Programming Fundamentals Lab

Lab # 09

Nested if-else, Switch, Type Casting, Recursion

Instructor: Fariba Laiq

Email: fariba.laiq@nu.edu.pk

Course Code: CL1002

Semester Fall 2022

Department of Computer Science,
National University of Computer and Emerging Sciences FAST Peshawar Campus

Contents

Nested-if in C

Nested if statements mean an if statement inside another if statement.

Syntax:

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

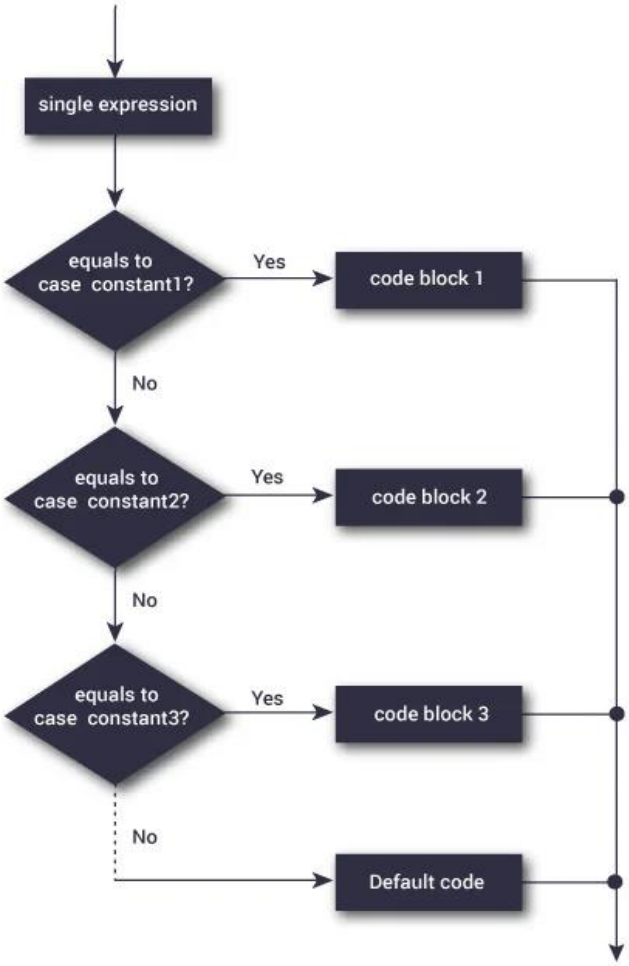
```
switch (expression)
{
   case constant1:
     // statements
     break;
   case constant2:
     // statements
     break;
   .
   .
   default:
     // default statements
}
```

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
```

Type Casting:

Type Casting is basically a process in C in which we change a variable belonging to one data type to another one. In type casting, the compiler automatically changes one data type to another one depending on what we want the program to do. For instance, in case we assign a float variable (floating point) with an integer (int) value, the compiler will ultimately convert this int value into the float value. The process of casting allows the programmers to make such types of conversions explicit or even force it in cases where it wouldn't happen normally.

There are two types of type conversion:

Implicit Type Conversion

Also known as 'automatic type conversion'.

- Done by the compiler on its own, without any external trigger from the user.

- Generally takes place when in an expression more than one data type is present. In such condition type conversion (type promotion) takes place to avoid loss of data.
- All the data types of the variables are upgraded to the data type of the variable with largest data type.

```
#include <stdio.h>

int main() {

   int  i = 17;
   char c = 'c'; /* ascii value is 99 */
   float sum;

   sum = i + c;
   printf("Value of sum : %f\n", sum );
}
```

Explicit Type Conversion

This process is also called type casting and it is user defined. Here the user can type cast the result to make it of a particular data type.

The syntax in C:

(type) expression

```
#include<stdio.h>
int main()
{
	double x = 1.2;
	// Explicit conversion from double to int
	int sum = (int)x + 1;
	printf("sum = %d", sum);
	return 0;
}
```

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 Case
2. Recursion/Function call to itself

Base Case: An initial simple definition which can not be expressed in terms of smaller version and we know the answer for that case.

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

```
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
#include <stdio.h>
int factorial(int n){
  if(n==1)
  return 1;
```

```
  else
  return n*factorial(n-1);
}
int main()
{
 int n, fact;
 printf("Enter a positive integer: ");
 scanf("%d",&n);
 fact=factorial(n);
 printf("Fact: %d\n",fact);

return 0;
}
```

Output:

Enter a positive integer: 5

Fact: 120

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