CONTROL STRUCTURES AND STATEMENTS IN C AND CPP

Control structures form the basic entities of a "**structured programming language**". We all know languages like C/C++ or Java are all structured programming languages. ***Control structures are used to alter the flow of execution of the program.*** Why do we need to alter the program flow ? The reason is "***decision making***"! In life, we may be given with a set of option like doing "Electronics" or "Computer science". We do make a decision by analyzing certain conditions (like our personal interest, scope of job opportunities etc). With the decision we make, we alter the flow of our life's direction. This is exactly what happens in a C/C++ program. We use control structures to make decisions and alter the direction of program flow in one or the other path(s) available.

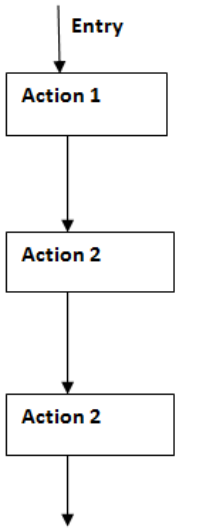
There are **three types** of control structures available in C and C++

1) Sequence structure (straight line paths)

2) Selection structure (one or many branches)

3)Loop structure (repetition of a set of activities)

All the 3 control structures and its flow of execution is represented in the flow charts given below.



Sequence Structure

Control statements in C/C++ to implement control structures

We have to keep in mind one important fact:- all program processes can be implemented with these 3 control structures only. That's why I wrote "***control structures are the basic entities of a structured programming language***". To implements these "control structures" in a C/C++ program, the language provides 'control statements'. So to implement a particular control structure in a programming language, we need to learn how to use the relevant control statements in that particular language.

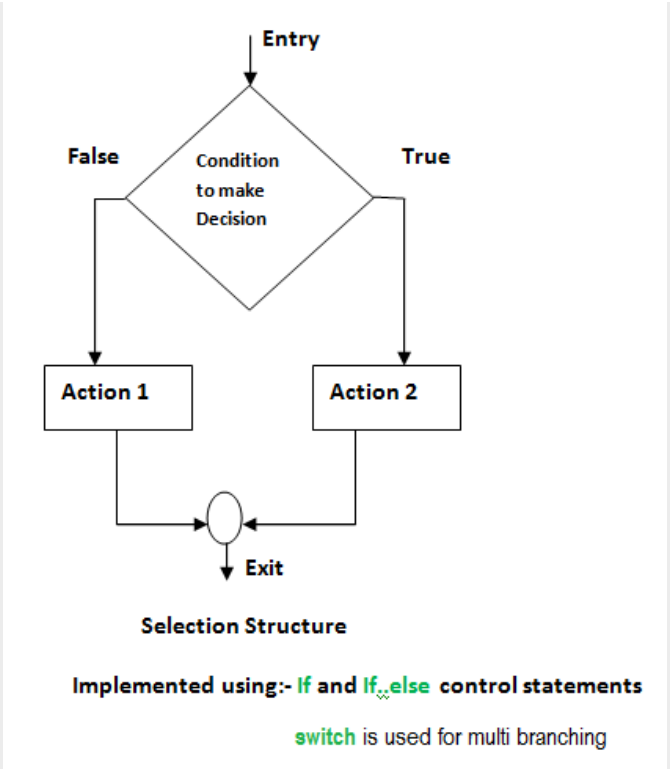
The control statements are:-

- **Switch**
- **If**
- **If Else**
- **While**
- **Do While**
- **For**

As shown in the flow charts:-

- Selection structures are implemented using **If** , **If Else** and **Switch** statements.
- Looping structures are implemented using **While**, **Do While** and **For** statements.

Selection structures



Selection structure

Selection structures are used to perform ‘**decision making**‘ and then branch the program flow based on the outcome of decision making. Selection structures are implemented in C/C++ with If, If Else and Switch statements. If and If Else statements are 2 way branching statements whereas Switch is a multi branching statement.

The simple If statement

The syntax format of a simple if statement is as shown below.

```
if (expression) // This expression is evaluated. If expression is TRUE
statements inside the braces will be executed
{
statement 1;
statement 2;
}
statement 1;// Program control is transfered directly to this line, if the
expression is FALSE
statement 2;
```

The expression given inside the brackets after **if** is evaluated first. If the expression is true, then statements inside the curly braces that follow if(expression) will be executed. If the expression is false, the statements inside curly braces will not be executed and program control goes directly to statements after curly braces.

Example program to demo “If” statement

Problem:-

A simple example program to demo the use of If, If Else and Switch is shown here. An integer value is collected from user.

If the integer entered by user is 1 – output on screen “UNITED STATES”. If the integer is 2 – output “SPAIN”, If the integer is 3 output “INDIA”. If the user enters some other value – output “WRONG ENTRY”.

Note:- The same problem is used to develop example programs for **“if else”** and **“switch**” statements

```
#include
void main()
{
int num;
printf("Hello user, Enter a number");
scanf("%d",&num); // Collects the number from user
if(num==1)
{
```

```
printf("UNITED STATES");
}
if(num==2)
{
printf("SPAIN");
}
if(num==3)
{
printf("INDIA");
}
}
```

The If Else statement.

Syntax format for If Else statement is shown below.

```
if(expression 1)// Expression 1 is evaluated. If TRUE, statements inside the curly braces are executed.
{ //If FALSE program control is transferred to immedate else if statement.
statement 1;
statement 2;
}
else if(expression 2)// If expression 1 is FALSE, expression 2 is evaluated.
{
statement 1;
statement 2;
}
else if(expression 3) // If expression 2 is FALSE, expression 3 is evaluated
{
statement 1;
statement 2;
}
else // If all expressions (1, 2 and 3) are FALSE, the statements that follow this else (inside curly braces) is executed.
{
statement 1;
statement 2;
}
other statements;
```

The execution begins by evaluation expression 1. If it is **TRUE**, then statements inside the immediate curly braces is evaluated. If it is **FALSE**, program control is transferred directly to immediate else if

statement. Here expression 2 is evaluated for TRUE or FALSE. The process continues. If all expressions inside the different if and else if statements are FALSE, then the last **else** statement (without any expression) is executed along with the statements 1 and 2 inside the curly braces of last **else** statement.

Example program to demo "If Else"

```
#include
void main()
{
int num;
printf("Hello user, Enter a number");
scanf("%d",&num); // Collects the number from user
if(num==1)
{
printf("UNITED STATES");
}
else if(num==2)
{
printf("SPAIN");
}
else if(num==3)
{
printf("INDIA");
}
else
{
printf("WRONG ENTRY"); // See how else is used to output "WRONG ENTRY"
}
}
```

Note:- Notice how the use of If Else statements made program writing easier. Compare this with above program using simple If statement only.

Switch statement

Switch is a multi branching control statement. **Syntax for switch statement is shown below.**

```
switch(expression)// Expression is evaluated. The outcome of the expression
should be an integer or a character constant
{
case value1: // case is the keyword used to match the integer/character
constant from expression.
```

```
//value1, value2 ... are different possible values that can come in
expression
statement 1;
statement 2;
break; // break is a keyword used to break the program control from switch
block.
case value2:
statement 1;
statement 2;
break;
default: // default is a keyword used to execute a set of statements inside
switch, if no case values match the expression value.
statement 1;
statement 2;
break;
}
```

Execution of switch statement begins by evaluating the expression inside the switch keyword brackets. The expression should be an integer (1, 2, 100, 57 etc) or a character constant like **'a'**, **'b'** etc. This expression's value is then matched with each case values. There can be any number of case values inside a switch statements block. If first case value is not matched with the expression value, program control moves to next case value and so on.**When a case value matches with expression value, the statements that belong to a particular case value are executed.**

Notice that last set of lines that begins with **default**. The word **default** is a **keyword in C/C++**. When used inside switch block, it is intended to execute a set of statements, if no case values matches with expression value. So if no case values are matched with expression value, the set of statements that follow **default:** will get executed.

Note: Notice the **break** statement used at the end of each case values set of statements. The word break is a**keyword in C/C++** used to break from a block of curly braces. The switch block has two curly braces { }. The keyword break causes program control to exit from switch block.

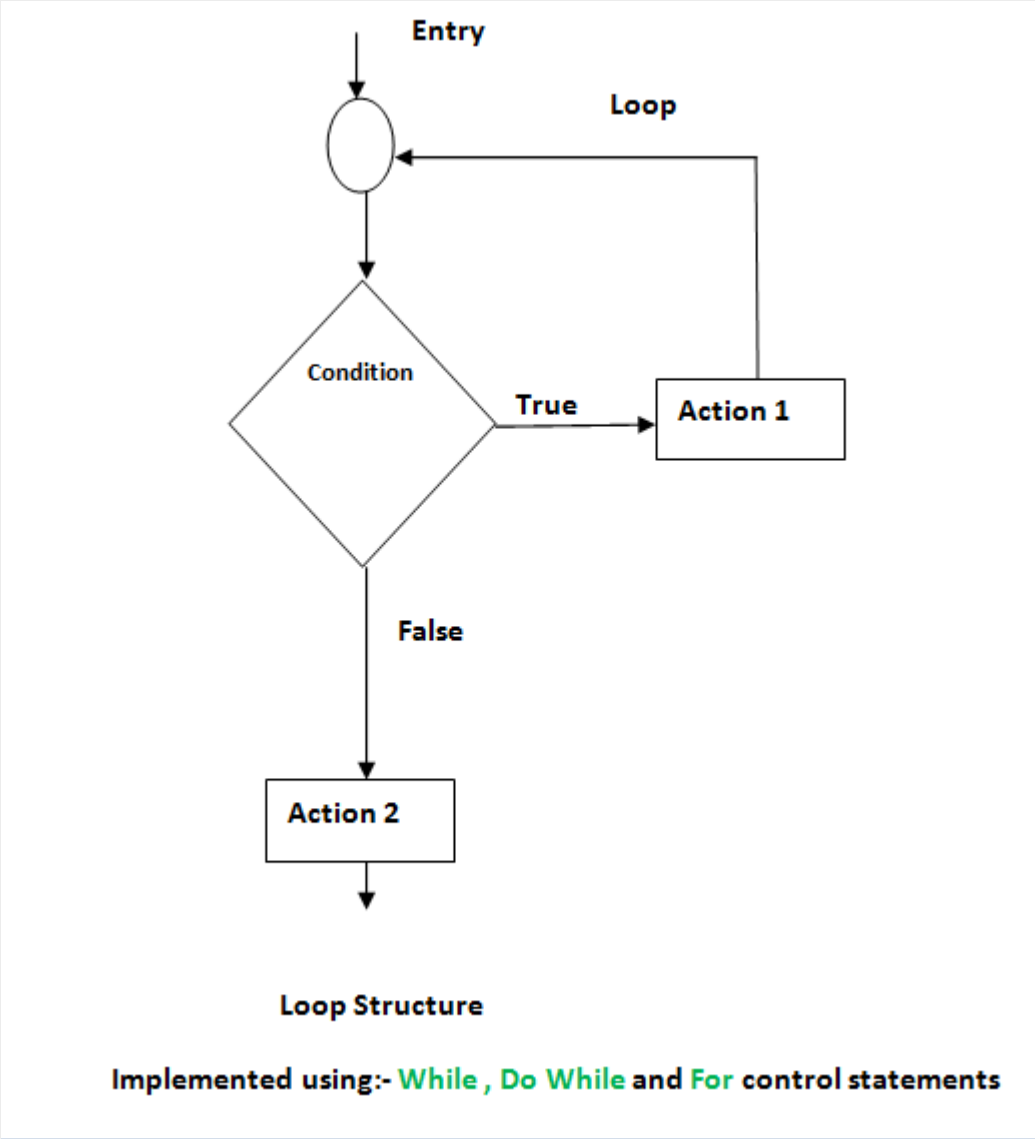
Example program to demo working of "switch"

```
#include
void main()
{
int num;
printf("Hello user, Enter a number");
scanf("%d",&num); // Collects the number from user
switch(num)
{
```

```
case 1:
printf("UNITED STATES");
break;
case 2:
printf("SPAIN");
break;
case 3:
printf("INDIA");
default:
printf("WRONG ENTRY");
}
}
```

Note:- Switch statement is used for multiple branching. The same can be implemented using **nested "If Else"**statements. But use of nested if else statements make program writing tedious and complex. Switch makes it much easier. Compare this program with above one.

Loop structures



Loop Structure

A loop structure is used to execute a certain set of actions for a predefined number of times or until a particular condition is satisfied. There are 3 control statements available in C/C++ to implement loop structures. **While, Do while and For statements.**

The while statement

Syntax for while loop is shown below:

```
while(condition)// This condition is tested for TRUE or FALSE. Statements
inside curly braces are executed as long as condition is TRUE
{
statement 1;
statement 2;
```

```
statement 3;
}
```

The condition is checked for TRUE first. If it is TRUE then all statements inside curly braces are executed.Then program control comes back to check the condition has changed or to check if it is still TRUE. The statements inside braces are executed repeatedly, as long as the condition is TRUE. When the condition turns FALSE, program control exits from while loop.

Note:- while is an entry controlled loop. Statement inside braces are allowed to execute only if condition inside while is TRUE.

Example program to demo working of "while loop"

An example program to collect a number from user and then print all numbers from zero to that particular collected number is shown below. That is, if user enters 10 as input, then numbers from 0 to 10 will be printed on screen.

Note:- The same problem is used to develop programs for do while and for loops

```
#include
void main()
{
int num;
int count=0; // count is initialized as zero to start printing from zero.
printf("Hello user, Enter a number");
scanf("%d",&num); // Collects the number from user
while(count<=num) // Checks the condition - if value of count has reached
value of num or not.
{
printf("%d",count);
count=count+1; // value of count is incremented by 1 to print next number.
}
}
```

The do while statement

Syntax for do while loop is shown below:

```
do
{
statement 1;
statement 2;
statement 3;
}
while(condition);
```

Unlike while, do while is an exit controlled loop. Here the set of statements inside braces are executed first. The condition inside while is checked only after finishing the first time execution of

statements inside braces. If the condition is TRUE, then statements are executed again. This process continues as long as condition is TRUE. Program control exits the loop once the condition turns FALSE.

Example program to demo working of "do while"

```
#include
void main()
{
int num;
int count=0; // count is initialized as zero to start printing from zero.
printf("Hello user, Enter a number");
scanf("%d",&num); // Collects the number from user
do
{
printf("%d",count); // Here value of count is printed for one time intially
and then only condition is checked.
count=count+1; // value of count is incremented by 1 to print next number.
}while(count<=num);
}
```

The for statement

Syntax of for statement is shown below:

```
for(initialization statements;test condition;iteration statements)
{
statement 1;
statement 2;
statement 3;
}
```

The for statement is an entry controlled loop. The difference between while and for is in the number of repetitions. The for loop is used when an action is to be executed for a predefined number of times. The while loop is used when the number of repetitions is not predefined.

Working of for loop:

The program control enters the for loop. At first it execute the statements given as initialization statements. Then the condition statement is evaluated. If conditions are TRUE, then the block of statements inside curly braces is executed. After executing curly brace statements fully, the control moves to the "iteration" statements. After executing iteration statements, control comes back to

condition statements. Condition statements are evaluated again for TRUE or FALSE. If TRUE the curly brace statements are executed. This process continues until the condition turns FALSE.

Note 1:- The statements given as "***initialization statements***" are executed only once, at the beginning of a for loop.

Note 2: There are 3 statements given to a for loop as shown. One for initialization purpose, other for condition testing and last one for iterating the loop. Each of these 3 statements are separated by semicolons.

Example program to demo working of "for loop"

```
#include
void main()
{
int num,count;
printf("Hello user, Enter a number");
scanf("%d",&num); // Collects the number from user
for(count=0;count<=num;count++)// count is initialized to zero inside for
statement. The condition is checked and statements are
executed.
{
printf("%d",count); // Values from 0 are printed.
}
}
```

Source : http://www.circuitstoday.com/control-structures-in-c-and-cpp