CL1002 – Programming Fundamentals Lab

Lab # 07

Control Structure (Switch & For Loop)

Instructor: Engr. Muhammad Usman

Email: usman.rafiq@nu.edu.pk

Department of Computer Science,

National University of Computer and Emerging Sciences FAST Peshawar

nested-if in C/C++

A nested if in C is an if statement that is the target of another if statement. Nested if statements mean an if statement inside another if statement. Yes, both C and C++ allow us to nested if statements within if statements, i.e, we can place an if statement inside another if statement.

Syntax:

```
if (condition1)
{
  // Executes when condition1 is true
  if (condition2)
  {
     // Executes when condition2 is true
  }
}
```

Example 1:

```
// C program to illustrate nested-if statement
#include <stdio.h>
int main()
{
  int i = 10;
  if (i > 0) {
      // Nested - if statement
      // Will only be executed if statement above
      // is true
      if (i < 15)
          printf("i is smaller than 15\n");
      else
          printf("i is greater than 15\n");
  }
  else
     printf("i is smaller than 0\n");
  return 0;
}
```

Output

```
i is smaller than 15
```

C switch Statement

The switch statement allows us to execute one code block among many alternatives.

You can do the same thing with the if...else..if ladder. However, the syntax of the switch statement is much easier to read and write.

Syntax of switch...case

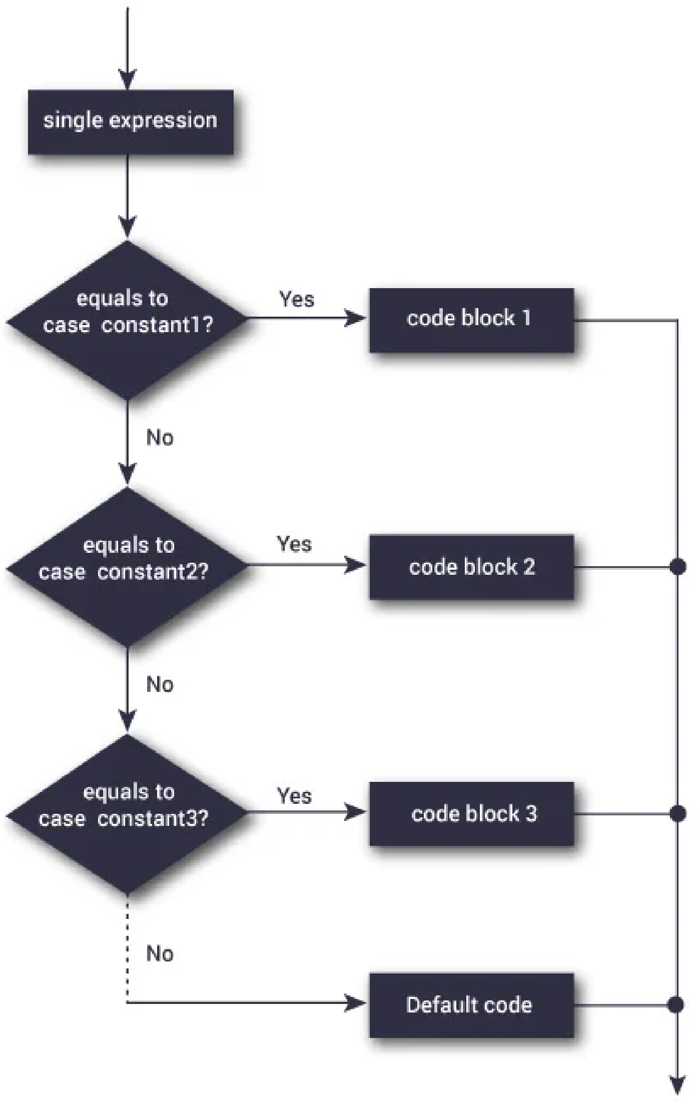
```
switch (expression)
{
   case constant1:
     // statements
     break;
   case constant2:
     // statements
     break;
   .
   .
   .
   default:
     // default statements
}
```

How does the switch statement work?

The expression is evaluated once and compared with the values of each case label.

- If there is a match, the corresponding statements after the matching label are executed. For example, if the value of the expression is equal to constant2, statements after case constant2: are executed until break is encountered.
- If there is no match, the default statements are executed.

switch Statement Flowchart



Example 2: Simple Calculator

```
// Program to create a simple calculator

#include <stdio.h>

int main() {

   char operation;

   int n1, n2;

   printf("Enter an operator (+, -, *, /): ");

   scanf("%c", &operation);
   printf("Enter two operands: ");

   scanf("%d %d",&n1, &n2);

   switch(operation)

   {

       case '+':

           printf("%d + %d = %.d",n1, n2, n1+n2);

           break;

       case '-':

           printf("%d - %d = %d",n1, n2, n1-n2);

           break;

       case '*':

           printf("%d * %d = %d",n1, n2, n1*n2);

           break;

       case '/':

           printf("%d / %d = %d",n1, n2, n1/n2);

           break;

       // operator doesn't match any case constant +, -, *, /

       default:

           printf("Error! operator is not correct");

   }

   return 0;

}
```

Output

```
Enter an operator (+, -, *, /): *

Enter two operands: 4 3
```

```
4 * 3 = 12
```

Control Structures (Repetition)

In programming, sometimes there is a need to perform some operation more than once or (say) n number of times. Loops come into use when we need to repeatedly execute a block of statements.
For example: Suppose we want to print “Hello World” 10 times. This can be done in two ways as shown below:

Manual Method

Manually we have to write printf for the C statement 10 times. Let’s say you have to write it 20 times (it would surely take more time to write 20 statements) now imagine you have to write it 100 times, it would be really hectic to re-write the same statement again and again. So, here loops have their role.

```
// C program to Demonstrate the need of loops
#include <stdio.h>
int main()
{
   printf("Hello World\n");
   printf("Hello World\n");
   printf("Hello World\n");
   printf("Hello World\n");
   printf("Hello World\n");
   return 0;
}
```

Using Loops

In Loop, the statement needs to be written only once and the loop will be executed 10 times as shown below.
In computer programming, a loop is a sequence of instructions that is repeated until a certain condition is reached.

```
// C program to Demonstrate the need of loops
#include <stdio.h>
int main()
{
   for (int i=0; i<5; i++){
       printf("Hello World\n");
   }
```

```
    return 0;
}
```

Advantage of loops in C

- It provides code reusability.
- Using loops, we do not need to write the same code again and again.
- Using loops, we can traverse over the elements of data structures (array or linked lists).

Types of C Loops

C programming has three types of loops:

1. for loop
2. while loop
3. do...while loop

for Loop

The for loop is used in the case where we need to execute some part of the code until the given condition is satisfied. The for loop is also called as a per-tested loop. It is better to use for loop if the number of iteration is known in advance.

The syntax of the for loop is:

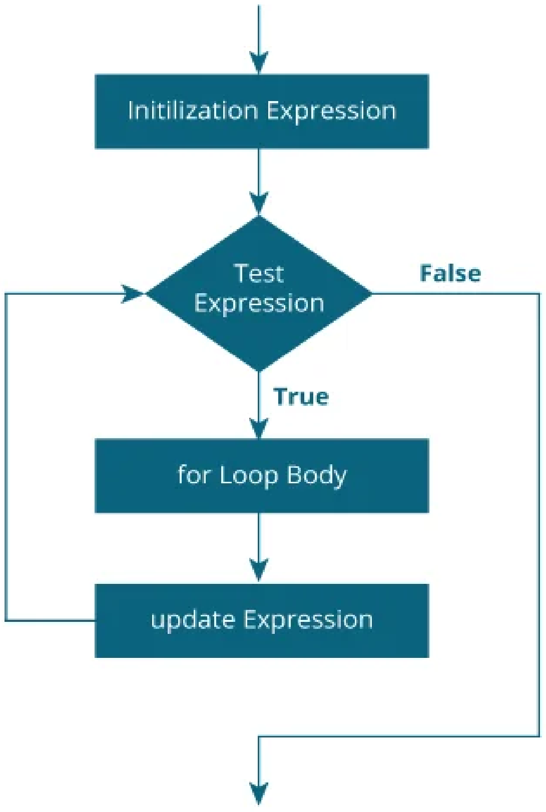
```
for (initializationStatement; testExpression; updateStatement)
{
    // statements inside the body of loop
}
```

How for loop works?

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of the for loop are executed, and the update expression is updated.
- Again the test expression is evaluated.

This process goes on until the test expression is false. When the test expression is false, the loop terminates.

for loop Flowchart



Example 3

```
// Print numbers from 1 to 10
#include <stdio.h>
int main() {
 int i;
 for (i = 1; i < 11; ++i)
 {
   printf("%d ", i);
 }
 return 0;
}
```

Output

```
1 2 3 4 5 6 7 8 9 10
```

1. i is initialized to 1.
2. The test expression i < 11 is evaluated. Since 1 less than 11 is true, the body of for loop is executed. This will print the 1 (value of i) on the screen.
3. The update statement ++i is executed. Now, the value of i will be 2. Again, the test expression is evaluated to true, and the body of for loop is executed. This will print 2 (value of i) on the screen.
4. Again, the update statement ++i is executed and the test expression i < 11 is evaluated. This process goes on until i becomes 11.
5. When i becomes 11, i < 11 will be false, and the for loop terminates.

Example 4

```c
// Program to calculate the sum of first n natural numbers
// Positive integers 1,2,3...n are known as natural numbers
#include <stdio.h>
int main()
{
   int num, count, sum = 0;
   printf("Enter a positive integer: ");
   scanf("%d", &num);

   // for loop terminates when num is less than count
   for(count = 1; count <= num; ++count)
   {
       sum += count;
   }
   printf("Sum = %d", sum);

   return 0;
}
```

Output

```
Enter a positive integer: 5
Sum = 15
```

Recursion

A function that calls itself is known as a recursive function. And, this technique is known as recursion.

Any recursive definition has two parts:

1. Base
2. Recursion

Base: An initial simple definition which can not be expressed in terms of smaller version.

Recursion: The part of the definition which can be expressed in terms of smaller versions of itself.

Working of Recursion in C

```
void recurse()
{
... .. ...
recurse();
... .. ...
}
int main()
{
... .. ...
recurse();
... .. ...
}
```

The figure below shows how recursion works by calling itself over and over again.

How recursion works in C programming

The recursion continues until some condition is met.

To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and the other doesn't.

Example 5

```
// Program to calculate the sum of first n natural numbers using
recursion
```

```
// Positive integers 1,2,3...n are known as natural numbers
#include <stdio.h>
int sumtill_n(int n){
   if(n<1)
   return 0;
   else
   return n+sumtill_n(n-1);
}
int main()
{
  int n,sum=0;
  printf("Enter a positive integer: ");
  scanf("%d",&n);
  sum =  sumtill_n(n);

  printf("Sum: %d\n",sum);

return 0;
}
```

Output

```
Enter a positive integer: 5
Sum = 15
```

References:

https://www.geeksforgeeks.org/cpp-loops/

https://www.programiz.com/c-programming/c-for-loop

https://www.programiz.com/c-programming/c-switch-case-statement