Fundamentals of C++ Programming

Date.....

Coolboymannu YT

Index

S.no	Topics	
1.	Introduction of C++	1-3
2.	Basic structure of C++	3
3.	Data type, variable, Keywords, expression, Operators, associativity	4-9
4.	Control structures	9-11
5.	function and Recursive function	12-13
6.	Array	14-15
7.	Data types, Pointers, storage class	16-17
8.	Structures, Union, enumerated	18-20
9.	Object Oriented Programming	21-28
	• Class	21
	• Constructor and Destructor	22-24
	• Encapsulation, Data Abstraction, Inheritance, Polymorphism	25-28
10	Exception handling	29
11.	Templates	30
12.	STL - standard Template library	31

YT - YouTube

Spiral

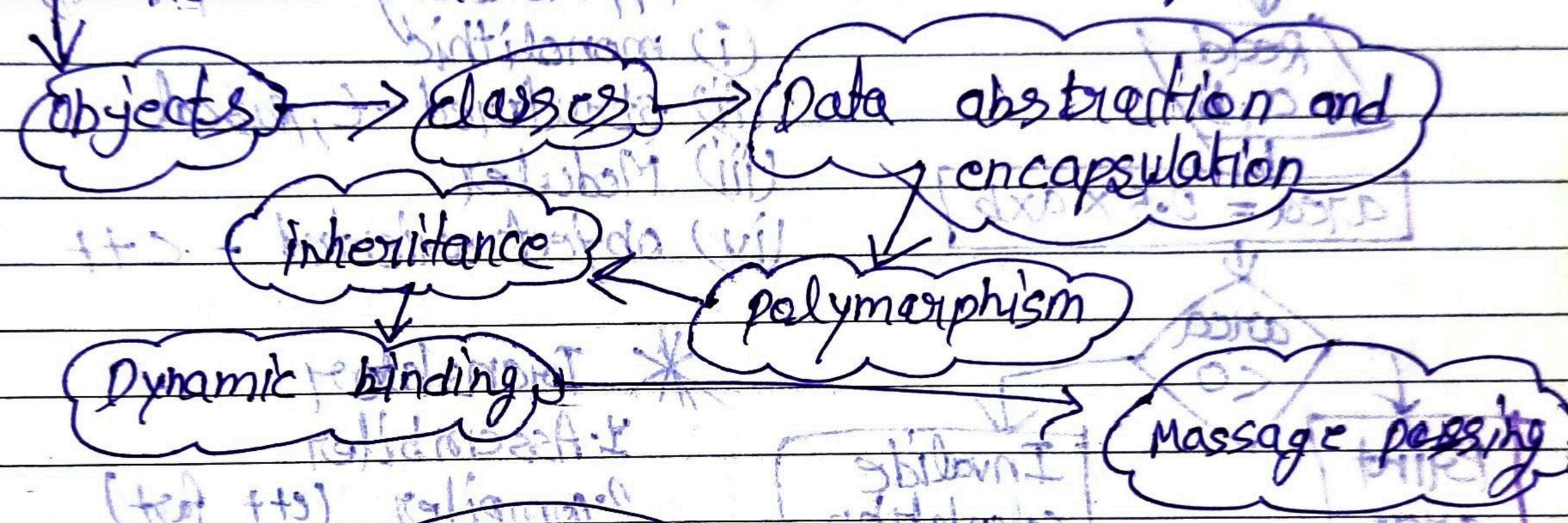
Date 1

C++

Introduction

- C++ is a comp lang that supports object-oriented programming

★ Object Oriented Programing
- Data is hidden and can't be accessed by external function.
- Obj may communicate with each other through functions.
- New data and functions can be easily added.



⇒ Algorithm → flow chart → Programing language

Algorithm ⇒ is a precise of sequence of instructions in order to be created out to rules a problem using any prog lang.

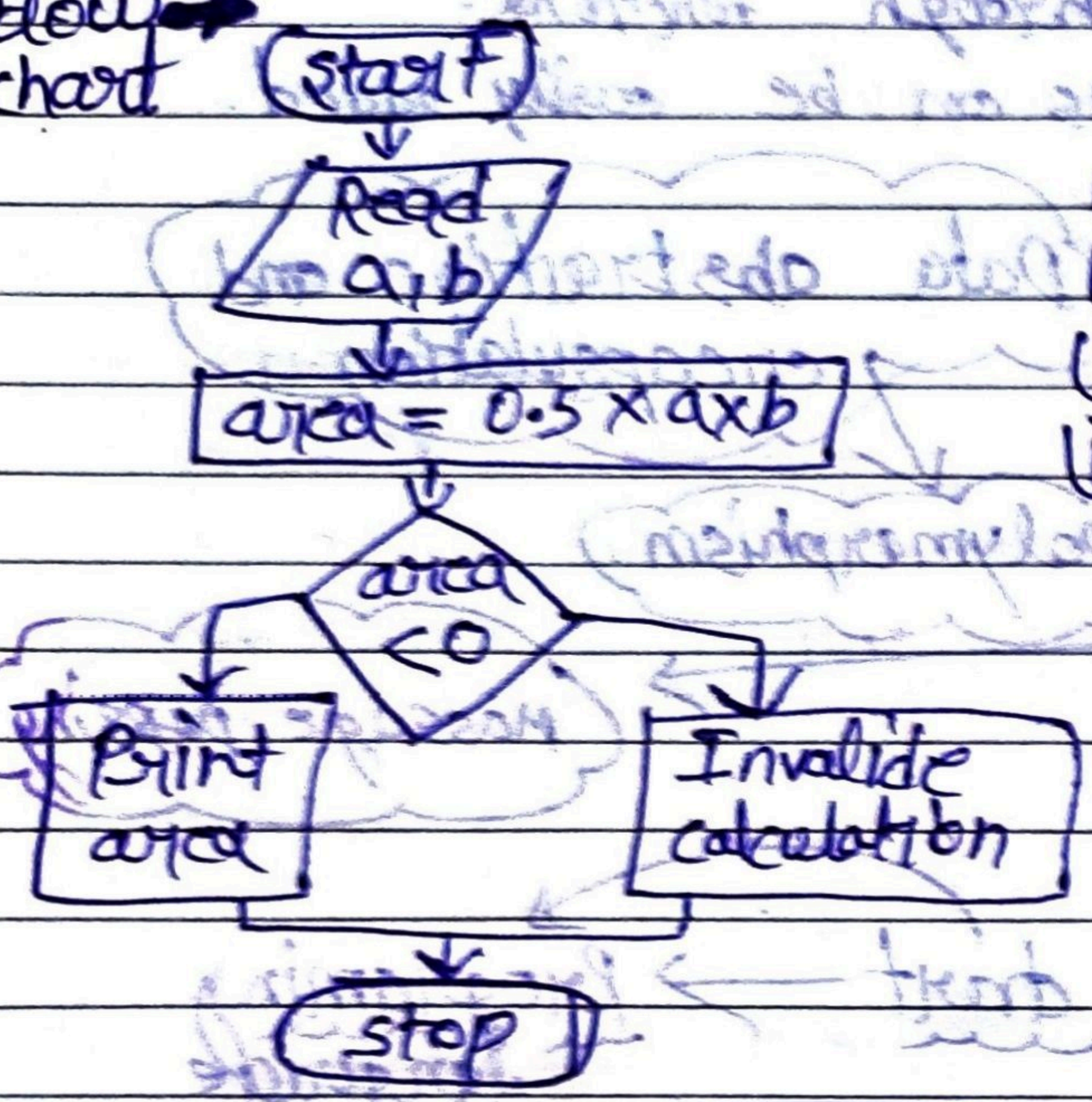
flow chart ⇒ is a pictorial representation of an algorithm use of flow of extution

Date.....

- → start / stop
- → input / output
- → conditions
- → processing
- → connector
- → documentation

→ link it to connect all

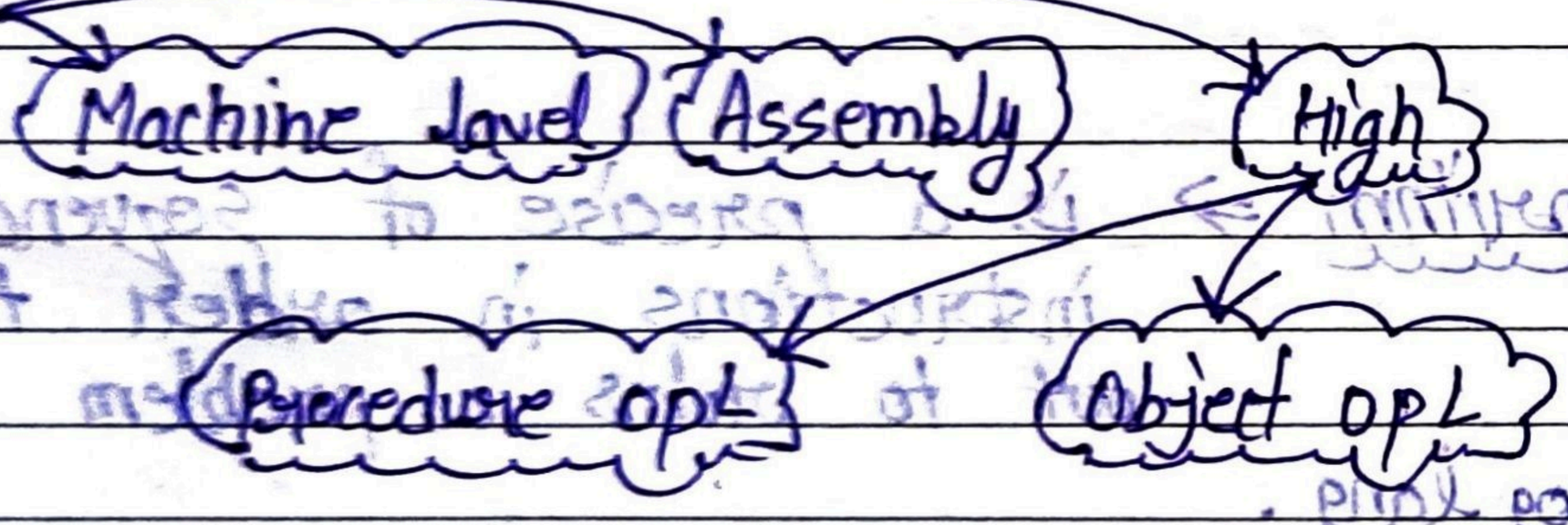
Flow chart →



Approaching of programs
(i) monolithic
(ii) structured - C, Pascal
(iii) Modular
(iv) object oriented - C++

* Translator
1. Assembler
2. compiler (C++ fast)
3. Interpreter (Python slow)

Language



- Global data access
- No data hiding
- It does not model real word problem very well

⇒ C, Pascal, fortran

- Data hiding
- Divided in to objects
- New data and function easily adding
- objects communicate each oth or easily

⇒ C++ Spiral

⇒ C++ ⇒ (i) Bizzarne stoustrup 1979 at Bill labs.

(ii) C++ is the superset of C [Karnighan Ritchie]
(iii) C++ have a grammar and own syntax rules. (English)
(iv) C++ is compiled lang
(v) statically typed lang = C++
Dynamically typed lang = Python

Ex ⇒
```
#including <iostream>
using namespace std;
int main()
{ int a,b,c;
int c = a+b ;
cout << c;
return 0;
}
```

• Basic structure of C++

⇒ Preprocessor Directive
<iostream> ⇒ Header file
main() ⇒ main function
; ⇒ end of a statement
return 0 ⇒ Exit / end of function

Imp. C++ is block of codes (groups of statments)
case sensitive lang
variable ⇒ to store the data.

Bug ⇒ Programming errors are know as bugs.

Re Bugging ⇒ the process of removing the errors is know as debugging.

(i) Syntactic errors — Grammer mistakes
(ii) Semantic " [logical errors] logic wrong giving / result
(iii) Run time " exception (handelling, removing process)

Date.....

(i) Data type, Veriable, Key words, expression, Operators, associativity, Precedence of operators.

(ii) control statments
(iii) Arrays
(iv) Fill system
(v) output and input statments

Cin → input
Cout → output
≫, ≪ → Extraction, insertion operator

⇒ Data type

		Typical size	
Integer	int	4 bytes	32bits (x8)
	short	2	16
	long	4	32
Floating point	float	4	32
	double	8	64
	long double	10	80
character	char	1	8
Bool	bool	1	8

int short int = 2 bytes
long int = 4
float float = 4
double = 8

[string = sequence of characters]

(True/false) [0 or 1]
(Bullyn)

[word length ⇒ number of instruction executed per second]

Date.....

ASCII ⇒ American standard code for information Interchange :
Table

7 bit ASCII ⇒ 2^7 ⇒ 128 ⇒ 0-127
8 bit ASCII ⇒ 2^8 ⇒ 256 ⇒ 0-256

0-256 we can use 256 character in 8 bits ASCII table.

⇒ Variable
Named units of storage through which we store the value into memory.

Naming conventions (identifier) ⇒ Giving name of variable

(i) the first character should be alphabet not the digit.
⇒ letter (letter / digit)

(ii) variable name could start with underscore also (_)
(iii) variable name could not be in the name of Keywords

⇒ Keywords
Reserved words defined in to the compiler.
like :- int , float , bool , char etc .

⇒ Expression
is a combination of operends of operators.

like :- a + b operator
two operends

Date.....

Convertion

⇒ Number → binary

2	42	0
2	21	1
2	10	0
2	5	1
2	2	0
2	1	1
	0	

Direction of writing remainder

binary → Number

$(0101010)_2$ ← flow of calculation

6 5 4 3 2 1 0

$+ \left[\begin{array}{l} 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 \\ 0\times2^5 + 0\times2^6 \end{array} \right.$

42 in binary $0 + 32 + 0 + 8 + 0 + 2 + 0$

0101010 $= 42$

⇒ Operators

- Scope Resolution operator [ ], ( ), { }
- Unary operator (++, --) ← increment
- Arithmatic (+, -, /, *, %) ← remainder
- Relational (<, >, <=, >=)
- Bitwise (&, |, <<, >>, ~, ^)
- Logical (&&, ||, !, ^)
- Comman (,)
- Assignment (=, +=, -=, *=, /=, %=)

- Arithmatic / Divide

$!(a==b) \doteq a!=b$
$!(a==b || a==c) = a!=b \&\& a!=c$
$!(a==b \&\& c>d) = a!=b || c<=d$ Date.....

- Logical operator ⇒ && (AND) || (OR) ! (NOT) ^ (XOR) (जोर)
- Logical ope are use to simplify the expration.

What is use Making logic circuits.

(2)

P	Q	P and Q	P or Q	P xor Q	Not P	Not Q
0	0	0	0	0		
0	1	0	1	1		
1	0	0	1	1		
1	1	1	1	0		

- Bitwise operator ⇒ (i) those are applicable only to integer.
  (ii) if we use bitwise ope in expression the It works on individual bits of variable.
  (iii) makes computeration of expression faster.

& → and
| → or
^ → XOR
! → NOT

>> << → Multiply by 2
Divide by 2

>> Right shift move the value of Binary) and Add zero
<< left shift " " " " " " "
→ like 6 in binary 110, then shifting and adding 0, 0 Now the meaning of 11 in binary is 4 (output)
$11 \Rightarrow (2\times1^1)+(2\times1^0) = 4$

Spiral

Date.....

⇒ Increment and decrement operator :→ unary operator
→ one variable

++

$x = 1$

Prefix	Postfix	$y = x++$
Preincrement	first give	Now y is 2 But
Add 1 ()	value then	x is 3 Because
simple add 1	add 1 in	value is 2 then Add
then show value	value the use	1, then result is 3

understand in Maths

$++x = [x+1]$

$x++ = y$

$\boxed{\begin{array}{l} y = x+1 \\ x = y+1 \end{array}}$

$y = x++$

+ +

⇒ ternary operator

formula → (condition ? expression 1 : expression 2)
true false

2 (cond ?((cond) ? exp1 : exp2) : (exp2);

⇒ comma operator [low precedence] low effect

int b = 4 ← final answer (output)

int a = (b, b+3)

output ⇒ 7

Date.....

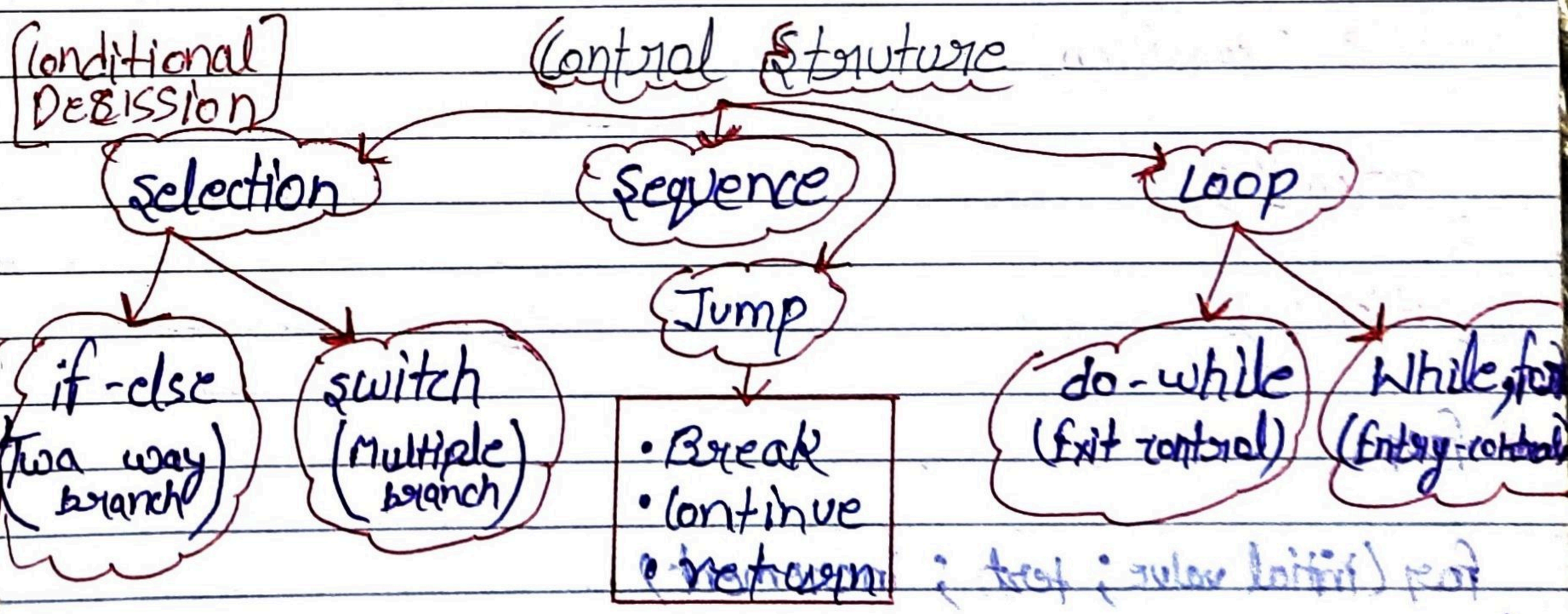
→ Assignment operator

1. = = ⇒ check value of 2 variable are equal
2. ! = ⇒ " " " " " " or not
3. > ⇒ " " is greater then ~~or equal to~~
4. > = ⇒ " " " " or equal to
5. < ⇒ " " " less then
6. < = ⇒ " " " " or equal to

~~Contn~~ Control Structures

operators for condition

< less than
> greater than
! = not equal to
= = equal to
< = less than or equal to
> = greater than or equal to



• expression like a>6, b=20 so on

Date.....

1) * if (expression)
{
action 1;
}
action 2;
action 3;

* if (expression)
{
action 1;
}
else
{ action 2;
}
action 3;

2) * do-while

do {
action 1;
}
while (condition);
action 2;

switch ( ){
case (): ...
case ( ) ...
default ...
}

* while

while (condition)
{
action 1;
}
action 2;

for (initial value; test; increment)
{
action 1;
}
action 2;

Date.....

Jumping statements :- if the flow of equation of statements are changed unconditionaly.

- Continue
- break
- return
- goto

1. continue

if (condition)
{continue;}
In loop
When condition will wright the whole program will be Re starting loop

2. Break

if (condition)
{break;}
When condition will be Right the whole program whale stop of exution

3. go to

go to label1;
⋮
label1 : [statement
cout << .....]
Re return 0;

Date.....

functions

⇒ A f^n is a coherent block of statements to perform a perticular given task.

caling function $\xrightarrow{\text{call to}}$ called function

[Return]

syntex

```
return type  functionname(arguments, arg)
{ statements;
  return (arguments);
}
```

Ex

```
int name(int, int)
{ cout << "Manish";
  return (Manish) }
int main( ){
      :
      :
int name(int, int);
return 0;
}
{ int name(int a, int d)
  return (a+b) }
```

→ Proto type fun

→ calling / Actual function/arguments

→ called / formal function

* Types of functions

1. call by value
2. call by reference (swaped)
3. call by pointer

Date.....

Recursive function

Recursive functions are the functions which calls it self again & again. This process is known as recursive functions.

- In recursion function we use selictions (if else)
- In Iteration functions we are repeation (for loop, while loop)

Factorial function

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

int factorial (int);  -> [Prototype]

int main(){
    int n, result;
    cout <<" enter a non negative number";
    cin >> n;
    result = factorial (n);  -> [calling fun]
    cout << result;
    return 0;
}
int factorial ( int n){  -> [called fun]
    if (n >1){
     return ( n* factorial (n-1);
     else {
     return 1}
    }
```

Date.....

Array

⇒ Data structure a way of representation of data.

- collection of similar types of data
- elements of an array could be accesed through its indexs.
- Relation beetwen array and pointer
  a Pointer Point the location of array's first element.

```
dataType arrayname[int Exp];
```

int const index = 10 One dimensional array.
* sales[index]

1. Initializing an array

```
for (index =0; index <10; index ++)
  { sales[index] = 0.0; }
```

2. Reading data into an array

```
for (index = 0; index <10; index ++)
  { cin >> sales[index];
```

3. Printing an array

```
for (index =0; index <10; index ++)
  { cout << sales[index] << " ";
```

Date.....

4. finding the sum and average of an array

```
int sum = 0;
  for (index = 0 ; index <10 ; index ++)
      { sum = sum + sales [index]
        average = sum /10 ; }
```

5. Larger element in the array

```
int max index = 0;
  for (index = 1 ; index <10 ; index++)
      if (sales [maxIndex] < sales [index]) {
          maxIndex = index ; }
  int largestSale = sales [maxIndex]
```

C++ Data types

- User-defined
  - structure struct
  - union union
  - class class
  - enumeration enum
- Built in
  - void void
  - integral int char
  - Floating float, double
- Derived
  - array
  - function
  - pointer *
  - reference &

Pointer in C++

- it is a special type of variable which is used to store the address of another variable.
- if we add asterisk (*) symbol with any variable at the time of declaring variable then this variable is called pointer variable

char* A / int* int1 / float* flo

int a = 2 / &a → it is pointing [address] reference
a का address

int* b → pointer ऐसा variable जो Address store करता है

b = &a → a का address b मे डाला जाता है।

Date.....

Storage Class.

A storage class in C++ defines the scope, lifetime, default initial value and storage space of a variable.

Storage class
- Automatic
  - auto — Automatic
  - register
- static
  - static
  - extern — external

	auto	register	static	extern
Storage	Memory	CPU reg.	Memory	Memory
Default value	Garbage	Garbage	0	0
Scope	Local	Local	Local	Global
Life	End the block →	End the block	Till end the program	End the block

Till end the program

Date.....

1 ⇒ structures - struct

```
struct struct_name {
    member_type member_name ;
    int A ;
    float B ;
    char D ; } ;
```

you can also define here structure name

```
struct {
    .....
    ..... } name ;
```

⇒ structure is a collection of simple variables
- The data items in a structure are called the member of the stru.
- Memory allocated when variable created.
- Total memory = all member's memory.

How to call a structure member?

~~first~~ • First make a variable
- A variable (•) structure variable name.

① struct man{
    int a; } ; →

② int main() {
    man manish ; → assigning (variable created)
    manish.a ; → calling.

Date.....

2 ⇒ Union union

⇒ It'share the memory in every members
- largest variable's memory it also of total union's memory

largest variable memory = union memory
↳ member

```
union man {
    int a
    char[10] ; } ;
```

typedef : you can make a own variable like

```
typedef man[10];
man Name;
```

↳ man is a user defined data type

typedef datatype variable-name

using ⇒ variable name new-variable

⇒ Union will show only last value correct because it can store only single value at a time.

Date.....

3 ⇒ enumerated (enum)

it can takes limited set of values. These values are referred by name (tag) The compiler assigns each tag an integer value internaly

```
enum    enum_name { tag-1, tage 2 ..... };
```

```
enum day{ Mon, Sun, Tue, Wed, Thr, Fri, Sat, };
  0     1    2    3    4    5    6
```

Date.....

Object Oriented programming

classes & objects

Three different pro. styles.
(i) structured
(ii) Modular
(iii) object oriented pro.

(i) structured - • very large & unsolvable.
• top down approach ↓

(ii) Modular - • a collection of similar type of function (python)

(iii) oops - • data security (main concern)

⇒ class

- user-defined data type
- It have functions & it's members
- By default access to members of a class is private
- three access specifiers - (private, protected, public)

```
class    Name {
          int a;
        public :
          int b;
        string x; };
```

Date.....

Properties of oops:
- Data encapsulation or information hiding

Constructor - is in when we create object of a class. Through constructor we initialize the data member of a class constructors may be defined within private and public section both but mostly we define construction in public section of a class.
- constructor do not return any value of constructor do not have any return statement.

The operator '::' is called the scope resolution operator
- This means that the scope of the function is restricted to the class - class name.

```
class_name :: function_name( ){ };
```

- the function can be define out of the class with the scope resolution operator.

Date.....

Constructors & Destructors.

⇒ C++ provides two special member function called constructor and destructors for automatic initialization and deletion of object.

- Rules for writing constructors.
  - A constructor name should be same as the class name.
  - It is declared in the public section.
  - It can be defined (coded) inside or outside the class, as any other method.
  - It is declared with no return type (not even void)

Types of constructor

1. Default
2. Parameterized
3. Copy

1. Default Constructor - The constructor with no parameter.

2. Parameterized - The constructor with parameter.
   Both are same
3. Copy constructor - one object with parameter is copied into another object.

Date.....

Destructor

- It's name is same as class name.
- It has no return type and no parameter.
- It is used to destroy class level variable.
- ~~Tilde~~ Tilde symbol (~) is used to declare a destructor.

C++

Date.....

OPPs.

Encapsulation - the process of combining data member and member function into a single entity like class.

- It is used to prevent the direct accessibility of data member and functions and this is done by using access specifier. public, private and protected.

Public - • Public element of a class can be accessed anywhere in program.

Private - • It is used to hide data member and member function from the other classes.

- Private element of a class can be accessed only inside in its own class.

Protected - It is approximately same as private but it allows the accessibility of data member

- and member function in the child class.

Date.....

Data Abstraction - • Abstraction means Data hiding.
• Members defined with a private access specifier which allow to Data Abstraction.

Inheritance - • The process of getting property of one class into another class.
• When a class inherits the property of a class it means it can access all the data member and member function of that class except private element.
• In this type of programming mainly two types of classes are used.
- Parent / Super / Base class.
- Child / Sub / Derived class.

Derived class : access specifier base class { }

Date.....

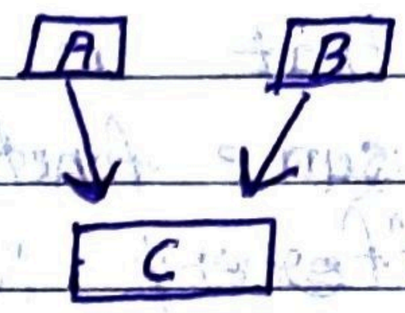
There are five types of inheritance.

Single

Base class
child class

- Two class are used

Multiple



Base class
child class

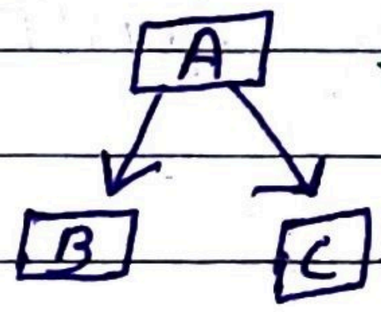
- More then 2 class is inherited by single class.

Multilevel

Base class
child / Base class
child class

- When first class is inherited by second class, second class is inherited by third class and so on....

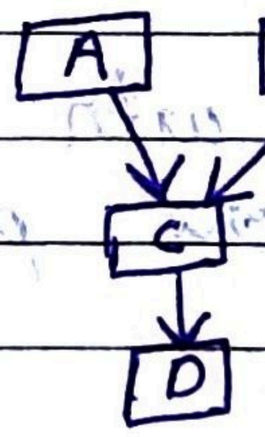
Hierarchical



Base class
derived class

- When a single class is inherited by two or more the two classes.

Hybrid



Base class
Base class of D, Derived of class A & B.

- The combination of two or more then two inheritance is called Hybrid....

Date.....

Polymorphism

Polymorphism - means 'one name, multiple forms'

It means one name many form so we can say that in this type of programming same function is used to perform different kind of operation.

Types of polymorphism

- compile time
  - function overloding
  - operator overloding
- Runtime
  - function overriding

Compile time

(i) function overloding - The function with same name and different parameter.

(ii) Operator overloding - It is a type of polymorphism in which an operator is overloded to give user defined meaning to it.

- By using an operator overloding we can change the meaning of operator.

Date.....

~~file~~ Exception handling

Exception → Known as runtime errors
Exception handling → Managing the error knowness

try block, throw, catch block

```
try{
    //code
    throw [anything / error]
}
catch {
    Error handling }
```

- if try block will raise any error then error or something else throw with 'throw' key word then 'catch' catch the throw argument finally catch block will execute.

Date.....

Templates

- A templates can be use to create a family of class and function
- It provides support for generic programming.
- A templates is defined with a parameter that would be replaced by a specified data type at the time of actual use of the class or function.

```
template <class mydatatype>
class classname {
    mydata-type name ; }

int main(){
    classname<int> a1;
return 0; }
```

→ mydatatype is a temporary data type when you will define the 'int' data type then int will replace 'mydatatype'.

```
template < class A>
A sum(A a , A b){
    return (a+b) }

int main(){
    int a=10 , b=5;
  c=sum <int>(a, b) ;
  cout<< c;
  return 0; }
```

Date.....

STL - Standard Template Library

Library -> Generic classes and functions.
- Reuse well tested components and classes.

Components of STL

Containers
- stores data
- use template classes

Algorithm
- sorting
- searching
- use template function

Iterator
- object points to an element in a container
- Handed just like pointers
- connects Algo. with containers