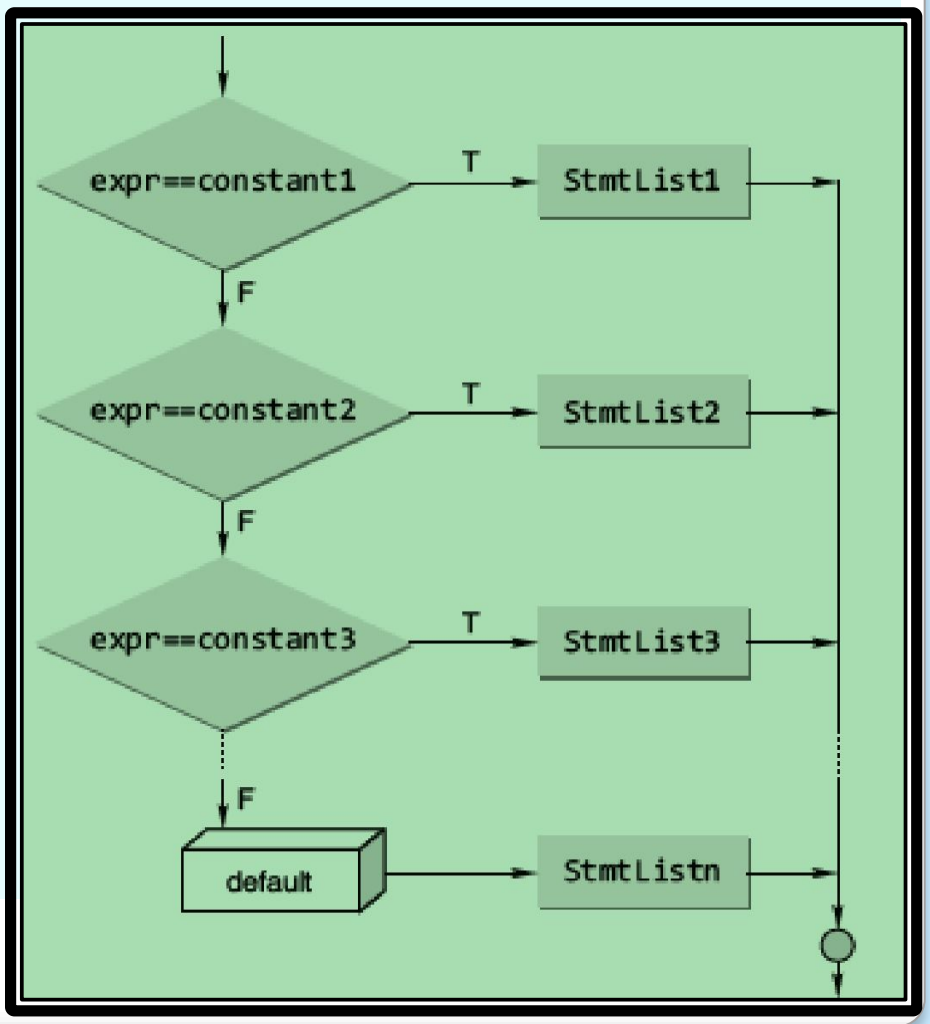
Control Statements

Control Statements Include

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

The Switch Statement

```
switch(expr)
{
case constant1:
   stmtList1;
break;
case constant2:
   stmtList2;
break;
case constant3:
   stmtList3;
break;
default: stmtListn;
}
```



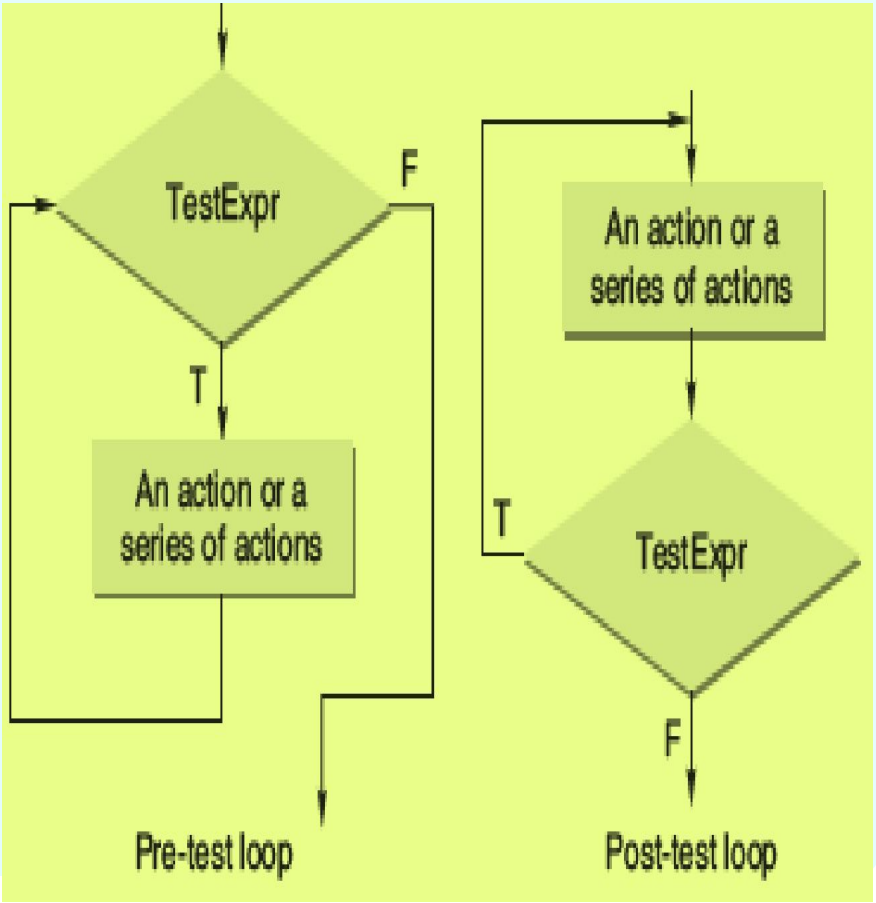
Use Switch case to create KIIT grade system where total marks is less than or equal to 100.

- 90-100 is ‘O’ grade
- 80-89 is ‘E’ grade
- 70 to 79 is ‘A’ grade
- 60 to 69 is ‘B’ grade
- 50 to 59 is ‘C’ grade
- 40 to 49 is ‘D’ grade
- below 40 is ‘F’ grade

```
#include<stdio.h>
int main()
{
int total_mark,tm;
printf("\nEnter total mark secured by a student: ");
scanf("%d",&total_mark);
tm=total_mark/10;
switch(tm)
{
        case 9: printf("\nSecured grade is O");
                break;
        case 8: printf("\nSecured grade is E");
                break;
        case 7: printf("\nSecured grade is A");
                break;
        case 6: printf("\nSecured grade is B");
                break;
        case 5: printf("\nSecured grade is C");
                break;
        case 4: printf("\nSecured grade is D");
                break;
        default: printf("FAIL");
}
return 0;
}
```

Iteration & Repetitive Execution

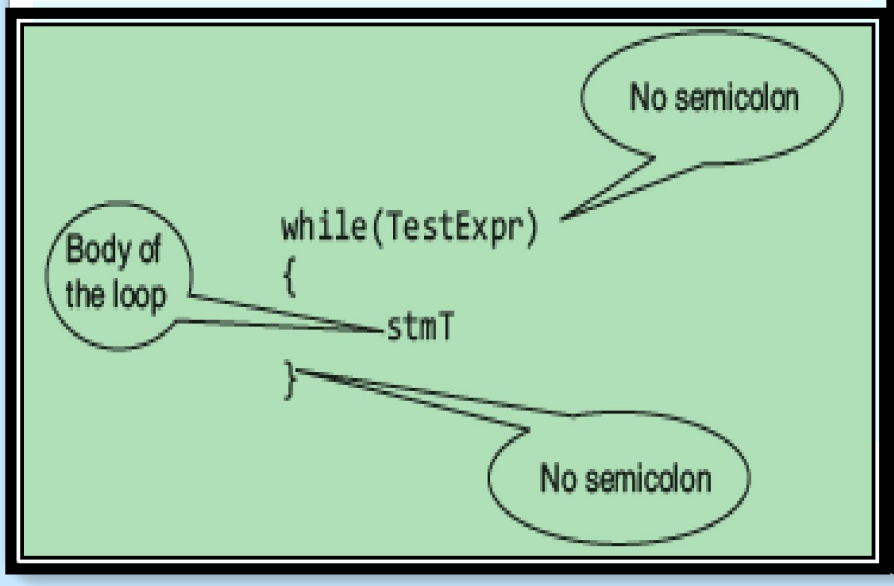
- A loop allows one to execute a statement or block of statements repeatedly. There are mainly two types of iterations or loops –
  - *unbounded iteration or unbounded loop (unefined iteration)*
  - *bounded iteration or bounded loop (fixed iteration)*
- A loop can either be a *pre-test loop or be a post-test loop* as illustrated in the diagram.

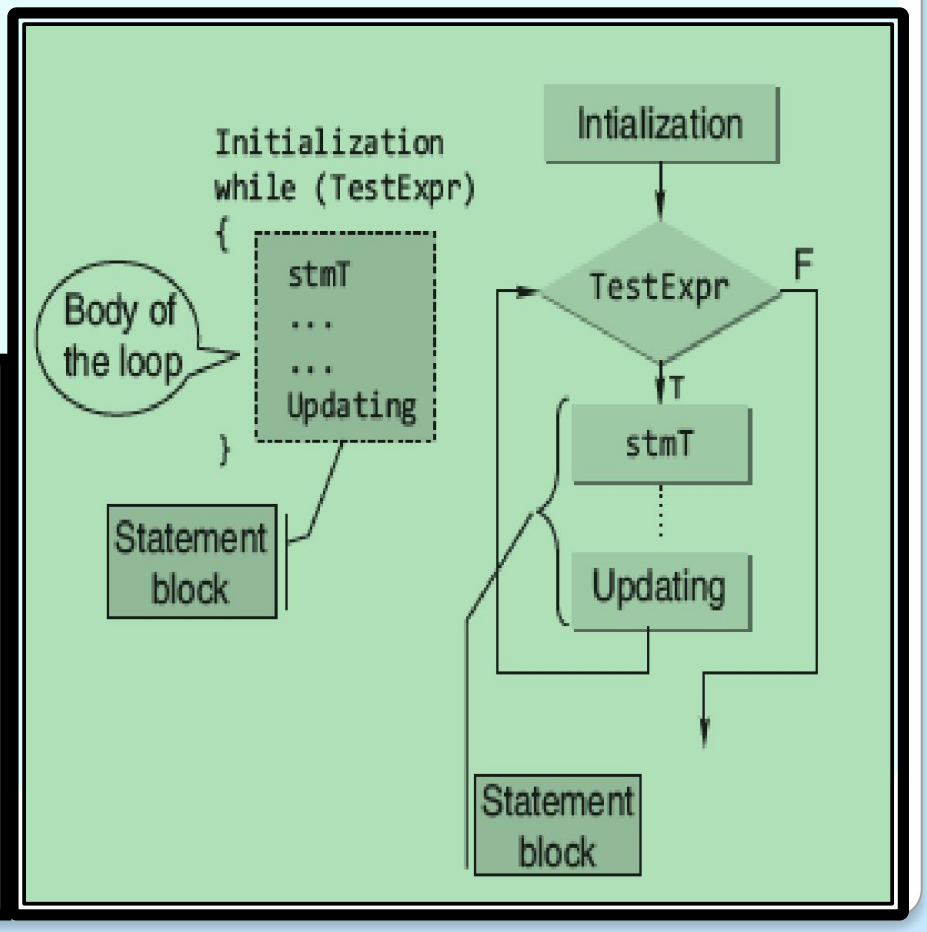


"While" Construct

Expanded Syntax of "while" and its Flowchart Representation

while statement is a pretest loop. The basic syntax of the while statement is shown below:





Loop features

- **Each loop construct will have three components**

1. **test expression**, where a condition will be checked, and the loop will be executed only if the condition is true. This can be checked pre or post loop execution.

   Hence there must be

   an exit condition

   an entry condition .

2. **Statements** inside the loop which need to be executed

3. **Updating statement** : this will make the loop moving forward (or backward).

An Example

```
#include <stdio.h>
int main()
{
   int c;
   c=5; // Initialization
   while(c>0)
      { // Test Expression
         printf(" \n %d",c);
         c=c-1; // Updating
      }
    return 0;
}
```

This loop contains all the parts of a while loop. When executed in a program, this loop will output

5
4
3
2
1

printing numbers from 1 to 10

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

C program to print the multiplication table of 2 from 1 to 10

```
include<stdio.h>
int main()
{
   int i=1;
   while(i<=10)
     { printf("2 * %d = %d\n",i,2*i);
      i++;
     }
   return 0;
}
```

WAP to print sum of digits in number

```
#include<stdio.h>
int main()
{
int num, sum = 0, r;
printf("Enter the number");
scanf("%d", &num);
while(num!=0)
{
r = num%10;
sum = sum+r;
num = num/10;
}
printf("Addition of the number is = %d",sum);
return 0;
}
```

- WAP to Print the reverse of a number

```
#include<stdio.h>
int main()
{
int num, rev = 0, r;
printf("Enter the number");
scanf("%d", &num);
while(num!=0)
{
r = num%10;
rev = rev*10+r;
num = num/10;
}
printf("Reverse of the number is = %d",rev);
return 0;
}
```

WAP to calculate factors of a positive integer?

```
#include<stdio.h>
int main()
{
int num, i = 1;
printf("Enter a number");
scanf("%d",&num);
while(i<=num)
{
if(num%i==0)
printf("%d\t",i);
i = i + 1;
}
return 0;
}
```

WAP to check whether a number n is prime number or not.

```
#include <stdio.h>
int main()
{
int n,i=2,test=1;
printf("\nEnter A Number :");
scanf("%d",&n);
while(i<=n/2)
{
      if(n%i==0)
      {
      test=0;
      break;
      }
      i=i+1;
}
if(test==1)
printf("\nThe Number %d is a prime number",n);
else
printf("\nThe Number %d is not a prime number",n);
return 0;
}
```

Infinite Loop

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

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

```
while(cond);
{
//code
}
```

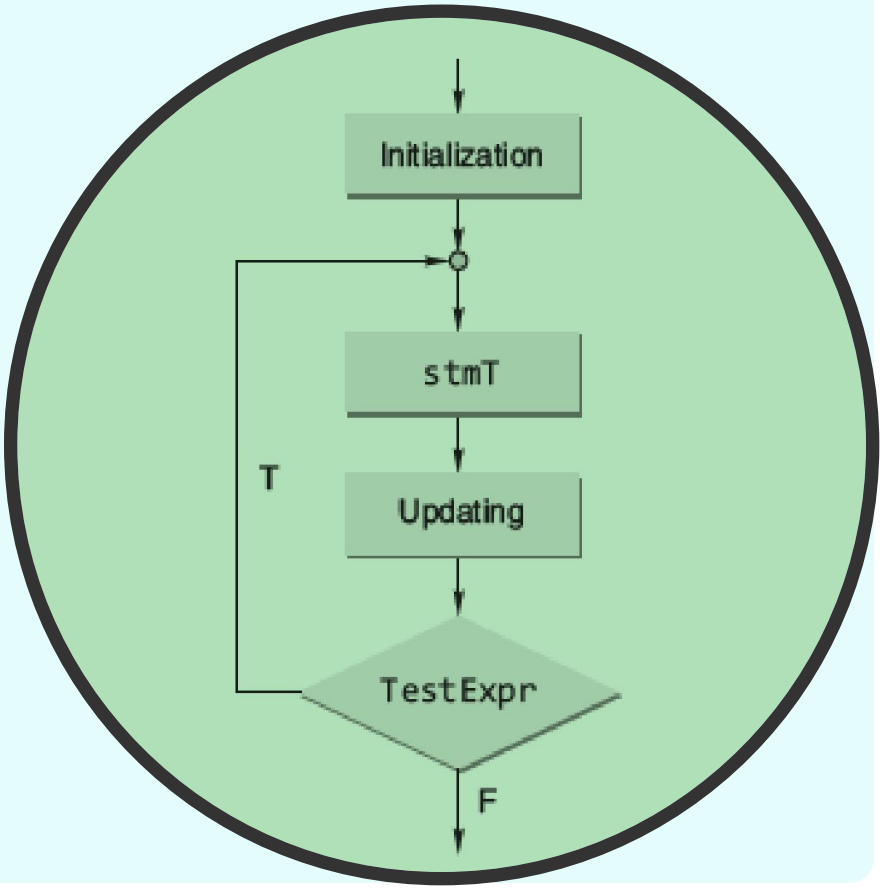
```
while(inp='y')
{
//code
}
```

"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

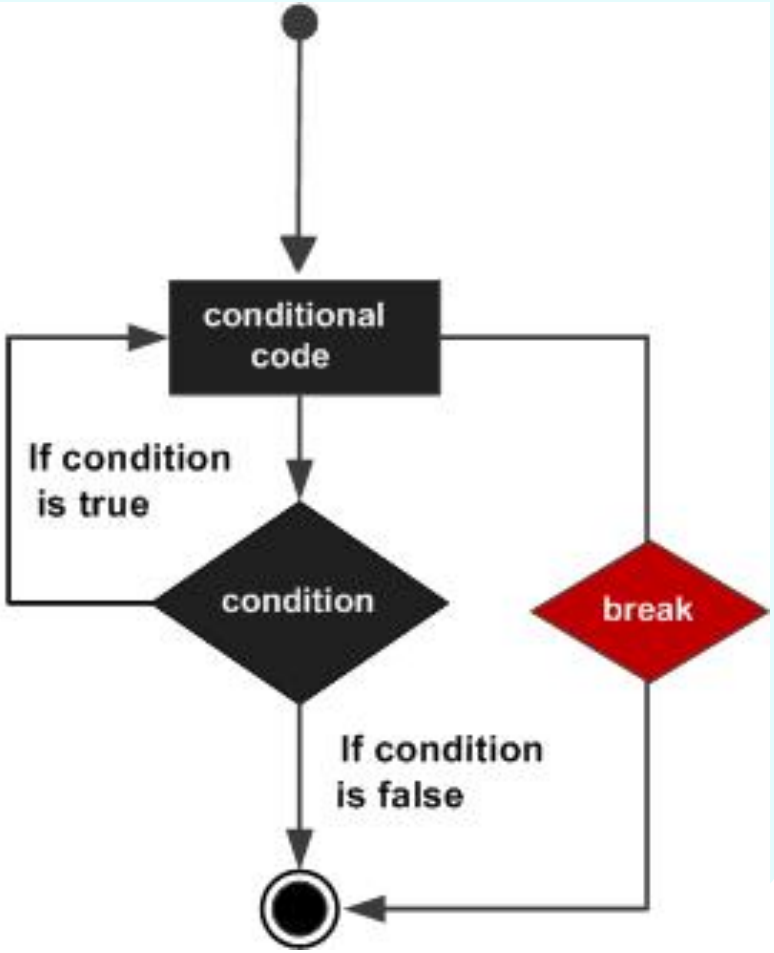
```
// Program to add numbers until user enters zero
#include <stdio.h>
int main()
{    int number, sum = 0;
// loop body is executed at least once
do
{    printf("Enter a number: ");
     scanf("%d", &number);
     sum += number;
}
while(number != 0);
printf("Sum = %d",sum);
return 0;
}
```

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

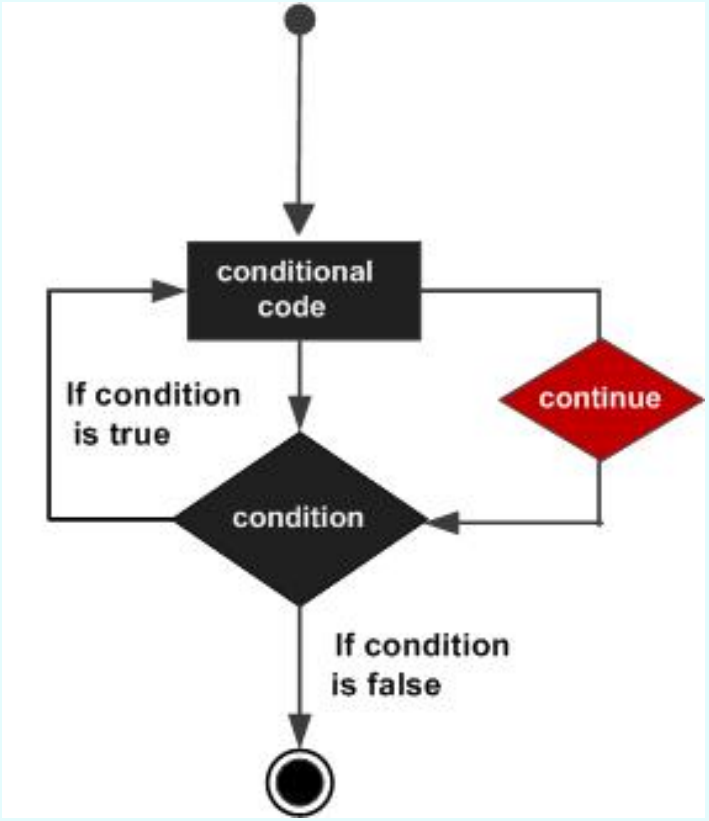
```
#include <stdio.h>
int main ()
{
   int a = 10;
   while( a < 20 )
   {
   printf("value of a: %d\n", a);
   a++;
   if( a > 15)
     { /* terminate the loop*/
     break;
     }
   }
return 0; }
```

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*

WAP to read an alphabet from the user and convert it into uppercase if the entered alphabet is in lowercase, otherwise display an appropriate message.

```
#include<stdio.h>
int main()
{
char ch;
printf("\n Enter an alphabet:");
scanf("%c", &ch);
if (ch>='a' && ch<='z')
{
ch=ch-32;
printf("\n The uppercase of the entered alphabet is %c", ch);
}
else
printf("\nThe entered character is not a lower case alphabet");
return 0;
}
```

PROGRAM #

Find out an entered alphabet is vowel or consonant

If entered character is alphabet (small or capital range of alphabets)

    If ch is vowel (a, e, i, o, u both in capital or small)

        print vowel

    else

        print conso

else

enter valid alphabets

```
#include<stdio.h>
int main()
{
char ch;
printf("\n Enter an alphabet: ");
scanf("%c", &ch);
if ((ch>='a' && ch<='z') || (ch>='A' && ch<='Z'))
{
    if (ch=='a' || ch=='A' || ch=='e' || ch=='E' || ch=='i' || ch=='I' || ch=='o' ||
ch=='O' || ch=='u' ||ch=='U')
        printf("\nThe entered character %c is a vowel", ch);
    else
        printf("\nThe entered character %c is a consonant", ch);
}
else
    printf("\nThe entered character %c is not an alphabet",ch);

return 0;
}
```

Assignment

Use Switch case to create KIIT grade system where total marks is less than or equal to 100.

- 90-100 is ‘O’ grade
- 80-89 is ‘E’ grade
- 70 to 79 is ‘A’ grade
- 60 to 69 is ‘B’ grade
- 50 to 59 is ‘C’ grade
- 40 to 49 is ‘D’ grade
- below 40 is ‘F’ grade

Assignment (While Loop)

WAP to print numbers from 1 to 10

WAP to print the multiplication table of 2 from 1 to 10

WAP to print sum of digits in number

WAP to Print the reverse of a number

WAP to calculate factors of a positive integer?

WAP to check whether a number n is prime number or not.