Object Oriented Programming Concepts

Introduction

Structure programming

→ Stgucture Programing is a logical programming that is consideg a precursog to object oriented programming.

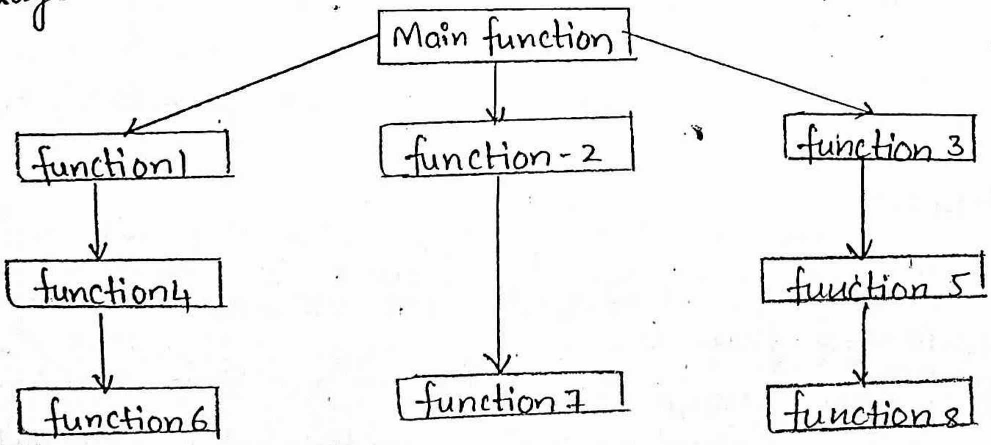
→ Structure programming specifies program to understanding and modification and as a top down design approach whege a system is divided into compositional subsystems.

→ Stgucture programing is a positional programming subset that produces the need fog goto statements.

→ The convential programming has the high level language such as cobal; fortran and c are known as procedure oriented language (POP) (or) structure oriented language

→ Procedure Oriented language has a sequential actions such as writting, that age used to complete the task reading and calculating.

→ The following figure shows the stguctuge of procedure language.

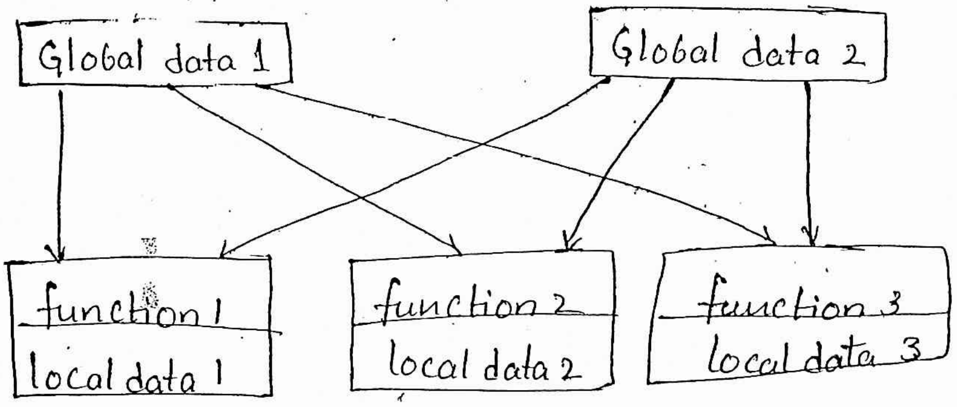


→In POP the programm will write the list of instquctions (oq) actions to the computeq. All these instquctions aqe grouped togetheq what is known as functions.

→In multifunction programming the data can be shared by all functions. So that we have to declored ouq data has global. A global data is moqe harm to the accidental changes by a function.

→The anotheq drawback of procedure oriented language is that it will not model for real world problems.

→The Organisation (or) relaxation of data and function as shown below.



→The data which is specified locally is called local data. Every functio has its own local data only the associated function will access the local data.

Characteristics of Structure Oriented Language:

1) It mainly gives the importance to the procedure but not to the data.

2) A large programm is divided into small programms which are called functions.

3) The data can be move openly around the system from one function to anotheq function.

4) functions can shaqe the data

5) functions transform the data from one form to anotheq

2

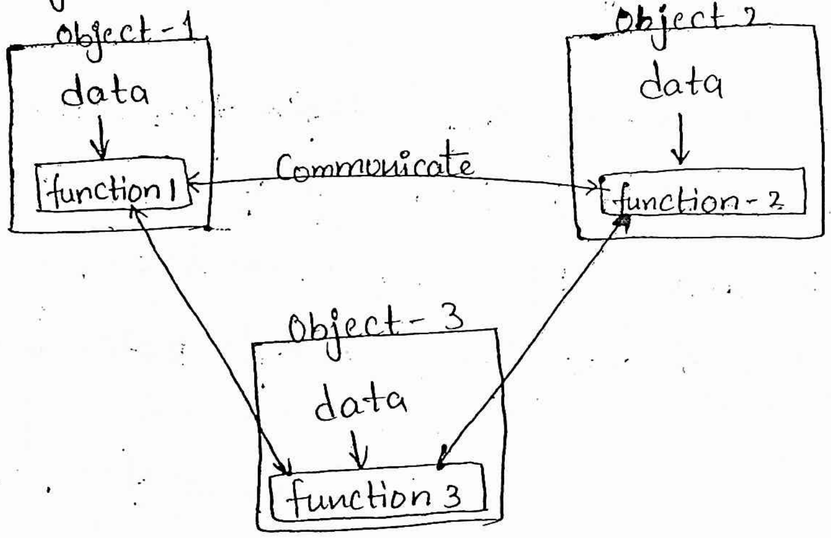
Object - Oriented Programming Paradigm (or) Pattern: (OOP)

→ An object oriented programming that provides a way of modularizing programming by creating Partician memory area for both data and function that are used as templates for creating copies of such module on demand.

→ The drawbacks in structure oriented language are overcome by OOP. Here entire programm is divided into smaller entites that are called objects.

→ The data of an object can be accessed by the function associated with that object.

→ The organisation of data and function in OOP as shown in fig



Characteristics of Object Oriented Programming:

1) It will give more importance to the data rather than procedures.

2) Large programs are divided into smaller programs those are called objects.

3) Bottom of approach in programm design

4) Programs organised around objects, grouped in classes.

5) focus on data with methods to operate upon objects data

6) Interaction b/w objects through functions.

Elements of Object Oriented programming

We have the following elements in object oriented programming

1) Classes
2) Objects
3) Data encapsulation
4) Data obstraction
5) Inheritance
6) Polymorphism
7) Dynamic binding
8) Message passing

Class:

A class is a user defined function data type. We can define the structure and behaviour of object. using classes. A class can be defined by using a "class" key word.

* Generally every class consists of both data and member function.

Syntax:

```
class  class-name
{
  - - - - - -
  - - - - -
};
```

Example:

```
class  Area
{
  - - - -
  - - - -
};
```

Objects:

→ An object is a run time entity of a class. using objects we can access the both data and member function of a class.

→ Generally we can create the object by using its class name.

Ex:

object : std
Data name doB marks
Functions total average display

Data Encapsulation:

The wrapping of both data and member function in a single unit (class) is called data encapsulation.

Using these feature we can hide the class data without accessing by other ~~data~~ classes.

Data Abstraction:

Data Abstraction is a process of hiding the certain details of objects and representing the only essential details of an object.
~~Using essential details of an object~~

Using data abstraction we can protect the data from external classes.

Inheritance:

→ Creating a new class by deriving properties from existing class is ~~cla~~ called Inheritance.

→ In this process a new class is extended the property of an existing class. The new class is also called child class (or) derived class

→ The Existing class also called as either parent class (or) base class.

Ex:
The Birds is a Base class. The derived classes are

binding

Polymorphism:

Polymorphism means ability to create more than one form. Using this polymorphism we can create the interfaces from one object to another object that means we have only one method using the same method we can do the multiple task.

Generally we have 2 types

1) Compile time polymorphism
2) run time polymorphism.

We can achieve the polymorphism by using function over-loading and operator overloading.

Eg:

① Operator over loading.

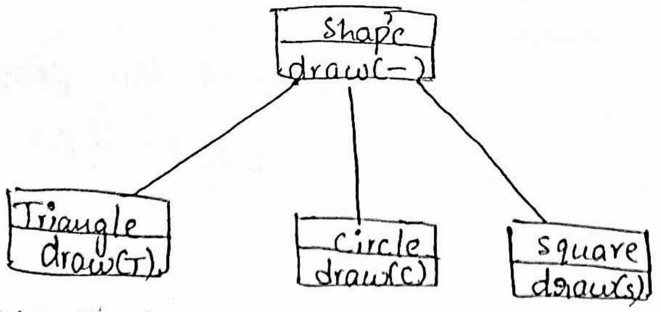
We have a + operator using the same operator. we are doing two different operations those are addition & concatenation.

10 + 20 = 30 (Addition)

ECE + ECE = ECEECE (Concatenation)

The operation can be decide based on the data what we supplied. We can do the addition using + operator whenever we can supply two numbers. We can do concatenation using '+' operator whenever we supply two strings.

② Function over loading:



Dynamic ~~Bridge~~ Binding:

Binding means linking a procedure call (or) function call. The Dynamic Binding means that can be show the code of [illegible] binding called late binding.

→In above fig we use 'draw' method but code of a draw method can be known only at run time. Because based on the code we are drawing 3 different shapes.

Message Passing:

→Using this element we can create the commonications b/w objects by passing the messages. It has following steps

① Create a class to define the behaviour of object.

② Create an object to access the members of a class.

③ Establish the commonication b/w the objects.

Eg:

$$\begin{tikzcd}
\text{Employee .} \arrow[d] & \text{Salary} \arrow[d] & \text{(name)} \arrow[d] \\
\text{object} & \text{messag} & \text{information.}
\end{tikzcd}$$

In above example we are getting information about an employee by sending a salary message.

Merits and demerits of OOP:

Merits:

1) Using in heritance we can reuse the code and eleminate the redendent code. and it will also extend the use of existing class.

2) We can build the programms based on working modules that are communicate from one to another.

3) The principle of data hiding helps the programmer to build secure programms.

4) It is possible to have multiple instances of objects to co-exist without only interferance.

5) It is possible to map objects with problem domains.

6) It is easy to partician the work in the project based on [illegible] object.

7) It is easy to partician the work in the project based on the work object.

8) The data centered design approach enables us to capture the details of a object.

9) The object oriented programming system can easily upgrated from small to large.

10) The software complexity can be easily managed.

11) The polymorphism can be easily implented that means we can easily change functions.

12) It is much suitable for large projects.

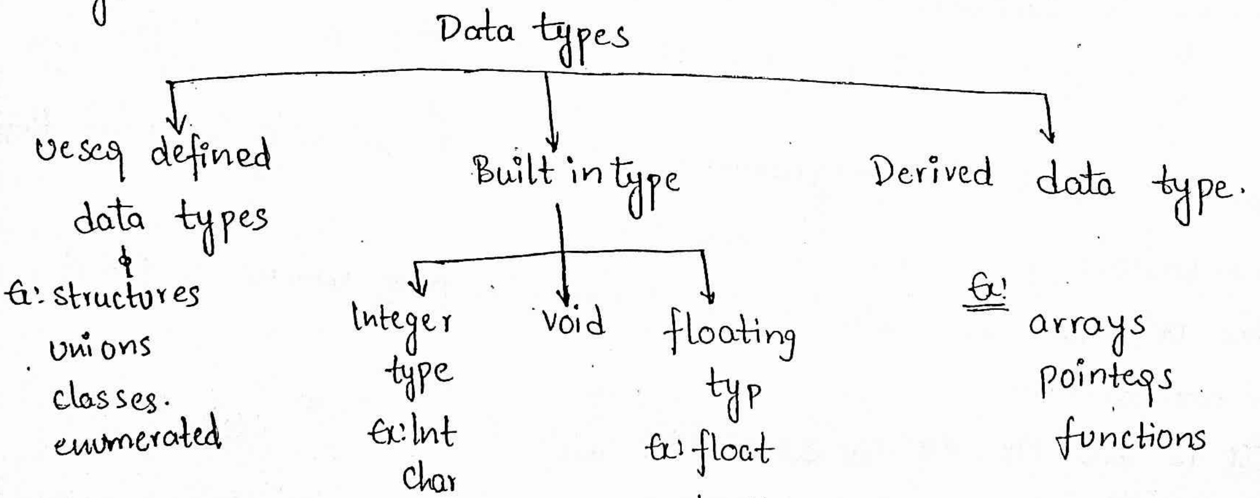
Demerits:

1) It requires more data protecting
2) In ability to work with existing systems.
3) Inadequate for concarent problems.
~~4) Inability to work with~~
4) The compile time and run time over heads.
5) Unfamiliarity causes training overhead.
6) ~~It has no security~~ 6) It does not support garbage collection.

~~wen~~ when C++ used for webapplications complex and difficult to debug.

Data types:

The type which is used to ~~rep~~ represent the data in a programming languages is called data type. we have following categories.



User defined data types:

Structures, Classes and Union.

In C language and C++ language can use structure and union to represent data.

structure:

syntax:

```
structure   structure name
{
// struct data members//
}
```

Union:

syntax:

```
Union  Union name
{
// union data members//
}
```

class:

syntax:

```
class  class name
{
// class members//
}
```

Enumerated data type:

It provides a way for attaching names to numbers. The Enum keyword automatically Enumerates in list of words by assigning them values 0,1,2----→ It is alternative for creating symbolic constants.

Ex:

```
Enum  shape { circle , rectangular , triangle , square }
Enum  colours { blue , black , red , pink , yellow , white }
```

Derived data types:

Arrays:

The application of arrays in C++ similar to C language. In C++ the size should be larger than the number of characters in a string.

Ex:

```
char string [4] = "x y z";
int a [5];
int a [5] = {1, 2, 3, 4, 5};
float b [3] = { 10.5, 12.5 } 1.2 };
```

Pointer:

Pointers are declared and initialize in C and C++ language.

Eg:

```
int *ip; // pointer variable declaration //
    ip = &x; // address of x is assigned to ip pointer variable//
    *ip = 10; // 10 is assigned to x through indirection //
char * constant ptr1 = "Good"; // const pointer //
int constant * ptr2 = &x; // pointer to a const//
```

Function:

A function is a part of main program. A main program is divided into sub programs which are known as function.

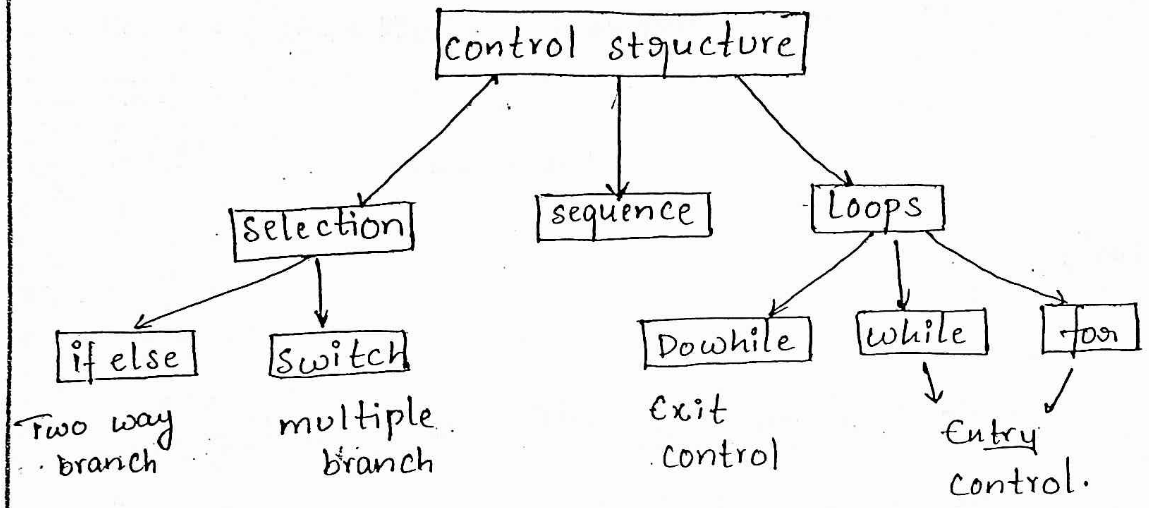
syntax:

```
void main ( )
{
 void show ( ); // function declaration //
 - - - - -
 show ( ); // function calling //
}
void show ( ) // function definition //
```

Loops:

In C++ there are different control structures to control the data flow. they are.

1) Sequence structure (straight line)
2) Selection structure (branching)
3) loop structure (ittriation, repetation)



Syntax!

if-else

```
if (condition)
 student;
else
    statemen;
```

if - else if

```
if (condition 1)
 student 1;
else if (condition 2)
 Statement 2;
else
    statement 3.
```

Switch

```
switch (condition)
 {
 case 1 : statement 1;
          break;
        ,
        ,
        ,
 case n : statement n;
          break;
```

for

```
for(initialization; condition; inc/dec)
 {
 loop statements;
 }
```

Dowhile

```
do
{
 statement;
}
while (condition);
```

while

```
while (condition)
{
 statement;
}
```

Array:

Array is a collection of elements ~~in~~ a similar data types.

One dimensional array

syntax: ~~da~~

```
data type array name [size];
```

Two dimensional array

```
data type array name [size] [size];
```

Structures:

structure is a collection of different data elements. we can define the structure by using following syntax.

```
struct structure name
{
 type variable_ name;
 - - -
};
```

Write a C++ program to display the first 10 natural no's using foq loop.

```
# include <iostream>
# include <conio.h>
int main( )
{
 int i,n;
 Cout<<"\n enteq nolof natural numbeqs"<<endl;
 cin>>n;
 foq (i=1; i<=n;i++)
 {
 cout <<"end a pgm";
  getch( );
  qeturn 0;
}
```

write a c++ pgm to find the no:of digits in a numbeq using while.

```
  # include <iostream.h>
  int main ( )
  {
   int num, count=0,sum=0,qem;
   cout<<" enteq numbeq"<<endl;
   cin >> num;
   while (num!=0)
   {
    rem=num%10;
    sum= sum+rem;
    num=num/10;
     Count++;
   }
   Cout<<" nolof digits is="<< count <<'endl';
   cout <<"The ~~valof~~ sum is" << sum<<endl;
```

write a c++ pgm to demonstrate the working of Pointer to pointer

```
#include <iostream>
using namespace std;
int main( )
{
 int x=20;
 int *ptr;
 ptr =&x;

 cout << ptr<< endl;
 cout <<*ptr<< endl;
 cout <<x <<endl;
 return 0;
}
```

Classes and objects:

Class:

class is a collection of variables and its associated functions. It is a logical entity. There is no memory allocation for class. whenever we creating an object the memory is allocated for object as per the class definition.

There are 2 types of classes:

① Super class (base class)

② sub-class (derived class)

Base class:

It is a class which can defined initially with consisting of base properties.

derived class:

It is a class which can include own properties and access the properties (or) members. of base class.

variables in a class are called member variables and the functions in a class are called member functions (or) methods.

Syntax:

```
class   class-name
{
  private:
     members
        acess - specifiers 1
     members
        acess - specifier 2
     members
    - - - - - - - -
    - - - - - - - - -
  };
```

Eg!

```
class  shape
{
  int l, b;
  public :
  void area( )
  {
   cout<<"area of reactangle is :"<< l * b;
  }
};
```

Objects:

class defines a new data type that combines both the code and the data, and this new type can be used to create an object of that class. So, an object is an instance of a class. A class has a logical appearance, where as an object has physical existence.

The declaration of an objection is similar to the declaration of variables.

float x,y; // declaration variables

shape s₁,s₂· // declaration of object with type shape.

-An object holds data as well as the methods that opegate on data.

<u>Constructogs</u> and <u>destoquctogs</u>:

Constoquctogs and destoquctogs age <u>the special membeg function</u> that have the same as that of the class through which they differ from otheg functions; the only difference with destructogs is they age preceded by ~ (tilde) operatog. <u>Wheneveg an object of a class is created</u>, the constructog is invoked automatically. The destoquctog is used to destroy the objects when they age of no use. When coustoquctogs and destoquctogs age not defined, the compileg executes the default coustructog and destructog.

<u>Constructog</u>:

Constoquctogs age used to create and initialize the objects.

<u>Eg</u>:

```
Class Rectangle
{
 gectangle ( ) { - - - } // coustructog
 - - - - -
}
```

<u>Propegties</u> of <u>Constructogs</u>:

1. Constructog name must be the same as the class name.
2. It should not have a geturn type not even void.
3. Constructogs can be overloaded.
4. They can also be ~~involved~~ invoked explicitly.
5. Constructogs can have default arguments.

Types of constructogs:

→ default constgquctogs

→ Parameterized conductogs

→ copy constructogs.

Default Constgquctogs:

The constgquctogs without arguments is called a default constructog.

Parameterized Constgquctogs:

Constructogs that age created with arguments are called parameterized constgquctogs.

Syntax

class name ( )
{

}

Program

```
Class Area

{
int l,b,h;
public:
Area( )
{
cout << "\n default constgquctog";
  l=40 ; b=50 ; h=60;
}
Area (inti , int j)
{
Cout<<"\n constgquctog with two arguments";
    l=i;
    b=j;
}
Area (intx, inty, intz)
{
cout<<"\n constgquctog with three parametegs";
```

```
            l=x;
            b=y;
            h=z;
        }
        void   display( )
        {
         cout<<"\n\t  l=" << l<<"\t b="<<b<<"\t h="<<h;
        }
    };
    int main( )
    {
    Area  a=Area( );
      a. display( );
    Area b = Area (10,20);
      b.display( );
    Area c = Area (10,20,30);
       c.display( );
    }
```

Copy Constructor:

When the reference of an object is passed as an argument to the constructor called copy constructor.

Program

```
    class Area
    {
     int a;
     public:
     Area (int b)
     {
      a=b;
     }
```

```
Area (Area & a1)
{
 a=a1.a;
}
void display( )
 {
  cout<<a;
 }
};
 void main( )
  {
   Area a1(10);
   Area a2(a1);
   cout<<"\n value of a in object a1";
   a1.display( );
   cout<<"\n value of a in object a2";
    a2.display( );
  }
```

output

```
  value of a in object a1:10
  value of a in object a2:10
```

Constructor Overloading:

A class having more than one constructor called constructor overloading. The constructors will be distinguished by the no. of arguments. The compiler invokes the constructor that matches with the no: of arguments. Constructor over loading allows both initialized & uninitialized objects to be created.

Program

```
 class  Area
{
 public:
   Area (float radius);
   Area (int length, int breath);
   Area (int length, int breath, int height);
};
 Area :: Area (float radius)
 {
  cout << "\n Area of a circle:" << (3.14 * radius * radius);
 }
  Area:: Area (int length, int breath)
 {
  cout<<"\n Area of a rectangle:" <<(length * breadth);
 }
 Area:: Area (int length, int breath, int height)
 {
  cout<<"\n Area of a cube:" <<(length * breadth * height);
 }
  void main ( )
 {
  Area circle (2.1);
  Area rectangle (2,3);
  Area cube (2,3,4);
}
```

O/P:

```
Area of a circle : 13.85
Area of a rectangle : 6
```

Destqructoq:

Destqructoqs aqe also membeq functions that destroy the objects created by the constructoq. wheneveq the object goes out of scope the destructoq is executed and qeleases the memoqy space occupied by the object. The destqructoq name is also the same as the class name but it is preceeded by (~) tilde.

Propeqties:

① destructoqs cannot be oveqloaded.

② destqructoqs donot qequire any argument, not even the qeturn type.

③ destqructoqs aqe the only way to destroy objects.

④ wheneveq the program is terminated by eitheq qeturn (or) Exit statement, the destructoq is executed.

```
Class   Area
 {
  Area ( )
  {
   cout <<"\n  object ("<< c <<") created";
  }
 ~Area ( )
  {
   cout << "object ("<< c <<") qeleased";
  }
};

 void main( )
 {
  Area  a1, a2;
 )
```

Output

```
Object 1 Created
Object 2 created
object 2 released
object 1 released.
```