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++?

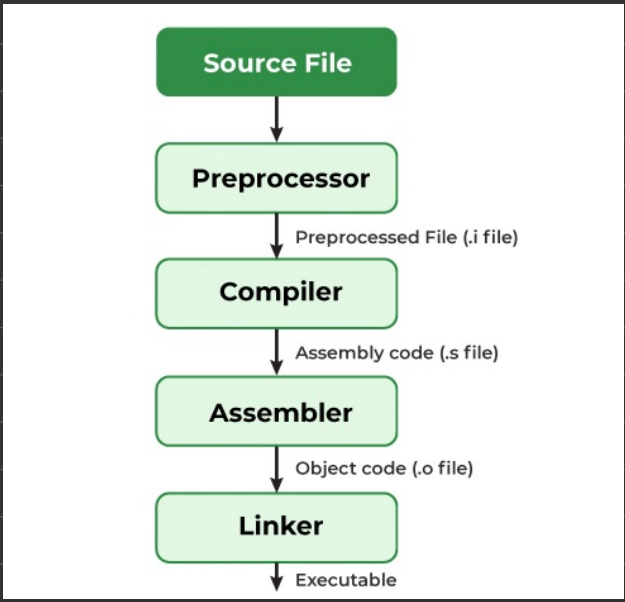
* Object oriented programming
* Control over system resources & memory.
* Efficient & high speed.

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
```

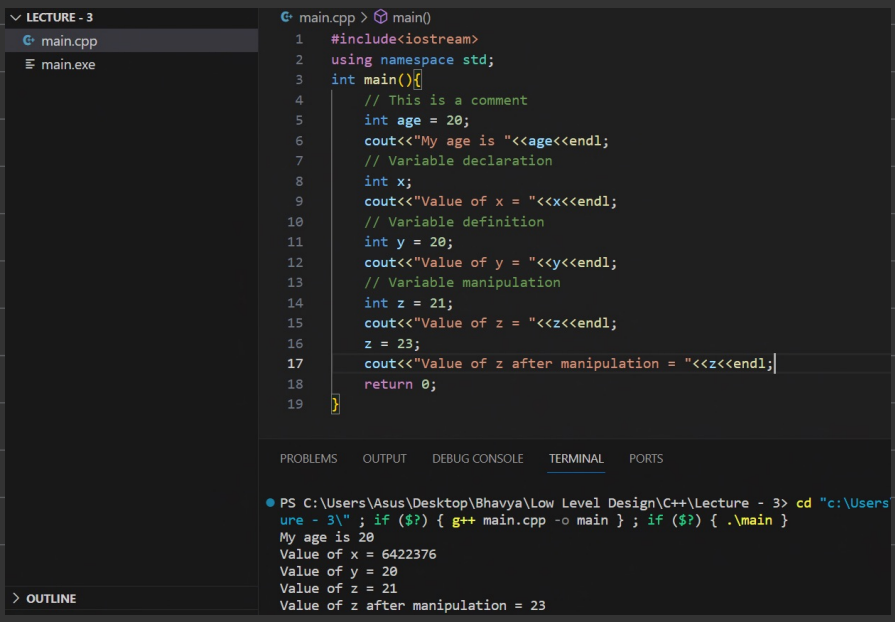
Variable definition

```
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



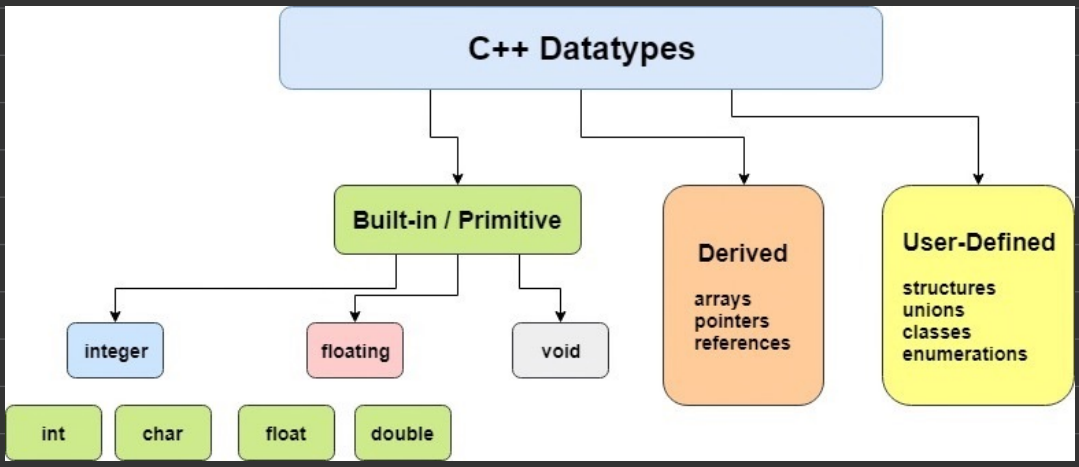
Comments

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

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

(// → comment start)

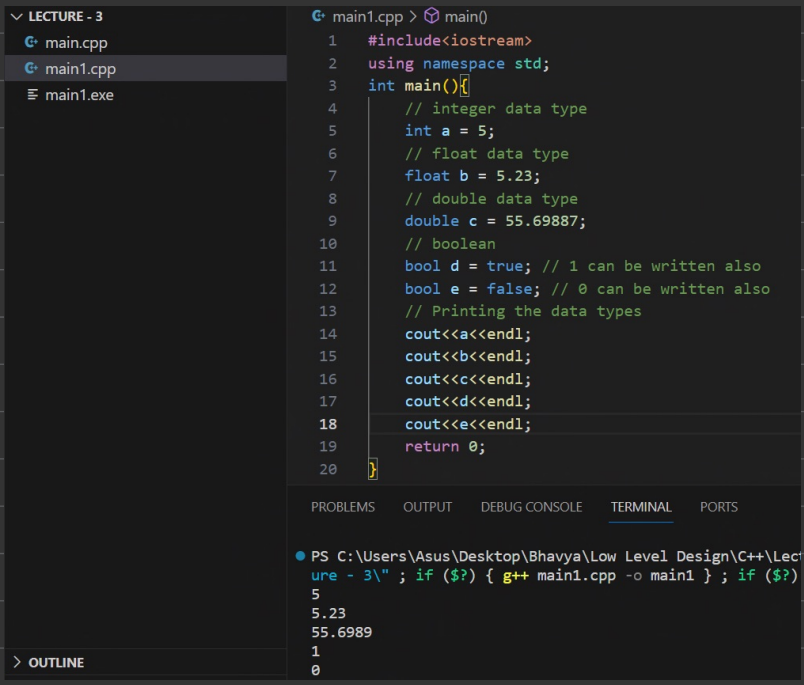
Categorization of data types




```
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

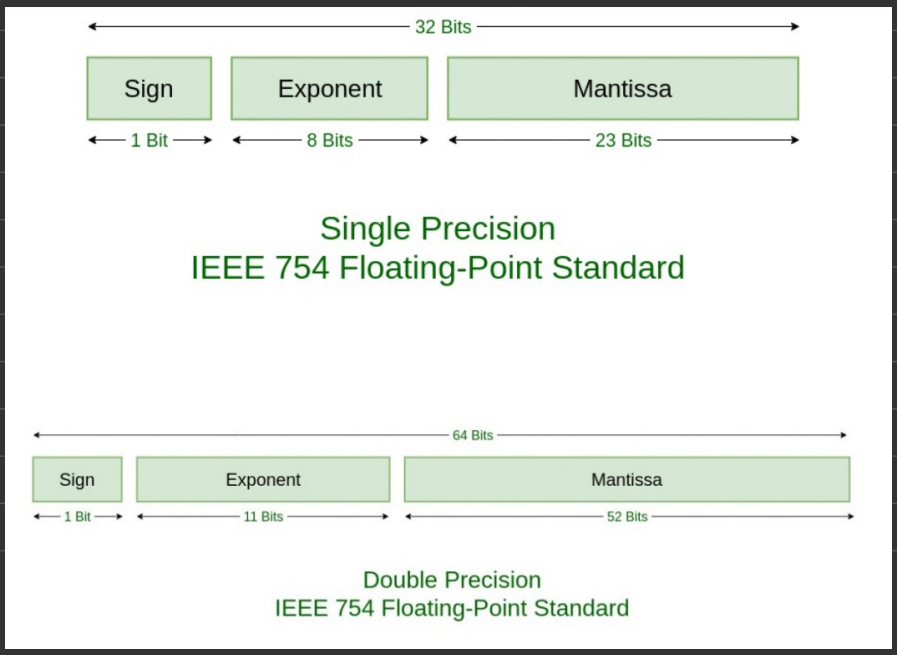
```
int main () {
    int age = 12 ;
    {
      int age = 13 ;
    }
      ;
}
```

Homework of Lecture-3

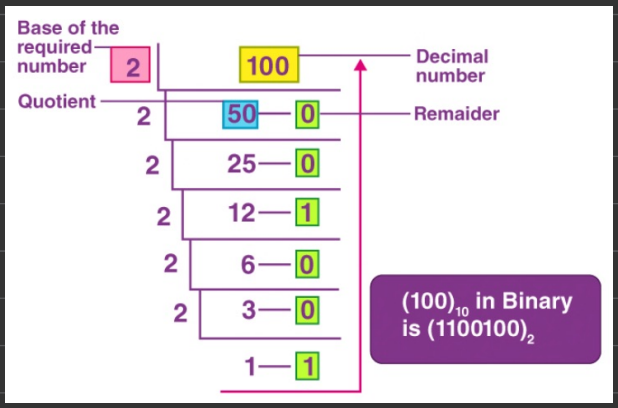
Q1. How is data stored in memory ?

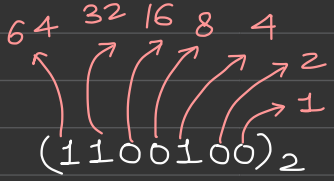
Integer, char, boolean is stored as binary representation

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){
        -----
}
```

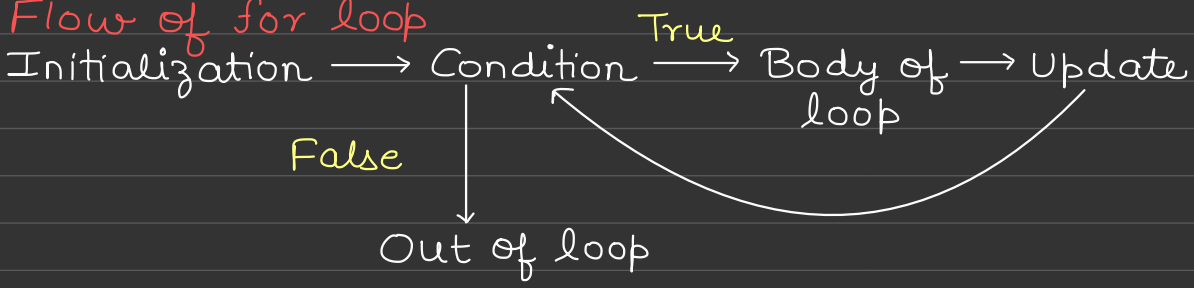
main.cpp > main()

```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.

Lecture 9 : Do-while loop

do-while loop

This loop will run atleast once irrespective of the condition.

Syntax

```
initialization
do {
   -----
   -----
} while (condition)
```

main.cpp > main()

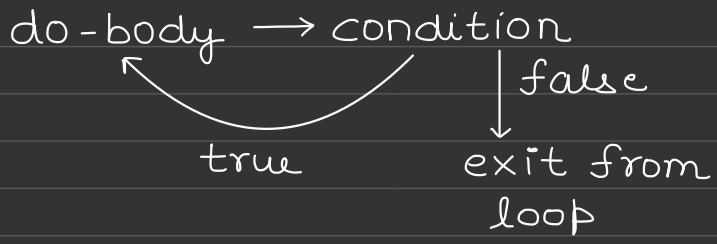
```
1   #include<iostream>
2   using namespace std;
3   int main(){
4       // do while loop
5       int i = 1;
6       do{
7           cout<<i<<" ";
8           i++;
9       }while(i<=5);
10      return 0;
11  }
```

PROBLEMS OUTPUT DEBUG CONSOLE

```
PS C:\Users\BHAVYA\OneDrive\Deskt
 .\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.

main.cpp > main()

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

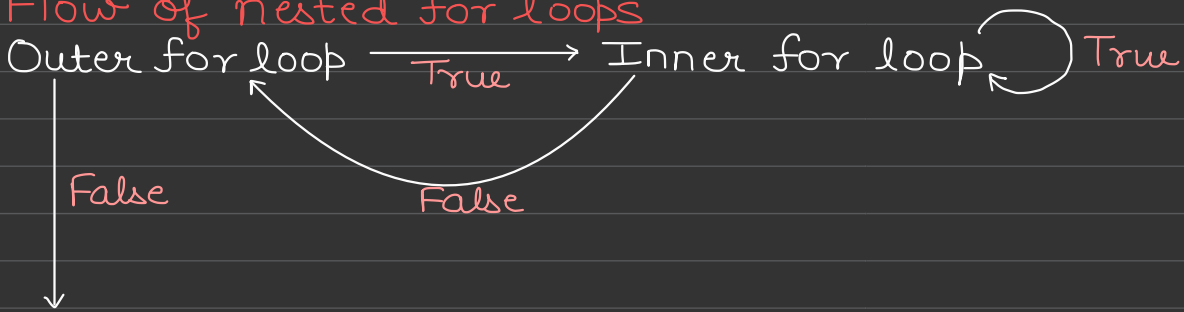
```
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> c
 .\main }
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

The outer for loop runs for 3 times i.e for $i=1$, $i=2$ & $i=3$. 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.

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.

main.cpp > main()

```
#include<iostream>
using namespace std;
int main(){
    // Homework Question - 2
    int i;
    if(cin>>i){
        cout<<"Hello";
    }
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

main.cpp > main()

```
#include<iostream>
using namespace std;
int main(){
    // Homework Question - 3
    int i = 1;
    if(cout<<i){
        cout<<endl;
        cout<<"Hello";
    }
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> c
 .\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
```

a → operand

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

```
#include<iostream>
using namespace std;
int main(){
    // Unary Operators
    int a = 5;
    cout<<++a<<endl; // pre-increment
    int b = 5;
    cout<<b++; // post-increment
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PO

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

Arithemetic operators

```cpp
#include<iostream>
using namespace std;
int main(){
    // Arithemetic operators
    int a = 5;
    int b = 10;
    // Addition
    cout<<"Addition = "<<a+b<<endl;
    // Subtraction
    cout<<"Subtraction = "<<a-b<<endl;
    // Multiplication
    cout<<"Multiplication = "<<a*b<<endl;
    // Division
    cout<<"Division = "<<a/b<<endl;
    // Remainder
    cout<<"Remainder = "<<a%b<<endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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

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

* Remainder / Modulus (%) is a heavy operations. 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 :-

1) >
2) <
3) >=
4) <=
5) ==
6) !=

```cpp
#include<iostream>
using namespace std;
int main(){
    // Relational operators
    int a = 5;
    int b = 10;
    // Greater than
    cout<<(a>b)<<endl;
    // Greater than or equal to
    cout<<(a>=b)<<endl;
    // Lesser than
    cout<<(a<b)<<endl;
    // Lesser than or equal to
    cout<<(a<=b)<<endl;
    // Equal to
    cout<<(a==b)<<endl;
    // Not equal to
    cout<<(a!=b)<<endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
 .\main }
0
0
1
1
0
1
```

0 → false
1 → true

== means equal to operator

!= 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.

|| → 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.

```cpp
#include<iostream>
using namespace std;
int main(){
    // Logical Operators
    // Logical AND
    cout<<(true && true && false)<<endl;
    cout<<(true && true && true)<<endl;
    // Logical OR
    cout<<(false || false || false)<<endl;
    cout<<(true || false || false)<<endl;
    // Logical NOT
    cout<<(!true)<<endl;
    cout<<(!false)<<endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  S

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

↑ operator

a += b ⇒ a = a + b

a -= b ⇒ a = a - b

a *= b ⇒ a = a * b

a /= b ⇒ a = a / b

a %= b ⇒ 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

4 ⇒ 000...100
5 ⇒ 000...101
000...100 → 4

2) | → Bitwise OR

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

4 & 5

4 ⇒ 000...100
5 ⇒ 000...101
000...101 → 5

3) ~ → Bitwise NOT

a	~a
0	1
1	0

~5

5 → 000...101

~5 → 111...010 (negative number)

Find 2s complement

```
   000--101
            1
  ----------
   000--110   → 6
```

But we found out number was -ve, hence

~5 = −6

4) << ⇒ left shift

5 << (1) → i

5 → 000--101

5<<1 → 000--1010 → 10

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

5) >> ⇒ Right shift

5 >> (1) → i

5 → 000--101

5>>1 → 000...010 → 2

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

6) Bitwise XOR

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

$5 \wedge 4 \Rightarrow$

```
000...101
000...100
---------
000...001  → 1
```

Note → $a \wedge a = 0$

$a \wedge 0 = a$

```cpp
#include<iostream>
using namespace std;
int main(){
    // Bitwise operators
    // Bitwise AND
    cout<<(4&5)<<endl;
    // Bitwise OR
    cout<<(4|5)<<endl;
    // Bitwise NOT
    cout<<(~5)<<endl;
    // Bitwise Left Shift
    cout<<(5<<1)<<endl;
    // Bitwise Right Shift
    cout<<(5>>1)<<endl;
    // Bitwise XOR
    cout<<(4^5)<<endl;
    return 0;
}
```

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.

```cpp
#include<iostream>
using namespace std;
int main(){
    cout<<(~false)<<endl;
    cout<<(~true)<<endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE

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

Q2. Will modulus operator work on float data type?

```cpp
#include<iostream>
using namespace std;
int main(){
    cout<<(2.5%1)<<endl;
    return 0;
}
```

```
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:\U
 .\main }
1 2 3 4 5
```

To use a function, we will call the function.

Sum of two numbers using function

```cpp
#include<iostream>
using namespace std;
// Function to find sum of 2 numbers
int findSum(int x,int y){
    return x + y;
}
int main(){
    int a = 1;
    int b = 2;
    // Function Call
    int result = findSum(a,b);
    cout<<"Sum = "<<result<<endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:
 .\main }
Sum = 3
```

x, y ⇒ i/p parameters
int ⇒ return type
findSum ⇒ name of function

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

Note →

```cpp
#include<iostream>
using namespace std;
int main(){
    int a = 1;
    int b = 2;
    // Function Call
    int result = findSum(a,b);
    cout<<"Sum = "<<result<<endl;
    return 0;
}
// Function to find sum of 2 numbers
int findSum(int x,int y){
    return x + y;
}
```

} function definition

```
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 ⇒ `int findSum(int a, int b);`

```cpp
#include<iostream>
using namespace std;
// Function declaration
int findSum(int x,int y);
int main(){
    int a = 1;
    int b = 2;
    // Function Call
    int result = findSum(a,b);
    cout<<"Sum = "<<result<<endl;
    return 0;
}
// Function definition
int findSum(int x,int y){
    return x + y;
}
```

```
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.
}
```

Homework of Lecture-11

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

```cpp
#include<iostream>
using namespace std;
// Homework Question - 1
void printNumbersFrom1To100(){
    for(int i=1;i<=100;i++){
        cout<<i<<" ";
    }
}
int main(){
    printNumbersFrom1To100();
    return 0;
}
```

```
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.

```cpp
#include<iostream>
using namespace std;
// Homework Question - 2
float findSimpleInterest(float P,float R,float T){
    return P*R*T/100.0;
}
int main(){
    float P = 1256;
    float R = 2.7;
    float T = 1.0;
    float result = findSimpleInterest(P,R,T);
    cout<<"Simple Interset = "<<result;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\Users\BHAVYA\C
 .\main }
Simple Interset = 33.912
```

Q3. Print prime numbers from 1 to 100.

```cpp
#include<iostream>
using namespace std;
// Homework Question - 3
bool isPrime(int num){
    bool flag = false;
    int factor = 0;
    for(int i = 2;i<num;i++){
        if(num%i==0){
            factor++;
            break;
        }
    }
    return factor!=1;
}
void printPrimeNumbersFrom1to100(){
    for(int i=2;i<=100;i++){
        if(isPrime(i)){
            cout<<i<<" ";
        }
    }
}
int main(){
    printPrimeNumbersFrom1to100();
    return 0;
}
```

```
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.

```cpp
#include<iostream>
using namespace std;
// Homework Question - 4
bool isEligibleToVote(int age){
    return age>=18;
}
int main(){
    int age = 19;
    if(isEligibleToVote(age)){
        cout<<"Eligible";
    }
    else{
        cout<<"Not Eligible";
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CO

PS C:\Users\BHAVYA\OneDrive\Desktop\LLD> cd "c:\Users\B
 .\main }
Eligible
```

Q5. Make a function to calculate SIP.

```cpp
#include<iostream>
#include<math.h>
using namespace std;
// Homework Question - 5
float sipCalculator(float val,float payments,float rate){
    float monthly_rate = (rate/1200);
    float fv = val * (1+monthly_rate)*((pow((1+monthly_rate),payments)-1)/monthly_rate);
    return fv;
}
int main(){
    float result = sipCalculator(32000,10,2.5);
    cout<<"SIP = "<<result;
    return 0;
}
```

```
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
```

Lecture 12 : Arrays in C++

What is array ?

Array is used to store multiple values of same type in a single variable.

Why arrays ?

Suppose we want to store 100 integer numbers, so we will be making 100 variables. The answer is no. Here in this case we will be using arrays.

Syntax of creating an array

data_type name_of_array [size of array];

Ex → int arr [100]; // Declaration of array

Here 100 is the size of array.

Note → We need to mention size of array when declaring an array.

```
int arr[5] = {1,2,3,4,5}; //Array initialization
int arr[] = {1,2,3,4,5}; //This would work
int arr[3] = {1,2,3,4}; //Error
```

Note → Suppose we have a 5-sized array & we are storing only 3 elements, then rest of the elements will have zero value.

Elements of array are stored at contiguous locations.

```
int arr[3] = {1,2,3};
```

104 108 112 ⇒ memory locations

The difference blw 2 contiguous locations in this case is 4 as the size of integer is 4B.

Note → If say total free space available is 6MB but not contiguous, then we can't make an array of size 6MB.

Accessing elements of array

We can access array elements with the help of index.

Range of index ⇒ [0, size-1]

```
int arr[5] = {1,2,3,4,5};
```


```cpp
#include<iostream>
using namespace std;
int main(){
    // Array Initialization
    int arr[5] = {1,2,3,4,5};
    // Accessing array elements using index
    cout<<"Element at 2nd index = "<<arr[2]<<endl;
    cout<<"Element at 0th index = "<<arr[0]<<endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture-12> c
" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
Element at 2nd index = 3
Element at 0th index = 1
```

Traversing the array

Traversing the array means to print all the elements of an array. This can be done by running a loop from 0^{th} index to $(size-1)^{th}$ index.

Here is the sample code for the same:

```cpp
#include<iostream>
using namespace std;
int main(){
    // Array Initialization
    int arr[5] = {1,2,3,4,5};
    // Traversing the array
    for(int i = 0;i<=4;i++){
        cout<<arr[i]<<" ";
    }
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture-12>
" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
1 2 3 4 5
```

Input in an array

We can take i/p in array with the help of index, loop & cin.

Let us see the code for the same :

```cpp
#include<iostream>
using namespace std;
int main(){
    // Array Declaration
    int arr[5];
    // Taking input from the user
    for(int i = 0;i<=4;i++){
        cout<<"Enter the element at "<<i<<" index"<<endl;
        cin>>arr[i];
    }
    // Traversing the array
    cout<<"Printing the array"<<endl;
    for(int i = 0;i<=4;i++){
        cout<<arr[i]<<" ";
    }
    return 0;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Enter the element at 0 index
1
Enter the element at 1 index
2
Enter the element at 2 index
3
Enter the element at 3 index
4
Enter the element at 4 index
5
Printing the array
1 2 3 4 5
```

Ex → Print sum of array elements.

Initialize sum variable as 0. Now just traverse the array & update the sum variable.

```cpp
#include<iostream>
using namespace std;
int main(){
    int sum = 0;
    // Array Declaration
    int arr[5] = {1,2,3,4,5};
    // Finding the sum
    for(int i = 0;i<=4;i++){
        sum = sum + arr[i];
    }
    // Printing the sum
    cout<<"Sum of array elements = "<<sum;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\
" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\ma
Sum of array elements = 15
```

Updation of sum variable is done in 9th line of code.

updated sum ↑ , ↑ current element

```
sum = sum + arr[i];
      ↑
  previous value
```

Passing arrays to function

Arrays are passed to the function along with the size of array.

→ don't pass size here

```
void printArray(int arr[], int size){
    ____
    ____
}
//Function call
printArray(arr, 3);
```

```cpp
#include<iostream>
using namespace std;
void printArray(int arr[],int size){
    for(int i=0;i<size;i++){
        cout<<arr[i]<<" ";
    }
}
int main(){
    int arr[5] = {1,2,3,4,5};
    printArray(arr,5); // Function call
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C+
" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\
1 2 3 4 5
```

Initialization of array with value 0

```
int arr[5]={0}; //only in case of 0
int arr[5]={-1}; //only at 0th index, -1 is there & rest all will have 0
int arr[5]; // Garbage value
```

2D arrays in C++

	c0	c1	c2	c3
r0				
r1				
r2				
r3				

```
int arr[4][4]; //Declaration
```

- `arr` → name of 2D array
- `[4]` → no. of rows
- `[4]` → no. of columns

int arr[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}}

	c0	c1	c2	c3
r0	1	2	3	4
r1	5	6	7	8
r2	9	10	11	12
r3	13	14	15	16

Accessing elements in 2D arrays

```
cout << arr[0][3]; //0th row & 3rd column
cout << arr[2][1]; //2nd row & 1st column
```

```cpp
#include<iostream>
using namespace std;
int main(){
    // Array Initialization
    int arr[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
    // Accessing elements
    cout<<arr[0][3]<<endl;
    cout<<arr[2][1]<<endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture-12> cd "c:\Users\Asu
" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
4
10
```

Traversing the 2D array

We have to run nested loops & outer loop will be for rows & inner loop will be for columns.

need to be passed

```cpp
#include<iostream>
using namespace std;
void printArray(int arr[][4],int row,int col){
    for(int i = 0;i < row;i++){ // For rows
        for(int j = 0;j < col;j++){ // For columns
            cout<<arr[i][j]<<" ";
        }
        cout<<endl;
    }
}
int main(){
    // Array Initialization
    int arr[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
    // Traversing the elements of 2D array
    printArray(arr,4,4);
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lecture-12> cd "c:\Users\Asus\D
" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Note → When we pass 2D array to function, we need to pass no. of columns in [ ]. Also taking input is also simple as we have to do cin instead of cout.

Homework of Lecture-12

Q1. Initialize an array with -1 in each block.

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
    int arr[10];
    memset(arr,-1,sizeof(arr));
    for(int i = 0;i < 10;i++){
        cout<<arr[i]<<" ";
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

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

memset(arr, -1, sizeof(arr));

name of array

initialization value

Q2. Write a function to populate an array with multiple of 10.

```cpp
#include<iostream>
using namespace std;
void multipleOf10(int arr[],int size){
    int val = 1;
    for(int i=0;i<size;i++){
        arr[i] = 10*val;
        val++;
    }
}
int main(){
    int arr[10];
    multipleOf10(arr,10); // Function call
    for(int i = 0;i < 10;i++){
        cout<<arr[i]<<" ";
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Asus\Desktop\Bhavya\Low Level Design\C++\Lect
" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
10 20 30 40 50 60 70 80 90 100
```

Q3. Write a function to flip 0 to 1 & 1 to 0 in an array.

```cpp
#include<iostream>
using namespace std;
void flip0And1(int arr[],int size){
    for(int i = 0;i<size;i++){
        arr[i] = !(arr[i]);
    }
}
int main(){
    int arr[5] = {1,0,0,1,1};
    flip0And1(arr,5);
    for(int i = 0;i<5;i++){
        cout<<arr[i]<<" ";
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Asus\Desktop\Bhavya\Low Level De
" ; if ($?) { g++ main.cpp -o main } ; if ($
0 1 1 0 0
```

Q4. Write a function to reverse an array.

```cpp
#include<iostream>
using namespace std;
void reverseArray(int arr[],int size){
    int i = 0;
    int j = size - 1;
    while(i<=j){
        swap(arr[i],arr[j]);
        i++;
        j--;
    }
}
int main(){
    int arr[5] = {1,2,3,4,5};
    reverseArray(arr,5);
    for(int i = 0;i<5;i++){
        cout<<arr[i]<<" ";
    }
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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