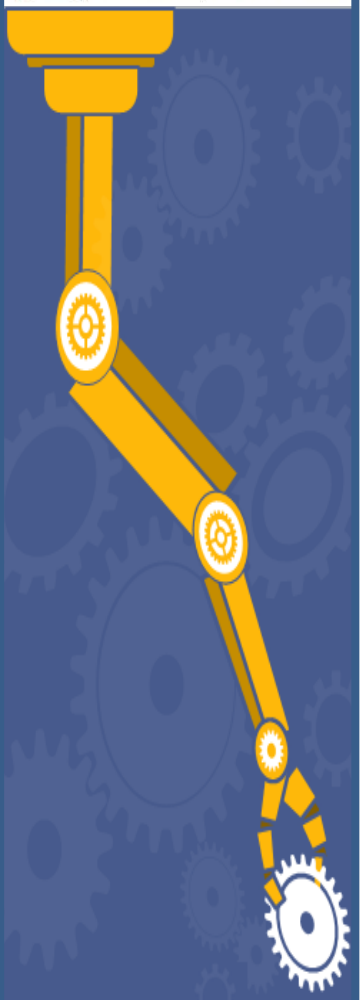
VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)



BCSE102L- Structured and object-oriented programming

Dr. P.Keerthika
Associate Professor
School of Computer Science & Engineering
VIT, Vellore.

BCSE102L- Structured and object-oriented programming

Module:1	**C Programming Fundamentals**	**2 hours**
Variables - Reserved words – Data Types – Operators – Operator Precedence - Expressions - Type Conversions - I/O statements - Branching and Looping: if, if-else, nested if, if-else ladder, switch statement, goto statement - Loops: for, while and do…while – break and continue statements.		
Module:2	**Arrays and Functions**	**4 hours**
Arrays: One Dimensional array - Two-Dimensional Array – Strings and its operations. User Defined Functions: Declaration – Definition – call by value and call by reference - Types of Functions - Recursive functions - Storage Classes - Scope, Visibility and Lifetime of Variables.		
Module:3	**Pointers**	**4 hours**
Declaration and Access of Pointer Variables, Pointer arithmetic – Dynamic memory allocation – Pointers and arrays - Pointers and functions.		
Module:4	**Structure and Union**	**2 hours**
Declaration, Initialization, Access of Structure Variables - Arrays of Structure - Arrays within Structure - Structure within Structures - Structures and Functions – Pointers to Structure -		
Module:5	**Overview of Object-Oriented Programming**	**5 hours**
Features of OOP - Classes and Objects - "this" pointer - Constructors and Destructors - Static Data Members, Static Member Functions and Objects - Inline Functions – Call by reference - Functions with default Arguments - Functions with Objects as Arguments - Friend Functions and Friend Classes.		
Module:6	**Inheritance**	**5 hours**
Inheritance - Types of Inheritance: Single inheritance, Multiple Inheritance, Multi-level Inheritance, Hierarchical Inheritance - Multipath Inheritance - Inheritance and constructors.		
Module:7	**Polymorphism**	**4 hours**
Function Overloading - Operator Overloading – Dynamic Polymorphism - Virtual Functions - Pure virtual Functions - Abstract Classes.		
Module:8	**Generic Programming**	**4 hours**
Function templates and class templates, Standard Template Library.		
	Total Lecture hours:	**30 hours**

BCSE102L- Structured and object-oriented programming – Text Books and Reference Books

Text Book(s)	
1.	Herbert Schildt, C: The Complete Reference, 4th Edition, McGraw Hill Education, 2017
2.	Herbert Schildt, C++: The Complete Reference, 4th Edition, McGraw Hill Education, 2017.
Reference Books	
1.	Yashavant Kanetkar, Let Us C: 17th Edition, BPB Publicaitons, 2020.
2.	Stanley Lippman and Josee Lajoie, C++ Primer, 5th Edition, Addison-Wesley publishers, 2012.

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

BCSE102P- Structured and object-oriented programming Laboratory

	Indicative Experiments
1.	Programs using basic control structures, branching and looping
2.	Experiment the use of 1-D, 2-D arrays and strings and Functions
3.	Demonstrate the application of pointers
4.	Experiment structures and unions
5.	Programs on basic Object-Oriented Programming constructs.
6.	Demonstrate various categories of inheritance
7.	Program to apply kinds of polymorphism.
8.	Develop generic templates and Standard Template Libraries.

Text Book(s)	
1.	Robert C. Seacord, Effective C: An Introduction to Professional C Programming, 1st Edition, No Starch Press, 2020.
Reference Book(s)	
1.	Vardan Grigoryan and Shunguang Wu, Expert C++: Become a proficient programmer by learning coding best practices with C++17 and C++20's latest features, 1st Edition, Packt Publishing Limited, 2020.

BCSE102L- Structured and Object-Oriented Programming

- **Module-6: Inheritance**
  - **Inheritance- Introduction & Types**
  - **Single Inheritance**
  - **Multiple Inheritance**
  - **Multilevel Inheritance**
  - **Hierarchical Inheritance**
  - **Multipath/Hybrid Inheritance**
  - **Inheritance and Constructors**

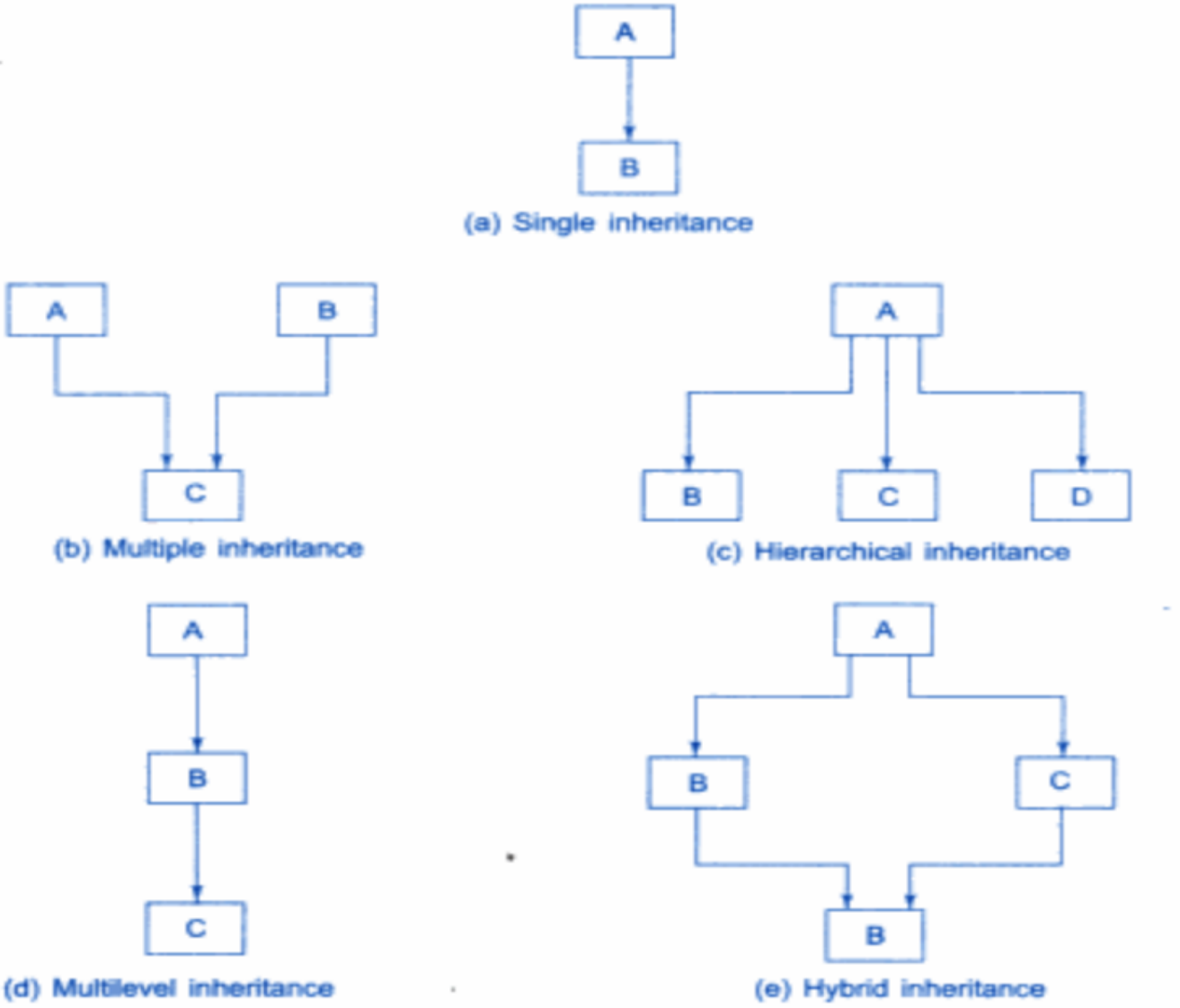
Inheritance

- The capability of a class to inherit the properties of some other class is known as Inheritance.
- There are two types of classes in inheritance:
- **Parent Class or Base Class** :- It is a class whose property is inherited by a subclass. For example "Account" is a parent class.
- **Child Class or Derived Class** :- It is a class that inherits the property of another class. For example, Savings and current account are child class.

Why Inheritance and when to use??

- We use inheritance in C++ when both the classes in the program have the same logical domain and when we want the class to use the properties of its superclass along with its properties.
- For example, there is a base class or parent class named "Animal," and there is a child class named "Dog," so, here dog is an animal, so in "Dog class," all the common properties of the "Animal" class should be there, along with its property of dog animal.

Types of Inheritance



Defining Derived class

- Derived class can be defined by specifying its relationship with the base class in addition to its own details.
- **General Form of defining derived class:**

```
class derived-class-name  : visibility-mode base-class-name
{
      .....//
      .....//   members of derived class
      .....//
};
```

Modes of Inheritance

- Public Inheritance

```
class B : public A
{
   ----------
   ----------
};
```

- class B : public A tells the compiler that we are inheriting class A in class B in public mode

- What to understand??
- All the public members of class A becomes public members of class B.
- All the protected members of class A becomes protected members of class B.
- Private members are never inherited.

Modes of Inheritance

- Private Inheritance

```
class B : private  A
{
   ----------
   ----------
};
```

- class B : private A tells the compiler that we are inheriting class A in class B in private mode
- What to understand??
- All the public members of class A becomes private members of the class B.
- All the protected members of the class A becomes private members of class B.
- Private members are never inherited.

Modes of Inheritance

- Protected Inheritance

```
class B : protected  A
{
   ----------
   ----------
};
```

- class B : protected A tells the compiler that we are inheriting class A in class B in protected mode
- What to understand??
- All the public members of class A becomes protected members of class B.
- All the protected members of class A becomes protected members of class B.
- Private members are never inherited.

General syntax of inheritance in C++

- **General Form:**

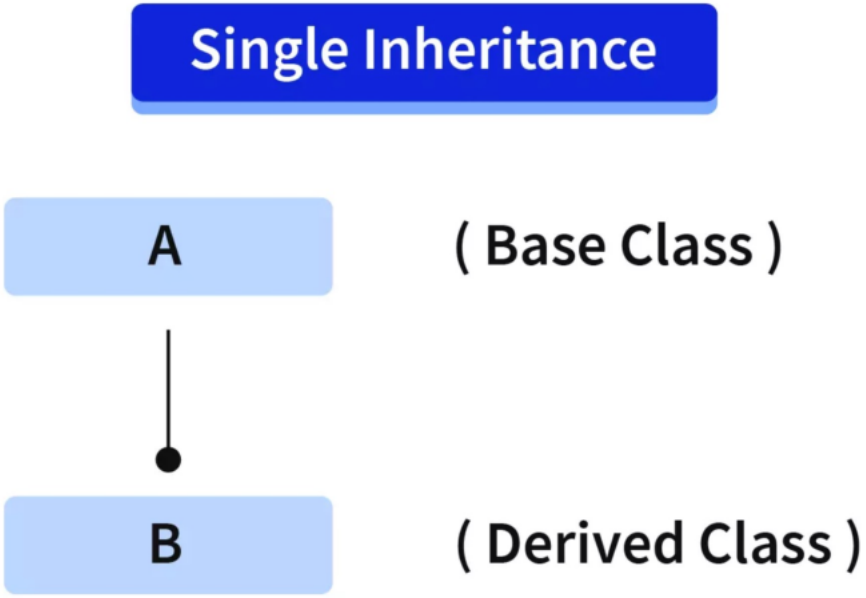
```
class Parentclass_name{
    Accessspecifier:
    Datamemebers;
    memberfucntions(){

    }
};
class childclass_name : Modeofinheritance Parentclass_name1,Modeofinheritance
    Parentclass_name2...{
    Accessspecifier:
    Datamemebers;
    memberfucntions(){

    }
};
```

Single Inheritance

- When the derived class inherits only one base class, it is known as Single Inheritance.



Example -1 for Single inheritance

```
class super
{
public :
int sup_a;
void sup_show( )
{
cout<<"Hello from show of super"<<endl;
}
};
```

```
class sub :public super
{
};
void main( )
{
sub o1;
o1.sup_a=20;
o1.sup_show ( );
}
```

Example -II for Single inheritance

```
class super
{
int sup_a; // private
public :
void sup_input(int x)
{
sup_a=x;
}
void sup_show( )
{
cout<<"sup_a="<<sup_a<
<endl;
}
};
```

```
class sub :public super
{
};
void main( )
{
int i;
sub o1;
cin>>i;
o1.sup_input(i);
o1.sup_show( );
}
```

Example -III

```
class super // Base Class
{
int sup_a;
public :
void sup_input(int x)
{
sup_a = x;
}
void sup_show( )
{
cout<<"sup_a="<<sup_a<<e
ndl;
}
};
```

```
class sub : public super
{  // Derived Class
int sub_a;
public :
void sub_input(int x)
{
sub_a=x;
}
void sub_show( )
{
cout<<"sub_a="<<sub_
a<<endl;
}
};
```

Example -III

```
int main( )
{
int i,j;
sub o1;
cin>>i; // data member for super class
cin>>j; // data member for sub class
o1.sup_input(i);
o1.sub_input(j);
o1.sup_show( );
o1.sub_show( );
return 0;
}
```

Observation:

After inheriting class super in class sub, the class sub has total 4 public members' functions.

- **Two of its own**
- **Two inherited**

- The class sub also has one private data member sub_a.
- In the main all 4 functions are called by an object of sub class.

Example -IV

```
class super // Base Class
{
int sup_a;
public :
void sup_input(int x)
{
sup_a = x;
}
void sup_show( )
{
cout<<"sup_a="<<sup_a<
<endl;
}
};
```

```
class sub :public super
{  // Derived Class
int sub_a;
public :
void sub_input(int x)
{
sup_input(x*2);
}
void sub_show( )
{
sup_show();
cout<<"sub_a="<<sub_a
<<endl;
}
};
```

Example -IV

```
int main( )
{
int i,j;
sub o1;
cin>>i; // data member for
super class
o1.sub_input(i);
o1.sub_show( );
return 0;
}
```

OUTPUT :

10

sup_a=20

sub_a=0

Example -V

```
class super // Base Class
{
int sup_a;
public :
void sup_input(int x)
{
sup_a = x;
}
void sup_show( )
{
cout<<"sup_a="<<sup_a<
<endl;
}
};
```

```
class sub :public super
{  // Derived Class
int sub_a;
public :
void sub_input(int x, int y)
{
sup_input(y); sub_a=x;
}
void sub_show( )
{
sup_show();
cout<<"sub_a="<<sub_a<
<endl;
}
};
```

Example -V

```
int main( )
{
sub o1;
o1.sub_input(20,50);
o1.sub_show( );
return 0;
}
```

OUTPUT :

sup_a=50

sub_a=20

Example -VI

```
class super // Base Class
{
int sup_a;
public :
void input(int x)
{
sup_a = x;
}
void show( )
{
cout<<"sup_a="<<sup_a<
<endl;
}
};
```

```
class sub :public super
{  // Derived Class
int sub_a;
public :
void input(int x)
{
sub_a=x;
}
void show( )
{
cout<<"sub_a="<<sub_
a<<endl;
}
};
```

Example -VI

```
int main( )
{
sub o1;
o1.input(50);
o1.show( );
return 0;
}
```

OUTPUT :

sub_a=50

Here both the classes have same function signatures so priority is given to **derived class.** So input and show of derived class will be called.

Example -VII

```
class super
{
protected :
void show( )
{
      cout<<"Hello      from
      super class "<<endl;
}
};
```

In public inheritance protected members of base are inherited in derived class and remains protected. Protected members cannot be used outside the class similar to private members. So, the error.

```
class sub :public super
{
};
int main( )
{
      sub obj;
      obj.show( );
      return 0;
}
```

OUTPUT:

It will give ERROR

show cannot access protected member declared in class 'super'

DEMONSTRATION OF SINGLE LEVEL PRIVATE INHERITANCE

Modes of Inheritance(Recall)

- Private Inheritance

```
class B : private  A
{
   ----------
   ----------
};
```

- class B : private A tells the compiler that we are inheriting class A in class B in private mode
- What to understand??
- All the public members of class A becomes private members of the class B.
- All the protected members of the class A becomes private members of class B.
- Private members are never inherited.

Example -I

```
class super // Base Class
{
public :
void sup_show( )
{
cout<<"Hello from
super"<<endl;
}
};
```

```
class sub :private super
{  // Derived Class
  -------
};

int main( )
{
sub obj;
obj.sup_show( );
return 0;
}
```

ERROR : 'sup_show' cannot access public member declared in class 'super'.

Example -II

```
class super // Base Class
{
public :
void sup_show( )
{
cout<<"Hello from super"<<endl;
}
};
```

```
class sub :private super
{  // Derived Class
 public :
void sub_show( )
{
sup_show();
cout<<"Hello from sub"<<endl;
}
};
int main( )
{
sub obj;
obj.sub_show( );
return 0;
}
```

Hello from super
Hello from sub

Example -III

```
class super // Base Class
{
int num;
};
```

```
class sub : super
{  // Derived Class
 public :
void sub_show( )
{
cout<<"num="<<num<<endl;
}
};
int main( )
{
sub obj;
obj.sub_show( );
return 0;
}
```

ERROR :super : : num cannot access private member declared in class super'

Example -IV

```
class super // Base Class
{
protected :
int num;
void input(int x)
{
num = x;
}
};
```

Num =100

```
class sub :private super
{  // Derived Class
 public :
void sub_show( )
{
input(100);
cout<<"num="<<num<<endl;
}
};
int main( )
{
sub obj;
obj.sub_show( );
return 0;
}
```

DEMONSTRATION OF SINGLE LEVEL PROTECTED INHERITANCE

Modes of Inheritance(Recall)

- Protected Inheritance

```
class B : protected  A
{
   ----------
   ----------
};
```

- class B : protected A tells the compiler that we are inheriting class A in class B in protected mode
- What to understand??
- All the public members of class A becomes protected members of class B.
- All the protected members of class A becomes protected members of class B.
- Private members are never inherited.

Example -I

```
class super // Base Class
{
public :
void show( )
{
cout<<"Hello_from
super"<<endl;
}
};
```

```
class sub :protected
super
{ // Derived Class
  -------
};

int main( )
{
sub obj;
obj.show( );
return 0;
}
```

ERROR : 'show' cannot access public member declared in class 'super'.

DEMONSTRATION OF ACCESSING PRIVATE MEMBERS IN PRIVATE INHERITANCE

Example -I

```
class super // Base Class
{
int num;
public :
void input(int x)
{
num = x;
}
int getnum( )
{
return num;
}
};
```

SQUARE: 2500

```
class sub :private super
{  // Derived Class
public :
void show( )
{
input(50);
cout<<"SQUARE="<<getnu
m()*getnum()<<endl;
}
};
int main( )
{
sub obj;
obj.show( );
return 0;
}
```

Example -II

```
class super // Base Class
{
int num;
public :
void input(int x)
{
num = x;
}
int getnum( )
{
return num;
}
void super_show( )
{
cout<<"Num in super CLASS is"<<num<<endl;
}
};
```

```
class sub :private super
{  // Derived Class
public :
void show( )
{
input(50);
cout<<"Num in sub CLASS is"<<getnum( )<<endl;
getnum()=getnum( )*getnum( );
super_show();
}
};
int main( )
{
sub obj;
obj.show( );
return 0;
}
```