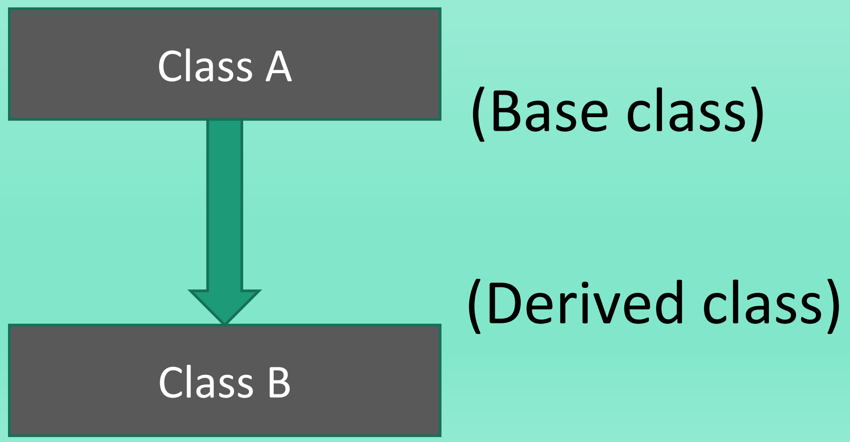
INHERITANC
E

Inheritance in c++

- Inheritance can be defined as the process of creating a new class from one or more existing classes.
- The existing class is known as base class or parent class or super class whereas newly created class is known as derived class or child class or sub class.
- Inheritance provides a significant advantage in terms of code reusability.
- Consider the diagram shown below



The general form of defining a derived class is

```
class derived-class-name: access-specifier base-class-name
{
//members of the derived class
};
```

Here access is one of the three keywords: public,private and protected.

The access specifier determines how elements of the base class are inherited by the derived class.

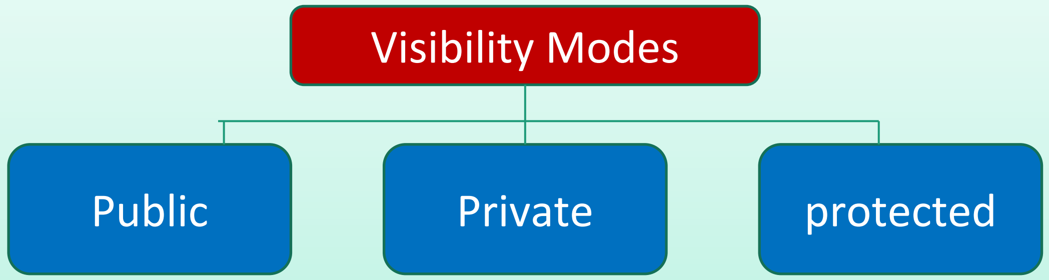
Advantages of Inheritance

Reusability:

- ✔ inheritance helps the code to be reused in many situations.
- ✔ Using the concept of inheritance,the programmer can create as many derived classes from the base class as needed while adding specific features to each derived class as needed.
- ✔ Saves Time and Effort:

The above concept of reusability achieved by inheritance saves the programmer time and effort.The main code written can be reused in various situations as needed.

Visibility modes can be classified into three categories



Modes of inheritance

1:Public Mode:If we derive a sub class from a public base class.Then the public member of the base class will become public in the derived class and the protected members of the base class will become protected in derived class.

2.Protected Mode:If we derive a sub class from a protected base class.Then both public member and protected members of the base class will become protected in derived class.

3.Private Mode:If We derive a sub class from a private base class.Then both public member and protected members of the base class will become private in derived class

Note: It is important to understand that if the access specifier is private,public members of the base become private members of the derived class. If access specifier is not present,it is private by default.

Visibility Of Inherited Members

BASE CLASS VISIBILITY	DERIVED CLASS VISIBILITY		
	PUBLIC DERIVATION	PRIVATE DERIVATION	PROTECTED DERIVATION
PRIVATE	NOT INHERITED	NOT INHERITED	NOT INHERITED
PROTECTED	PROTECTED	PRIVATE	PROTECTED
PUBLIC	PUBLIC	PRIVATE	PROTECTED

Types of inheritance

The process of inheritance is broadly classified into following categories

1:Single inheritance

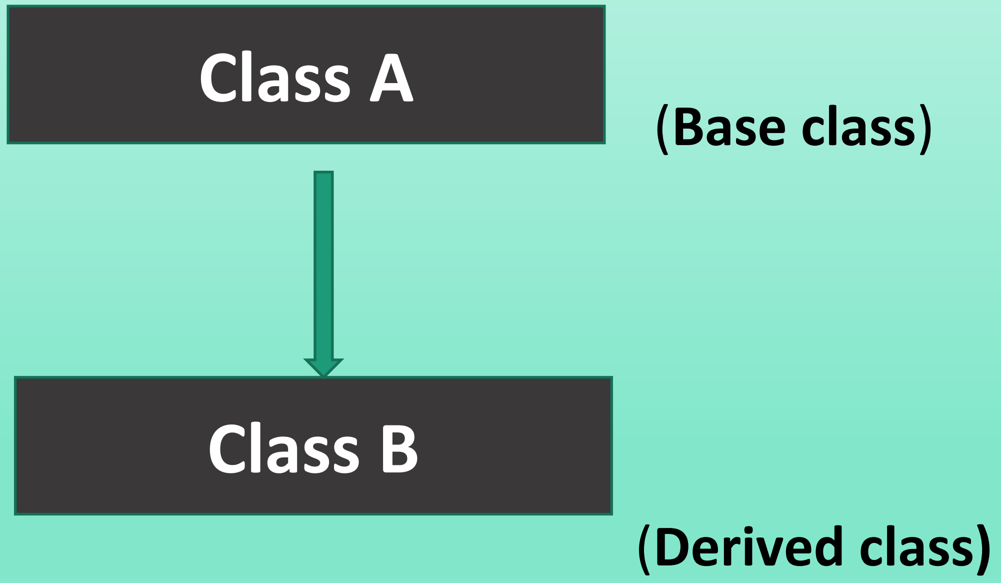
2:Multilevel inheritance

3:Multiple inheritance

4:Hierarchical inheritance

5: Hybrid inheritance

Single inheritance:The process in which a derived class inherits traits from only one base class,is called single inheritance.In single inheritance,there is only one base class and one derived class.The derived class inherits the behavior and attributes of the base class.The derived class can add its own properties,ie data members(variables) and functions.It can extend or use properties of the base class with out any modification to the base class.



.

Syntax:

```
class base_class
{
};
class derived_class:visibility-mode  base_class
{
};
```

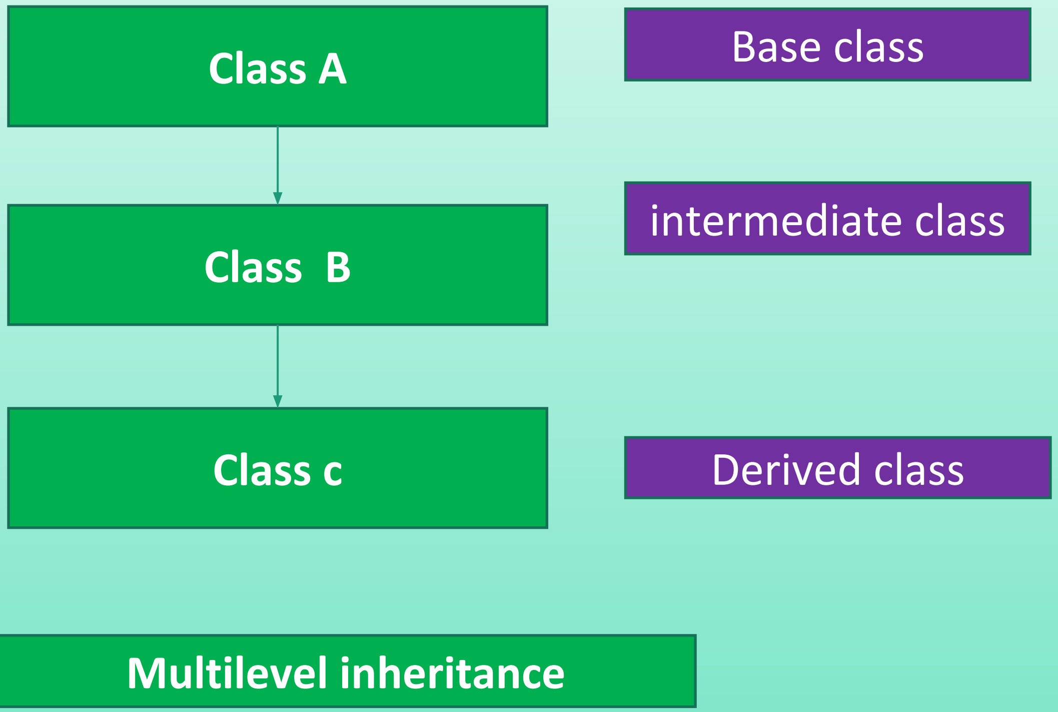
Program to illustrate the concept of single inheritance

```
#include<iostream>
using namespace std;
class stu                                        //base class
{
private:
int  id;
char name[10];
public  :
void getstu()
{
cout<<“enter student id and name”;
cin>>id>>name;
}
void putstu()
{
cout<<“id=“<<id<<endl;
cout<<“name=“<<name<<endl;
}
};
```

```
class  phy:public stu
{



float h,w;
public:
     void getphy()
{
cout<<“enter height and weight”;
cin>>h>>w;
}
void  putphy()
{
cout<<“height=“<<h<<endl;
cout<<“weight=“<<w;
}
};
void  main()
{
phy p;
p.getstu();
p.getphy();
p.putstu();
p.putphy();
}
```

Multilevel inheritance

The process in which a derived class inherits traits from another derived class,is called multilevel inheritance.



syntax:

```
class A
{
…….
};
class B:public A
{
……….
};
class C:public B
{
…….
};
```

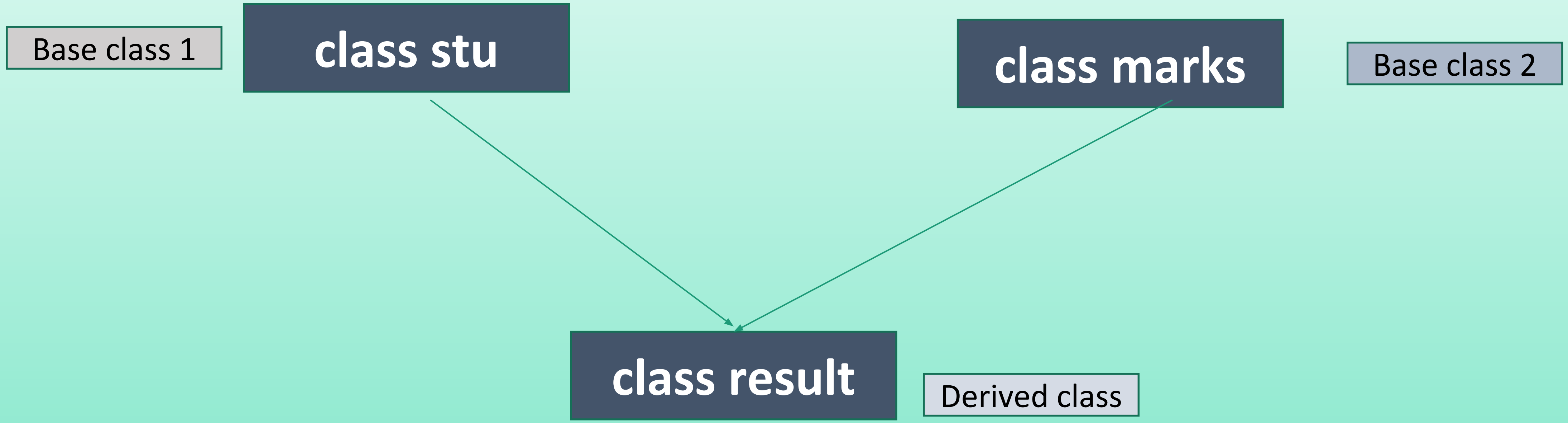
Here,class B is derived from class A.Due to this inheritance,class B adopts the features of base class A.Additionally,class B also contains its own features,Further,a new class,class c is derived from class B,Due to this ,class c adopts the features of class B,as wel as features of class A.

Program to illustrate the concept of multilevel inheritance

```
#include<iostream>
using namespace std;
class ONE
{
protected:
int n1;
};
class TWO:public ONE
{
protected:
int n2;
};
class THREE:public TWO
{
public:
void input()
{
```

```
cout<<“enter n1 n2”;
cin>>n1>>n2;
}
void sum()
{
int sum=0;
sum=n1+n2;
cout<<“sum is”<<sum;
}
};
int main()
{
THREE t1;
t1.input();
t1.sum();
}
```

Multiple inheritance:The process in which derived class inherits traits from serval base classes, is called multiple inheritance. In multiple inheritance,there is only one derived class and several base classes.we declare the base classes and derived class as given.



Syntax:

```
class base_class1{
};
class base_class2{
};
```

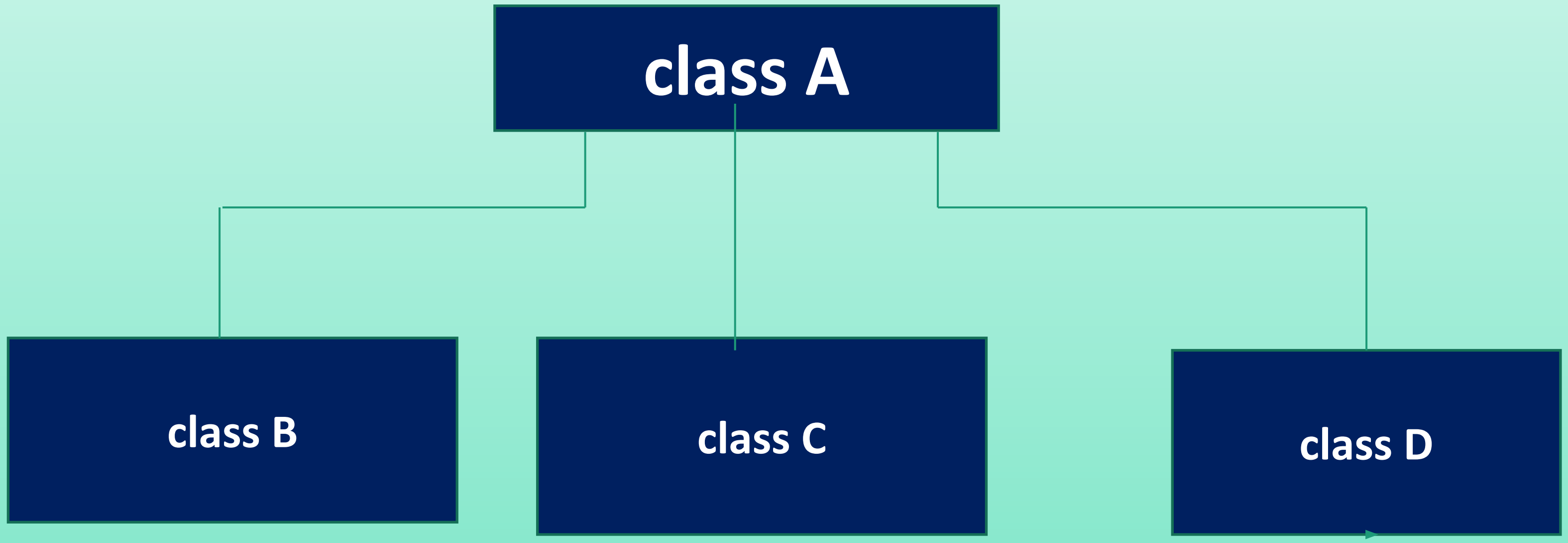
Program to illustrate the concept of multiple inheritance

```
class  stu
{
int  id;
char name[20];
public:
void getstu()
{
cout<<“enter stu id,name”;
cin>>id>>name;
}
void  putstu()
{
cout<<“id=“<<id<<endl;
cout<<“name=“<<name<<endl;
}
};
```

```
class marks{
protected:
int m1,m2,m3;

public:
void getmarks()
{
cout<<“enter marks for 3 subjs”;
cin>>m1>>m2>>m3;
}
void putmarks()
{
cout<<“m1=“<<m1<<endl;
cout<<“m2=“<<m2<<endl;
cout<<“m3=“<<m3<<endl;
}
};
class result:public stu,public marks
{
int tot;
float avg;
public:
void show()
{
tot=m1+m2+m3;
avg=tot/3.0;
cout<<“tot=“<<tot<<endl;
cout<<“avg=“<<avg;
}
};

int main()
{
result r;

r.getstu();
r.getmarks();
r.putstu();
r.putmarks();
r.show();
}
```

Hierarchical Inheritance

In this type of inheritance,more than one sub class is inherited from a single base class.i,e more than one derived class is created from a single base class.



```
class derived_class:visibility-mode base_class1,visibility-mode
base-class2
{
};
```

syntax:

```
class base-class-name
{
data members;
member functions;
};
class derived-class-name1:visibility mode base-class-name
{
data members;
member functions;
};
class derived-class-name2:visibility mode base-class-name
{
data members;
member functions;
}
```

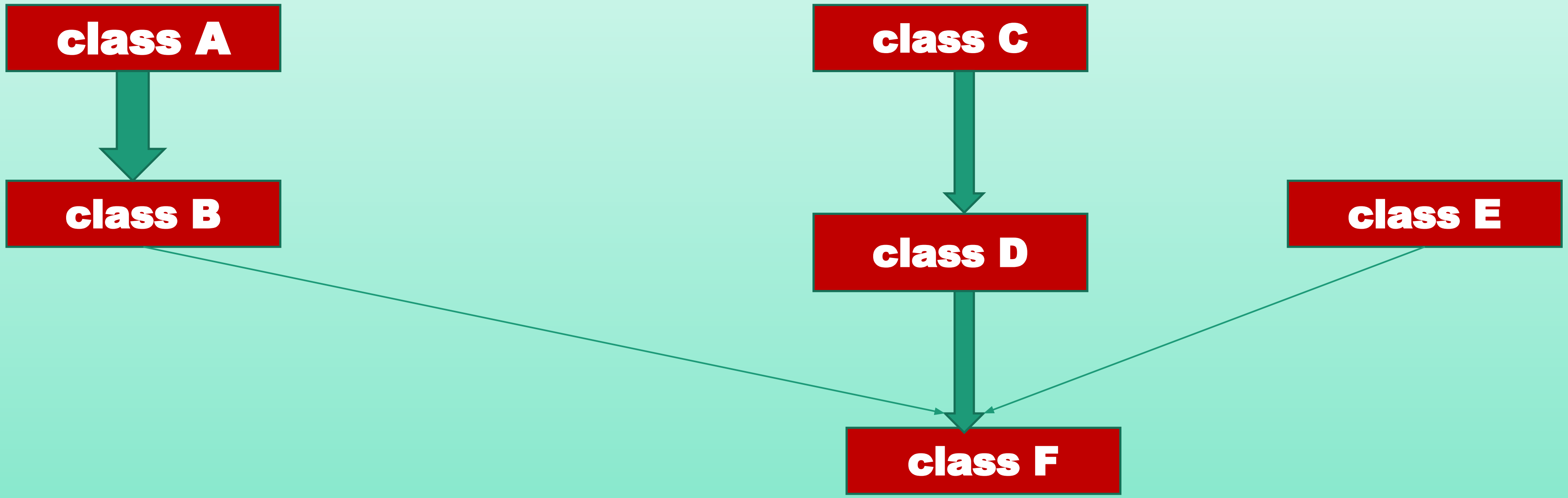
Note: visibility mode can be either private, public or protected

//Program to illustrate the concept oh hierarchical inheritance

```
#include<iostream>
using namespace std;
class A
{
public:
int x,y;
void getdata()
{
cout<<“\n enter the value for xand
y:\n”;
cin>>x>>y;
}
};
class B:public A
{
public:
void product()
{
cout<<“\n product=“<<x*y;
}
};
```

```
class C:public A
{
public:
void sum()
{
cout<<“\n sum=“<<x+y;
}
};
int main()
{
B obj1;
C obj2;
obj1.getdata();
obj1.product();
obj2.getdata();
obj2.sum();
}
```

Hybrid inheritance:The proper combination of one or more type of inheritance happening together is known as hybrid inheritance.The following diagram explains the concept in better way.



In the above diagram,the derivation of class B from class A is single inheritance.The derivation of class F from class D which is derived from class C is the multilevel inheritance,now the derivation of class Ffrom class B,D and E is

Program to illustrate the concept of hybrid inheritance

```
#include<iostream>
using name space std;
class stu
{
int id;
char name[20];
public:
void getstu()
{
cout<<“enter stu id and name:”;
cin>>id>>name;
}
};
class marks:public stu
{
protected:
int m,p,c;
public:
void getmarks()
{
cout<<“enter 3 subjects
marks:”;
cin>>m>>p>>c;
}
};
class sports
{
protected:
int spmarks;
public:
void getsports()
{
cout<<“enter sports
marks”:
cin>>spmarks;
}
};
class result:public
marks,public sports
{
int tot
float avg;
public:
void show()
{
tot=m+p+c;
avg=tot/3.0;
cout<<“tot”<<tot<<end;
cout<<“avg=“<<avg<<endl;
cout<<“avg+sports
marks”<<avg+spmarks;
}
};
int main()
{
result r;
r.getstu();
r.getmarks();
r.getsports();
r.show();
}
```

THANK YOU