Problem Solving and Programming using C (PSPC)

Dr. M. Raja Roy
Associate Professor
Anil Neerukonda Institute of Technology and Sciences
www.anits.edu.in
www.mrrtechnical.co.in

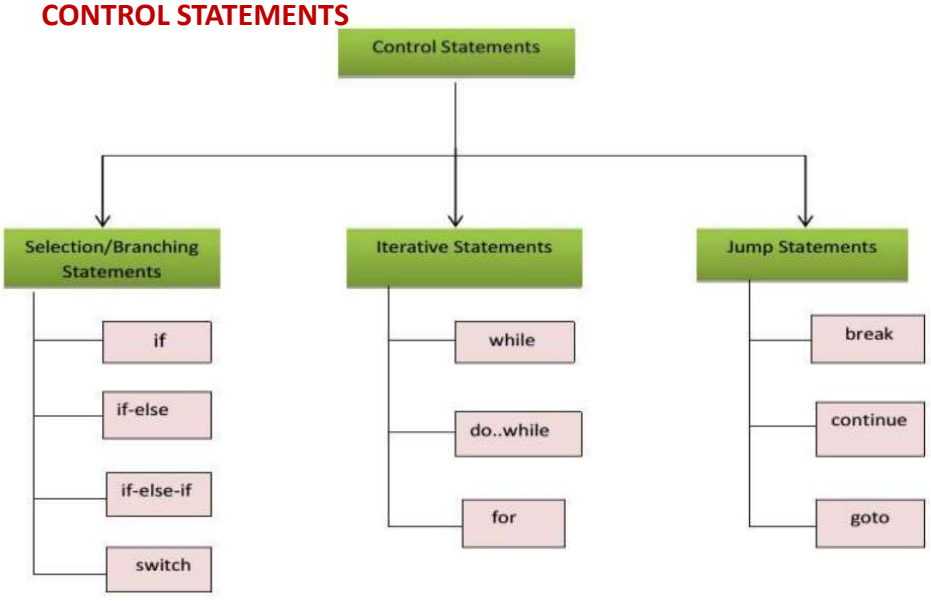
UNIT-II

CONTROL STATEMENTS

Control statements enable us to specify the order in which the various instructions in the program are to be executed.

They define how the control is transferred to other parts of the program. Control statements are classified in the following ways:

CONTROL STATEMENTS



a) If statement

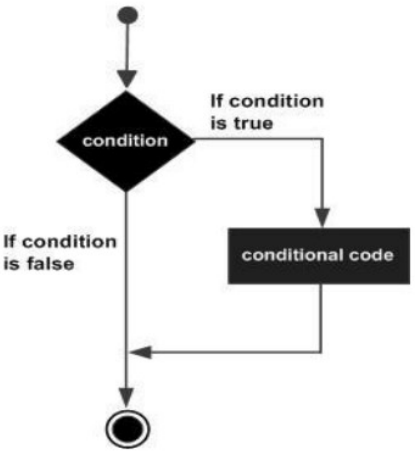
Syntax:

```
if(boolean_expression)
{
  /* statement(s) will execute if the Boolean expression is true */
}
```

If the Boolean expression evaluates to true then the block of code inside the if statement will be executed.

If boolean expression evaluates to false then the first set of code after the end of the if statement (after the closing curly brace) will be executed.

Flow Diagram:



Relational Operators - <, >, <=, >=, ==, !=

Logical Operators - and, or, not

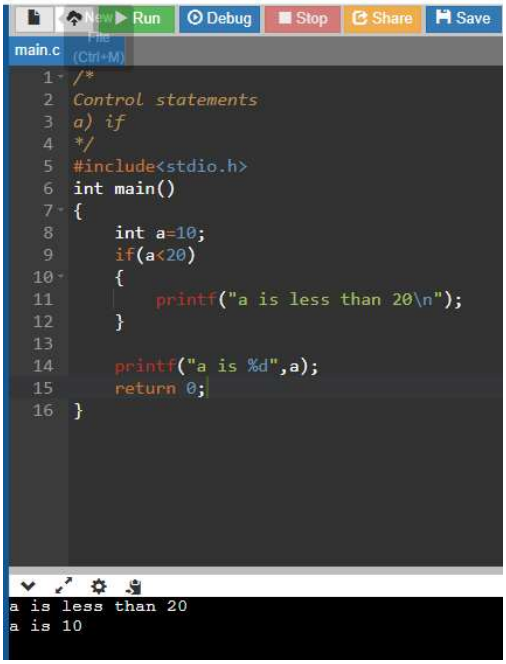
Example :

```
#include <stdio.h>
int main ()
{
 int a = 10;
 if( a < 20 )
 {
        printf("a is less than 20\n" );
 }
 printf("value of a is : %d\n", a);
 return 0;
}
```

When the code is compiled and executed, it produces following

a is less than 20
value of a is : 10

Example :



a) If-else statement

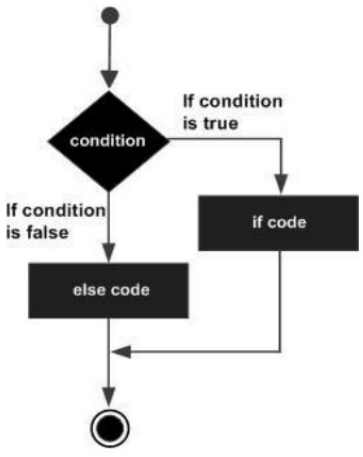
Syntax:

```
if(boolean_expression)
{
  /* statement(s) will execute if the Boolean expression is true */
 }
 else
{
  /* statement(s) will execute if the boolean expression is false */
}
```


a) If-else statement

If the Boolean expression evaluates to true then the if block of code will be executed otherwise else block of code will be executed.

Flow Diagram:



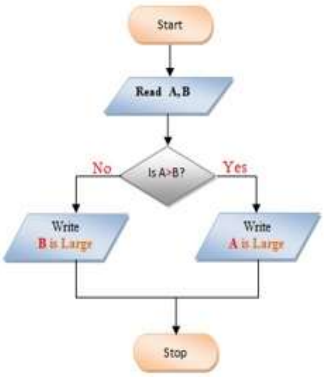
Example

Greatest of two numbers

Algorithm

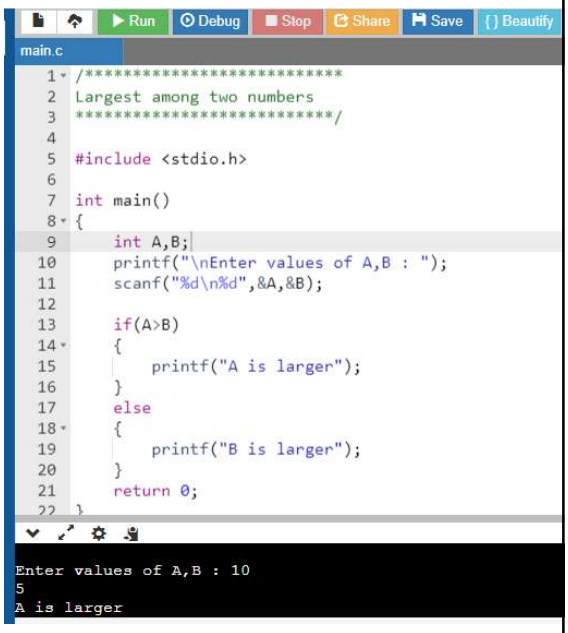
1. Start
2. Read A,B
3. If A > B then
   Print A is large
   else
   Print B is large
4. Stop

Flowchart



Example

```
#include <stdio.h>
int main()
{
   int A,B;
   printf("\nEnter values of A,B : ");
   scanf("%d\n%d",&A,&B);

   if(A>B)
   {
      printf("A is larger");
   }
   else
   {
      printf("B is larger");
   }
   return 0;
}
```



If else-if ladder Statement

The if-else-if ladder statement is an extension to the if-else statement.

It is used in the scenario where there are multiple cases to be performed for different conditions. In if-else-if ladder statement, if a condition is true then the statements defined in the if block will be executed, otherwise if some other condition is true then the statements defined in the else-if block will be executed, at the last if none of the condition is true then the statements defined in the else block will be executed.

Syntax :

```
if(condition1)
{
        //code to be executed if condition1 is true
}
else if(condition2)
{
        //code to be executed if condition2 is true
}
else
{
        //code to be executed if all the conditions are false
}
```

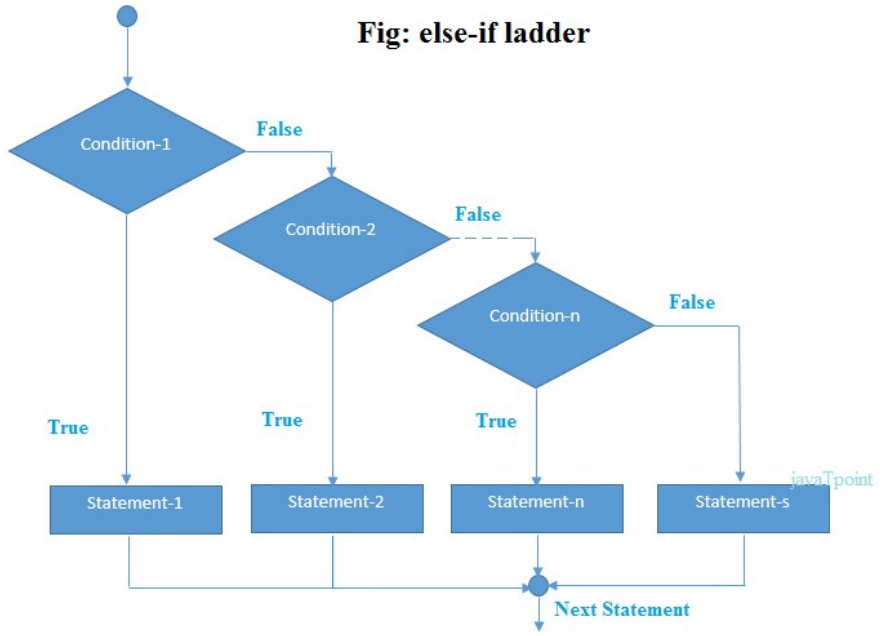


Fig: else-if ladder

Program to calculate the grade of the student according to the specified marks

```
#include <stdio.h>
int main()
{
   int marks;
   printf("Enter your marks : ");
   scanf("%d",&marks);
   if(marks > 85 && marks <= 100)
   {
      printf("Congrats ! you scored grade A ...");
   }
   else if (marks > 60 && marks <= 85)
   {
      printf("You scored grade B + ...");
   }
   else if (marks > 40 && marks <= 60)
   {
      printf("You scored grade B ...");
   }
   else if (marks > 30 && marks <= 40)
   {
      printf("You scored grade C ...");
   }
   else
   {
      printf("Sorry you are fail ...");
   }
}
```

Out Put :

```
Enter your marks : 86
Congrats ! you scored grade A ...
```

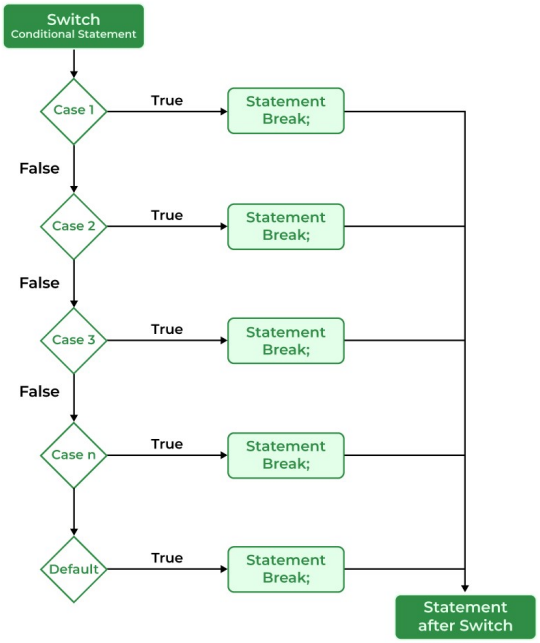
switch statement

The switch statement in C language is used to execute the code from multiple conditions. It is like if else-if ladder statement.

The syntax of switch statement :

```
switch(label)
{
        case label1:
                //code to be executed;
                break;  //optional
        case label2:
                 //code to be executed;
                 break;  //optional
        default:
                //code to be executed if all cases are not matched;
}
```


Flow Chart for switch



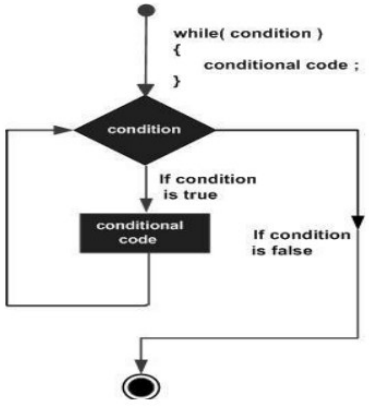
Switch in C

```
#include<stdio.h>
void main()
{
   int number=0;
   printf("Enter a number:");
   scanf("%d",&number);

   switch(number)
   {
      case 10:
      printf("number is equals to 10");
      break;

      case 50:
      printf("number is equal to 50");
      break;

      case 100:
      printf("number is equal to 100");
      break;

      default:
      printf("Number is not equal to 10, 50 or 100");
   }

printf("\nOut of switch");

}
```

Out Put :
Enter a number:10
number is equals to 10
Out of switch

Iterative statements or Loop statements :

A loop statement allows us to execute a statement or group of statements multiple times and following is the general formm of a loop statement in most of the programming languages:



while loop in C

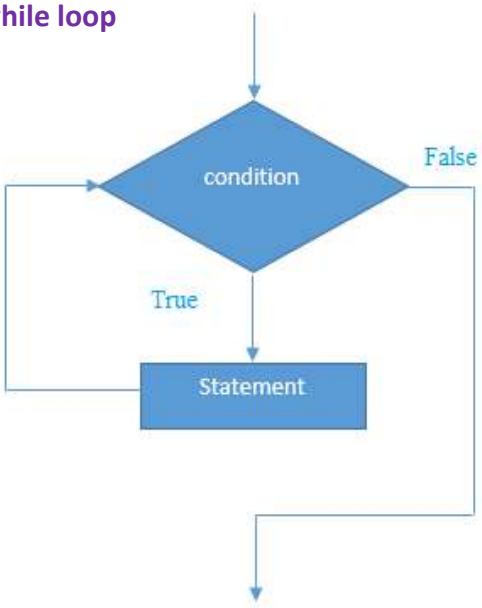
The while loop in C language is used to iterate the part of program or statements many times.

In while loop, condition is given before the statement. So it is different from the do while loop. It can execute the statements 0 or more times.

The syntax of while loop in c language is given below.

```
while(condition)
{
        //code to be executed
}
```


Flow Chart for while loop



while loop in C

Example : Program to list of Numbers

```
#include <stdio.h>
void main()
{
int i=1;
        while(i<=10)
        {
                printf("%d \n",i);
                i++;
        }
}
```

Out Put :

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


do while loop in C

To execute a part of program or code several times, we can use do-while loop of C language. The code given between the do and while block will be executed until condition is true.

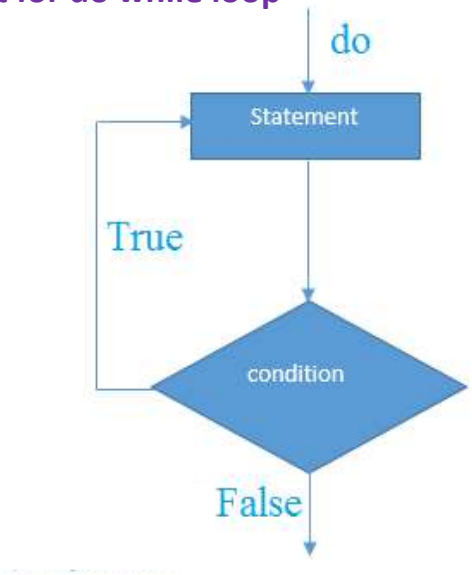
In do while loop, statement is given before the condition, so statement or code will be executed at lease one time. In other words, we can say it is executed 1 or more times.

do while loop in C

The syntax of do while loop in c language is given below.

```
do
{
        //code to be executed
}
while(condition);
```


Flow Chart for do while loop



do while loop in C
Example : Program to list of Numbers

```
#include <stdio.h>
void main()
{
int i=1;
do
{
 printf("%d \n",i);
 i++;
}
while(i<=10);

}
```

Out Put :

1
2
3
4
5
6
7
8
9
10

do while loop in C
Example : Program to list of Numbers

```
#include <stdio.h>
void main()
{
int i=100;
do
{
 printf("%d \n",i);
 i++;
}
while(i<=10);

}
```

Out Put :

100

for loop in C

The for loop in C language is also used to iterate the statement or a part of the program several times, like while and do-while loop.

But, we can initialize and increment or decrement the variable also at the time of checking the condition in for loop.

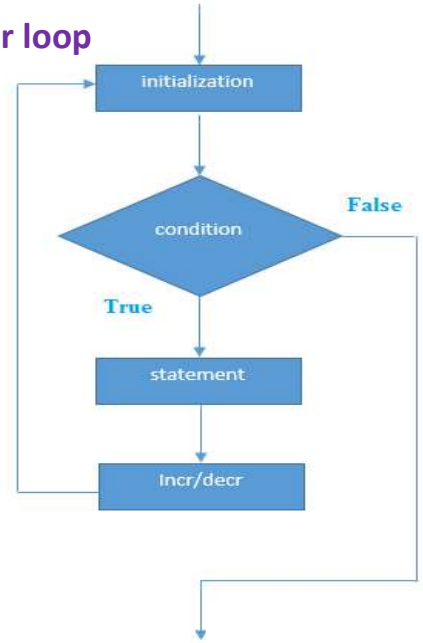
This loop may execute the statement 0 or more times.

for loop in C

The syntax of for loop in c language is given below:

```
for(initialization;condition;incr/decr)
{
        //code to be executed
}
```

Flow Chart for for loop



for loop in C
Example : Program to list of Numbers

```
#include <stdio.h>
void main()
{
int i=0;
   for(i=1;i<=10;i++)
   {
      printf("%d \n",i);
   }
}
```

Out Put :

1
2
3
4
5
6
7
8
9
10

break statement

The break statement in C language is used to break the execution of loop (while, do while and for) and switch case.

In case of inner loops, it terminates the control of inner loop only.

Syntax:

break;

break statement

The break statement in C language is used to break the execution of loop (while, do while and for) and switch case.

In case of inner loops, it terminates the control of inner loop only.

Syntax:

break;

Dr. M. Raja Roy, Associate Professor, ANITS www.mrrtechnical.co.in

Break statement in for loop
Example : Program to list of Numbers

```
#include <stdio.h>
void main()
{
int i=0;
   for(i=1;i<=10;i++)
   {
      if(i==5)
      {
        break;
      }
      printf("%d \n",i);
   }
}
```

Out Put :

1

2

3

4

<- Loop breaks at 5.

5 onwards remaining will not be printed.

continue statement

The continue statement in C language is used to continue the execution of loop (while, do while and for). It is used with if condition within the loop.

In case of inner loops, it continues the control of inner loop only.

Syntax:

continue;

continue statement in for loop
Example : Program to list of Numbers

```
#include <stdio.h>
void main()
{
int i=0;
   for(i=1;i<=10;i++)
   {
      if(i==5)
      {
        continue;
      }
      printf("%d \n",i);
   }
}
```

Out Put :

1
2
3
4 5 will not be printed.
6 Loop continues to 6.
7
8
9
10

goto statement

The goto statement is known as jump statement in C language. It is used to unconditionally jump to other label. It transfers control to other parts of the program.

It is rarely used today because it makes program complex.

Syntax:

goto label;

Goto in Switch

```
#include<stdio.h>
void main()
{
   int number=0;
   printf("Enter a number:");
   scanf("%d",&number);

   switch(number)
   {
      case 10:
      printf("number is equals to 10");
      goto endswitch;

      case 50:
      printf("number is equal to 50");
      goto endswitch;

      case 100:
      printf("number is equal to 100");
      goto endswitch;

      default:
      printf("Number is not equal to 10, 50 or 100");
   }

   endswitch:
   printf("\nOut of switch");

}
```

Out Put :
Enter a number:10
number is equals to 10
Out of switch