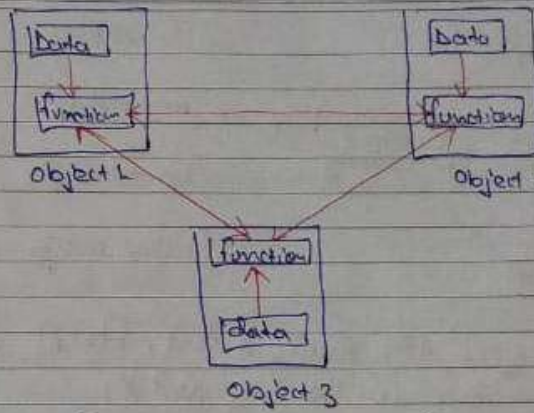
OOPS using C++

Unit 01

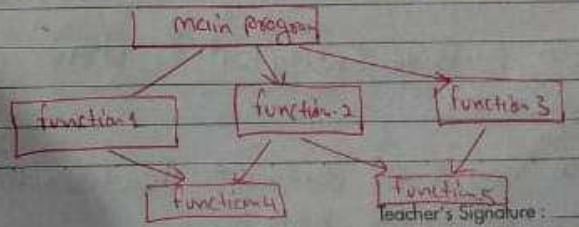
Introduction and features

Q. Differentiate between object oriented programming and procedural oriented programming.

Ans OOPS ⇒ • यह प्रोग्राम को बहुत सारी Entity में विभाजित करता है। जिन्हे object कहते हैं,
- इसमे data तथा function को, एक unit मे Bind किया जाता है। जिन्हे object कहते है।
- यह Bottom-up designing approach को follow करता है,
- इसका use large तथा Complex Application को design करने के लिए किया जाता है।
- इसमे object एक दूसरे से Communicate message passing के द्वारा करता है,
- इसमे Complex user define datatype create किये जाता है।
- इसकी सहायता से Complex application के structure तथा working को easily समझा जा सकता है,
- यह different concept जैसे Inheritance, Polymorphism, Encapsulation, आदि को follow करता है,
- C++, java, .net आदि language इस Approach को follow करती है,



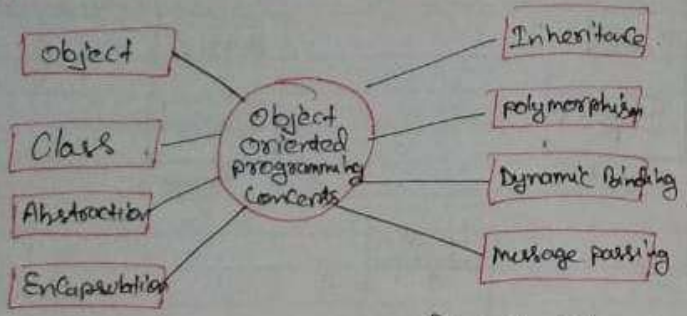
POPS ⇒ • इसमे program बहुत सारे function मे विभाजित होता है।
- इसमे data global define होता है। जिसे कोई भी function easily access कर सकता है।
- यह Top-down designing Approach को follow करता है,
- इसकी सहायता से medium size की Application को design किया जाता है,
- इसमे function parameter pass करके एक-दूसरे से communicate करते है।
- इसमे large, तथा complex Application का structure को समझने मे difficulty होती है,
- इसमे simple user define datatype create किये जाते हैं,
- C, Pascal, fortran, Basic आदि language इसे follow करती है,



Teacher's Signature :

Q. Explain Basic Concepts of OOPs?

Ans



1) Object ⇒ • Object एक Entity होती है, जिसकी State, behaviour तथा identity होती है।
- Object एक Run-time Entity होती है। जो program में Actual work perform करता है।
- इसे Active Entity भी कहते है।
- हर object का एक data structure होता है। जिसे Attributes कहते है।

जैसे Object → Account.
Attribute → Name, Account No., Balance etc.

Class ⇒ • Class उन object का collection होता है। जिनका same Attribute तथा common behaviour होता है।
- Class एक passive Entity होती है। जिसमे work object करता है।
- Class एक way है जो data तथा function को एक unit में Bind करता है।

Expt. No. Date
Page No.

जैसे → furniture एक class है। table, chair, sofa इसके object है।

3) Data Encapsulation ⇒ • एक single unit में data तथा function को wrapp करना Encapsulation कहलाता है।
- Encapsulation की सहायता से different class object communicate करते है।
- Student class encapsulate data & function such as rollno, name, Input(), Show() etc.
- Encapsulation की सहायता से C++ में data hiding की जाती है।

4) Data Abstraction ⇒ • OOPs में complexity को manage करने के लिए Abstraction concept का use किया जाता है।
- यह एक process है। जिसकी सहायता से बिना internal detail, जाने Essential features को use किया जा सकता है।

Ex ⇒ television remote.

5) Inheritance ⇒ • यह एक process है जिसमे पहले से मोजूद class से नयी class create की जाती है।
- नयी class को drive class, कहते है, तथा पहले से मोजूद class को base class कहते है।
- जब एक class दूसरी class की property (function member or data member) को Access करता है, तो इसे inheritance कहते है।
- इसमे derived class, base class के सभी feature को Access करता है।

Teacher's Signature :

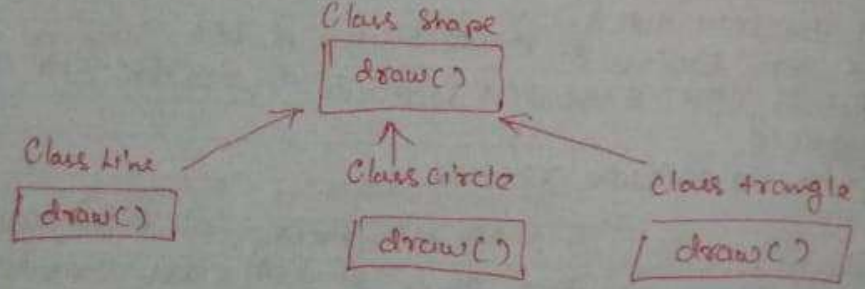
Types of inheritance

1) Single inheritance
2) multi Level Inheritance
3) multiple Inheritance
4) Hierarchical Inheritance
5) Hybrid Inheritance

6) polymorphism ⇒ polymorphism दो word से मिलकर बना है। poly + morphism

जिसमे poly का मतलब बहुत सारे, morphism का मतलब forms.

- polymorphism एक technique है जिसमे same name के function को बार-बार use किया जाता है।
- इसमे same name द्वारा different task perform किये जाते है।



7) Dynamic Binding ⇒ • Binding की सहायता से different function को आपस मे Link किया जाता है।
- function Call की सहायता से different function Link किये जाते है।

Expt. No. Date Page No.

Binding दो प्रकार की होती है।

1) Static Binding or Early Binding
2) Dynamic Binding or Late Binding

1) Static Binding ⇒ . इसमे function call compile time पर की जाती है। इसकी सहायता से program efficient तथा faster हो जाते है।

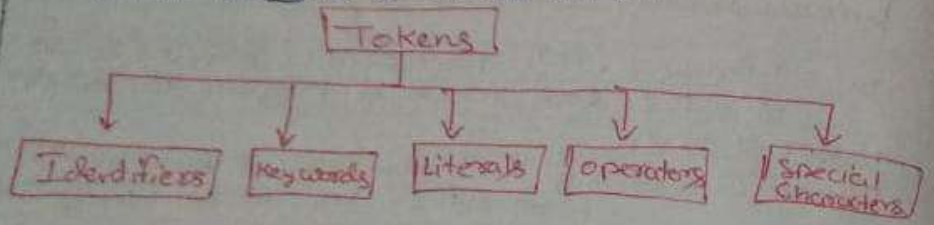
2) Dynamic Binding ⇒ इसमे function call Run time पर की जाती है। यह more flexible होता है।

8) Message passing ⇒ in OOPs भिन्न object आपस मे Link message passing की सहायता से होते है।

- Message passing मे message एक object से दूसरे object मे send तथा receive किया जाता है।
- इसका use object को communicate करने मे किया जाता है।

(6) What do you mean by tokens: Explain it's types. (7)

Ans ⇒ • Token Character का Collection होता है। जिनका प्रयोग Specific task perform करने के लिए किया जाता है।
• C++ में पाँच प्रकार के tokens होते हैं।



1) Identifiers ⇒ • यह Alphanumeric Character की एक String होती है।
• इनका प्रयोग Variable तथा function का नाम देने के लिए किया जाता है।
Ex ⇒ Area, total, Var etc.

2) Keywords ⇒ • यह एक reserved words होते हैं, जिनकी predefine meaning होती है।
• इनका प्रयोग special purpose के लिए किया जाता है,
• Keyword program में user define identifier की तरह work करते हैं।
Ex ⇒ int, long, if, goto, for, Catch etc.

3) Literals or Constants ⇒ • यह वह token होते हैं, जिनकी value fixed होती है।
• इनकी value को Execution time पर Change नहीं किया जा सकता है।

Expt. No. (8) Date
Page No.

विभिन्न प्रकार के होते हैं,

• Literals

a) Integer Constants ⇒ यह +Ve तथा -Ve decimal value होती है।
Ex ⇒ 365, -127, +84

b) Character Constant ⇒ जो भी value single quotes के अंदर define होती है, उसे Character Constant कहते हैं।
Ex ⇒ 'a', '2' 'in'

c) Floating Constant ⇒ • यह integer value का fraction part होता है, जिसे e or E से denote किया जाता है,
Ex ⇒ 4.523, 1.4e7, 4.7e-12

d) String Constant ⇒ • Character का collection string कहलाता है।
• इसे double quotes के अन्दर define किया जाता है,
जैसे "Aman", "A", "2"

Q ⇒ What is operator? Explain it's types.
Ans ⇒ operator एक symbol होते हैं। जो Expression को operate करते हैं।
operator निम्न प्रकार के होते हैं,
1) Arithmetic operators (+, -, *, /)
2) Assignment operators (=, +=, -=, *=)
3) Increment/decrement operators (++, --)
4) Relational operators (<, <=, >, >=)
5) Logical operators (&&, ||, !)
6) Conditional operators (?:)
7) Special operators (,)

Teacher's Signature :

Q> Explain the term class & object with Examples. (9)

Ans> Class ⇒ • data तथा function को एक Single unit में Bind करना Class कहलाता है।

- Class object का Collection होता है।
- यह एक Passive Entity होती है। जिसमे Actual work Object करता है।
- जो Variable Class के अन्दर define होते है, उन्हे data member तथा function को member function कहते है।
- Class में data तथा function को तीन प्रकार से define करते है।

1 Public
2 Private
3 Protected

Syntax ⇒

```
Class <Classname>
{
  private:
  data members;
  member function;
  public:
  data members;
  member function;
  protected:
  data members;
  member function;
};
```

Expt. No. (10) Date

Page No.

Ex ⇒

```
Class Student
{
  int rollno;
  float marks;
  public:
  void getdata (int x, float y);
  void display (void);
};
```

Object ⇒ • object किसी भी Class का एक instance part होता है।

- object एक Active Entity होती है, जो program में Actual work Perform करता है।
- Class एक user define datatype है, जबकि object Class का एक instance part है।
- object Run time Entity है।

Syntax ⇒

```
Class <Classname>
{
  -- // Body of class
  --
};  S1, S2, S3; } object creation
```

Ex ⇒

```
Class Student
{
  --
  --
};  S1, S2, S3;
```

यहाँ S1, S2, S3 Student Class के object है।

Teacher's Signature :

Q. Explain Constructor and types of Constructor.

Ans ⇒
- Constructor एक Special member function होता है, जिसकी सहायता से Class के object को initialize किया जाता है,
- Constructor Class का एक special member function होता है, जो Automatic ~~Error~~ Call हो जाता है जब object create किया जाता है,
- Constructor function का नाम Class के नाम के समान ही होता है,
- Constructor को Public section में declared किया जाता है,
- Constructor का कोई भी return type नहीं होता है, और यह कोई value return भी नहीं करता है,
- Constructor को derived class inherit नहीं कर सकती है,

Syntax ⇒

```
Class Classname
{
---
===
public:
Classname ( Parameter list); // Constructor declaration
---
};
Classname :: Classname (Parameter list); // Constructor
{                                           defination
---
---
}
```

Ex ⇒

```
Class Bank
{
private:
int x, y;
public:
Bank();   // Constructor declaration
};
```



```
Bank :: Bank()
{                    // Constructor defined
 x = 0; y = 0;
}
```

There are three types of Constructor

Constructor
- Default Constructor
- Parameterized Constructor
- Copy Constructor

1) Default Constructor ⇒ वह Constructor जिसमें कोई भी Parameter Pass नहीं किये जाते है, उसे Default Constructor कहते है,

Syntax ⇒

```
Class first
{
private:
---
===
public;
first ();     // Default Constructor
};
first :: first()   // Constructor without a
{                     arguments
---
}
```

(2) Parameterized Constructors ⇒ वह Constructor जिसमे Arguments pass किये जाते है, उसे parametrized Constructor कहते है।

Syntax ⇒

```
Class Bank
{
private:
int x, y;
public:
Bank(int a, int b);   // parameterized constructor
};
Bank::Bank(int a, int b);  // Constructor with two arguments.
{
x = a;
y = b;
}
Bank B1;
```

(3) Copy Constructor ⇒ जब कोई Constructor अपनी class के refrence को Argument की तरह use करता है। तो उसे Copy Constructor कहते है।

- Copy Constructor अपनी class का reference use करता है।

Syntax ⇒

```
Sample S1;      // default Constructor used
Sample S2 = S1; // Copy constructor is invoked
```

Ex ⇒

```
Classname::Classname(Classname &ptr)
```

Ex ⇒

```
X::X(X &ptr)
```


Q ⇒ Write a Program to display series of fibonacci number
or
Write a program for Constructor
or
Write a program for class & object
or
Write a program for default Constructor.

Ans ⇒

```
#include <iostream.h>
#include <conio.h>
Class fibonacci
{
  private:
  long int a, b, c;
  public:
  fibonacci();
  Void increment();
  Void display();
};
fibonacci::fibonacci()
{
  a = 0;
  b = 1;
  c = a+b;
}
Void fibonacci::increment()
{
  a = b;
  b = c
  c = a+b;
}
```

```
Void fibonacci :: display()
{
 cout << c <<'\t';
}
 Void main()
{
 fibonacci f;
for (int i=0; i<=15; i++)
{
 f.display();
 f.increment();
}
}
```

Output ⇒ 1, 1, 2, 3, 5, 8, 13, ...

Expt. No. Date

Page No.

Q. Explain Destructors with Example :

Ans ⇒
- Destructor का use Constructor द्वारा Create किये गये object को destroy करने के लिए किया जाता है।
- destructor का नाम भी class के नाम के समान होता है। लेकिन इसमें destructor function से पहले tilde (~) sign का use किया जाता है।
- destructor को public Section में define किया जाता है।
- इसका कोई भी return type नहीं होता है। ना ही ये कोई Arguments लेता है।

Syntax ⇒

```
Class classname
{
private :
// data variables
// methods
public :
Classname();    // Constructor
~classname();   // destructor
// methods
};
```

Q. Write a program to demonstrate destructor function

Ans ⇒

```
# include <iostream.h>
Class sample
{
public :
sample()
{
 cout << "object is Born";
}
```

Teacher's Signature

```
~Sample()        // destructor Created.
{
Cout<<"object Dies";
};
Void main()
{
Sample S;        // Constructor is called
Cout<<" main terminated";
}
```

Q⇒ What do you mean by inline function?

Ans⇒
- जब function Class के अन्दर define किया जाता है तो उसे inline function कहते हैं।
- छोटे-छोटे function को Class के अन्दर define किया जाता है,
- इन function की सारी functioning class के अन्दर ही की जाती है।

Syntax⇒

```
Class Classname
{
private:
int a, b, c;
public:
Void int (int x, int y, int z)  // inline function.
{
x = a;
y = b;
z = c;
} };
```


Q What do you mean by friend function? Explain with Example

Ans⇒
- friend function किसी भी Class का member नहीं होता है, किन्तु इसे special permission होती है, private और protected data को Access करने की।
- friend function को friend keyword के द्वारा declared किया जाता है।
- friend function को Class के object के द्वारा call नहीं किया जाता है।
- friend function को Class में private or public किसी भी part में declare किया जा सकता है।
- friend function की सहायता से non member function को Access किया जा सकता है।
- friend function को Class के अन्दर declared or define किया जा सकता है।

Q⇒ WAP to access the private data of a class by non member function through friend.

Ans⇒

```
#include <iostream.h>
#include <conio.h>
Class Item
{
int x;
int y;
public:
Void setdata()
{
x = 14;
y = 11;
}
friend float sample(Item i)  // friend declared
};
```

Teacher's Signature :

```
float Sample (Item i)
{
 return float ( ix + i.y)/2.0;
}
Void main()
{
 Clrscr();
 Item a;        // Object a
 a. Setdata();
Cout << " mean Value =" << Sample(a) << "\n";
}
```

Output ⇒ mean value = 12.5

Q ⇒ What do you understand by operator overloading ? Give with Example.

Ans ⇒
- operator को नया नयी meaning provide करना operator overloading कहलाता है।
- operator overloading में user define datatype, inbuilt datatype के समान work perform करता है।
- operator को overload operator function की सहायता से किया जाता है।
- जो operator पहले से language में available है केवल उन्ही को overload किया जाता है।
- इससे operator की meaning को change नही किया जाता है जैसे + operator का प्रयोग - के लिए नही किया जाता है।

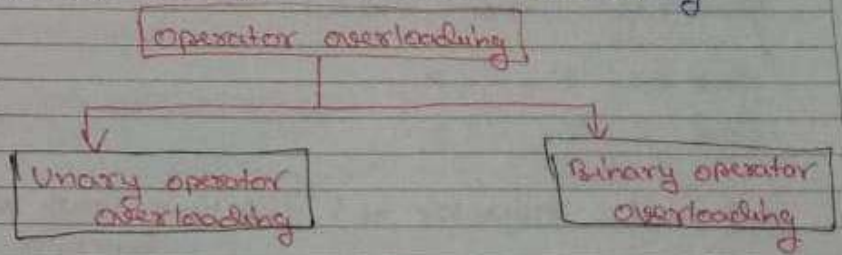
- operator function को operator keyword की सहायता से create किया जाता है।

Syntax ⇒ (it's the type of value returned by the specified operation)

```
Return type Classname :: operator op (arguments list)
{
  function body
}
```

(op ↓ operator being overloaded)

There are two type of operator overloading



Unary operator overloading ⇒

- unary operator single operand पर कार्य करता है।
- Unary operator बिना किसी formal argument के member function को overload करता है।
- unary operator (+), (−), (++), (−−) and (!) है।

Q ⇒ Write a program in C++ to overload a unary minus operator.

Ans ⇒

```
# include < iostream.h>
# include < conio.h>
 Class first
 {
   int x, y;
```

Teacher's Signature :

```
public:
void getdata(int a, int b);
void display();
void operator -();   // declaring overload unary minus
};
void first :: getdata(int a, int b)
{
  x=a; y=b;
}
void first :: display (void)
{
  cout << "x=" << x;
  cout << "y=" << y;
}
void first :: operator -() . // defining unary minus
                             operator
{
  x=-x;
  y=-y;
}
void main()
{
  clrscr();
  first f;
 f.getdata (17, -34)
 -f;      // Calling operator -() function.
cout << "After Calling unary operator f=";
f.display();
getch();
}
```


Binary operator overloading ⇒ • Binary operator दो operands पर काम करता है।

- Binary operator member function की सहायता से overload किया जाता है। इसमें एक formal argument pass किया जाता है।
- Arithmetic operators +, -, *, /, % overloaded in Binary.

Ex ⇒ Write a program in C++ to overload a binary operator
or
Write a program to add Complex number.

```
# include <iostream.h>
# include <conio.h>
  class Complex
   {
      int a, b;
   public:
     void getdata()
   {
     cout << " Enter the value of Complex no.";
     cin >> a >> b;
   }
   Complex operator +(Complex ob)
   {
     Complex t;
     t.a = a + ob.a;
     t.b = b + ob.b;
     return(t);
   }
```

Teacher's Signature :

```
Complex operator - (Complex ob)
 {
   Complex t;
   t.a = a - ob.a;
   t.b = b - ob.b;
   return(t);
 }
  Void display ()
{
 Cout << a << "+" << b << "i" << "\n";
 } };
Void main()
{ Clrscr();
Complex ob1, ob2, result, result1;
 ob1.getdata();
 ob2.getdata();
 result = ob1 + ob2;
 result = ob2 - ob2;
Cout << "Input values";
 ob1.display();
 ob2.display();
Cout << "Result:";
 result.display();
 result1.display();
getch();
}
```


Q) What do you understood by function overloading? Give an example illustrating its use in a C++ program.

Ans: जब एक single class में same name का function, different parameter (Argument) की सहायता से प्रयोग किया जाता है तो उसे function overloading कहते हैं। इसमें function को बार-बार use किया जाता है, or भिन्न class

- वस्तुतः सारे function एक class के अन्दर same name को share करते हैं तथा उनके Argument different-different होते हैं तो उसे function overloading कहते हैं।
- function overloading की सहायता से same operation के लिए different function name को Eleminate करके केवल एक single name दिया जाता है।

Syntax ⇒

```
Class Classname
{
int a, b, c;

Void sum (int x, int y)
{
 ---
}

Void sum (int *p, int q, int r)
{
 ---
 ---
}
 };
```

Syntax ⇒

```
int Sum (int a, int b)
{
 ---
}
float Sum (int p, int q)
{
 ---
}
```

Teacher's Signature :

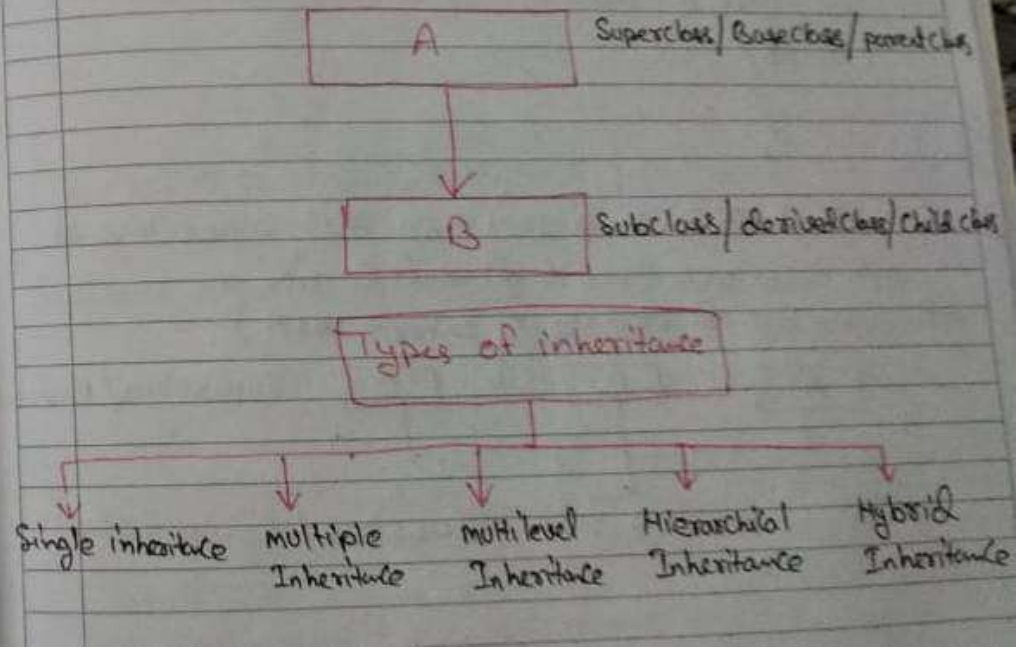
WAP for function overloading. (25)

Ans
```
#include <iostream.h>
#include <conio.h>
int add (int x, int y)
{
  return (x+y);
}
float add (float x, float y)
{
  return (x+y);
}
void main()
{
int a=6, b=10, c;
float p=14.00, q=10.00, r;
c = add (a, b);
r = add (p, q);
Cout << "c=" << c;
Cout << "r=" << r;
getch();
}
```

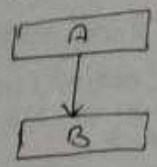
Expt. No. (26) Date Page No.

Q) What is Inheritance? Explain different types of inheritance with example.

Ans
- Inheritance एक process है जिसकी सहायता से existing class (base class) से new class (derived class) create की जाती है।
- यह एक process है जिसकी सहायता से एक class दूसरी class की property को inherit करती है।
- derived class को subclass, child class तथा base class को superclass, parent class कहते हैं।
- Inheritance को Code reusability भी कहते हैं।
- derived class एक base class की तरह भी work करता है जिससे subclass भी create की जाती है।



1) Single Inheritance ⇒ इसमे एक subclass तथा एक ही superclass होती है, subclass, superclass की property को inherit करती है।



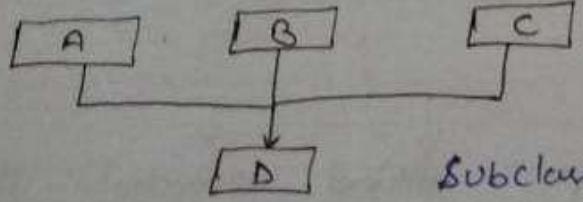
Superclass

Subclass

Syntax ⇒

```
Class A          // Base class / superclass
{
- -
- -
};
Class B : public A    // derived class or Subclass
{
- -
- -
};
```

2) Multiple Inheritance ⇒ इसमे बहुत सारी Superclass से एक Subclass create की जाती है। और वह subclass सारी superclass की property को Inherit करती है।



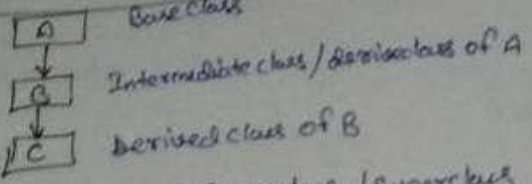
Superclass / Baseclass

Subclass / derived class

Syntax ⇒

```
Class A
{
- -
- -
};
Class B
{
- -
- -
};
Class C
{
- -
- -
};
Class D : public A, public B, public C
{
- -
- -
- -
};
```

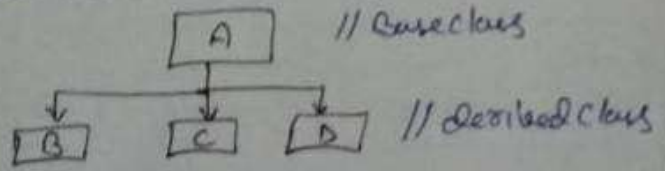
3) multilevel Inheritance ⇒ इसमे एक derived class से दूसरी derived class create की जाती है। derived class intermediate derived class की property को Inherit करती है।



Syntax ⇒

```
Class A    // Baseclass / superclass
{
==
};
Class B: public A   // Subclass / derived class
{
==
};
Class C: public B   // Sub-Subclass / derived class
{
==
};
```

4) Hierarchical inheritance ⇒ इसमें एक Base Class से बहुत सारी derived class Create की जाती है। ये सारी derived class Base Class की property को Inherit करती है,



Syntax ⇒

```
Class A
{
==
};
Class B: public A
{
==
};
```

```
Class C: public A
{
==
};
Class D: public A
{
==
};
```

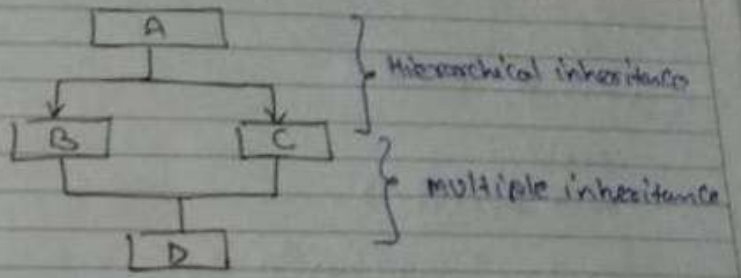

Expt. No. Date

Page No.

5) Hybrid Inheritance ⇒ इस inheritance में दो प्रकार की inheritance को combined किया जाता है।

Hybrid Inheritance Hierarchical और Multiple Inheritance का Combination है।

Hybrid Inheritance = Hierarchical + multiple Inheritance



Syntax ⇒

```
Class A
{
==
};
Class B: public A
{
==
};
Class C: public A
{
==
};
Class D: public B, public C
{
==
};
```

Teacher's Signature

Q. WAP for Inheritance

Ans.


```
include <iostream.h>
include <conio.h>
Class A
{
 public:
 void display()
 {
 Cout << "mainclass";
 }
 };
Class B : public A
{
 public:
 Void display1()
 {Cout << "Subclass of A";
 }
 };
Class C : public B
{ public:
 Void display2()
 {Cout << "Subclass of B";
 }
 };
Void main()
{
 C c1;   object create
 c1.display2();
 c1.display1();
 c1.display();
 getch();
}
```


Q. What do you mean by polymorphism? Explain its types.

Ans. • Polymorphism दो Greek words से मिलकर बना है।

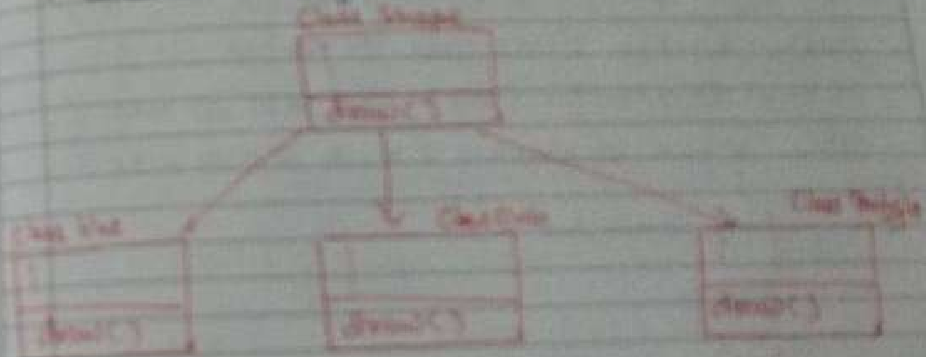
polymorphism = poly + morphos

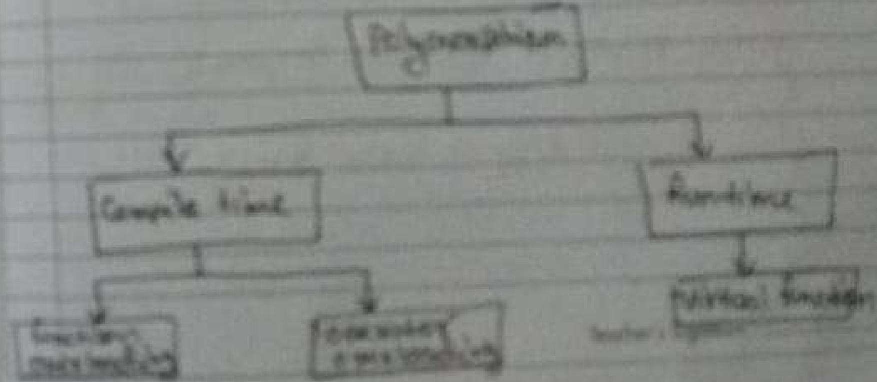
poly का मतलब बहुत और morphism का मतलब forms है।

• polymorphism का मतलब एक नाम अनेक काम है।

• Polymorphism एक ability है जिसमें same name के function द्वारा different-different task perform किये जाते हैं।

• Inheritance polymorphism को support करता है।





Q What is a Virtual function? Explain with Example.

Ans ⇒
- Virtual function वह function होता है। जो really Exist नहीं करता है, किन्तु वह program के कुछ part में work perform करता है।
- Virtual function एक member function होता है। जिसे Virtual keyword से declare किया जाता है।
- Virtual function को Base class में define किया जाता है तथा derived class इसे override करती है।
- Virtual function static member नहीं होता है।
- इसे object pointer की सहायता से Access किया जाता है।
- Virtual function other class का friend होता है।
- Virtual constructor Create नहीं किया जा सकता है।
- किन्तु Virtual destructors Create किया जा सकता है।

Syntax ⇒

```
Class sampleclass
{
private:
- - - -
- - - -
public:
Virtual returntype functionname (arguments) //virtual function
{
- - -
- - -        // Body of Virtual function
}
};
```


Ex ⇒

```
Class Sample
{
 private:
 int x;
 float y;
 public:
Virtual void display();  // Virtual function.
Virtual int Sum();
 };
```

WAP for virtual (overriding) function.

```
# include <iostream.h>
# include <conio.h>
  Class Sample
  {
   public:
  Virtual void display()
   {
   Cout << " Sample class";
   }
    };
Class Sample1 : public Sample
{
 public:
  Virtual void display()
 {
  Cout << " Sample1 Class";
 }
   };
```

Teacher's Signature :

```
void main()
{
    Sample *a = new Sample;
    SampleL *b = new SampleL;
    a -> display();
    b -> display();
  getch();
  }
```

output:

Sample Class.

SampleL Class.

Expt. No.