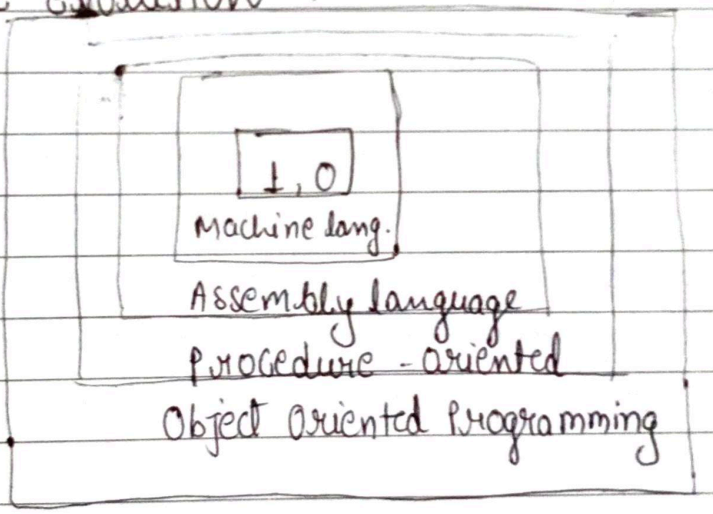
Date 12 / 2 / 22

Saarthi

## Main Quality Issues -:

- Correctness -: Software behaves exactly as intended.
- Maintainability -: easy to repair improve the software.
- Reusability -: reuse of existing programs or software components.
- Interoperability -: exchange of data with other software.
- Portability
- Security
- Openness.
- User friendliness -: Easy to use.

## Software Evolution -:




## Procedure - Oriented Programming (POP)

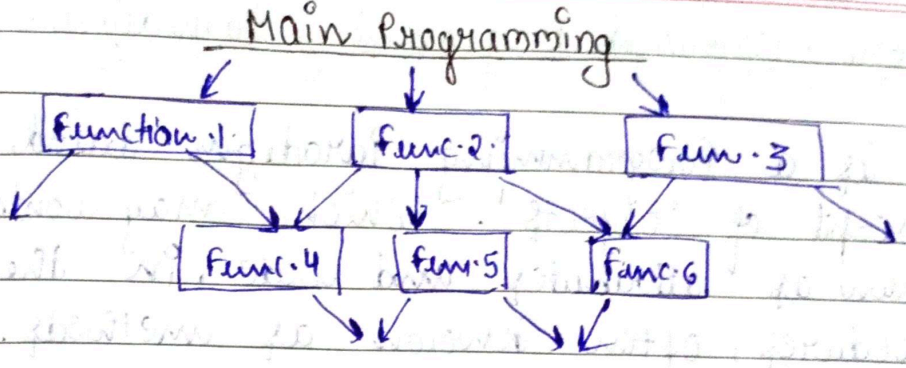
→ In POP the Problem is viewed as the sequence of things to be done such as reading, calculator and printing.

→ Primary focus on functions.

→ ex -: C, Paiscal, FORTAN

Date ___/___/_____

Saathi




## Characteristics of POP

→ Emphasis is on doing things (algorithms).
→ Large programs are divided into smaller programming knows as functions.
→ Most of the functions share global data.
→ Data move openly around the system from function to function.
→ Functions transform data from one form to another.
→ Employs top-down approach in program design.

## Problems in POP

→ POP is not suitable for large complex software.
→ There are two main problems -:
(1) First, Functions have unrestricted access to global data.
(2) Not suitable for real-world Modeling -:

Attributes (Data) — Behaviour (Functions)

Date ___/___/___

Saathi

# Object - Oriented Programming Paradigm -:

- OOP is a Programming Paradigm based on the concept of 'objects'. which may contain data often known as attributes and code, in the form of procedures, often known as methods.

## Characteristics of OOP -:

1. Class
2. Object
3. Encapsulation
4. Data Hiding
5. Data Abstraction
6. Inheritance
7. Polymorphism

(1.) Class -: It is a user-defined data-type that defines the abstract characteristics of an object, that includes ~~all~~ attributes (data) and methods or functions.

(2) Object -: → Object are the basic run-time entities in an object oriented system refers to a particular instance of class.

→ The object can be combination of data members and member functions.

| Class | Object |
|---|---|
| • fruit | apple |
| • vehicle | Car |

Page No.

Date ___/___/_____

Saathi

What kinds of things become objects in OOP?

Four categories may be -:

(1) Physical objects
(2) Computer-related objects
(3) Data storage
(4) Human entity

POP vs OOP example

```
int main()
{
int l, b, h;
Calc - Floor - Area(l,b);
Calc — 4wall- Area(l,b,h);
}
```

```
Class room
{
int l, b, h;
public:
Calc - Floor - Area();
Calc - 4wall - Area();
};
```

(3) Encapsulation ~~and Data Hiding~~
Encapsulation is a process of combining data members and functions in a single unit called class.

Due to encapsulation the data is not accessible to the outside world and only those functions which are wrapped in the class can access it.

(4) Data Hiding
Data Hiding is a process of hiding or preventing data from accidental alteration.

Page No.

Date ___/___/_____

Saathi

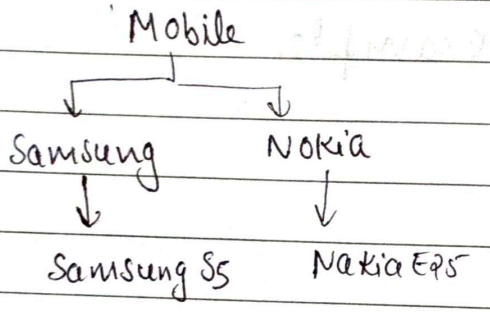
(5) Data Abstraction

Providing only essential information to the outside world and hiding their background details.

(6) Inheritance

Inheritance is the process by which objects of one class acquires the properties of objects of another class.

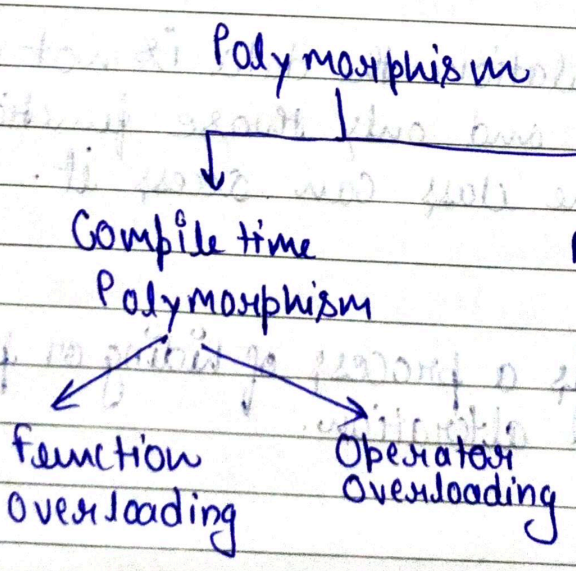
Ex:-




(7) Polymorphism

- Poly = many and Morph = form

So, Polymorphism is the ability where same entity behaves differently in different scenarios




Page No.

Saathi

Date __/__/____

Function Overloading -: When there are multiple functions with same name but different parameters.

Operator Overloading -: The existing operators work for user-defined types like objects & structures.

# INTRODUCTION TO C++

```
#include <iostream>
using namespace std;
int main()
{
cout << "Every age has a language of its own\n";
return();
}
```

- In Header file, contains only declaration of pre-defined functions.
- library file contains actual code.

WAP to read two numbers & display sum?

```
#include <iostream>
using namespace std;
int main()
{ int a, b;
cout << "Enter two numbers a & b";
cin >> a;
cin >> b;
cout << "sum is" << a+b;
return 0; }
```

Page No.

Saathi

Date ___/___/_____

```
# include <iostream>
using namespace std;
int main()
{
 Cout << "Hii \n";
 return 0;
}
// First, CPP C++ Program
```

→ Header file was only declaration of pre-defined function not actual code.
→ Actual code - library files.

Q→ WAP to read two no's from user display sum?

```
# include <iostream>
using name space std;
int main()
int a, b, sum;
Cout << "Enter no's";
 Cin >> a >> b;
  Cout << "sum" << a+b;
  return 0;
```

Q→ WAP to check whether no. is palindrome or not?

```
# include <iostream>
using name space std;
int main()
int n, num, digit, rev = 0;
Cout << "enter no:" >> Cin >> num;
 n = num;
 do
 {
```

Saathi

Date ___/___/___

```
digit = num %. 10;
rev = (rev *10) + digit;
num = num/10;
}
while (num!=0);
Cout << "The reverse of no"
if (n == rev)
cout << "It is palindrome."
else
Cout << "It is not"
```

Structure (struct)

It is user defined data type it allows us to combine different data items within a single type

```
struct student
{ [ char name];
  int age;
  float CGPA;
  };
```

members of structure

| | 50B | 9B | 4B |
|---|---|---|---|
| dev 2000 | Rohit\0 | 20 | 9.5 |
| | name | age | CGPA |

```
int a ; int *p = &a
struct student s1; [*p = q]
struct student s2 = { "Rohit", 20, 9.5" }
```

while declaration

- Accessing structure members using . (dot operator)

```
Cout << s2.name;
Cout << s2.age;
Cout << s2.CGPA;
// Cin >> s2.name;
struct student *ptr;
ptr = &s2;
*ptr [*ptr ≈ s2]
```

```
scanf("%s", &
scanf("%s", s1.name);
scanf("%d", &s1.age);
scanf("%d", &s1.CGPA);
```

```
Cout << (*ptr).name ; // ptr → name
                      ptr → age
                      ptr → CGPA
```

*6- Create an array of structure of student type of size 10.

```
struct student A[10];
for(i=0; i<10; i++)
{cin >>A[i].name;
 cin >> A[i].age;
 cin >> A[i].CGPA;
}
```

Imp. dot Operator → Variable se access krte hai member ko.

Arrow Operator → Jab structure ko se pointer banaya or pointer se access krte hai member ko.

* Q-: Create an array of structure array 10.

```
struct student A[10];
 ctr.A[0].name;
  A[0].age;
  A[0].CGPA;
  or
```

```
for (i=0 ; i<10 ; i++)
{   cin >> A[i].name;
    cin >> A[i].age;
    cin >> A[i].CGPA;
};
```

Union

Also user defined data type that allows us to combine diff. data items in a common memory.

```
union student
{
  char name[50];
  int age;
  float CGPA;
}
```

- In union memory unit be allocated only for largest data type.

Q→ WAP to calculate area of circle

myself:-

```
#include <iostream>
#define pi 3.14 //MACRO
using namespace std;
int main() int r; int r; float r;
{cin >> cout << Enter value of radius r;
 cin >> r;
 cout << pi*r*r; "Area of circle" << pi*r*r;
 return 0; }
```

Saathi

Date __/__/____

Sir's method

```
#include <iostream>
using namespace std;
int main()
{
  int r;
  float A;
Const float PI = 3.14F;
  A = PI * r * r;
Cout << "Area = " << A;
  return 0;
   }
cout << Enter radius;
  cin >> r;
```

F → float

```
# define fun(a,b) {a*b
```

2+3 * 1+2 = 7

```
cout << fun(2+3, 1+2)
```

Function :- Set of statements which can perform certain task is known as function.

return type | fun name | arguments

```
void swap(int x, int y); // declaration
int main()
{
  int a = 5, b = 10;
```

actual arguments

```
  swap(a,b) // func. call
  return 0;
}
```

Page No.

Saathi

→ formal arguments

```
void temp(int x, int y) // declaration
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}
```

fun body / definition

(1) take something & return something
(2) take something but return ~~something~~ nothing
(3) Take ~~something~~ nothing but " something
(4) Take nothing & return nothing.

Date 23/2/22

```
int n;
cin>>n;
```

→ static memory allocated with stack ~~memory~~ area of RAM.

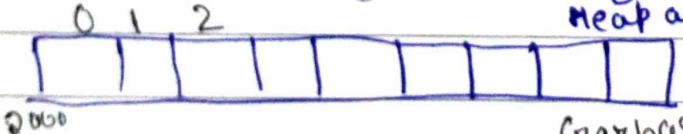
```
int age[n] : // variable length array (VLA)
```

Malloc
Calloc

Dynamic Memory Allocation in C

Malloc -:

```
Malloc ( n * size of (int) );
```




Dynamic memory allocated in the Heap area

- Return type of malloc / calloc / ralloc → : void*

* malloc memory is initialize with → garbage value.

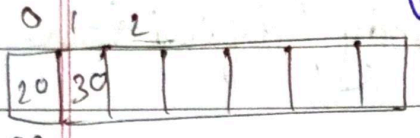
Saathi

Date ___/___/_____

```
int * = int* malloc ( n * scanf(int));

        for( i=0 ; i<n ; i++)

                cin >> p[i];
```



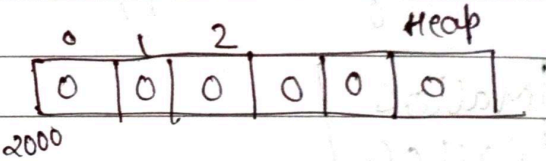

```
if ( p = = Null)
cout << " Memory can't be allocated";
```

## Calloc -:

* Calloc ~~allo~~ initialize memory with = 00.
  calloc में 2 argument होते हैं।

```
int*p = ( int p* ) Collac ( n , size of (int));
```

* Calloc takes 2 arguments & Malloc takes only one argument.




```
if (P = = Null)

        P =(int*)realloc (P, n*size of (int))
```

free (p);

Wild Pointer -: Initialized Pointers are known as wild Pointer

dangling Pointer -: earlier P is was pointing to valid location, but that memory is deleted, now P is still there but location is no more valid.

Page No.

Saathi

Date

```
int *ptr // wild pointer
    if (p == Null)
    ptr = p
     free(p);
     cout << p; //Dangling Pointer
```

## Reference Variables in C++ :-

- A reference variable is an alias (alternate name) for an existing variable.

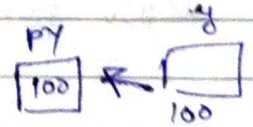
```
int x :     // x is variable
int & alias_x = x ; // alias_x is a reference
```

- To create reference variable we simply put '&' sign and equate it to an existing variable of same data type.

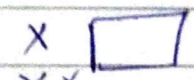
Differences between references and Pointers

(1) A reference may not occupy any space as it is an alternative name for a variable .. but pointer has their own memory locations.

```
int x :
int & xx = x : // a reference
```




```
int y :
int *py = &y // a pointer
```



Date ___/___/_____

Saathi

(2) A reference must be initialized at place of declaration. Its declaration cannot be deferred from its initialization. The initialization of a pointer cannot be deferred.

```
int x;
int &xx; // Error not Possible.
  xx = x;
```

(3) A reference once created and bound to a variable, cannot be reinitialized to another variable. A pointer can be reinitialize any time.

```
int x, y;
int &xy = x; // a reference
  xy = y; // This doesn't mean by xy has become
          a reference by y.
```

```
int x, y
 int *pxy = &x;
   pxy = &y;
```

(4) ~~There~~ There is no concept of NULL reference. There exist NULL pointer.

(5) A function returning a reference can appear on the left hand side of the assignment operator. No such concept with pointers.

Date ___/___/_____

Saathi

## Example of function returning reference -:

```
#include <iostream>
using namespace std;
static int a = 20;
int & setValues()
{
  return a; // return a reference to the variable a
}
int main()
{
  cout << "Value before change" << a << endl;
  setValues() = 500; // change the value of a
  cout << "Value after change" << a << endl;
  return 0;
}
```

## Pass-by-Reference

```
Void change (int&, int); // prototype
int main()
{
  int a = 10, b = 20;
  change (a, b); // function called
  cout << endl << "a =" << a << "b =" << b;
  return 0;
}
"function to change two parameters;"*/
```

Date ___ / ___ / _____

Saathi

Similarities b/w Reference & Pointers:

- Both can be used as formal parameters in a function.
- Both can be return from a function.
- A variable can have any no. of reference or pointers.

Advantages of Pass-by Reference

- It saves space, because no extra space is reserved for reference parameter.
- It saves time, because there is no data copy from calling function to called function.
- References are safer and easier to use.

Q→ Write three different C++ program to swap two variable

→ Call by value

→ Call by Pointer

→ Call by Reference