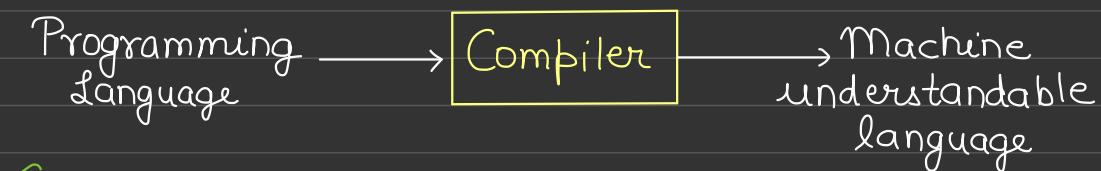


Lecture 1 : What the heck is C++?

Programming Language

We as a programmer wants to communicate with the computer so as to perform some specific task. We know that computer only understands binary language. We will be using programming language which would be converted to binary language by some process. This process is known as compilation and is done by the compiler.



C++

Here we are going to discuss C++ programming language. C++ is a high-level, general purpose programming language.

Why C++ ?

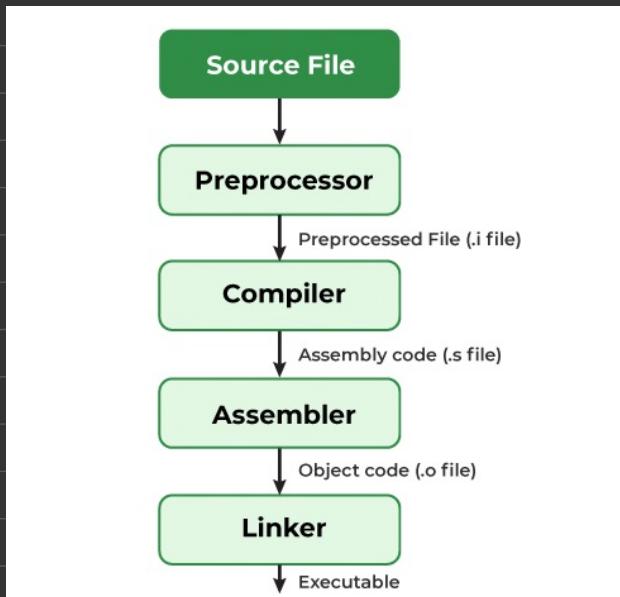
- * Object oriented programming
- * Control over system resources & memory.
- * Efficient & high speed.

Uses of C++

- * System / Software development
- * Game development
- * Real time system
- * Backend can be created.

Compilation process

Programmer writes a code which is known as the source code. This source code is given to the compiler for compilation process and now at the final stage, we get machine understandable language.



After all these four stages, we are getting an executable file with extension as .exe

Homework of Lecture-1

Q1. What is high level & low level programming language? Is C++ a low level or high level language?

High level language is a programming language that allows a programmer to write programs which are independent of a particular type of computer. It is easy to read, write & maintain as it is written in English like words.

Low level language is represented in form of 0s & 1s which are machine instructions.

C++ is a mid-level language that is both high level & low level language as when we require low level functionalities, it incorporates the features such as pointers concept, memory mapping etc.

Lecture 2 : Namaste Dunia program

Create a file named say main.cpp

```
main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     cout<<"Namaste Dunia"<<endl;
5     return 0;
6 }
```

int main()

This is the starting point of the program. The curly brackets represents the scope of int main

function.

return 0

This represents successful execution of the program & operating system gets to know about the success.

Note → Any non-zero value represents unsuccessful execution of the program.

`cout << "Namaste Dunia" << endl` → standard definition of cout is present in std namespace. Also std namespace is present in the iostream header file.

Now we need to include header file & also tell that we are using std namespace.

Note → cout is a keyword which is used to print something on the screen

→ insertion operator

`cout << "Hello" ;` → Termination of a line
→ print this content on screen

Double inverted comma indicates string.

Note → endl is a keyword which means to go to a new line.

`#include <iostream>`, using namespace std
iostream is a header file having input & output facilities, std namespace etc. Now with the line using namespace std, we are telling that we are using std namespace.

Output

The screenshot shows a dark-themed C++ IDE interface. On the left, there's a file tree with 'C++' expanded, showing 'main.cpp' and 'main.exe'. A handwritten note '↳ executable file' points to 'main.exe'. The main area contains the following C++ code:

```
#include<iostream> // Input , output facilities, std namespace
using namespace std; // We are using standard namespace here
int main(){}
cout<<"Namaste Dunia" << endl; // This is used to print content on the screen
return 0; // represents successful execution of the program
```

To the right of the code, the word 'Output' is written above a terminal window. The terminal shows the command line and its output:

```
PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++> cd "c:\Users\Asus\Desktop\Bhavya\Low
+ main.cpp -o main" ; if ($?) { .\main }
Namaste Dunia
PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++>
```

A handwritten arrow points from the terminal output towards the word 'Output'.

Homework of Lecture-2

Q1. Alternative of endl.

There is an escape sequence character named as \n.

cout << "Namaste Dunia\n" ; character → acting as new line

Q2. How to use cout without including the line using namespace std?

Std :: cout << "Namaste Dunia" ;

↳ This means use definition of cout present in Std.

Q3 What is the meaning of return -1 ?
This indicates unsuccessful execution of the program.

Q4 What is preprocessor directive ?
These are lines of source file where the 1st non-whitespace character is #. A preprocessor is a program that is invoked by compiler to process code before compilation

#include <iostream>

Lecture 3 : Variables & Datatypes

Variables

It is a named storage memory location. It is used to store data.

int marks = 50 ;
 ↑ name of variable
 ↑ data type ↑ data to store
cout << marks ; 50 will be the output

Datatypes

It specifies type of information & size of data.

int → integer (4B)
char → character (1B)
bool → boolean (1B)

Variable declaration

```
int x ; // No value stored  
cout << x ; // Some garbage value is printed
```

Variable definition

```
int y = 20 ; // Value is stored in variable  
cout << y ; // 20 is the output
```

Variable manipulation

```
int z = 23 ;  
cout << z ; // 23  
z = 24  
cout << z ; // 24
```

Code + Output

The screenshot shows a code editor interface with a dark theme. On the left, there's a file tree with 'LECTURE - 3' expanded, showing files 'main.cpp' and 'main.exe'. The main area displays the following C++ code:

```
main.cpp > main()  
1 #include <iostream>  
2 using namespace std;  
3 int main(){  
4     // This is a comment  
5     int age = 20;  
6     cout << "My age is " << age << endl;  
7     // Variable declaration  
8     int x;  
9     cout << "Value of x = " << x << endl;  
10    // Variable definition  
11    int y = 20;  
12    cout << "Value of y = " << y << endl;  
13    // Variable manipulation  
14    int z = 21;  
15    cout << "Value of z = " << z << endl;  
16    z = 23;  
17    cout << "Value of z after manipulation = " << z << endl;  
18    return 0;  
19 }
```

At the bottom, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected, showing the command line and its output:

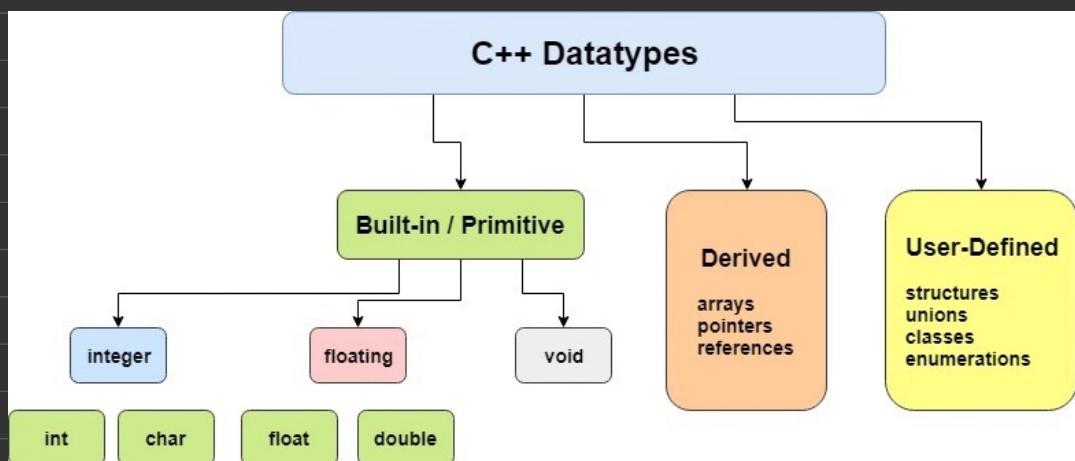
```
PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture - 3> cd "c:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture - 3\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }  
My age is 20  
Value of x = 6422376  
Value of y = 20  
Value of z = 21  
Value of z after manipulation = 23
```

Comments

Comments are ignored by the compiler & is for our understanding.

→ comment start
// This is a comment

Categorization of data types



```
int x = 5; // integer
float y = 3.14; // decimal
char z = 'a'; // character in single quotes
double a = 55.656; // decimal
bool b = 0; // false can be written
bool c = true; // 1 can be written
```

Note → integer data is stored in the form of 0s & 1s.

Code + Output

LECTURE - 3

- main.cpp
- main1.cpp
- main1.exe

```
main1.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){}
4     // integer data type
5     int a = 5;
6     // float data type
7     float b = 5.23;
8     // double data type
9     double c = 55.69887;
10    // boolean
11    bool d = true; // 1 can be written also
12    bool e = false; // 0 can be written also
13    // Printing the data types
14    cout<<a<<endl;
15    cout<<b<<endl;
16    cout<<c<<endl;
17    cout<<d<<endl;
18    cout<<e<<endl;
19
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture - 3\" ; if ($?) { g++ main1.cpp -o main1 } ; if ($?)
5
5.23
55.6989
1
0
```

> OUTLINE

Data types size

system dependent



Type	Bits	Range
int	16	-32768 to -32767
unsigned int	16	0 to 65535
signed int	16	-31768 to 32767
short int	16	-31768 to 32767
unsigned short int	16	0 to 65535
signed short int	16	-32768 to -32767
long int	32	-2147483648 to 2147483647
unsigned long int	32	-2147483648 to 2147483647
signed long int	32	0 to 4294967295
float	32	3.4E-38 to 3.4E+38
double	64	1.7E-308 to 1.7E+308
long double	80	3.4E-4932 to 3.4E+4932
char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127

Note → To check the size of the data type, we can use a function named as `sizeof`. It gives result in bytes.

```
int x = 5;  
cout << sizeof(x); // 4
```

```
// sizeof function  
cout << "Size of integer data type = " << sizeof(a) << endl;
```

Size of integer data type = 4

) output

Signed & unsigned integer

Signed integer means both +ve & -ve can be stored.
Unsigned integer means only +ve can be stored.

Range of signed integer \Rightarrow -2^{n-1} to $2^{n-1} - 1$

Range of unsigned integer \Rightarrow 0 to $2^n - 1$

$n \Rightarrow$ no. of bits

Note → As we can represent boolean by only 1 bit & rest 7 bits are wasted, then why we are storing in 1B. This is due to the fact 1B is smallest addressable memory.

Redefinition of variable

This concept won't work in the same scope.

Error

```
int main () {
    int age = 12 ;
    int age = 13 ;
    :
}
```

No error

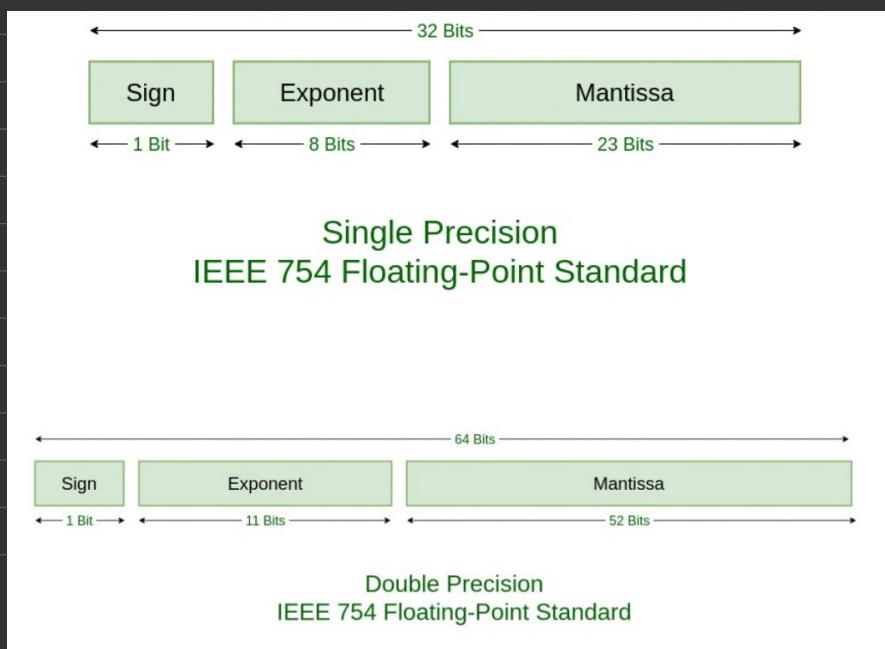
```
int main () {
    int age = 12 ;
    {
        int age = 13 ;
    }
}
```

Homework of Lecture-3

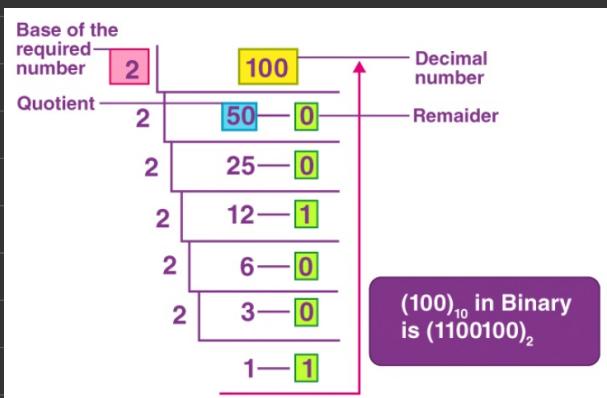
Q1. How is data stored in memory ?

Integer, char, boolean is stored as binary representation

Decimal numbers are stored with the help of IEEE 754 Standard



Q2. How to convert from decimal to binary and vice-versa?



64 32 16 8 4
 ↗ ↗ ↗ ↗ ↗
(1100100)₂

Now to convert to decimal, multiply each bit by weight.

$$64 + 32 + 4 = 100$$

Q3. How to create my own namespace?

Syntax to create namespace is

→ keyword

namespace namespace-name {

```
1 #include<iostream>
2 using namespace std;
3 namespace mynamespace{
4     void fun(){
5         cout<<"Hello";
6     }
7 }
8 namespace mynamespace1{
9     void fun(){
10        cout<<"Hello World";
11    }
12 }
13 using namespace mynamespace;
14 int main(){
15     fun();
16     return 0;
17 }
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C+
f (?) { g++ main.cpp -o main } ; if (?) { .\main
Hello

Lecture 4 : User input in C++

We can take input from the user with the help of `cin` keyword.

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // Declaration of variable
5     int age;
6     // Outputting the message on the screen
7     cout<<"Enter your age : ";
8     // Taking input from user and storing in the age variable
9     cin>>age;
10    // Printing the age
11    cout<<"Your age = "<<age;
12
13 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture - 4> cd "c
4\" ; if (\$?) { g++ main.cpp -o main } ; if (\$?) { .\main }
Enter your age : 20 → input from user
Your age = 20

From the above code snippet, we get to know that we are taking input in `age` variable
→ takes i/p & stores in age variable
`cin>>age`

Note → Remember to press enter after giving the input.

```
bool x ;  
cin >> x ; // Here we have to give 0 or 1 and  
not false or true.
```

It is important to note that definition of cin
is also present in std namespace.

Homework of Lecture-4

Q1. Explain cin.fail(), cin.ignore() &
getline function.

* cin.fail() returns true if last cin command
failed & false otherwise

* cin.ignore() is used to ignore specific
number of characters or characters upto
delimiter.

cin.ignore(n, delimiter)

* getline is used to read a line of text
from i/p stream.

```
String input ;  
getline(cin, input) ;  
→ present in std
```

Lecture 5 : Control flow statements

if statement

If the condition inside the if is true, then if block will be executed.

→ Rule

Syntax of if statement

→ multiple or single

if (condition) {

}

Note → If the condition is false, nothing will be executed.

```

C:\Users\Asus\Desktop\Bhavya\Low Level Design>main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     int budget = 25;
5     // if statement
6     if(budget>=20){
7         cout<<"You can buy chips"<<endl;
8     }
9     return 0;
10 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\Asus\Desktop\Bhavya\Low Level Design>g++ main.cpp -o main ; if ($?) {
You can buy chips
○ PS C:\Users\Asus\Desktop\Bhavya\Low Level Design>

```

if-else statement

If the condition inside the if is true, then if block will be executed, otherwise else block will be executed.

Syntax of if-else

if (condition) {

} → no condition

else {

}

```

C:\Users\Asus\Desktop\Bhavya\Low Level Design>main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     int age = 17;
5     if(age>=18){
6         cout<<"You can vote"<<endl;
7     }
8     else{
9         cout<<"You can not vote";
10    }
11 }
12

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\Asus\Desktop\Bhavya\Low Level Design>g++ main.cpp -o main ; if ($?) {
You can not vote
○ PS C:\Users\Asus\Desktop\Bhavya\Low Level Design>

```

if - else if statement

This is used to check multiple conditions one by one.

Syntax of if - else if

if (condition 1) {

} *→ Can be multiple*
else if (condition 2) {

}

Note → Once any condition is satisfied, then other conditions won't be checked.

if - else if - else statement
else statement is added in addition to the previous statement.

Syntax of if-else if- else if (cond1){

}
else if (cond2){ --- }

else{ --- }

```
main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // if else if statement
5     int marks = 60;
6     if(marks >= 80){
7         cout<<"Grade = A"<<endl;
8     }
9     else if(marks >= 70){
10        cout<<"Grade = B"<<endl;
11    }
12    else if(marks >= 60){
13        cout<<"Grade = C"<<endl;
14    }
15    return 0;
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PC

```
PS C:\Users\Asus\Desktop\Bhavya\Low Level Design 5\" ; if ($?) { g++ main.cpp -o main } ; if ($?) Grade = C
```

```
main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // if else if else statement
5     int marks = 59;
6     if(marks >= 80){
7         cout<<"Grade = A"<<endl;
8     }
9     else if(marks >= 70){
10        cout<<"Grade = B"<<endl;
11    }
12    else if(marks >= 60){
13        cout<<"Grade = C"<<endl;
14    }
15    else{
16        cout<<"You Failed"<<endl;
17    }
18    return 0;
19 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PC

```
PS C:\Users\Asus\Desktop\Bhavya\Low Level Design 5\" ; if ($?) { g++ main.cpp -o main } ; if ($?) You Failed
```

Nested if statement

One or more if inside another if is called as nested if.

Syntax of nested if

```
if(cond1){  
    --  
    if(cond2){  
        -- --  
    }  
}  
}
```

```
main.cpp > main()  
1 #include<iostream>  
2 using namespace std;  
3 int main(){  
4     // nested if statement  
5     int height = 160;  
6     int weight = 56;  
7     if(height >= 150){  
8         if(weight <= 60){  
9             cout<<"Good Category BMI"<<endl;  
10    }  
11 }  
12 return 0;  
13 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\L
5\" ; if (\$?) { g++ main.cpp -o main } ; if (\$?) { ./m
Good Category BMI

Note → We can add else & else if in the nested if statement.

Lecture 6 : Switch Statement

This acts as an alternative to using a series of if-else statements.

Syntax of switch statement

```
switch(expression){  
    case value1 :  
        -- --
```

```
        break;
```

; → multiple cases would be there
default :

```
-- --
```

This is fall through behaviour.

```
C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture - 5> cd "c:\U  
5\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { ./main }  
Enter your grade : B  
You have got good marks
```

Note → After each & every case except default, we need to write the break statement. If break is not written, then whenever a case is matched, all the subsequent cases would be executed.

default case would be executed only when no case is matched. It is not mandatory & can be discarded.

Conditions pertaining to switch

- 1) Expression inside switch can be int, char & enum only.
- 2) Case values should be unique.
- 3) No range checking in case.
↳ case (age ≥ 18)
- 4) Range checking can be done in expression.
- 5) Execution is done in a sequential fashion.

Homework of Lecture-6

Q1. What can not be written inside the

case values ?

- 1) Variables can't be written
- 2) Functions can't be written
- 3) Float is not allowed
- 4) Strings is not allowed

Lecture 7 : Ternary operator

Syntax of ternary operator

Condition ? expression_if_true : expression_if_false

```
C++ main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // Ternary operator
5     int age = 17;
6     age>=18?cout<<"You can vote":cout<<"You can not vote";
7     return 0;
8 }
```

Annotations on the code:

- A curly brace under the closing brace of the main function is labeled "Condition".
- An arrow points from the condition "age >= 18" to the text "true case".
- An arrow points from the condition "age >= 18" to the text "false case".

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture - 5> cd "5\" ; if (\$?) { g++ main.cpp -o main } ; if (\$?) { .\main }
You can not vote

Lecture 8 : Loops

If we want to do something repeatedly, we will be using loops.

for loop

Syntax of for loop

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

}

```
main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // for loop
5     cout<<"Printing Numbers from 1 to 5"<<endl;
6     for(int i = 1;i<=5;i=i+1){
7         cout<<i<<" ";
8     }
9     return 0;
10 }
```

PROBLEMS

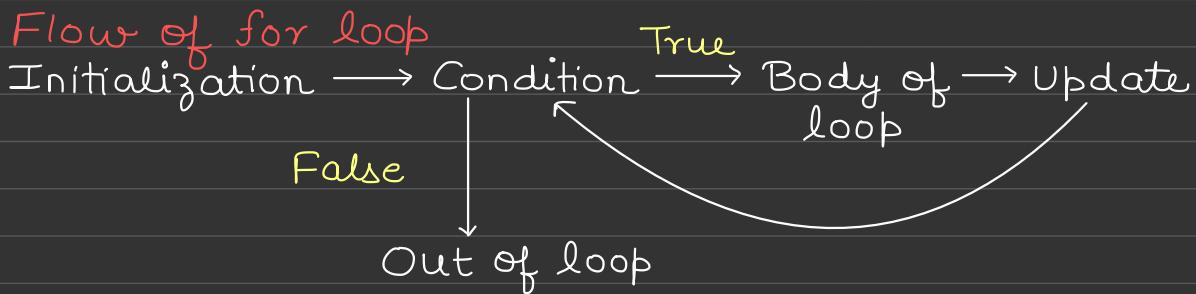
OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

- PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture 8\" ; if (\$?) { g++ main.cpp -o main } ; if (\$?) { .\main }
- Printing Numbers from 1 to 5
1 2 3 4 5



Note → We can write for loop without initialization, updation & condition. All of them are optional.

break keyword

It is used to exit from the loop.

```

1 #include<iostream>
2 using namespace std;
3 int main(){
4     // break keyword
5     for(int i = 1;i<=5;i=i+1){
6         cout<<i<<" ";
7         if(i == 3){
8             break;
9         }
10    }
11    return 0;
12 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Asus\Desktop\Bhavya\Low Level
8\" ; if ($?) { g++ main.cpp -o main } ; i
1 2 3
```

As $i == 3$ condition becomes true, break is encountered & hence we come out of the loop.

Output
1 2 3

continue keyword

It is used to skip one iteration of loop.

```

1 #include<iostream>
2 using namespace std;
3 int main(){
4     // continue keyword
5     for(int i = 1;i<=5;i=i+1){
6         if(i == 3){
7             continue;
8         }
9         cout<<i<<" ";
10    }
11    return 0;
12 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

- PS C:\Users\Asus\Desktop\Bhavya\Low Level
8\" ; if (\$?) { g++ main.cpp -o main } ;
1 2 4 5

Here 3 won't be printed as for $i = 3$, we are skipping that iteration.

Output

1 2 4 5

while loop

Syntax of while loop

initialization

while (condition) {

updation

}

```

C: main.cpp > ⌂ main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // while loop
5     cout<<"Printing numbers from 1 to 5 "<<endl;
6     int i = 1; // initialization
7     while(i <= 5){
8         cout<<i<<" ";
9         i=i+1; // updation
10    }
11    return 0;
12 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture 8\" ; if (\$?) { g++ main.cpp -o main } ; if (\$?) { .\main
Printing numbers from 1 to 5
1 2 3 4 5

It is just another way of writing the code.

Homework of Lecture-8

Q1 Print counting from 1 to 100.

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     for(int i = 1;i <= 100;i++){
5         cout<<i<<endl;
6     }
7     return 0;
8 }
```

Q2 Print counting from 100 to 1.

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     for(int i = 100;i >= 1;i--){
5         cout<<i<<endl;
6     }
7     return 0;
8 }
```

Q3 Print name 50 times-

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     for(int i = 1;i <= 50;i++){
5         cout<<"Bhavya"<<endl;
6     }
7     return 0;
8 }
```

Q4 Print counting from 0 to -10.

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     for(int i = 0;i >= -10;i--){
5         cout<<i<<endl;
6     }
7     return 0;
8 }
```

Q5 Print table of 7.

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     for(int i = 1;i <=10;i++){
5         cout<<7*i<<endl;
6     }
7     return 0;
8 }
```

Q6 Print characters from A-Z & a-z.

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     for(char ch = 'A';ch <= 'Z';ch++){
5         cout<<ch<<endl;
6     }
7     return 0;
8 }
```

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     for(char ch = 'a';ch <= 'z';ch++){
5         cout<<ch<<endl;
6     }
7     return 0;
8 }
```

Q7 How to run for loop without initialization, condition & updation?

```
for( ; ; ) {  
    cout << "Hello"; }  
} This is an infinite  
loop.
```

Lecture 9 : Do-while loop

do-while loop

This loop will run atleast once irrespective of the condition.

Syntax

initialization

```
do {
```

} while (condition)

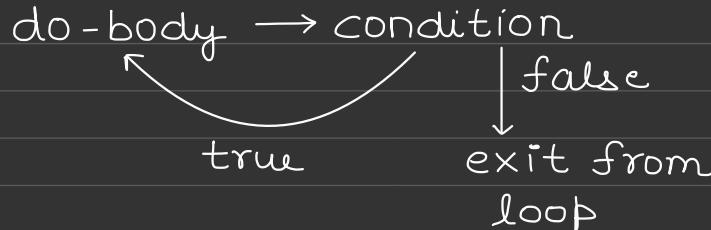
```
C++ main.cpp > main()  
1 #include<iostream>  
2 using namespace std;  
3 int main(){  
4     // do while loop  
5     int i = 1;  
6     do{  
7         cout << i << " ";  
8         i++;  
9     }while(i <= 5);  
10    return 0;  
11 }
```

PROBLEMS OUTPUT DEBUG CONSOLE

```
PS C:\Users\BHAVYA\OneDrive\Desktop  
.\\main  
1 2 3 4 5
```

In the 1st iteration, the condition is not checked.

Flow of do-while loop



Note → do while loop is not used much as compared to other loops.

Nested Loops

This simply means loop inside a loop.

```
C++ main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // nested loops
5     for(int i=1;i<=3;i++){
6         for(int j=1;j<=5;j++){
7             cout<<j<< " ";
8         }
9         cout<<endl;
10    }
11    return 0;
12 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

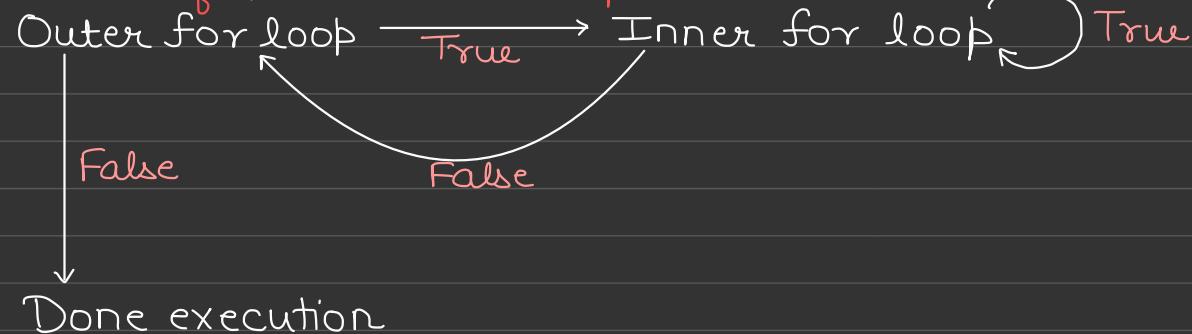
```
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> .\main
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

The outer for loop runs for 3 times i.e for $i = 1, 2 \& i = 3$. For each i , the inner loop runs for 5 times i.e $j = 1, j = 2, j = 3, j = 4 \& j = 5$.

Hence we can see the output as

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

Flow of nested for loops



Here in the flow we are checking how the control goes from one loop to another depending upon the conditions.

Homework of Lecture-9

Q1. What is the output of the following code?

```
for (int i = 0 ; i <= 5 ; i++) ;  
{  
    cout << "Hello";  
}
```

It is important to note ; after the for loop bracket which indicates there is no body of the for loop. Hence Hello would be printed only once & it is not due to the for loop.

```
G+ main.cpp > main()  
1 #include<iostream>  
2 using namespace std;  
3 int main(){  
4     // Homework Question - 1  
5     for(int i=0;i<=5;i++);  
6     {  
7         cout<<"Hello";  
8     }  
9     return 0;  
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd  
.\\main }  
Hello
```

Q2. What is the output of the following code?

```
int i;
if(cin>>i){
    cout<<"Hello";
}
```

If we give any value as i/p & it is a success, then if block would be executed, thereby printing Hello on the screen.

```
C++ main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // Homework Question - 2
5     int i;
6     if(cin>>i){
7         cout<<"Hello";
8     }
9     return 0;
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD
.main }
123
Hello

```
C++ main.cpp > main()
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // Homework Question - 3
5     int i = 1;
6     if(cout<<i){
7         cout<<endl;
8         cout<<"Hello";
9     }
10 }
11 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD
.main }
1
Hello

Note → If inside if condition cout is written & printing is done successfully, then if block would get executed.

Lecture 10 : Operators in C++

Operators are symbols by which we can perform mathematical & logical operations.

Unary operators

Pre-increment & post-increment

Pre-increment means first increment & then use.
Post-increment means first use & then increment.

```
int a = 5;  
cout << (++a); // pre-increment
```

```
int b = 5;  
cout << (b++); // post-increment
```

```
1 #include<iostream>  
2 using namespace std;  
3 int main(){  
4     // Unary Operators  
5     int a = 5;  
6     cout<<++a<<endl; // pre-increment  
7     int b = 5;  
8     cout<<b++; // post-increment  
9     return 0;  
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PC

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd ".\main"

6
5

6 is the output in case of pre-increment.

5 is the output in case of post-increment. Here value of b becomes 6.

Note → In case of pre-decrement, first decrement & then use.
In case of post-decrement, first use & then decrement

Arithmetic operators

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // Arithmetic operators
5     int a = 5;
6     int b = 10;
7     // Addition
8     cout<<"Addition = "<<a+b<<endl;
9     // Subtraction
10    cout<<"Subtraction = "<<a-b<<endl;
11    // Multiplication
12    cout<<"Multiplication = "<<a*b<<endl;
13    // Division
14    cout<<"Division = "<<a/b<<endl;
15    // Remainder
16    cout<<"Remainder = "<<a%b<<endl;
17    return 0;
18 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\_main"
Addition = 15
Subtraction = -5
Multiplication = 50
Division = 0
Remainder = 5
```

* It is important to note that $5/10$ is coming out to be 0. This is due to concept of typecasting.

$\frac{(\text{int})}{(\text{int})} \rightarrow \text{int}$, $\frac{(\text{float})}{(\text{int})} \rightarrow \text{float}$

* Remainder / Modulus ($\% \cdot$) is a heavy operation. Try to avoid it if possible.

Relational operators

These are used in comparing the values and result in only 2 values i.e true or false only.

The operators are \div

- 1) >
 - 2) <
 - 3) > =
 - 4) < =
 - 5) = =
 - 6) | =

```

1 #include<iostream>
2 using namespace std;
3 int main(){
4     // Relational operators
5     int a = 5;
6     int b = 10;
7     // Greater than
8     cout<<(a>b)<<endl;
9     // Greater than or equal to
10    cout<<(a>=b)<<endl;
11    // Lesser than
12    cout<<(a<b)<<endl;
13    // Lesser than or equal to
14    cout<<(a<=b)<<endl;
15    // Equal to
16    cout<<(a==b)<<endl;
17    // Not equal to
18    cout<<(a!=b)<<endl;
19
20    return 0;
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

.\main
0
0
1
1
0
1

```

$0 \rightarrow \text{false}$

$1 \rightarrow \text{true}$

$= =$ means equal to operator

\neq means not equal to operator

Logical operators

If we want to check multiple conditions, we will be using these operators.

$\&\&$ → Logical AND

If all conditions are true, it results in true else it gives false.

$\| \rightarrow$ Logical OR

If any one of the condition is true, then result is true.

$!$ → Logical NOT

It is used as negation. It makes true as false and false as true.

```

1 #include<iostream>
2 using namespace std;
3 int main(){
4     // Logical Operators
5     // Logical AND
6     cout<<(true && true && false)<<endl;
7     cout<<(true && true && true)<<endl;
8     // Logical OR
9     cout<<(false || false || false)<<endl;
10    cout<<(true || false || false)<<endl;
11    // Logical NOT
12    cout<<(!true)<<endl;
13    cout<<(!false)<<endl;
14    return 0;
15 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS S

```

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:/User
. \main "
0
1
0
1
0
1

```

It is important to note the below cases :-

(cond₁ || cond₂ || cond₃)

If cond₁ is true, then cond₂ & cond₃ are not checked, this is known as Short circuit.

(cond₁ && cond₂ && cond₃)

If cond₁ is false, then cond₂ & cond₃ are not checked, this is known as Short circuit.

Assignment operator

$a = b$

\curvearrowleft operator

$a += b \Rightarrow a = a + b$

$a -= b \Rightarrow a = a - b$

$a *= b \Rightarrow a = a * b$

$a /= b \Rightarrow a = a / b$

$a \% = b \Rightarrow a = a \% b$

These are acting as shorthand notation.

Bitwise operators

These works on the bit level.

1) & → Bitwise AND

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

4 & 5

$$\begin{array}{r} 4 \Rightarrow 000\dots100 \\ 5 \Rightarrow 000\dots101 \\ \hline 000\dots100 \end{array} \rightarrow 4$$

2) | → Bitwise OR

a	b	a b
0	0	0
0	1	1
1	0	1
1	1	1

4 | 5

$$\begin{array}{r} 4 \Rightarrow 000\dots100 \\ 5 \Rightarrow 000\dots101 \\ \hline 000\dots101 \end{array} \rightarrow 5$$

3) $\sim \rightarrow$ Bitwise NOT

a	$\sim a$
0	1
1	0

~ 5

$$5 \rightarrow 000__101$$
$$\sim 5 \rightarrow 111__010 \quad (\text{negative number})$$

Find 2s complement

$$\begin{array}{r} & | \\ 000__101 & \\ \hline 000__110 & \rightarrow 6 \end{array}$$

But we found out number was -ve, hence
 $\sim 5 = -6$

4) $<< \Rightarrow$ Left shift

$$5 << 1 \rightarrow i$$

$$5 \rightarrow 000__101$$
$$5 << 1 \rightarrow 000__1010 \rightarrow 10$$

In short multiply by 2^i (kind of but not sure)

5) $>> \Rightarrow$ Right Shift

$$5 >> 1 \rightarrow i$$

$$5 \rightarrow 000__101$$
$$5 >> 1 \rightarrow 000__010 \rightarrow 2$$

In short divide by 2^i (kind of but not sure)

6) Bitwise XOR

a	b	$a \wedge b$
0	0	0
0	1	1
1	0	1
1	1	0

$$5 \wedge 4 \Rightarrow \begin{array}{r} 000__101 \\ 000__100 \\ \hline 000__001 \rightarrow 1 \end{array}$$

Note $\rightarrow a \wedge a = 0$
 $a \wedge 0 = a$

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // Bitwise operators
5     // Bitwise AND
6     cout<<(4&5)<<endl;
7     // Bitwise OR
8     cout<<(4|5)<<endl;
9     // Bitwise NOT
10    cout<<(~5)<<endl;
11    // Bitwise Left Shift
12    cout<<(5<<1)<<endl;
13    // Bitwise Right Shift
14    cout<<(5>>1)<<endl;
15    // Bitwise XOR
16    cout<<(4^5)<<endl;
17
18 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\\"
.\\main }
4
5
-6
10
2
1

Homework of Lecture-10

Q1. On what data type bitwise operators will work?

It will work with integer & boolean data types.

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     cout<<(~false)<<endl;
5     cout<<(~true)<<endl;
6     return 0;
7 }
```

PROBLEMS OUTPUT DEBUG CONSOLE

```
PS C:\Users\BHAVYA\OneDrive\Desktop
.\main }
-1
-2
```

Q2. Will modulus operator work on float data type?

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     cout<<(2.5%1)<<endl;
5     return 0;
6 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

```
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\Users\BHAVYA\OneDrive\Desktop\LLD\" ; if
.\main }
main.cpp: In function 'int main()':
main.cpp:4:15: error: invalid operands of types 'double' and 'int' to binary 'operator%'
    cout<<(2.5%1)<<endl;
          ^~~^~
```

Lecture 11 : Functions in C++

Functions take some i/p data, process the data & gives o/p.

It is a way to group code into a single unit. It helps in organizing code, making it readable and maintainable.



Syntax of function → very intuitive → not mandatory
return_type function_name(p1, p2 ...){
----- } function body

?

Note → If function does not return anything, we can use void.

```
1 #include<iostream>
2 using namespace std;
3 // Function to print counting from 1 to 5
4 // Function definition and declaration
5 void printCounting(){
6     for(int i=1;i<=5;i++){
7         cout<<i<<" ";
8     }
9 }
10 int main(){
11     // Function Call
12     printCounting();
13     return 0;
14 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\\"
.\\main }
1 2 3 4 5

To use a function, we will call the function.

Sum of two numbers using function

```
1 #include<iostream>
2 using namespace std;
3 // Function to find sum of 2 numbers
4 int findSum(int x,int y){
5     return x + y;
6 }
7 int main(){
8     int a = 1;
9     int b = 2;
10    // Function Call
11    int result = findSum(a,b);
12    cout<<"Sum = "<<result<<endl;
13    return 0;
14 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:
.\\main"
Sum = 3

$x, y \Rightarrow$ i/p parameters
 $\text{int} \Rightarrow$ return type
 $\text{findSum} \Rightarrow$ name of function

We are storing returned value from the function in result variable in main() function.

Note →

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     int a = 1;
5     int b = 2;
6     // Function Call
7     int result = findSum(a,b);
8     cout<<"Sum = "<<result<<endl;
9     return 0;
10 }
11 // Function to find sum of 2 numbers
12 int findSum(int x,int y){
13     return x + y; 3 function definition
14 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\Users\BHAVYA\On
.\\main"
main.cpp: In function 'int main()':
main.cpp:7:18: error: 'findSum' was not declared in this scope
 int result = findSum(a,b);
 ^~~~~~

We have to declare the function above main function else it will give the error.

Function declaration \Rightarrow $\text{int findSum(int } a, \text{ int } b);$

```

1 #include<iostream>
2 using namespace std;
3 // Function declaration
4 int findSum(int x,int y);
5 int main(){
6     int a = 1;
7     int b = 2;
8     // Function Call
9     int result = findSum(a,b);
10    cout<<"Sum = "<<result<<endl;
11    return 0;
12 }
13 // Function definition
14 int findSum(int x,int y){
15     return x + y;
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POR
 PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd ".\main"
 Sum = 3

Now there won't be any error if we declare the function before calling the function.

Note → It is a good practice to use camel case when naming functions.

findSumOfThreeNumbers
 small ← ↴ capital ←

Using return in case of void return type

void func_name(---){
 ;
 ;
 ;}

return; // No value sent here.

3

Homework of Lecture-11

Q1. Print numbers from 1 to 100 with the help of functions.

```
1 #include<iostream>
2 using namespace std;
3 // Homework Question - 1
4 void printNumbersFrom1To100(){
5     for(int i=1;i<=100;i++){
6         cout<<i<<" ";
7     }
8 }
9 int main(){
10     printNumbersFrom1To100();
11     return 0;
12 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

```
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\Users\BHAVYA\OneDrive\Desktop\LLD\" ; if ($?) { g++ main.cpp -o main } ; if (?) { .\main }
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 4
7 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100
```

Q2. Make a function to find simple interest.

```
1 #include<iostream>
2 using namespace std;
3 // Homework Question - 2
4 float findSimpleInterest(float P,float R,float T){
5     return P*R*T/100.0;
6 }
7 int main(){
8     float P = 1256;
9     float R = 2.7;
10    float T = 1.0;
11    float result = findSimpleInterest(P,R,T);
12    cout<<"Simple Interest = "<<result;
13    return 0;
14 }
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

```
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\Users\BHAVYA\OneDrive\Desktop\LLD\" ; if ($?) { g++ main.cpp -o main } ; if (?) { .\main }
Simple Interest = 33.912
```

Q3. Print prime numbers from 1 to 100.

```
1 #include<iostream>
2 using namespace std;
3 // Homework Question - 3
4 bool isPrime(int num){
5     bool flag = false;
6     int factor = 0;
7     for(int i = 2;i<num;i++){
8         if(num%i==0){
9             factor++;
10            break;
11        }
12    }
13    return factor!=1;
14 }
15 void printPrimeNumbersFrom1to100(){
16     for[int i=2;i<=100;i++){
17         if(isPrime(i)){
18             cout<<i<<" ";
19         }
20     }
21 }
22 int main(){
23     printPrimeNumbersFrom1to100();
24     return 0;
25 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

Q4. Eligible to vote or not.

```
1 #include<iostream>
2 using namespace std;
3 // Homework Question - 4
4 bool isEligibleToVote(int age){
5     return age>=18;
6 }
7 int main(){
8     int age = 19;
9     if(isEligibleToVote(age)){
10         cout<<"Eligible";
11     }
12     else{
13         cout<<"Not Eligible";
14     }
15     return 0;
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CC

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\Users\BHAVYA\OneDrive\Desktop\LLD"> .\main

Eligible

Q5. Make a function to calculate SIP.

```
1 #include<iostream>
2 #include<math.h>
3 using namespace std;
4 // Homework Question - 5
5 float sipCalculator(float val,float payments,float rate){
6     float monthly_rate = (rate/1200);
7     float fv = val * (1+monthly_rate)*((pow((1+monthly_rate),payments)-1)/monthly_rate);
8     return fv;
9 }
10 int main(){
11     float result = sipCalculator(32000,10,2.5);
12     cout<<"SIP = "<<result;
13     return 0;
14 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

```
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\Users\BHAVYA\OneDrive\Desktop\LLD\" ; if ($?) { g
.\main }
SIP = 323684
```