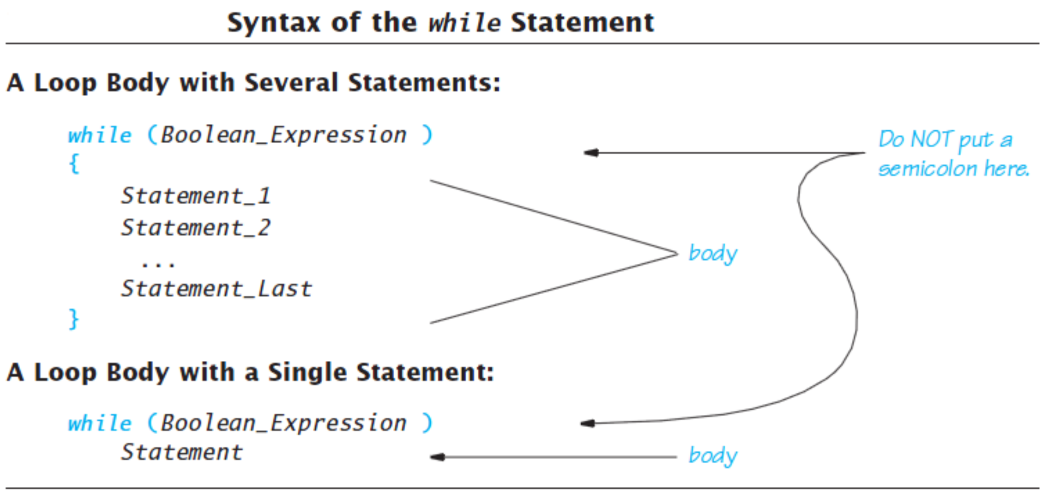
## Repetition ………………………………………………………………………..

A portion of a program that repeats a statement or group of statements until some condition occurs is called a loop. The C++ language has a three of ways to create loops:

1. while statement ( or while loop)
2. do-whil statement
3. for statement

In the following *syntax of while loop*:



**Syntax of the `while` Statement**

**A Loop Body with Several Statements:**

```
while (Boolean_Expression )
{
      Statement_1
      Statement_2
        . . .
      Statement_Last
}
```

**A Loop Body with a Single Statement:**

```
while (Boolean_Expression )
      Statement
```

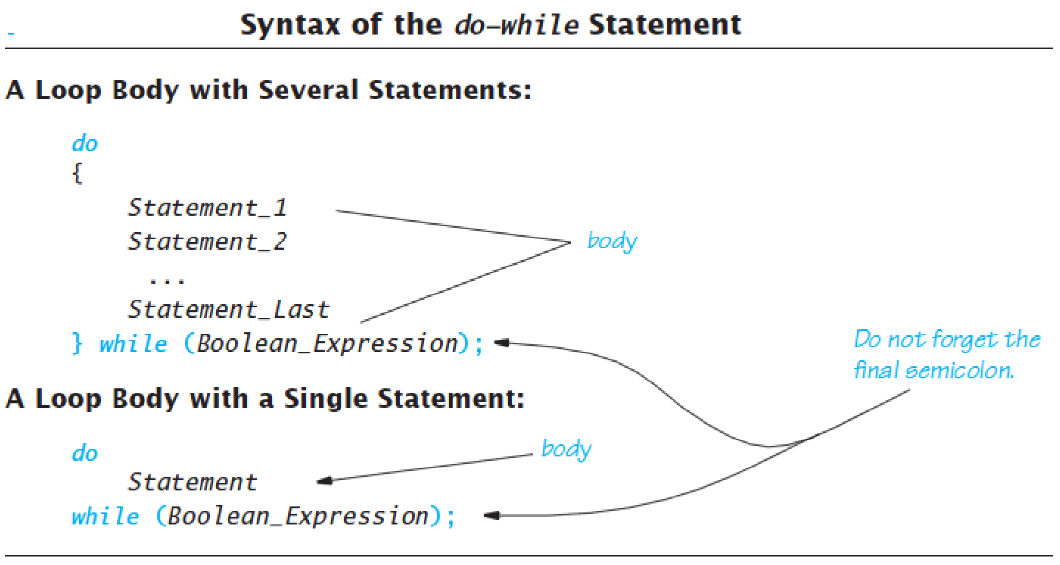
In a while statement, the statement is executed repeatedly as long as the Boolean_Expression evaluates to a non-zero value. This means that somewhere in the while statements(body) must be a statement altering the tested expression's value.

*Exercise* 1: Show the output of the following program:

```
#include <iostream>
void main( ){
   int count_down;
   cout<< "How many greetings do you want? ";
   cin>>count_down;
   while (count_down> 0)
      {
        cout<< "Hello ";
        count_down = count_down - 1;
      }//while
}//main
```

*Exercise*2: Rewrite the previous program by using for loop statement instead of while statement.

In the following t*he syntax of do-while loop:*



**Syntax of the *do-while* Statement**

**A Loop Body with Several Statements:**

```
do
{
     Statement_1
     Statement_2
       . . .
     Statement_Last
} while (Boolean_Expression);
```

**A Loop Body with a Single Statement:**

```
do
     Statement
while (Boolean_Expression);
```

A do-while statement is similar to a while statement except that the loop body is always executed at least once.

*Exercise*3: Rewrite the program in exercise 1 by using do-while statement instead of while statement.

*Exercise 4:* In the following program to print the numbers 1 to 100 :

```
#include <iostream.h>
void main() {
   int x;
  for ( x=1; x <= 100; x++)
     cout<< x << " ";
 }
```

Rewrite the previous program by using do-while statement instead of for statement.

*Exercise* 5: Rewrite the previous program by using while statement instead of for statement.

☒ ***Break and continue statements***

Two useful statements in connection with repetition statements are the *break* and *continue* statements.

The loop exits when the condition at the beginning becomes false and also can be exited at any point through the use of a break statement.

*The syntax of the* ***break*** *statement is*

```
break;
```

A break statement, as its name implies, forces an immediate break, or exit, from the switch, while, for, and do-while statements.

*Example*: Execution of the following while loop is terminated immediately if a number greater than 76 is entered:

```
while(count<=10)  {
   cout<<"Enter a number:";
   cin>>num;
   if (num>76){
      cout<< "stop\n";
      break;
     }
  else
    cout<<"go \n";
  count++;
 }
```

The continue statement is applies only to loops created with while, do-while, and for statements.

*The syntax of the **continue** statement is*

```
          continue;
```

When continue is encountered in a loop, the next iteration of the loop begins immediately, this means execution is transferred automatically to the top of the loop.

**Example:**

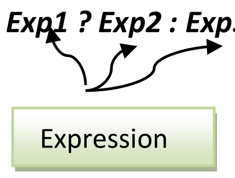
```
count = 1;
while(count<10){
    cout<<"Enter a grade:";
    cin>>grade;
    if  (grade<0  || grade>100)
      continue;
    total=total+grade;
    count++;
}
```

☒ **Additional Information** ......................................................................................

- ☒ clrscr()  function for clear screen (conio.h)
- ☒ ? Operator

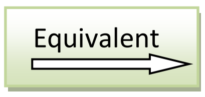
Syntax:

***Exp1 ? Exp2 : Exp3;***



For example:

```
x = 10;
y = x>9 ? 100 : 200;
```



```
x = 10;
 if  (x>9)          y = 100;
 else               y = 200;
```

☒ In the following functions for reading and writing character.

| **Function** | **Operation** | **Header file** |
|---|---|---|
| ch = getchar() | Reads a character from the keyboard, wait for carriage return | stdio.h |
| putchar(ch) | Writes a character (ch) to the screen | |
| ch=getch( ) | Read a character from the keyboard | conio.h |
| putch (ch) | Writes a character (ch) to the screen | |

☒ in the following tables some library functions

1. Character Functions

| Header file | Function | Operation |
|---|---|---|
| ctype.h | isalnum(ch) | Returns true if ch is either a letter or a digit |
| | isalpha(ch) | Returns true if ch is a letter |
| | isdigit(ch) | Returns true if ch is a digit |
| | islower(ch) | Returns true if ch is a lowercase letter |
| | ispunct(ch) | Returns true if ch is a punctuation character |
| | isspace(ch) | Returns true if ch is a whitespace character: space, tab, carriage return, new line, or form feed |
| | isupper(ch) | Returns true if ch is an uppercase letter |
| | toascii(ch) | Returns ASCII code for ch |
| | tolower(ch) | Returns the lowercase version of ch if ch is an uppercase letter; otherwise returns ch |
| | toupper(ch) | Returns the uppercase version of ch if ch is a lowercase letter; otherwise returns ch |

## 2. Arithmetic Functions

| Header file | Function | Operation |
|---|---|---|
| math.h | cos(x) | Returns the cosine of an angle of *x* radians. |
| | sin(x) | Returns the sine of an angle of *x* radians. |
| | tan(x) | Returns the tangent sine of an angle of *x* radians. |
| | ceil | Rounds *x* upward, returning the smallest integral value that is not less than *x*. |
| | fmin(x,y) | Returns the smaller of its arguments: either *x* or *y*. |
| | fmax(x,y) | Returns the larger of its arguments: either *x* or *y*. |
| | fdim(x,y) | Returns the *positive difference* between *x* and *y*. that's mean return x-y if x>y, and zero otherwise. |
| | exp | Returns $e^x$ |
| | Fabs | Returns the absolute value |
| | floor(x) | Rounds *x* downward, returning the largest integral value that is not greater than *x*. |
| | Fmod | Returns x modulo y for arguments x and y |
| | trunc(x) | Rounds *x* toward zero, returning the nearest integral value that is not larger in magnitude than *x*. |
| | round(x) | Returns the integral value that is nearest to *x*, with halfway cases rounded away from zero. |
| | pow(base,exponent) | Returns *base* raised to the power *exponent*: $base^{exponent}$ |
| | sqrt(x) | Returns the *square root* of *x*. |
| | abs(x)<br>fabs(x)<br>labs(x) | Returns the *absolute value* of *x*: $\|x\|$. |
| | rand() | Generate random number |
| | srand(seed) | Initialize random number generator using the argument passed as *seed*. |
| | abort() | Abort current process |
| | exit() | Terminate calling process |
| | div(x,y)<br>ldiv(x,y)<br>lldiv(x,y) | Returns the integral quotient and remainder of the division of *x* by *y* ( x/y) |
| | labs(x)<br>llabs(x)<br>abs(x) | Returns the *absolute value* of *x*: $\|x\|$. |

☒ Formatted Console I/O

The functions **printf( )** and **scanf( ) (functions in stdio.h)** perform formatted output and input in various formats. Both functions can operate on any of the built-in data types, including characters and numbers.

Syntax:

**printf (*control_string*, argument_list);**

The *control_string* consists of two types of items:

1. composed of characters that will be printed on the screen.
2. format specifiers that define the way the variables are displayed, begins with a percent sign (%) and is followed by the format code. There must be exactly the same number of variables (argument_List) as there are format specifiers.

In the following table some of format specifiers:

| Code | Format |
|---|---|
| %c | display character |
| %d, %i, %u | displays numbers in integers |
| %f | display number in floating point |
| %s | Display string. |
| %n | Display the number of characters will be printed |

A format specifiers may take modifiers that placed between the percent sign and the format code, which represent the *minimum field width*.

- for floating-point data : **%10.4f:** displays a number at least ten characters wide with four decimal places.
- for strings: **%5.7s** displays a string at least five and not exceeding seven characters long.
- for integer types : % 3d display a number at least 3 digits.

for example the following statement:

```
printf("I like %c%s", 'C', "++ very much!");
```

will be display *:* I like C++ very much!

**scanf( )** is can read all the built-in data types.

suntax : **scanf(*control_string*, & variable);**

The *control_string* determines how value are read into the variable. The input format specifiers are preceded by a % sign and tell **scanf( )** what type of data is to be read next. in the following some of format specifiers:

| Code | Meaning |
|---|---|
| %c | Read a single character. |
| %d, %i | Read a decimal integer. |
| %e, %f | Read a floating-point number. |
| %s | Read a string. |

Example:

```
 printf ("Enter your age : " );
scanf ("%d", &i);
printf ("my age %d years old.\n", i);
```

Problems:

1. Write program that inputs a number TOTAL and a sequence of numbers. The program stops inputting numbers when the sum of the numbers exceeds TOTAL and then outputs how many numbers were input.

2. Write program to input a number TOTAL and compute and output the value of S. The progrm should stop when S > TOTAL. Also output how many terms of S were used:

$$s = \sum_{i=2}^{n} 1/i$$

3. Show the output of the following :

a.

```
#include <stdio.h>
 void main(void){
   int i;
   for(i=1; i<5; i++)
      printf("%8d %8d %8d\n", i, i*i, i*i*i);
}
```

b.

```
#include <stdio.h>
void main(void){
   printf("%.4f\n", 123.1234567);
   printf("%3.8d\n", 1000);
 }
```

c.

```
#include <stdio.h>
void main(void) {
    printf("right-justified:%8d\n", 100);
    printf("left-justified:%-8d\n", 100);
 }
```

4. Write program to read 3 integer numbers by using **scanf( ) and print average.**

**5.** Write program to inputs characters from the keyboard and displays them in reverse case that is, it prints uppercase as lowercase and lowercase as uppercase. To stop the program, enter (.).

6. Write a program that accepts a character as input data and determines whether the character is a lowercase letter.
(Hint: A lowercase letter is any character that's >= 'a' and <='z')

7. Modify the program written for exercise 6 to also determine whether the entered character is an uppercase letter.

8. Write a program that first determines whether an entered character is a lowercase or an uppercase letter . If the letter is lowercase, determine and print its position in the alphabet. For example, if the entered letter is c, the program should print 3 because c is the third letter in the alphabet.

9. Write a program that accepts a character as input data and determines whether the character is a letter, digit or special character by display appropriate message.

10. Modify a program written for exercise 7 to entered N character and determine whether the character is letter , digit or special character.

11. How create header file in c++?