Subprogram

In computer science, subprogram (also called subroutine, procedures, functions or method) is a portion of code within the larger program that perform a specific task.

In programming language such as c, c++ etc. subprogram may also be simply called as function.

Advantages

1) The composition of complex programming task into simplest.
2) Reducing the duplicity of code within the program.
3) Enable the reuse of code within the program.
4) Dividing the large programming task among various program or various stages of project.
5) Hiding implementation detail from user of the subprogram.

Disadvantages

1) Use of the subprogram imposes the computational step in the call mechanism itself.
2) The subprogram typically requires standard house-keeping code, both act entry to exit form.

e.g. return_type function_name(arguments /* void if there is no argument */)

```
{
        /* some code */
}
```

```
void abc(void)
{
        /* some code */
}
int xyz(void)
{
        printf("zzzz");
        return 0;
}
```

Coding Style

a) Choice of names
b) Indentation and spacing
c) Documentation (comments)
d) Function declaration
e) Data types
f) Global declaration
g) E-value and constant
h) Loops, loop test and parenthesis

C-Construct

if statement

A) if (expression)

statement; /* if there is only statement then there is no need of curly parenthesis */

B) if (expression)

```
{
        statement;
        statement;
        statement;
}
```

The statement simple or compound will be executed only if the expression is non-zero value i.e. if the expression is true.

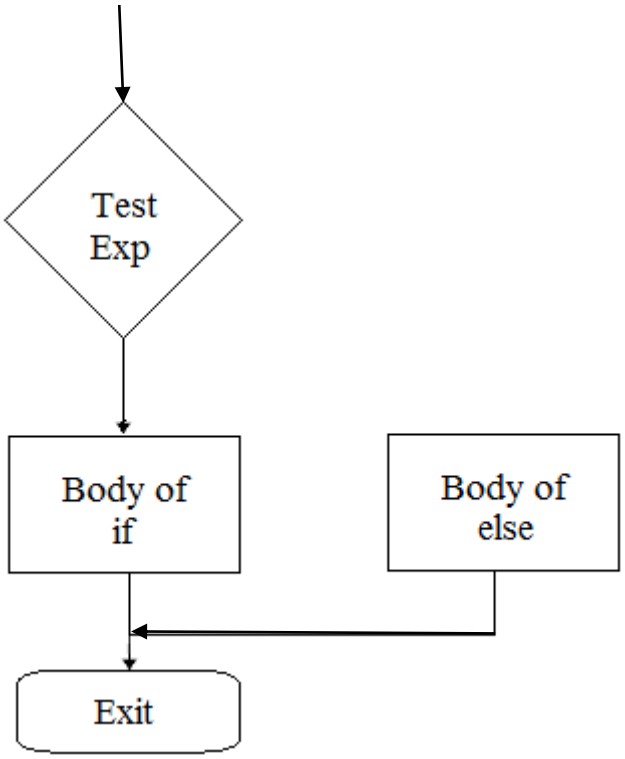
If the expression have the value of zero i.e. false then the statement will be ignored.

e.g. if(x>10)

```
{
        printf("%d",x);
}
```

e.g. if(-6)

```
{
        printf("%d",x);
}
```



e.g. if(x==6)

```
        {
                printf("%d",x);
        }
```

if(x!=10)

```
        {
                printf("%d",x);
        }
```

if else statement

if(expression)

```
        statement1;
```

else

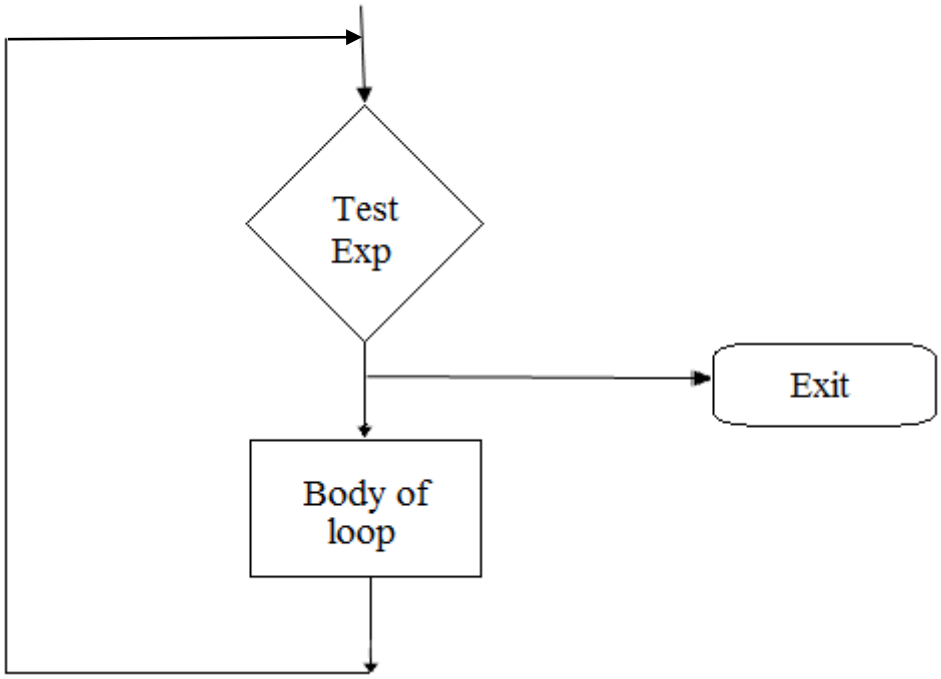
```
        statement2;
```

If the expression has non-zero value i.e. if the exp. is true then the exp 1 will be executed otherwise statement 2 will be executed.

The while statement

while(expression)

statement;

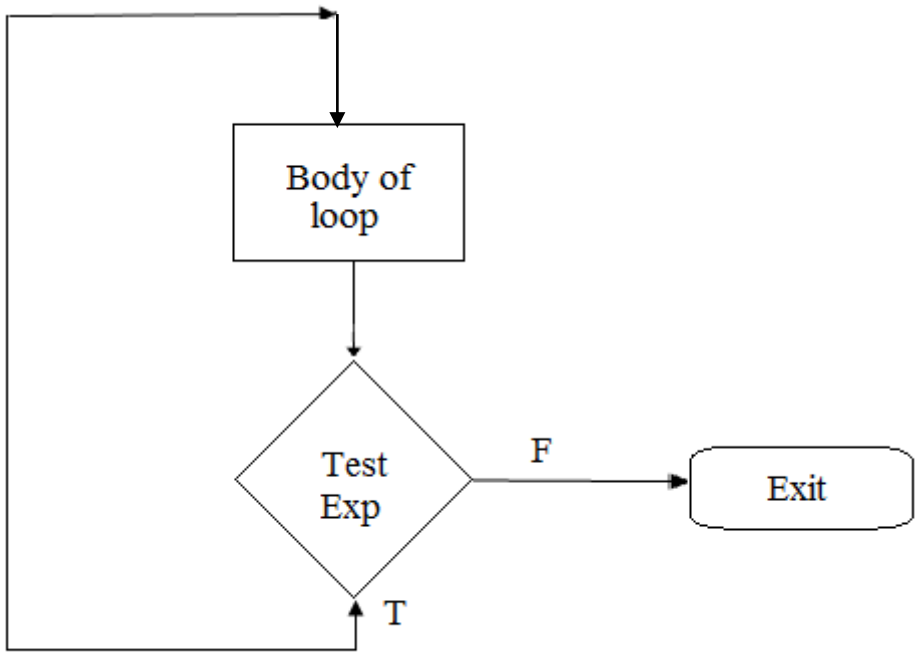


e.g. void main()

```
{
        int a=5;
        while(a>=2)
        {
                printf("%d",a);
                a--;
        }
}
```

do-while statement

```
do
{
        statement;
}while(expression);
```



The statement will be executed repeatedly as long as the value of expression is true.

Note: Statement will always execute at least once.

e.g.

```
void main()
{
        int a=5;
        do
        {
                printf("%d",a);
                a--;
        }while(a>=2);
}
```

for statement

```
for(initialization;condition;updation)
{
        Statement;
}
```

e.g.

```
void main()
{
        int i;
        for(i=5;i<=10;i++)
        {
                printf("%d",i*i);
        }
}
```

e.g.

```
void main()
{
        int i=5;
        for(;i<=10;)
        {
                printf("%d",i*i);
                i++;
        }
}
```

The switch statement

It causes a particular group of statement to be chosen from the several available. The selection is based upon the current value of expression which is included within the switch statement. Expression can be integer or character because individual character have equivalent integer value.

The embedded statement is generally a compound statement that statement that specify alternate forces of attraction. The case values must be unique within the switch statement

```
switch(a)
{
        case -1:        y=abs(x);
                        break;
        case 0:         y=sqrt(x);
                        break;
        case 5:         y=2*(x-1);
                        break;
        default:        y=0;

}
```

The Break Statement

Break statement is used to terminate the loop. If a break statement is included in a while loop, it jump out of the loop instantly.

e.g.

```
void main()
{
        int a=-5;
        if(a<0)
        {
                printf("Error");
                break;
        }
}
```

The Continue Statement

The continue statement is used to bypass the remainder of the currently pass through the loop, the loop does not terminate rather remaining statement skipped and the computation proceed directly from the beginning of the loop.

```
void main()
{
```

```
    int n,count,x,avg,sum=0;
    printf("Enter the no. of students=");
    scanf("%d",&n);
    for(count=1;count<=n;count++)
    {
        printf("x=");
        scanf("%d",&x);
        if(x<0)
            continue;
        sum=sum+x;
    }
    avg=sum/n;
    printf("Average=%d",avg);

}
```

goto statement

The goto statement is used to alter the normal execution of programs. The general form is goto level1; where level1 is identifier that is used to target the statement

```
main()
{
    int a=2;
    if(a>2)
        goto xyz;
    printf("programming");
    xyz: printf("language");
}
```

Output will be language, if we take value of a<=2, then the output would be programminglanguage.