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++?

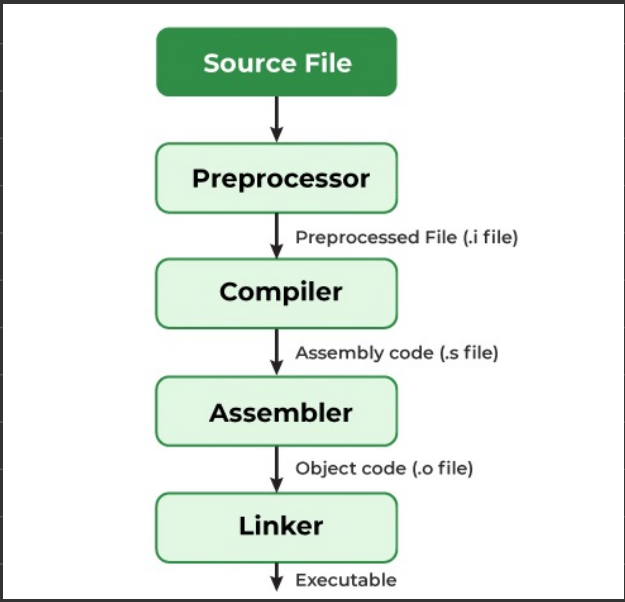
* Object oriented programming
* Control over system resources & memory.
* Efficient & high speed.

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()

```cpp
#include<iostream>
using namespace std;
int main(){
    cout<<"Namaste Dunia"<<endl;
    return 0;
}
```

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

cout << "Hello" ; → Termination of a line
↑ insertion operator
↳ 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

Homework of Lecture-2

Q1. Alternative of endl.

There is an escape sequence character named as \n.

```
cout <<"Namaste Dunia\n" ;
```

acting as new line character

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

Data types
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
```

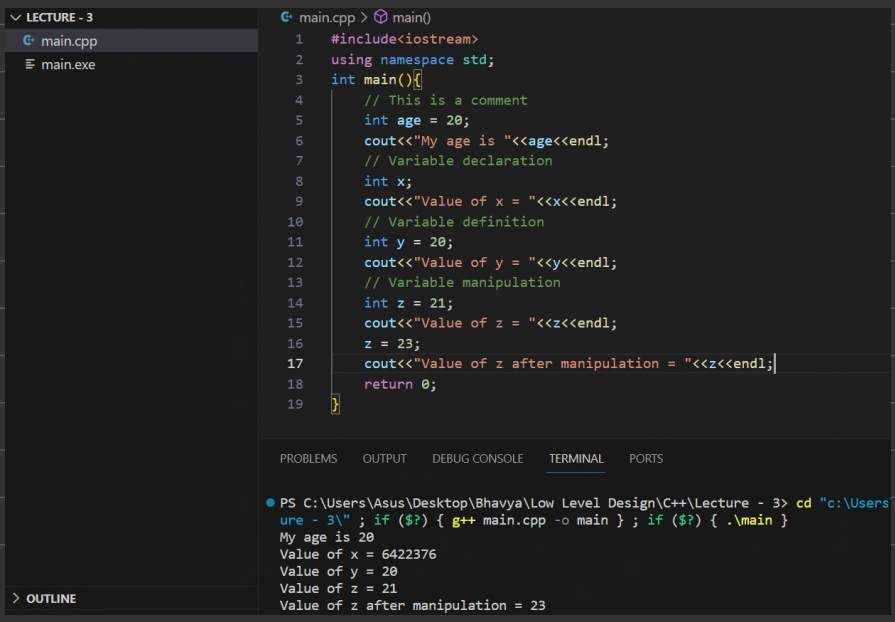
Variable definition

```
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



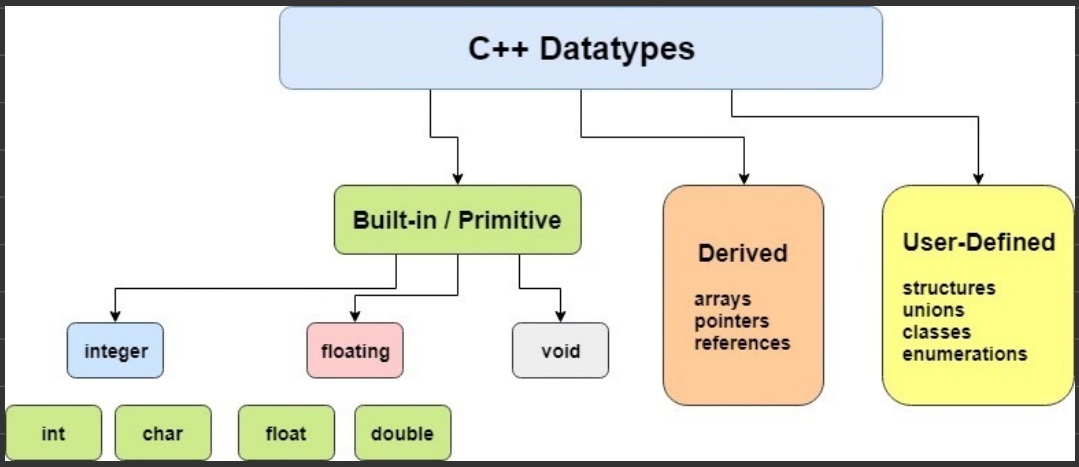
Comments

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

```
// This is a comment
```

(// → comment start)

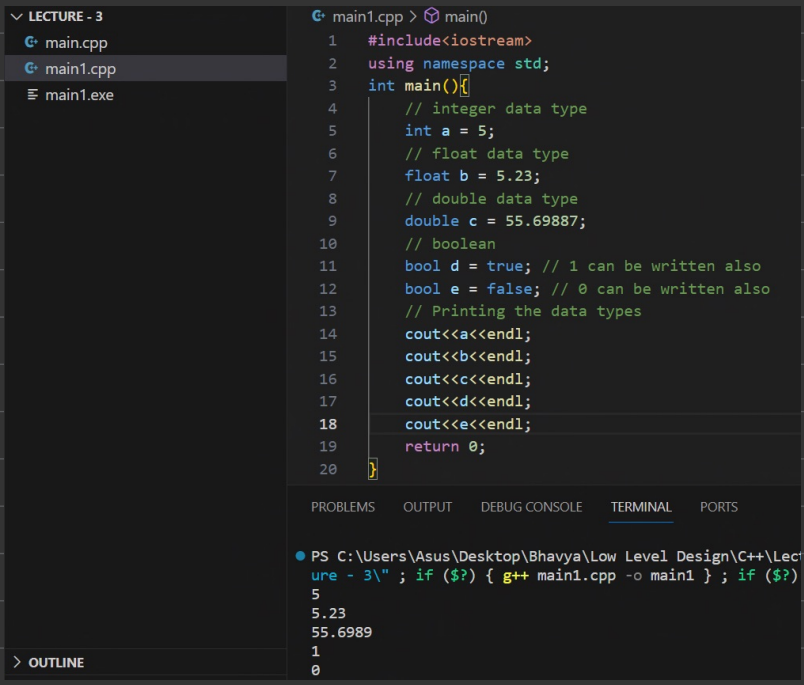
Categorization of data types




```
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



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;
```

output

```
Size of integer data type = 4
```

Signed & unsigned integer

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

Range of signed integer ⇒ -2^{n-1} to $2^{n-1}-1$
Range of unsigned integer ⇒ 0 to 2^n-1

n ⇒ 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

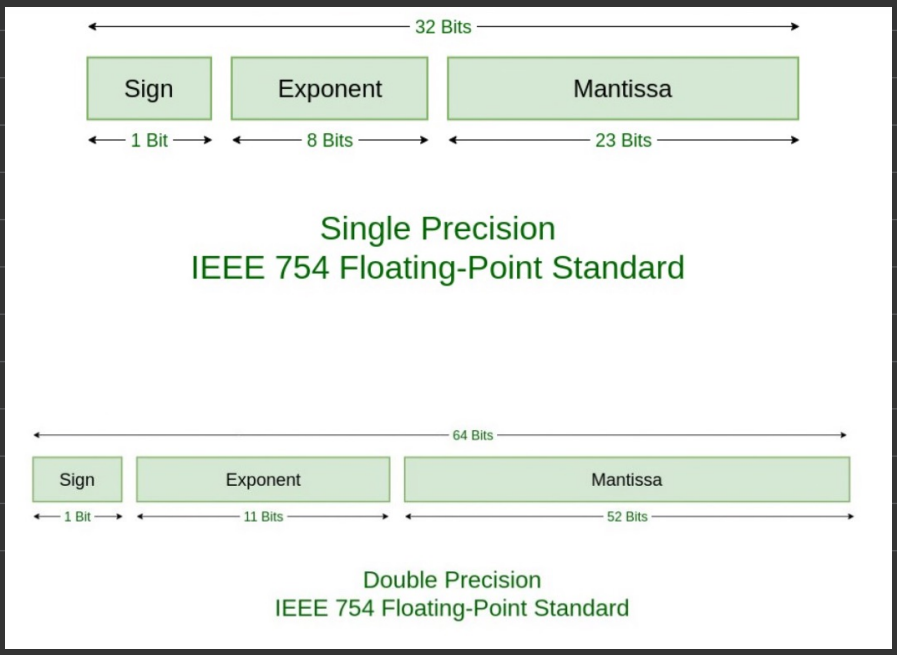
```
int main () {
    int age = 12 ;
    {
      int age = 13 ;
    }
      ;
}
```

Homework of Lecture-3

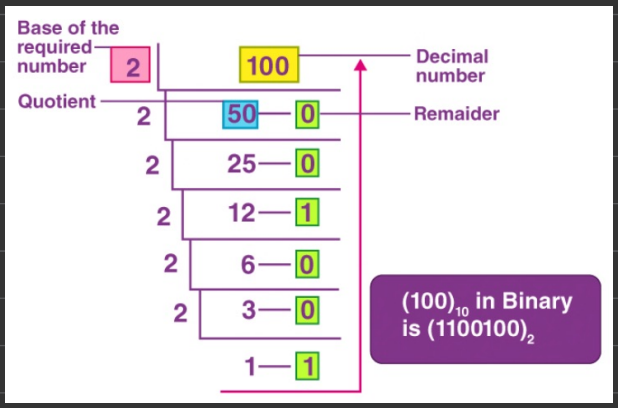
Q1. How is data stored in memory ?

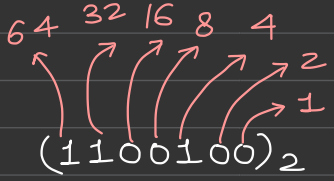
Integer, char, boolean is stored as binary representation

Decimal numbers are stored with the help of IEEE 754 standard



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





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

namespace (keyword) 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.

```cpp
#include<iostream>
using namespace std;
int main(){
    // Declaration of variable
    int age;
    // Outputing the message on the screen
    cout<<"Enter your age : ";
    // Taking input from user and storing in the age variable
    cin>>age;
    // Printing the age
    cout<<"Your age = "<<age;
    return 0;
}
```

```
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

cin >> age → takes i/p & stores in age variable

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, delimitor)
```

* 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.

```cpp
#include<iostream>
using namespace std;
int main(){
    int budget = 25;
    // if statement
    if(budget>=20){
        cout<<"You can buy chips"<<endl;
    }
    return 0;
}
```

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

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 {
    ---
}
```

```cpp
#include<iostream>
using namespace std;
int main(){
    int age = 17;
    if(age>=18){
        cout<<"You can vote"<<endl;
    }
    else{
        cout<<"You can not vote";
    }
    return 0;
}
```

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

if-else if statement

This is used to check multiple conditions one by one.

Syntax of if-else if

```
if (condition 1) {
    ---
}
else if (condition 2){   → Can be multiple
    ---
}
```

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

```cpp
#include<iostream>
using namespace std;
int main(){
    // if else if statement
    int marks = 60;
    if(marks >= 80){
        cout<<"Grade = A"<<endl;
    }
    else if(marks >= 70){
        cout<<"Grade = B"<<endl;
    }
    else if(marks >= 60){
        cout<<"Grade = C"<<endl;
    }
    return 0;
}
```

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

if-else if-else statement

else statement is added in addition to the previous statement.

Syntax of if-else if-else

```
if (cond 1){
    ----
}
else if (cond 2){ ---- }
else { ---- }
```

```cpp
#include<iostream>
using namespace std;
int main(){
    // if else if else statement
    int marks = 59;
    if(marks >= 80){
        cout<<"Grade = A"<<endl;
    }
    else if(marks >= 70){
        cout<<"Grade = B"<<endl;
    }
    else if(marks >= 60){
        cout<<"Grade = C"<<endl;
    }
    else{
        cout<<"You Failed"<<endl;
    }
    return 0;
}
```

```
PS C:\Users\Asus\Desktop\Bhavya\Low Level Desig
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()

```
#include<iostream>
using namespace std;
int main(){
    // nested if statement
    int height = 160;
    int weight = 56;
    if(height >= 150){
        if(weight <= 60){
            cout<<"Good Category BMI"<<endl;
        }
    }
    return 0;
}
```

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.

```
main.cpp > main()
1   #include<iostream>
2   using namespace std;
3   int main(){
4       char grade;
5       cout<<"Enter your grade : ";
6       cin>>grade;
7       // switch statement
8       switch(grade){
9           case 'A' : cout<<"You have got excellent marks"; break;
10          case 'B' : cout<<"You have got good marks";break;
11          case 'C' : cout<<"You have got bad marks";break;
12          default : cout<<"You have failed";
13      }
14      return 0;
15  }

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS 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

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  }
```

true case

false case

condition

```
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){
        -----
}
```

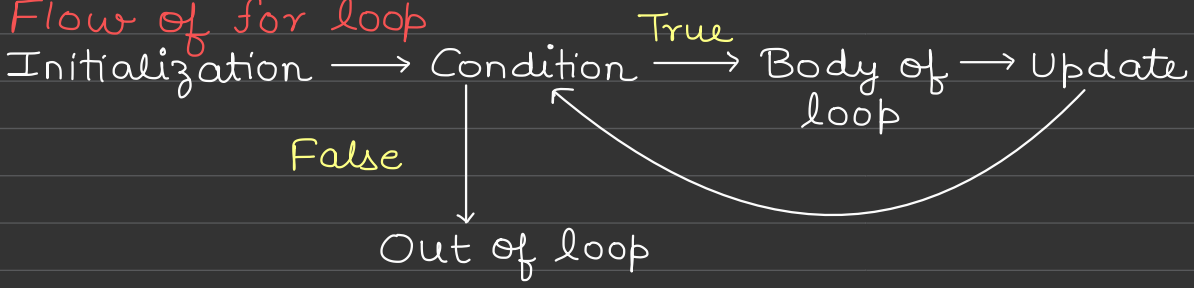
main.cpp > main()

```cpp
#include<iostream>
using namespace std;
int main(){
    // for loop
    cout<<"Printing Numbers from 1 to 5"<<endl;
    for(int i = 1;i<=5;i=i+1){
        cout<<i<<" ";
    }
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

Flow of for loop



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.

```cpp
#include<iostream>
using namespace std;
int main(){
    // break keyword
    for(int i = 1;i<=5;i=i+1){
        cout<<i<<" ";
        if(i == 3){
            break;
        }
    }
    return 0;
}
```

```
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.

```cpp
#include<iostream>
using namespace std;
int main(){
    // continue keyword
    for(int i = 1;i<=5;i=i+1){
        if(i == 3){
            continue;
        }
        cout<<i<<" ";
    }
    return 0;
}
```

```
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
}
```

main.cpp > main()

```cpp
#include<iostream>
using namespace std;
int main(){
    // while loop
    cout<<"Printing numbers from 1 to 5 "<<endl;
    int i = 1; // initialization
    while(i <= 5){
        cout<<i<<" ";
        i=i+1; // updation
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lectu
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.

```cpp
#include<iostream>
using namespace std;
int main(){
    for(int i = 1;i <= 100;i++){
        cout<<i<<endl;
    }
    return 0;
}
```

Q2 Print counting from 100 to 1.

```cpp
#include<iostream>
using namespace std;
int main(){
    for(int i = 100;i >= 1;i--){
        cout<<i<<endl;
    }
    return 0;
}
```

Q3 Print name 50 times.

```cpp
#include<iostream>
using namespace std;
int main(){
    for(int i = 1;i <= 50;i++){
        cout<<"Bhavya"<<endl;
    }
    return 0;
}
```

Q4 Print counting from 0 to -10.

```cpp
#include<iostream>
using namespace std;
int main(){
    for(int i = 0;i >= -10;i--){
        cout<<i<<endl;
    }
    return 0;
}
```

Q5 Print table of 7.

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

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

```cpp
#include<iostream>
using namespace std;
int main(){
    for(char ch = 'A';ch <= 'Z';ch++){
        cout<<ch<<endl;
    }
    return 0;
}
```

```cpp
#include<iostream>
using namespace std;
int main(){
    for(char ch = 'a';ch <= 'z';ch++){
        cout<<ch<<endl;
    }
    return 0;
}
```

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

```
for( ; ; ){
    cout << "Hello";
}
```

This is an infinite loop.