Control Statements

Control Statements Include

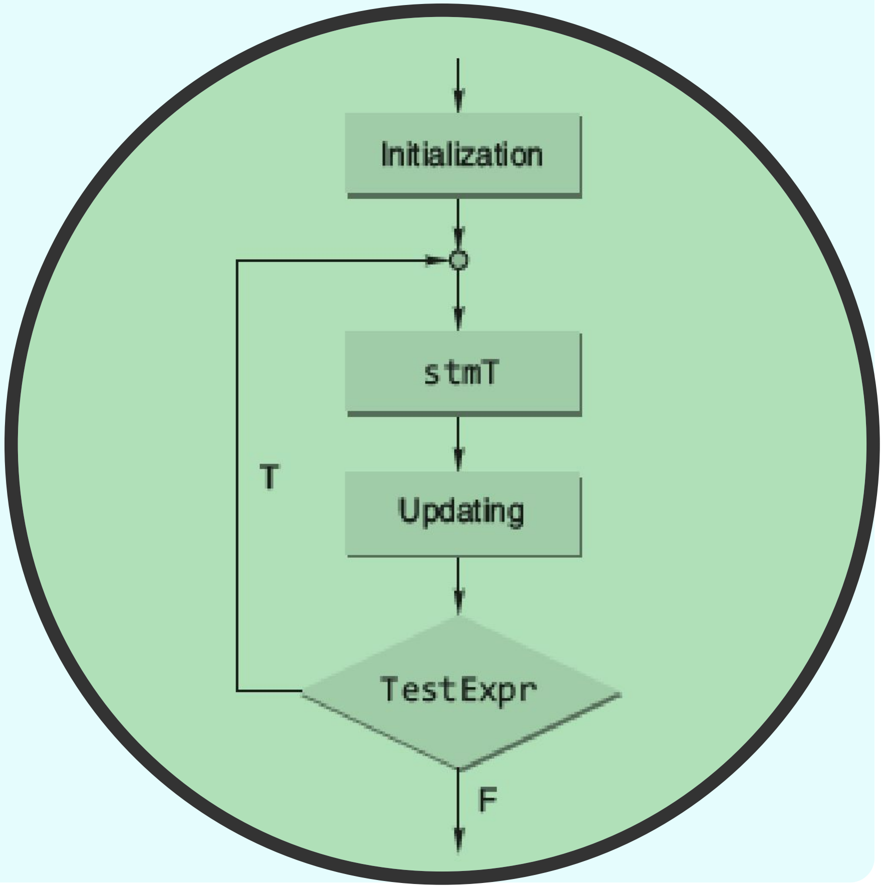
Selection Statements	Iteration Statements	Jump Statements
• if	• for	• goto
• if-else	• while	• break
• switch	• do-while	• continue

“do-while” Construct

The C do-while loop

The form of this loop construct is as follows:

```
do
{
  stmT; /* body of
   statements would be
   placed here*/
}while(TestExpr);
```



Point to Note

With a do-while statement, the body of the loop is executed first and the test expression is checked after the loop body is executed. Thus, the do-while statement always executes the loop body at least once.

Example - do while loop

```
// Program to add numbers until user enters zero
#include <stdio.h>
int main()
{    double number, sum = 0;
// loop body is executed at least once
do
{        printf("Enter a number: ");
     scanf("%lf", &number);
      sum += number;
}
while(number != 0.0);
printf("Sum = %.2lf",sum);
return 0;
}
```

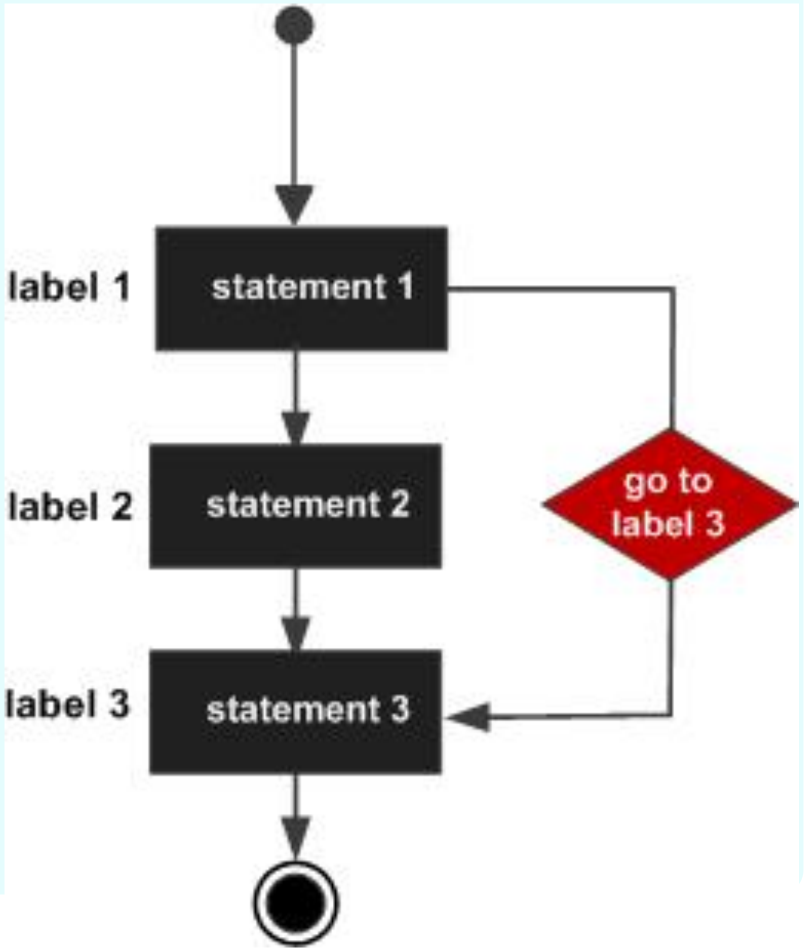
Output

```
Enter a number: 1.5
Enter a number: 2.4
Enter a number: -3.4
Enter a number: 4.2
Enter a number: 0
Sum = 4.70
```

Go Statement

The control is unconditionally transferred to the statement associated with the label specified in the goto statement. The form of a goto statement is

```
syntax
goto label_name;
     .. .
label_name: statement;
```



Go to example

```
#include <stdio.h>
int main ()
{ /* local variable definition */
int a = 10;
/* do loop execution */
LOOP:do
{
   if( a == 15)
   { /* skip the iteration */
   a = a + 1;
   goto LOOP;
    }
printf("value of a: %d\n", a);
a++; }
while( a < 20 );
return 0; }
```

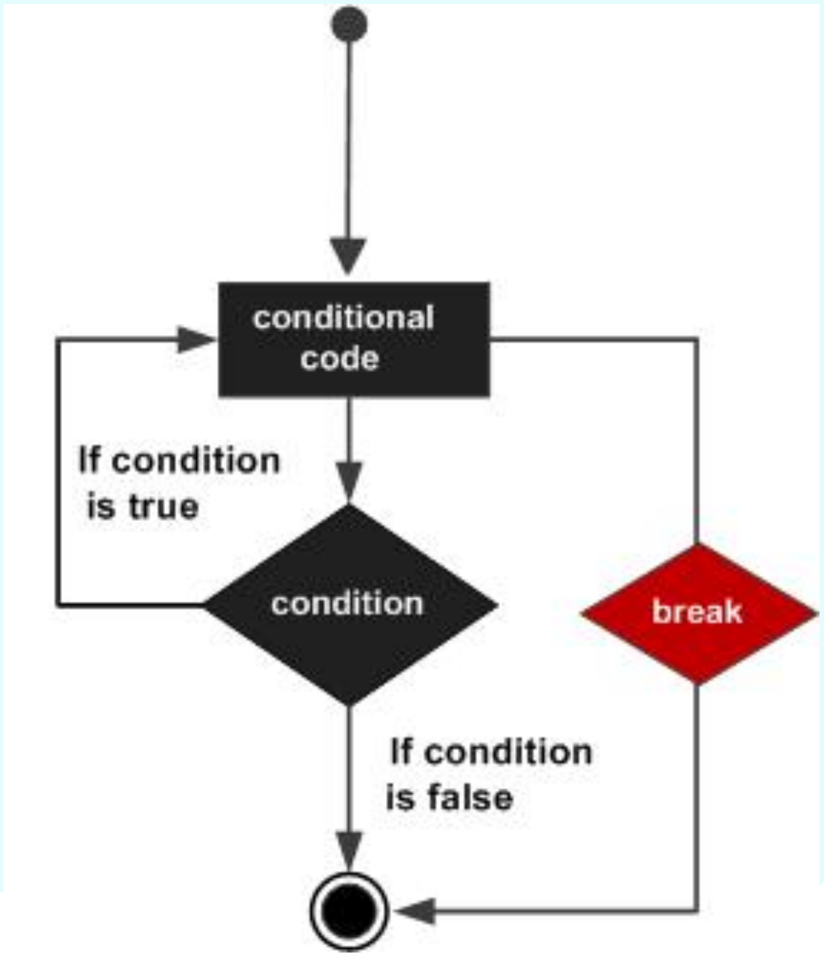
```
Output
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19

Note that 15 is missing
```

SPECIAL CONTROL STATEMENTS

- “break” statements
- “continue” statements

break

- When a **break** statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.
- It can be used to terminate a case in the **switch** statement.



Example

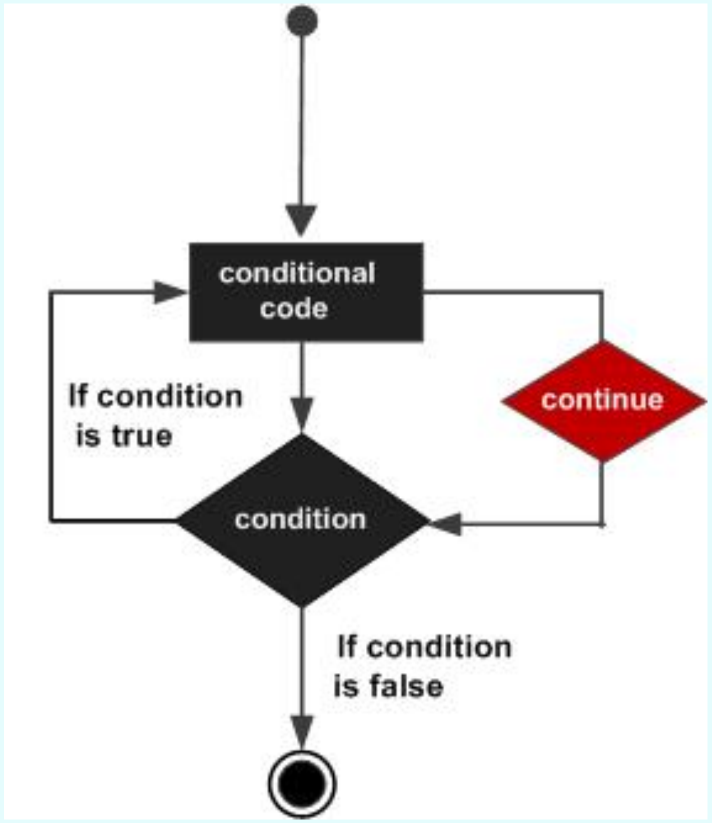
```
#include <stdio.h>
int main ()
{
   int a = 10;
   while( a < 20 )
   {
   printf("value of a: %d\n", a);
   a++;
   if( a > 15)
     { /* terminate the loop*/
     break;
     }
   }
return 0; }
```

Output

```
Value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
```

Continue

- The **continue** statement in C programming works somewhat like the **break** statement.
- Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between.



Example

```
#include <stdio.h>
 int main ()
{
int a = 10;
/* do loop execution */
 do
{ if( a == 15)
     {
     a = a + 1;
     continue;
      }
printf("value of a: %d\n", a);
 a++;
} while( a < 20 );
return 0;
}
```

```
OUTPUT

value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Note 15 is missing !

“break” & “continue” statements

break	continue
1. It helps to make an early exit from the block where it appears.	1. It helps in avoiding the remaining statements in a current iteration of the loop and continuing with the next Iteration
2. It can be used in all control statements including switch construct.	2. It can be used only in loop constructs.

“break” & “exit” statements

break	exit
1. It is a keyword	1. It is a pre-defined function
2. It doesn’t require any header file as it is pre-defined in stdio.h header file in C	2. It requires header file stdlib.h
3. It terminates the loop	3. It terminates the program
4. It is often used only within the loop and switch case statement	4. It is often used anywhere within the program
5. It cannot be used as a variable name as it is a reserved word	5. It is not a reserved word so, it is often used as a variable name
6. In a C program, more than one **break** statement can be executed	6. In a C program, just one exit function will be executed

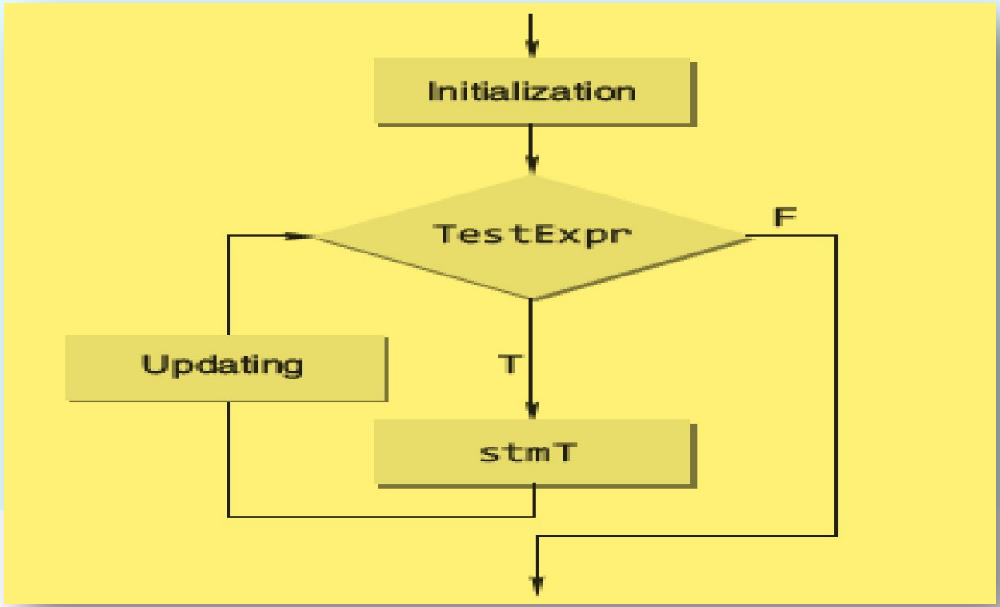
Common programming errors

- *Use of = instead of ==*
- *Forgetting to use braces for compound statement*
- *Dangling else*
- *Use of semicolon in loop*
- *Floating point equality*

“for” Construct

- “for’ loop is an entry condition loop.
- The general form of the for statement is as follows:

```
for(initialization; TestExpr; updating)
{
  stmT;
}
```



printing numbers from 1 to 10

```
#include<stdio.h>
int main()
{
int i;
printf("\nThe natural number's are: ");
for(i=1;i<=10;i++)
{
    printf("%d\n",i);
}
return 0;
}
```

Fibonacci Series

The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21

Program : Print Fibonacci Series up to n number of terms

```
#include <stdio.h>
int main()
{
   int i, n, t1 = 0, t2 = 1, nextTerm;
  printf("Enter the number of terms: ");
  scanf("%d", &n);
   printf("Fibonacci Series: ");
   for (i = 1; i <= n; ++i)
   {
     printf("%d, ", t1);
     nextTerm = t1 + t2;
     t1 = t2;
     t2 = nextTerm;
   }
 return 0;
}
```

WAP to print sum of the natural numbers

```
#include <stdio.h>
int main()
{
int i,N,sum = 0;
printf("\nEnter the value of N\n");
scanf("%d",&N);
for(i=1;i<=N;i++)
sum = sum+i;
printf("\nSum of the natural number is = %d",sum);
return 0;
}
```

Series: 1^2+2^2+3^2+4^2+..N^2

```
#include <stdio.h>
int main()
{
int i,N;
unsigned long sum = 0;
printf("\nEnter the value of N\n");
scanf("%d",&N);
for(i=1;i<=N;i++)
sum = sum+(i*i);
printf("\nSum of the series is = %ld",sum);
return 0;
}
```

Nested Loops

- A nested loop refers to a loop that is contained within another loop.
- If the following output has to be obtained on the screen

```
*
**
***
****
```

then the corresponding program will be

Code

```
#include<stdio.h>
int main()
{
int n,i,j;
printf("\nEnter the no. of rows
   =>");
scanf("%d",&n);
printf("\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=i;j++)
printf("* ");
printf("\n");
}
return 0;
}
```

Nested Loops

```
1
2 2
3 3 3
4 4 4 4
```

```
#include <stdio.h>
int main()
{
 int row, col;
//outer for is to print all rows
 for(row=1;row<=4;row++)
 {
     //inner for loop is to print all
   column values in one row
     for(col=1;col<=row;col++)
     {
          printf("%d \t", row);
     }
    printf("\n");//to go to next row
 }
 return 0;
}
```

WAP to print the following pattern for n rows. Ex. for n=5 rows

```
    *
   * *
  * * *
 * * * *
* * * * *
```

Code

```
#include<stdio.h>
int main()
{
int n,i,j,k;
printf("\nEnter how many rows =>");
scanf("%d",&n);
printf("\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n-i;j++)
printf(" ");
for(k=1;k<=i;k++)
printf("* ");
printf("\n");
}
return 0;
}
```

WAP to form a pyramid of numbers for a given number. Ex. for number 4

```
   1
  121
 12321
1234321
```

```
#include <stdio.h>
int main()
{
int n,i,j,k;
printf("\nEnter a number to form a
pyramid=>");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=n-i;j++)
printf(" ");
for(k=1;k<=i;k++)
printf("%d",k);
for(k=i-1;k>0;k--)
printf("%d",k);
printf("\n");
}
return 0;
}
```

WAP to print the following pattern for n rows. Ex. for n=4 rows

```
1
01
101
0101
10101
010101
```

```
#include <stdio.h>
int main()
{
int i,j,n;
printf("\nEnter The
Number Of Rows =>");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=i;j++)
printf("%2d",(i+j+1)%2
);
printf("\n");
}
return 0;
}
```

When to use which loop

- **For, while** loops are pre-test loops. **Do while** is post-test loop.
- For loop is usually when the number of iterations are known in advance. Like 0 to 75 etc.
- While and do while are for unknown iterations.
- While – when you don't even want to execute the body even once.
- Do – while when you want to execute the body

Assignment

- WAP to print Fibonacci Series up to n number of terms using for loop.
- WAP to find sum of the natural numbers for a given number of terms.
- WAP to print the following pattern for n rows. Ex. for n=4 rows

```
1
2 2
3 3 3
4 4 4 4
```

- WAP to print the following pattern for n rows. Ex. for n=4 rows

```
*
**
***
****
```

- WAP to print the following pattern for n rows. Ex. for n=4 rows

```
1
1 2
1 2 3
1 2 3 4
```