# Inheritance in C++

Dhruba J. Kalita

June 2022

# About Inheritance

- Inheritance is the process of creating new classes, called Derived Classes, from existing classes, called Base Classes.
- The derived class inherits all the capabilities of the base class but can new features and refinements of its own.
- One most important advantage of inheritance is that it permits code re-usability. It helps in the distribution of class libraries.

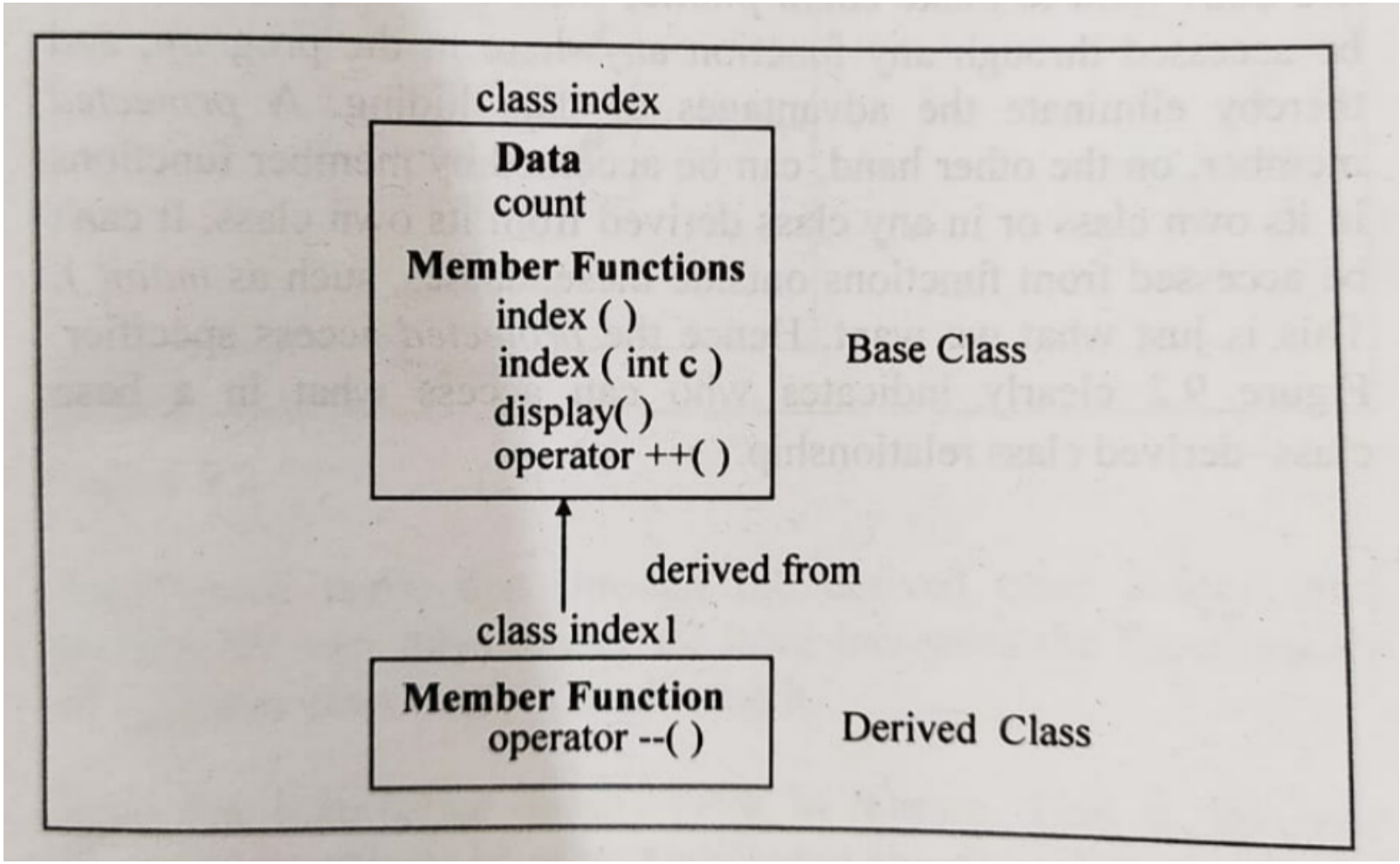
# An Example

```
#include<iostream>
using namespace std;
class index //base class
{
    protected:
        int count;
    public:
        index()
        {
            count=0;
        }
        index(int c)
        {
            count=c;
        }
        void display()
        {
            cout<<endl<<"count="<<count;
        }
```

```
        void operator ++()
        {
            count++;
        }
};
class index1:public index //derived class
{
    public:
        void operator--()
        {
            count--;
        }
};
int main()
{
    index1 i;
    ++i;
    ++i;
    i.display();
    --i;
    i.display();
    return 0;
}
```

```
    --i;
    i.display();
    return 0;
}
```

# Relationship between the Base class and the Derived Class



# Why Protected?

- The members of a derived class can access members of the base class if the base class members are public or protected
- Normally we don't want to make count public, since that would allow it to be accessed through any function anywhere in the program, and thereby eliminate the advantages of data hiding.
- Inheritance does not work in reverse.

# Implementation of stack

```
#include<iostream>
using namespace std;
const int MAX=25;
class stack   // Parent class
{
    protected:
        int s[MAX];
        int top;
    public:
        stack()
        {
            top=-1;
        }
        void push(int num)
        {
            top++;
            s[top]=num;
        }
        int pop()
```

```
        {
            int num;
            num = s[top];
            top--;
            return (num);
        }
};
class stack1:public stack
{
    public:
        void push(int num)
        {
            if(top == MAX-1)
                cout<<endl<<"Stack is full";
            else
                stack::push(num); //push() of the base class
        }
        int pop()
        {
            int n;
            if(top==-1)
            {
                cout<<endl<<"Stack is empty";
```

```
                    return 0;
                }
                else
                {
                    n=stack::pop(); //pop() of the base class
                    return (n);
                }
            }
};
int main()
{
    int n;
    stack1 stk;
    stk.push(10);
    stk.push(20);
    stk.push(30);
    n=stk.pop();
    cout<<endl<<n;
    n=stk.pop();
    cout<<endl<<n<<endl;
    return 0;
}
```

# More points

- A derived class can specify that a base class is public, or private by using the following notation-
  `class c:public b`
  `class a:private b`
- The public access specifier means that the protected members of the base class are protected members of the derived class and the public members of the base class are the public members of the derived class
- The private access specifier means that the protected and public members of the base class are private members of the derived class
- The default access specifier is private
- When we define an object of a derived class, the compiler executes the constructor function of the base class followed by the constructor function of the derived class. class.

# More points

- When a base class and a derived class have public member functions with the same name and parameter list, the function in the derived class gets a priority when the function is called as a member of the derived class object.
- A program can declare objects of both the base and derived classes. The two objects are independent of one other.

# Implementation of stack

```
#include<iostream>
using namespace std;
class one //base class
{
    private:
        int a;
    protected:
        int b;
    public:
        int c;
};
class two: public one //publicly deived class
{
    public:
        void function1()
        {
            int z;
            z=a; //error not accessible;
            z=b; // works
```

# Implementation of stack

```
            z=c; // works
        }
};
class three:private one // privately derived class
{
    public:
        void function2()
        {
            int y;
            y=a; // not eccessible
            y=b; // works
            y=c; // works
        }
};
int main()
{
    int x;
    two second; // Object of class two
    x=second.a; // not accessible
    x=second.b; // not accessible
```

# Implementation of stack

```
1       x=second.c; // works
2
3       three third; // Object of class three
4       x=third.a; // Not accessible
5       x=third.b; // Not accessible
6       x=third.c; // Not accessible
7   }
```


Thank You !