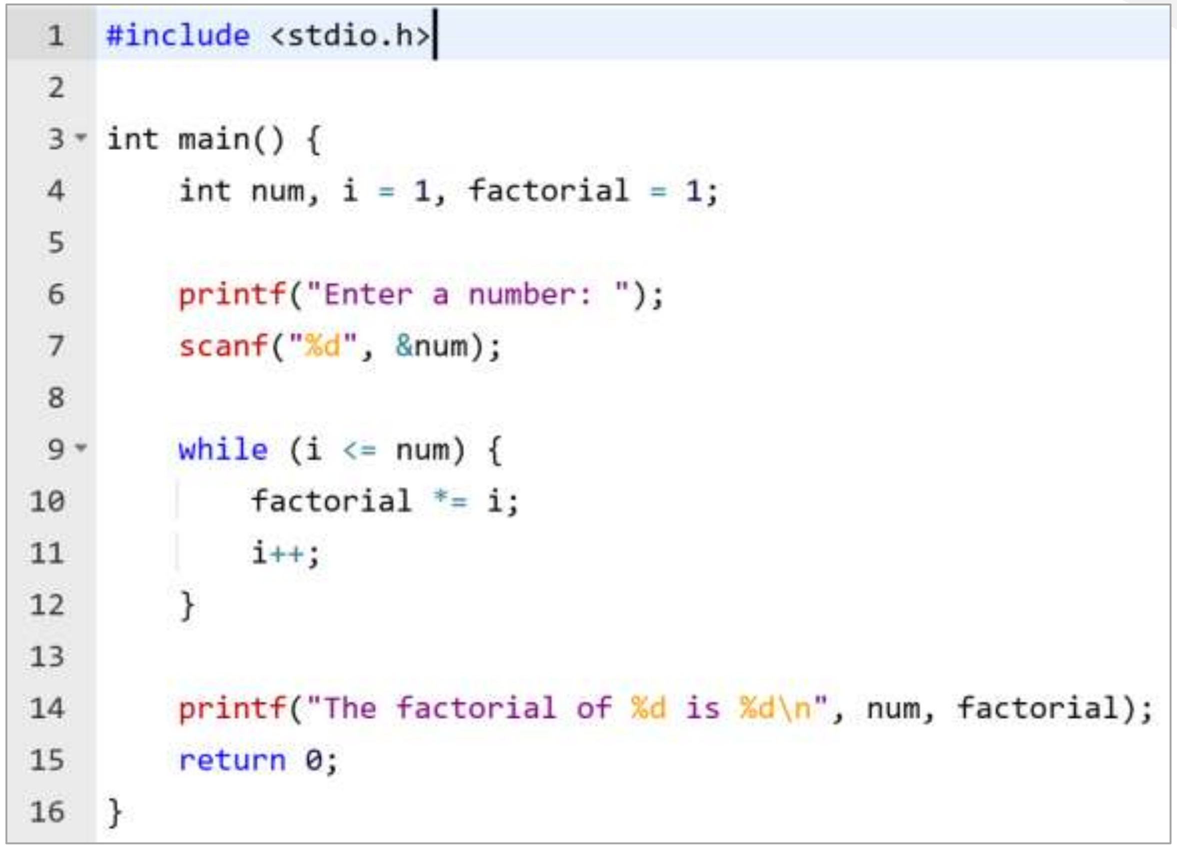


Terminal

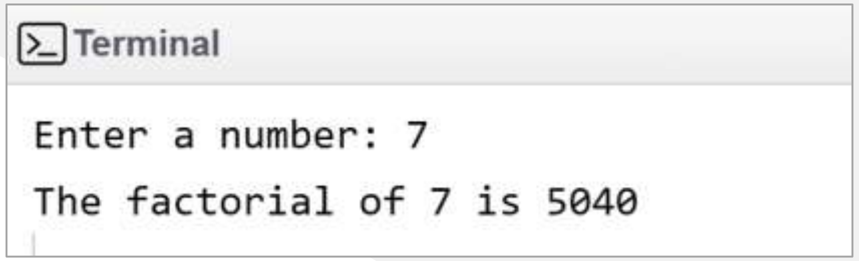
```
Enter positive numbers (0 to stop):
10
20
30
0
The sum is: 60
```

C Programs

- **Finding the factorial of a number:**



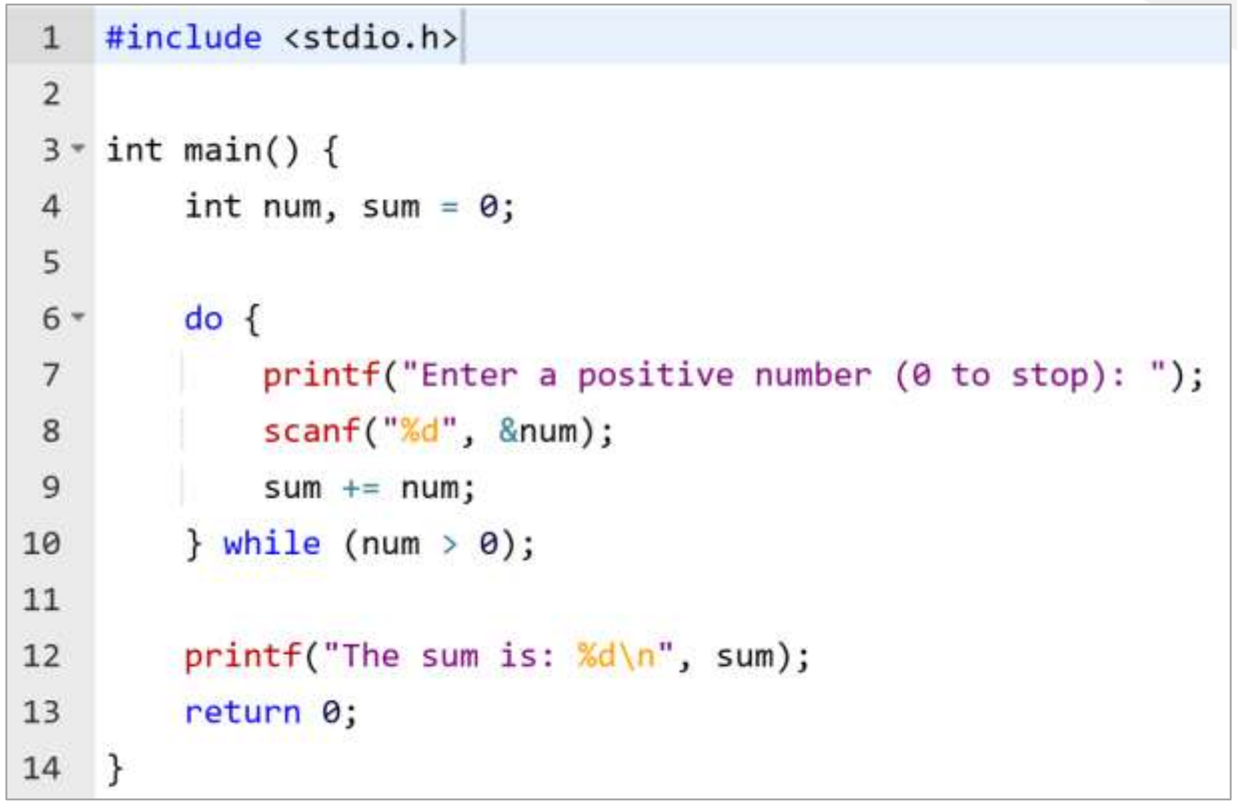
```c
#include <stdio.h>

int main() {
    int num, i = 1, factorial = 1;

    printf("Enter a number: ");
    scanf("%d", &num);

    while (i <= num) {
        factorial *= i;
        i++;
    }

    printf("The factorial of %d is %d\n", num, factorial);
    return 0;
}
```



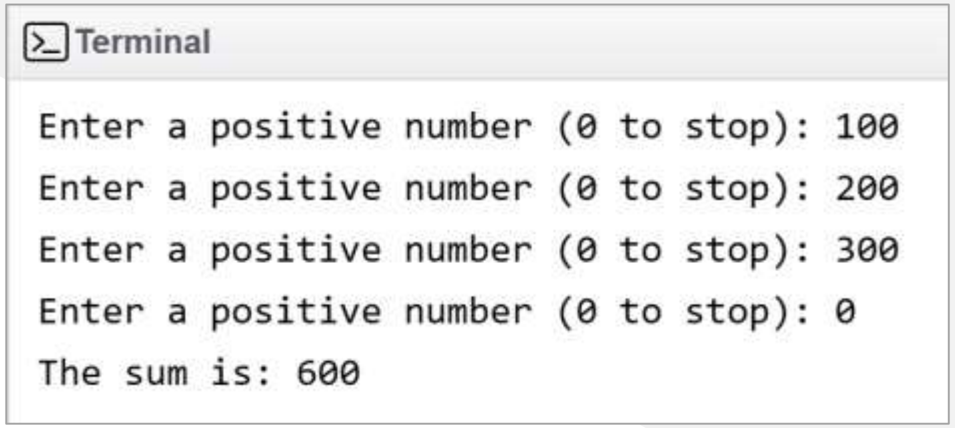
```
Enter a number: 7
The factorial of 7 is 5040
```

C Programs

- **Reading and summing positive numbers, ensuring at least one input:**



```c
#include <stdio.h>

int main() {
    int num, sum = 0;

    do {
        printf("Enter a positive number (0 to stop): ");
        scanf("%d", &num);
        sum += num;
    } while (num > 0);

    printf("The sum is: %d\n", sum);
    return 0;
}
```

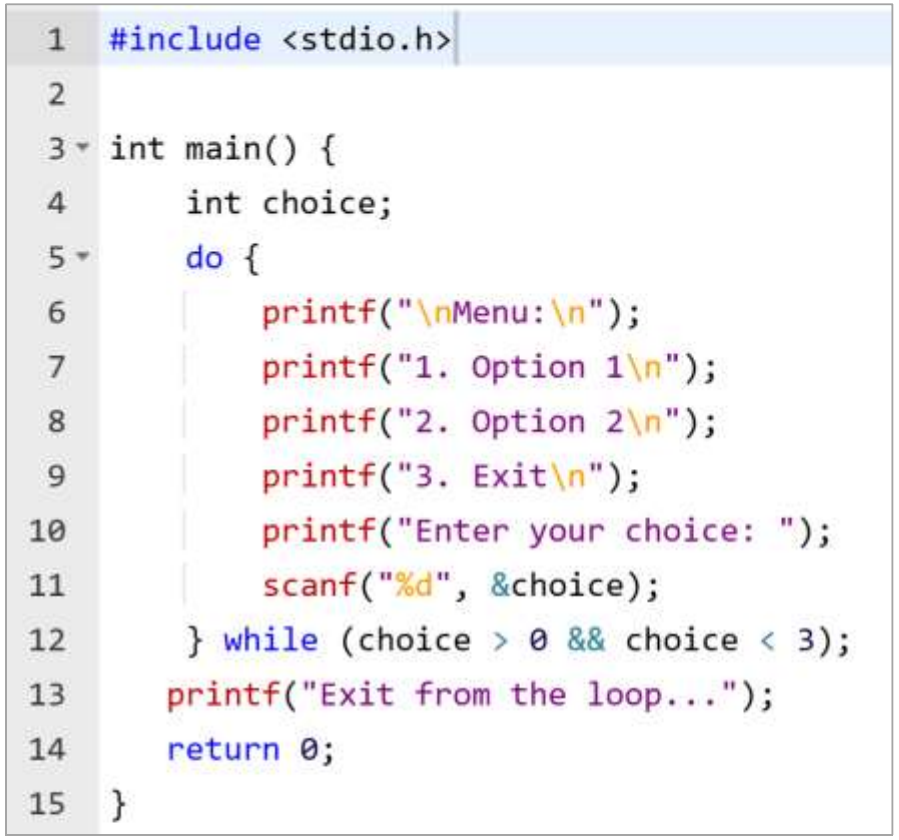


Terminal

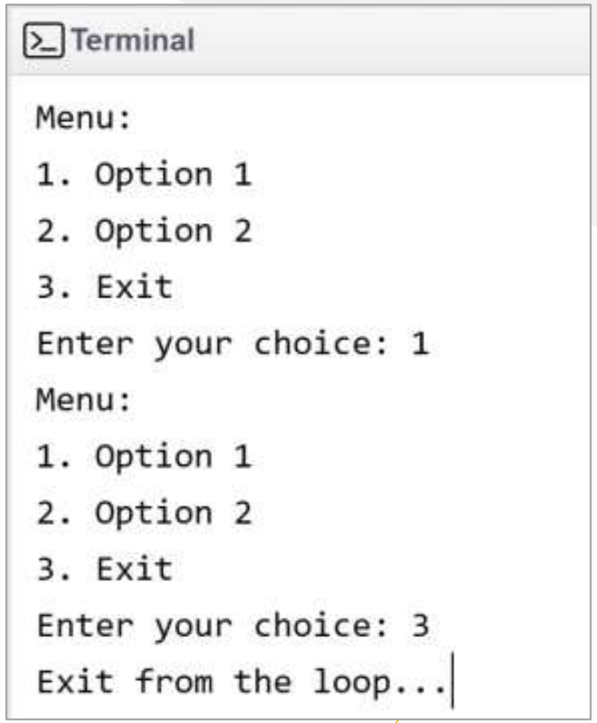
```
Enter a positive number (0 to stop): 100
Enter a positive number (0 to stop): 200
Enter a positive number (0 to stop): 300
Enter a positive number (0 to stop): 0
The sum is: 600
```

C Programs

- **Repeatedly displaying a menu until a valid choice is made:**



```c
#include <stdio.h>

int main() {
    int choice;
    do {
        printf("\nMenu:\n");
        printf("1. Option 1\n");
        printf("2. Option 2\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
    } while (choice > 0 && choice < 3);
   printf("Exit from the loop...");
   return 0;
}
```



```
Menu:
1. Option 1
2. Option 2
3. Exit
Enter your choice: 1
Menu:
1. Option 1
2. Option 2
3. Exit
Enter your choice: 3
Exit from the loop...
```

? THE END