③
```
#include <iostream>
using namespace std;
int glo=6;
void sum(){
int a;
cout<<glo;
}
int main(){
int glo=9;
glo=78;
//int a=14;
// int b=15;
int a=14, b=15;
float pi=3.14;
char c='d';
sum();
cout<<glo;
return 0;
}
```

DATE 2021 (2/12)

Point: Local precedence more than global

=> bool istrue = false;
cout<<istrue;

true = 1

Rules to Declare Variables in C++:

* Variable names in C++ can range from 1 to 255 characters.
* All variable names must begin with a letter of the alphabet or an underscore(_).
* After the first initial letter, variable names can also contain letters and numbers.
* Variable names are case sensitive.
* No spaces or special characters are allowed.
* You cannot use a C++ keyword (a reserved word) as a variable name.

continue	or eq	volatile
co-await (since C++20)	or_eq	wchar_t
co-return (since C++20)	private	while
co-yield (since C++20)	protected	xor
	public	xor_eq
decltype (since C++11)	reflexpr (reflection TS)	

DATE 201

DATA TYPE SIZE AND RANGE IN C++ (For C++ 64BIT)

DATA TYPE	SIZE (IN BYTES)	RANGE
short int	2	-32768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 to 4	1 wide character

SALMAN PAPER PRODUCT

Page No.

(2) Assignment operators: (-->) used to assign values or variables

```
int a=3, b=9
char d='d'
```

DATE 2 0 1

(3) Comparision operators: (Following are comparison operators)

```
Cout<< "the value of a==b is"<<(a==b)<<endl;
Cout<< "the value of a!=b is"<<(a!=b)<<endl;
Cout<< "the value of a>=b is"<<(a>=b)<<endl;
Cout<< "the value of a<=b is"<<(a<=b)<<endl;
Cout<< "the value of a>b is"<<(a>b)<<endl;
Cout<< "the value of a<b is"<<(a<b)<<endl;
```

!= not equal

(4) Logical operators:

```
Cout<<"Following are the logical operators in C++"<<endl;
Cout<<" The value of this logical and operator
    (a=b) && (a<b) is :" <<((a==b) && (a<b))
    <<endl;

Cout<<" The value of this logical or operator
    (a=b) || (a<b) is: "<<((a=b) || (a<b))<<endl;

Cout<<" The value of this logical not operator
    (!(a==b) is: <<(!(a==b))<<endl;
```

H.w: (Make a C++ program to use these operators)

Practical

DATE 2 0 1

```
#include<iostream>
using namespace std;
int C = 45;
int main(){
 int a, b, c;
 cout<<" Enter the value of a: "<<endl;
 cin a;
 cout<<" Enter the value of b: "<<endl;
 cin b;
 ~~cout<<" Enter the :~~
 C = a+b;
 cout<<" The sum is "<<c<<endl
 cout<<" The global c is "<<::C;
 return 0;
}
```

↑ Build in Data types

```
Float d = 34.4F;
long double e = 34.4;
cout<<" The value of d is "<<d<<endl<<" The
      value of e is "<<e;
```

(By default it will choose option of double however if we put (F) then it choose Float)

#Literals: Float, double and long Double *

```
cout<<" The size of 34.4 is "<<sizeof(34.4)<<endl;
cout<<" The size of 34.4F is "<<sizeof(34.4F)<<endl;
cout<<" The size of 34.4F is "<<sizeof(34.4F)<<endl;
cout<<" The size of 34.4l is "<<sizeof(34.4l)<<endl;
cout<<" The size of 34.4L is "<<sizeof(34.4L)<<endl;
```

⇒ Reference Variables:- (&Y)

```
Float x=455;
Float &Y=x;
Cout<<x<<endl;
Cout<<Y<<endl;
```

DATE 201

⇒ Typecasting:- (changing variable) int → float:

One Data type to another.

```
int a=45;
Float b=45.46;
Cout<<"The value of a is" <<(Float)a;
Cout<<"The value of b is" <<(int)b;
        Or//
Cout<<"The value of a is"<<Float(a);
Cout<<"The value of b is"<< int(b);
int c= int(b);

Cout<<"The Expression is"<<a+b<<endl; => 90.46
Cout<<"The Expression is "<<a+ int(b)<<endl; => 90
Cout<<"The Expression is "<<a+(int)b <<endl; => 90
```

Constants

Values Change:

```
int main() {
  int a=34;
  Cout<<" The Value of a was"<<a;
  int a=45;
  Cout<<" The value of a is"<<a;
  return 0;
}
```

[Constant int a=3] → Constant number
Keyword — Can't change number

a=45; Can't Concluded b/c it Constant number

C++ Control Structures:

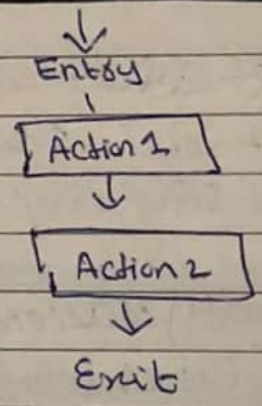
Lec#4

DATE 2 0 1

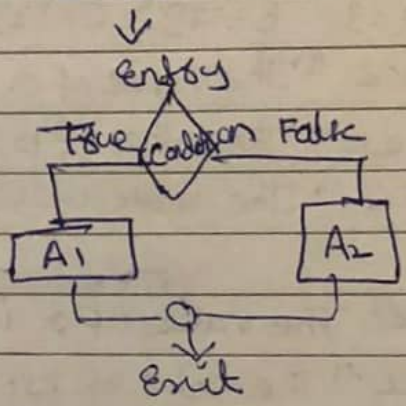
1. Sequence Structure
2. Selection Structure
3. Loop Structure

} Basic Control Structure

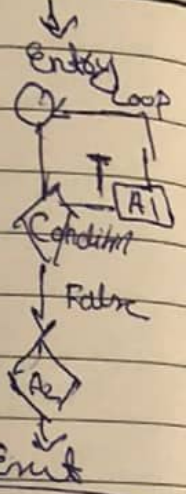
=> 1. Sequence Structure:



2. Selection Structure



3. Loop Structure



1. If-else Statement:

```
if(i<3){ int i=0
    Cout<<i;
    i++;
}
```

2) => if-else-ladder

```
if((age<18)&&(age>0)){
```

practical

```
else if (age<1){
    Cout<<"You are
    not yet born"
    <<endl;}
```

practical:

```
int main(){
    int age;
    Cout<<"Tell me your age"<<endl;
    Cin>>age;
    if (age<18){
        Cout<<"You can not come
        to my party"<<endl;}
    else if (age==18){
        Cout<<"You are a kid and
        you will get a kid pass
        to the party"<<endl;}
    else{
        Cout<<"You can
        Come to the party"<<endl;}
    return 0;
}
```

```
Switch (Expression) {
case 1:
    {
    action 1;
    }
default
    {
    action 4;
    }
}
int age; cout << "Tell me your age"; cin >> age;
Switch (age)
{
case 18:
    cout << "You are 18" << endl;
    break;
case 22:
    cout << "You are 22" << endl;
    break;
case 2:
    cout << "You are 2" << endl
default:
    cout << "No Special Cases" << endl;
    break;
}
cout << "Done with Switch Case";
return 0;
}
```

Loops in C++ (Block of Statements / Do work Repeatedly)

Lec#5

three types of loops in C++

1. For loop
2. while loop
3. Do while loop

DATE 19/12/21

* **Syntax For Loop :-**

```
For(initialization; condition; updation)
    {
        Loop body (C++code)
    }
```

Ex:

=> `For (int i=0; i<40; i++)`

`i<=40`

```
    {
        Cout<<i<<endl;
    }
```

Example of infinite For Loop

```
For (int i=1; 34<=40; i++)
{   cout<<i<<endl;
}
```

* **Syntax For while loop**

```
While(condition);
{
    C++Statements;
}
```

Ex: Printing 1 to 40 using while loop

```
while(i<=40){   int i=1;
    Cout<<i<<endl;
    i++;
```

Example of infinite while loop:-

```
while(true){
    Cout<<i<<endl;
    i++;
}
```

DATE 2021 23/12

* C++ Pointers and Arrays

Pointer arithmetic Formula:-

$address_{new} = address_{current} + i * Size of(datatype)$

$(p+i)$ → $32 + 1 * 4 = 36$

```
int* P= marks;
 Cout<<" The Value of *P is "<<*P<<endl;
 Cout<<" The value of *(P+1) is"<<*(p+1)<<endl;
 Cout<<" The Value of *(p+2) is "<<*(p+2)<<endl;
 Cout<<" The Value of *(p+3) is "<<*(p+3)<<endl;
```

Lec#8

* Structure in c++:-

Structure is a user-defined datatype that is available in c++. Structures are used to Combine different types of data-types, just an array used to Combine Same type of data types.

Practical

```
Struct employee
{
 int eID;
 char Favchar;
 Float Salary;
};
 int main() {
 struct employee harry;
 harry.eId=1;
 harry.Favchar='C';
 harry.Salary=120000000;
 Cout<<"The Value is"
       <<harry.eID
       <<endl;
 Cout<<"The Value is"
       <<harry.Favchar
       <<endl;
 Cout<<"The Value is"
       <<harry.Salary<<endl;
 return 0;
 }
```

SALMAN PAPER PRODUCT

Function Prototype

The Function prototype is the template of the Function which tells the detail of the Function e.g (name parameters) to the Compiles

DATE 28/12/21

Ex:-

```
int Sum (int a, int b);
```

```
=> int Sum (int a, int b); // Acceptable
=> int Sum (int a, b); // Not Acceptable
=> int Sum (int, int); // Acceptable
```

=> Formal Parameters: → Imaginary just

The Variables which are declared in the Function are Called Formal parameter.
(the Variables "a" and "b" are Formal parameters)

=> Actual Parameters: → Real

The Values which are passed to the Function are called actual parameter. (i.e the Variables "num1" and "num2" are the actual parameters)

```
Void g();
    Cout << "\n Hello Good morning";
```

Void as return type means that this Function will not return anything, and this Function has no parameters.

=> ... variables without using the keyword "struct" and the name of the struct is

Practical:

DATE 2 0 1

```
typedef struct employee
{ int eID;
  char favchar;
  float salary;
}ep;
int main() {
ep harry;
  struct employee Shubam;
  struct employee RohanDass;
  harry.eID=1;
  harry. favchar= 'C';
  harry. Salary= 120000000;
Cout<<" The Value is"<<harry.eID<<endl;
Cout<<" The Value is"<<harry.favchar<<endl;
Cout<<" The Value is"<<harry.Salary<<endl;
  return 0;
  }
```

* Unions in C++:-

Unions are similar to structures but they provide better memory management then structure. Unions use shared memory so only 1 variable can be used at a time.

Practical:

```
union money
{
  int rice;
  char coal;
  float Pounds;
};
int main(){
  union money mn1;
  mn1.rice=34;
  return 0;
}
```

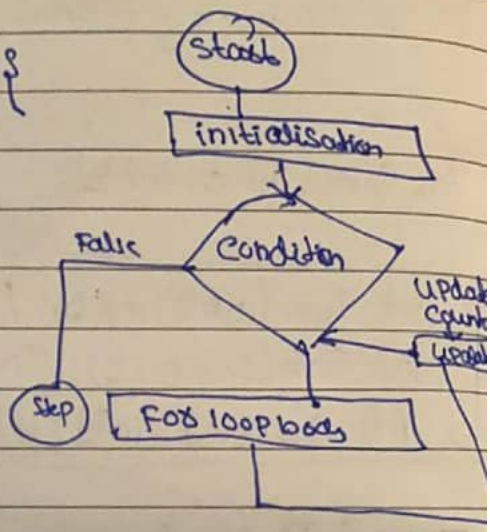
- we can use 1 variable at a time otherwise the compiler will give us garbage value
- The compiler chooses the datatype has maximum memory allocation

"More about Loops and others"

DATE 2 0 1

For Loop

```
For(initialisation; condition; update) {
        //body
}
```

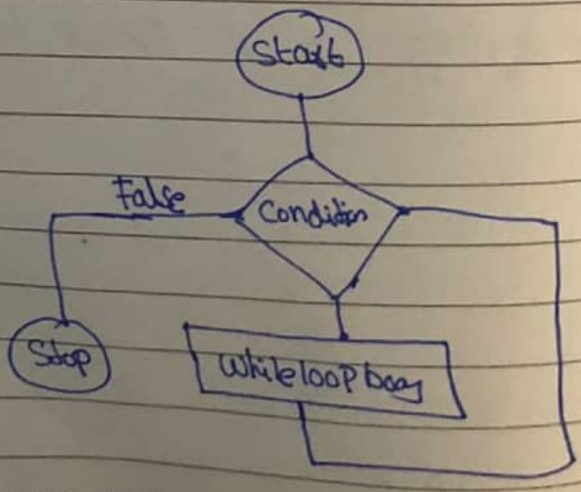


Ex:

```
#include <iostream>
using namespace std;
int main() {
  int n;
  cin>>n;
  int sum=0;
  For (int count=1; count<=10; count++) {
        sum= sum+count;
  }
  cout<<sum<<endl;
    return 0;
}
```

While Loop

```
While (condition is true) {
        // body
}
```



Break and Continue Statement

To Control the Flow of loops on meeting Some Specified Condition

DATE 2 0 1

Ex:-

```
#include <iostream>
using namespace std;
int main()
{
  int Pocketmoney = 3000;
  for(int date=1; date<=30; date++)
    {
      if(date%2==0){
          Continue;
      }
    if (Pocketmoney==0){
          break;
    }
  Cout<<"Go out today!" <<endl;
  Pocketmony= Pockey money-300;
    }
  return 0;
```

- **Continue:-**
  Skip to the next iteration of the loop
- **Break:-**
  Terminate the loop

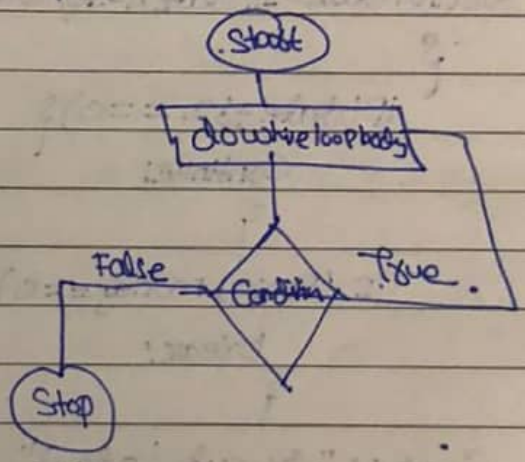
Ex:

```
#include<iostream>
using namespace std;
int main() {
    int n;
    cin>>n;
while(n>0) {
    cout<<n<<endl;
    cin>>n;
    }
return 0;
}
```

DATE 2 0 1

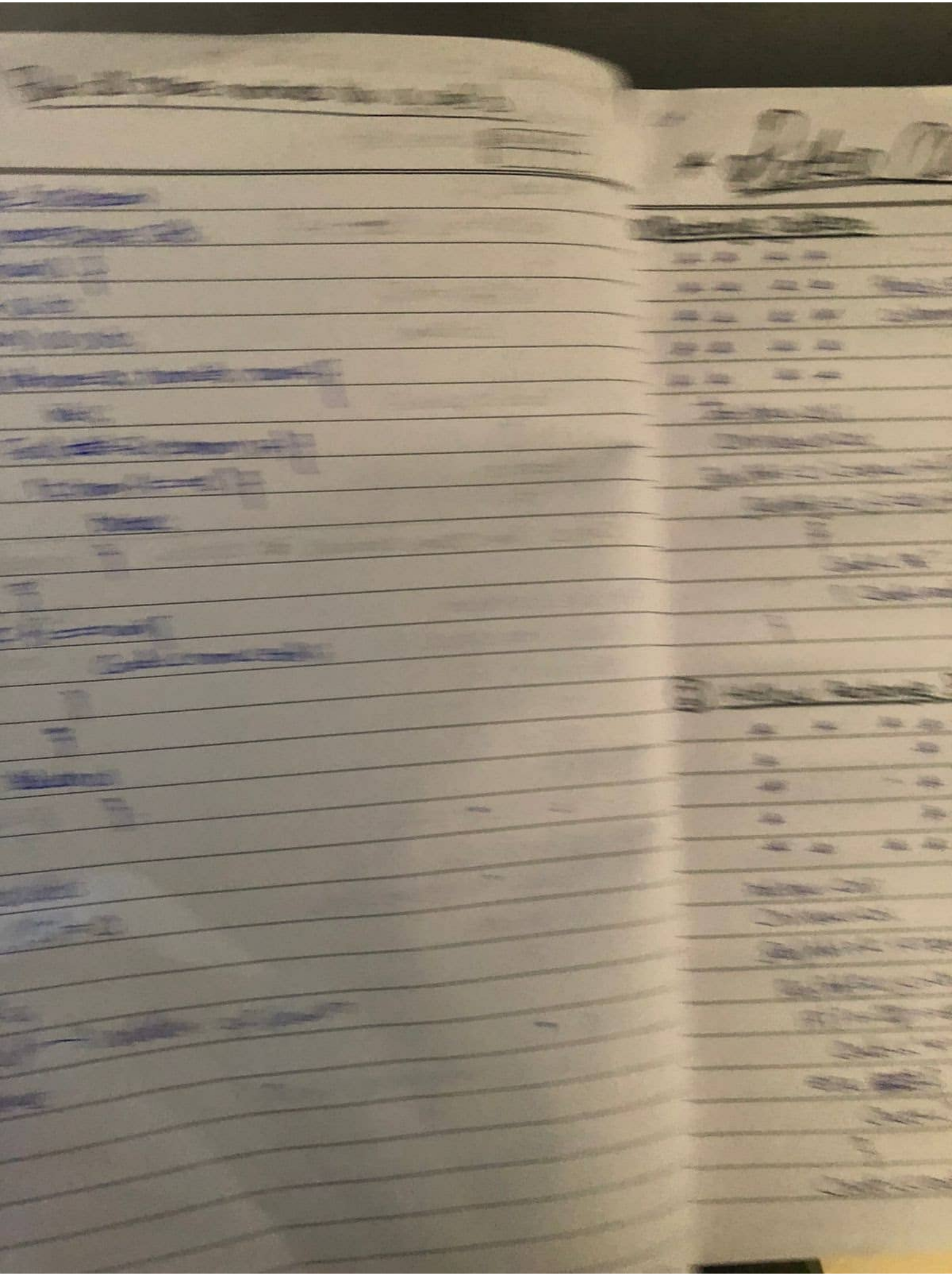
Do Loop

```
do {
    //body
} while(Condition);
```



Ex:

```
#include<iostream>
using namespace std;
int main() {
    int n;
    cin>>n;
do {
    cout<<n<<endl;
    cin>>n; } while(n>0);
return 0;
}
```



Pattern Questions:

DATE 2 0 1

① Rectangle Pattern:

```
* * * *
* * * *
* * * *
* * * *
* * * *
```

Rows: 5
Columns: 4

```
Int row, col;
Cin>>row>>col;
For(int i=1; i<=row; i++){
    For(int j=1; j<=col; j++)
        {
        Cout<<"*";
        } Cout<<endl;
}
```

② Hollow Rectangle Pattern:

```
* * * *
*     *
*     *
*     *
* * * *
```

Rows: 5
Columns: 4

```
int row, col;
Cin>>row>>col;
for(int i=1; i<=row; i++){
  For(int j=1; j<=col; j++){
    if(i==1||i==row||j==1||j==col){
      Cout<<"*";}
    else {
      Cout<<" ";}
    }
  Cout<<endl;
}
```

③ Inverted Half Pyramid:

n = 5

```
5 * * * * *
4 * * * *
3 * * *
2 * *
1 *
```

n to 1 "*"

```
int n;
Cin>>n;
For(int i=n; i>=1; i--){
  For(int j=1; j<=i; j++){
    Cout<<"*";
    }
  Cout<<endl;
}
```

Print number 1 to 100 skip the 3//

```
# include <iostream>
  using namespace std;

  int main() {
     For(int i=0; i<100; i++)
     {
       if (i%3==0){
          Countinue;
       }
     Cout<<i<<endl
     }
   return 0
   }
```

XOR

Check the prime number if or not.

```
#include <iostream>
 using namespace std;
 int main() {
      int n;
     cin>>n;
   int i;
  For(i=2; i<n; i++){
     if(n%i=0) {
      Cout<<"non prime"<<endl;
        break;
       }
  }
 IF(i==n) {
     Cout<<"prime number"<<endl;
                 }
     return 0;
      }
```

n divisible to i
non-prime
$n \% i = 0$
nonprime

1,2,3,4. n-1
P

* Enums in C++:-

Enums are users-defined types which consist of named constants. (use to make the program more readable).

DATE 23/12

Practical:

```
int main() {
enum Meal{breakFast, lunch, dinner};
Meal m1 = lunch;
cout<<m1;
return 0;
}
```

* Functions in C++:-

Functions are the main part of top-down structured Programming. we break the code into small pieces and make Functions of that code.

Practical:-

```
int Sum(int a, int b); {
int c = a+b;
return c;
}
int main() {
int num1, num2;
Cout<<" Enter first number"<<endl;
Cin>>num1;
Cout<<" Enter Second number"<<endl;
Cin>>num2;
Cout<<" The Sum is" << Sum(num1, num2);
return 0;
}
```

* (Syntax do while loop)

do { c++ Statements;

} while(condition)

Ex: int i=1;

```
do {
    Cout<<i<<endl;
    i++;
} while(i<=40);
```

H.w: make the table of 6 using these 3 loops.

Video:-11 is useful

Break and Continue:-

Break:-

```
int main(){
  For(int i=0; i<40; i++)
   {
    Cout<<i<<endl;
   if(i==2){
      break;
    }
   Cout<<i<<endl;
```

Continue:-

```
For(int i=0; i<40; i++)
 if(i==2){
     continue;
   }
 Cout<<i<<endl;
    }
```

DATE 20/12

Practical:.

```
int main() {
    int masks [] = {23, 45, 56, 89}
    Cout << masks[0] << endl;
    Cout << masks[1] << endl;
    Cout << masks[2] << endl;
    Cout << masks[3] << endl;
    return 0;
    }
```

=> Example:.

```
int math masks [4];
 mathmasks [0] = 2278;
 math masks [1] = 738;
  math masks [2] = 378;
  math masks [3] = 578;
Cout << " These are math masks" << endl;
  Cout << math masks [0] << endl;
  Cout << math masks [1] << endl;
  Cout << math masks [2] << endl;
  Cout << math masks [3] << endl;
```

Using For Loop:.

```
For (int i=0; i<4; i++)
{ Cout << "The value of masks" << i << "is" << masks [i] << endl; }
```

Manipulators (operators → control data display)

=> endl
=> setw — (space leger)

```
#include<iostream>
#include<iomanip>
using namespace std;
int main() {
int a=3, b=78, c=1233;
cout<< "The value of a is without setw": "<<a<<endl;
cout<< "The value of b is without setw" <<b<<endl;
cout<< "The value of c is without setw"<<c<<endl;

cout<< "The value of a is"<<setw(4)<<a<<endl;
cout<< "The value of b is"<<setw(4)<<b<<endl;
cout<< "The value of c is"<<setw(4)<<c<<endl;
return 0;
}
```

Operator Precedence:

```
int a=3 , b=4;
int c=(a*5)+b;
cout<<c;
return 0;
```

→ ((((a×5)+b)-45)+87)

website:- cppreference.com
C++ operator precedence

Lect#3

=> Header Files:-

DATE 2 0 1

=> two types of header files:-

1. System header files: It comes with compiler
2. Users defined header files :- It is written by the programmer

#include<iostream> is System header file

#include "this.h" / This will produce an Error if this.h is no present in the current directory

(Reference website => cpp reference for header files)
(Cppreference.com)

(word deprecated means (that file which is change nowdays but not include)
(Khatam hone wali)

=> Operators in C++:-

Following are the types of operators in c++

1. Arithmetic operators:-

i.e int a=47 , b=67

```
Cout<<" The value of a+b is"<<a+b;
Cout<<" The Value of a-b is"<<a-b;
Cout<<" The value of a.b is"<<a*b;
Cout<<" The value of a/b is"<<a/b; => (Float/Points mein aye)
Cout<<" The vale of a%b is"<<a%b; => % modulo
Cout<<" The value of a++ is"<<a++; (one increase)
Cout<<" The Value of a-- is"<<a--; (one decrease)
Cout<<" The Value of ++a is"<<++a;
Cout<<" The Value of --a is"<<--a;
```

→ (number then increment)
→ (number then decrement)
→ (increment then number)
→ (decrement then number)

SALMAN PAPER PRODUCT

Page No.

LIST OF RESERVED KEYWORDS IN C++:

- This is a list of keywords reserved in C++.
- Since they are used by the language itself,

DATE 2021 13/12

these keywords are not available for re-definition or overloading.

⇒ In short, you cannot create a variable with these names.

alignas (since C++11)	default(1)	register(2)
alignof (since C++11)	delete(1)	reinterpret_cast
and	do	requires (since C++20)
and_eq	double	return
asm	dynamic_cast	short
atomic_cancel (TM TS)	else	signed
atomic_commit (TM TS)	enum	sizeof(1)
atomic_noexcept (TM TS)	explicit	static
auto(1)	export(1)(3)	static_assert (since C++11)
bitand	extern(1)	static_cast
bitor	false	struct(1)
bool	float	switch
break	for	synchronized (TM TS)
case	friend	template
catch	goto	this
char	if	thread_local (since C++11)
char8_t (since C++20)	inline(1)	throw
char16_t (since C++11)	int	true
char32_t (since C++11)	long	try
class(1)	mutable(1)	typedef
compl	namespace	typeid
concept (since C++20)	new	typename
const	noexcept (since C++11)	union
consteval (since C++20)	not	unsigned
constexpr (since C++11)	not_eq	using(1)
constinit (since C++20)	nullptr (since C++11)	virtual
const_cast	operator	

Lec#2

⇒ Basic Input/output in C++:

DATE 13/12/2021

C++ Comes with libraries which help us in performing input/output. In C++ "Sequence of bytes" corresponding to input and output are commonly known as "streams".

- **Input stream:** Direction of flow of bytes takes place from input device (for ex. Keyboard) to the main memory.
- **output stream:** Direction of flow of bytes takes place from main memory to the output device (for example display).

⇒ Practical:

```
#include<iostream>
using namespace std;
int main()
{
int num1, num2;
Cout<<" The value of num1: \n" ; /* "<<" is called Insertion operation. */
Cin>>num1; /* ">>" is called Extraction operation */

Cout<<" Enter the Value of num2:\n")
Cin>>num2;

Cout<<" The Sum is" <<num1+num2);
return 0;
}
```

• DERIVED DATA TYPES IN C++:

- Array
- Function
- Pointer

Covered today

DATE 20/12/21

* PRACTICAL ON PC:

①
```
#include<iostream>
using namespace std;
int main(){
 int a=14;
 int b=15;
Cout<<" This is tutorial 4 - Here the value of
      a is "<<a<<". The value of b is "<<b;
 return 0;
}
```

we can also use this
int a=14, b=15;

②
```
#include<iostream>
using namespace std;
int main(){
 //int a=14;
 //int b=15;
int a=14, b=15;
Float pi=3.14;
Char c='U';
Cout<<" This is tutorial 4. In Here the value of a is "<<a<<
      "\n The value of b is "<<b;
Cout<<"\n The value of pi is: "<<pi;
Cout<<"\n The value of c is: "<<c;
 return 0;
}
```

Single character only

Variable Scope:-

* Scope of a variable is the region in Code where the existence of variable is valid.

DATE 12/12/21

* Based on Scope we have local and global variables in C++.

=> Local Variables:- Local variables are declared inside the braces of any function and can be accessed only from there

=> Global Variables:- Global variables are declared outside any Function and can be accessed from anywhere.

- Can global and local variable have same name in C++?
  Yes, we will see!

• C++ DATA TYPES:-

- Data types define the type of data a variable can hold. For example an integer variable can hold integer data, a character type variable can hold character data etc.
- Data types in C++ are categorised in three groups:

* Built-in
* User-defined
* Derived

=> **Built-in:-**

- int → 0, 1, 4, 7, 8, 9, 10
- Float → 1.22, 1.33. Decimal numbers of low Precision
- Char → 'a', 'b'
- Double → 3.7489..... Decimal numbers of high Precision
- Bool → true, False.

=> **USER DEFINED DATA TYPES In C++:-**

- Struct
- Union
- Enum

Discussed Further

* C++. Course *

Lec#1

DATE 12 12 2021

① What is programming & why C++?
⇒ C++ (It is near to hardware).

[C++ Program (.cpp)] → [Compiler] → (exe)
Compilation

First program

```
#include <iostream>

int main(){
    std::cout<<"Hello world";
    return 0;
}
```

* Installation of VS Code:

* installation of G++:

- C++ → 1979 → Bjarne Stroustrope → Extension of C
- C++ : Fast programs, more Control over System resources, memory management.

C++ → High Performance
[→ 2011 → C++11 , 2014 → C++14 , 2017 → C++17]

⇒ Basic Structure of a C++ program:-

```
# include <iostream>      → Header File (input output)
int main (){             → Skip Empty lines as your wish
std::cout<<"Hello world";   point to screen
return 0;                → main to zero
}
```

1 (entry) main Function Func()

(Low Level) → Hardware ke nazdeek

(High Level) → Logically Sound & Hardware se door.

• **Variables:-** Container to store your Data.

Datatypes
- int → (whole, integers) −1, 0, 1
- Float → (Decimal points) 0.11, −0.12
- char → ('c', 'b')

DATE 12/12/2021

```
int Sum=6;
cout<<"Helloworld"
<<Sum;
```

i.e
a=1 → (a+b) → 1 million+
b=2

• **Comments:-** // This is created for storing Areas.

// Forward slash

```
int a=3;
int b=4;
Float C=3.1;
```

```
/* . . . . . . . —
. . . Text . . . -
. . . —  -
. . . . . . */
```

multiline comments.

Ctrl + // all comments in VS Code.

⇒ **VARIABLES IN C++:-**

* A Variable is a Container to hold data.
* Primary variable types in C++,
  1. int ⇒ −1, 1, 0, 7, 8, 1, 1, 2
  2. Float ⇒ 1.21, 3.77,
  3. char ⇒ 'a', 'D', 'e'
  4. Double ⇒ 1.218...] (Long Decimal number)
  5. Boolean ⇒ True/False

* int Sum=34 means Sum is an integer variable which holds value 34 in memory.

⇒ **SYNTAX FOR DECLARING VARIABLES IN C++:-**

- Data type variable-name=value;

Ex int a=4, b=6

int a=4
char letter='0'

- Based on Scope, variable can be classified into two types.
  1. Local Variables
  2. Global Variables

Scanned with CamScanner