BIT Notes

By Karna Bahadur Shrestha

Control Structure

In most of the C programs we have encountered so far, the instructions that were executed **in the same order in which they appeared within the program (sequentially).**

Each instruction was executed one and only once. Program of this type are very simple, since they do not include any logical control structures.

While programming, we often encounter a situation in which **decisions have to be made or repetitions are required.**

In realistic C programs, the program needs logical condition to determine **if certain conditions are true or false**, they do require the repeated execution of groups of statements and, they involve the execution of individual groups of statements on a selective basis.And all these operations can be carried out using the various control statements included in C.

In general, those statements which alters the **flow of execution of program are known as control statements**. C Language has following Control making statements:

Decision Making Statements
In realistic C program, it may require a logical test to be carried out at some particular point within the program.

And depending on the outcome of the logical test, several possible actions will be carried out, this is known as **branching or conditional branching statements**. Different decision making statements are:

- if statement
- if…..else statement
- if……else if statement
- Nested if …else statement
- switch statement

if Statement:

The simplest form of the control statement is the If statement. It is very frequently used in decision making and allowing the flow of program execution. The if structure has the following syntax

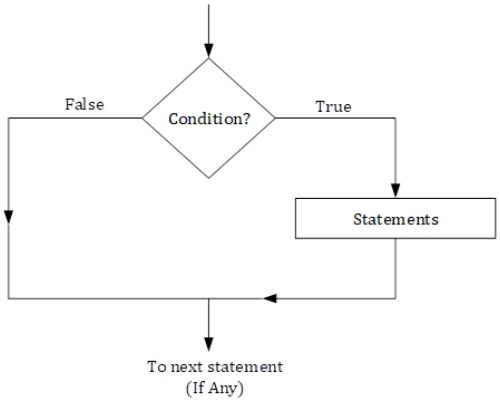
```
if (condition){

      statements;

      }
```

The statement is any valid C‟ language statement and the condition is any valid C‟ language expression, frequently logical operators are used in the condition statement. The condition part should not end with a semicolon, since the condition and statement should be put together as a single statement. The command says if the condition is true then performs the following statement or If the condition is fake the computer skips the statement and moves on to the next instruction in the program.

Example prograı




```
# include <stdio.h>
void main ()
{
    int numbers;
    printf ("Type a number:")
    scanf ("%d", &number);
    if (number < 0)
        number = -number;
    printf ("The absolute value is %d \n", number);
}
```

The above program checks the value of the input number to see if it is less than zero. If it is then the following program statement which negates the value of the number is executed. If the value of the number is not less than zero, we do not want to negate it then this statement is automatically skipped. The absolute number is then displayed by the program, and program execution ends.

The If else Statement:

The syntax of the If else construct is as follows:-

```
if (condition){
      statements;
   }
else{
      statements;
}
```

The if else is actually just on extension of the general format of if statement. If the result of the condition is true, then program statement 1 is executed, otherwise program statement 2 will be executed. If any case either program statement 1 is executed or program statement 2 is executed but not both when writing programs this else statement is so frequently required that almost all programming languages provide a special construct to handle this situation.

```
void main ()
{
                                          // start of the main
    int num;                          // declare variable num as integer
    printf ("Enter the number");      // message to the user
    scanf ("%d", &num);          // read the input number from keyboard
    if (num < 0)                 // check whether number is less than zero
```

```
                printf ("The number is negative"); // if it is less than zero then it is
                negative
        else                                        // else statement
                printf ("The number is positive")  // if it is more than zero then it is
                positive
}
```

In the above program the If statement checks whether the given number is less than 0. If it is less than zero then it is negative therefore the condition becomes true then the statement The number is negative is executed. If the number is not less than zero the If else construct skips the first statement and prints the second statement declaring that the number is positive.

Compound Relational tests:

C language provides the mechanisms necessary to perform compound relational tests. A compound relational test is simple one or more simple relational tests joined together by either the logical AND or the logical OR operators. These operators are represented by the character pairs && // respectively. The compound operators can be used to form complex expressions in C.

Syntax

a) if (condition1 && condition2 && condition3)
b) if (condition1 || condition2 || condition3)

The syntax in the statement „a" represents a complex if statement which combines different conditions using the and operator in this case if all the conditions are true only then the whole statement is considered to be true. Even if one condition is false the whole if statement is considered to be false.

The statement „b" uses the logical operator or (||) to group different expression to be checked. In this case if any one of the expression if found to be true the whole expression considered to be true, we can also uses the mixed expressions using logical operators and and or together.

Nested if Statement:

The if statement may itself contain another if statement is known as nested if statement.

***Syntax*:**

```
if (condition1)

    if (condition2)

        statement;
```

if statement may be nested as deeply as you need to nest it. One block of code will only be executed if two conditions are true. Condition 1 is tested first and then condition 2 is tested. The second if condition is nested in the first. The second if condition is tested only when the first condition is true else the program flow will skip to the corresponding else statement.

The IF ELSE If Ladder:

When a series of many conditions have to be checked we may use the ladder else if statement which takes the following general form.

```
if(condition1)
{
     statement1;
}
else if(condition2)
{
     statement2;
}
else if(condition3)
{
     statement3;
}
.
.
.
else if(condition n)
{
     statement n;
}
else
{
     default statement;
}
statement x;
```

This construct is known as if else construct or ladder. The conditions are evaluated from the top of the ladder to downwards. As soon on the true condition is found, the statement associated with it is executed and the control is transferred to the statement – x (skipping the rest of the ladder. When all the condition becomes false, the final else containing the default statement will be executed.

/* Example program using If else ladder to grade the student according to the following rules.

Grade	Grade
80 to 100	Distinction
60 to 79	First
50 to 69	Second
35 to 49	Pass
Below 35	Fail

```
#include<stdio.h>
void main ()
{
int marks;
printf ("Enter marks\n");
scanf ("%d", &marks);
if (marks <= 100 && marks >= 80)//check whether marks is less than 100 or greater than 80
     {
     printf ("\n Distinction");
     }
else if (marks >= 60)    //else if the previous condition fails Check
     {
     printf("\n First class");          }

else if (marks >= 50)    //else if marks is greater than 50 print
     {
     printf ("\n second class");
     }
else if (marks >= 35)    //else if marks is greater than 35 print
     {
     printf ("\n pass class");          }
else
printf ("Fail"); //If all condition fail apply default condition
print Fail


}
```

In the first If condition statement it checks whether the input value is lesser than 100 and greater than

80.If both conditions are true it prints distinction. Instead if the condition fails then the program control is transferred to the next if statement through the else statement and now it checks whether the next condition given is whether the marks value is greater than 60 If the condition is true it prints first class and comes out of the If else chain to the end of the program on the other hand if this condition also fails the control is transferred to next if statements program execution continues till the end of the loop and executes the default else statement fails and stops the program.

The Switch Statement

Unlike the If statement which allows a selection of two alternatives the switch statement allows a program to select one statement for execution out of a set of alternatives. During the execution of the switch statement only one of the possible statements will be executed the remaining statements will be skipped. The usage of multiple If else statement increases the complexity of the program since when the number of If else statements increase it affects the readability of the program and makes it difficult to follow the program.

The switch statement removes these disadvantages by using a simple and straight forward approach.

The general format of the Switch Statement is

```
switch (Expression)
{
     case Constant1: Statement1;
                     break;

     case Constant2: Statement2;
                     break;
     .
     .
     .
     case ConstantN: StatementN;
                     break;

     default:        Default Statement;
                     break;
}
```

When the switch statement is executed the control expression is evaluated first and the value is compared with the case label values in the given order. If the label matches with the value of the expression then the control is transferred directly to the group of statements which follow the label. If

none of the statements matches then the statement against the default is executed.The default statement is optional in switch statement in case if any default statement is not given and if none of the condition

matches then no action takes place in this case the control transfers to the next statement of the if else statement.

Example

```
#include <stdio.h>
void main ()
{
 int num1, num2, result;
 char operator;

 printf ("Enter two numbers");
 scanf ("%d %d", &num1, &num2);
 printf ("Enter an operator");
 scanf ("%c", &operator);

 switch (operator)
     {
          case '+':
                    result = num1 + num2;
                    break;
           case '-':
                    result = num1 - num2;
                     break;
          case '*':
                     result = num1 * num2;
                    break;
           case '/':
                     if (num2 != 0)
                          result = num1 / num2;
                     else
                     {
                          printf ("warning : division by
                          zero \n");
                          result = 0;
                     }
                     break;
          default:
                    printf ("\n unknown operator");
                    result = 0;
                    break;
     }
 printf ("%d", result);
}
```

In the above program the break statement is need after the case statement to break out of the loop and prevent the program from executing other cases.

The GOTO statement

The goto statement is simple statement used to transfer the program control unconditionally from one statement to another statement. Although it might not be essential to use the goto statement in a highly structured language like C, there may be occasions when the use of goto is desirable.

Syntax

a>	b>
goto label;	label:
……	……
……	……
……	……
Label:	goto label;
Statement;	

The goto requires a label in order to identify the place where the branch is to be made. A label is a valid variable name followed by a colon.

The label is placed immediately before the statement where the control is to be transformed. A program may contain several goto statements that transferred to the same place when a program. The label must be unique. Control can be transferred out of or within a compound statement, and control can be transferred to the beginning of a compound statement. However the control cannot be transferred into a compound statement. The goto statement is discouraged in C, because it alters the sequential flow of logic that is the characteristic of C language.