FAST NATIONAL UNIVERSTIY OF COMPUTER AND EMERGING SCIENCES, PESHAWAR

DEPARTMENT OF COMPUTER SCIENCE

CL217 – OBJECT ORIENTED PROGRAMMING LAB

File Handling and Public Inheritence

LAB MANUAL # 08

Instructor: Fariba Laiq

SEMESTER SPRING 2023

File Handling in C++

File handling is used to store data permanently in a computer. Using file handling we can store our data in secondary memory (Hard disk).

- ios stands for input output stream.
- This class is the base class for other classes in this class hierarchy.
- This class contains the necessary facilities that are used by all the other derived classes for input and output operations.

ifstream	ios::in
ofstream	ios::out
fstream	ios::in | ios::out

ofstream: Handles output file streams (for writing data to files)

ifstream: Handles input file streams (for reading data from files)

fstream: Handles both input and output file streams (for reading from and writing to files)

Example Code:

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
    string line;
    ofstream fout;
    fout.open("sample.txt");
    while(true) {
        cout<<"enter a line: "<<endl;
        getline(cin, line);
        if(line!="-1") {
            fout<<line<<endl;
        } else {
            break;
        }
    }
    fout.close();
    ifstream fin;
    fin.open("sample.txt");
    while(getline(fin, line)) {
        cout<<line<<endl;
    }
    fin.close();
}
```

Output:

```
enter a line:
hi
enter a line:
hello
enter a line:
-1
hi
hello
```

Note: To read word by word instead of line by line you can use **fin >> word** instead of getline().

Example Code:

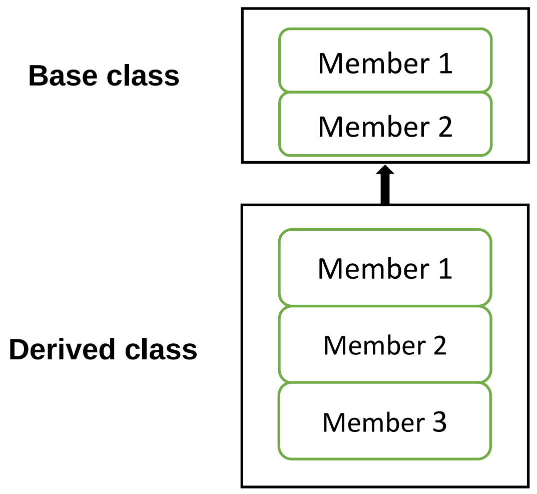
```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
        string line;
        fstream f;
        f.open("sample.txt", ios::out);
        while(true) {
                cout<<"enter a line: "<<endl;
                getline(cin, line);
                if(line!="-1") {
                        f<<line<<endl;
                } else {
                        break;
                }
        }
        f.close();
        f.open("sample.txt", ios::in);
        while(getline(f, line)) {
                cout<<line<<endl;
        }
        f.close();
}
```

Output:

```
enter a line:
hi
enter a line:
hello
enter a line:
-1
hi
hello
```

Inheritance in C++

- Inheritance is the second most important feature of Object-Oriented Programming.
- In inheritance the code of existing class is used for making new class.
- This saves time for writing and debugging the entire code for a new class.
- To inherit means to receive. In inheritance a new class is written such that it can access or use the members of an existing class. The new class that can access the members of an existing class is called ***derived class*** or ***child class***. The existing class is called the ***base class*** or ***parent class***.
- The derived class can use the data members and member functions of the base class. It can have its own data members and member functions. Thus, a derived can even be larger than a base class.
- The figure shows the relationship between a derived class and the base class.
- The arrow is drawn from derived class to the base class.
- The direction of arrow indicates that the derived class can access members of the base class but the base class cannot access members of its derived class.
- The figure shows that the derived class has only one member of its own i.e. member 3.



Defining Derived Classes

- ❖ The syntax for defining a derived class is slightly different from the syntax of the base class definition.
- ❖ The declaration of a derived class also includes the name of the base class from which it derived.
- ❖ The general syntax for defining a derived class is:

```
class  sub_class_name  :  specifier base_class_name
{
  members to derived class
}  ;
```

Suppose, the same function is defined in both the derived class and the based class. Now if we call this function using the object of the derived class, the function of the derived class is executed.

This is known as function overriding in C++. The function in derived class overrides (changes the definition) the function in base class.

};

```
#include<iostream>
using namespace std;
class Person {
        string name;
        int age;
    public:
        Person(string name, int age) {
            this->name=name;
            this->age=age;
        }
        string getName() {
            return name;
        }
        int getAge() {
            return age;
        }
        void displayInfo() {

        }
};

class Student: public Person {
    float gpa;
    public:
        Student(string name, int age, float gpa): Person(name, age) {
            this->gpa=gpa;
        }

        void displayInfo() {
            cout<<"name: "<<getName()<<"   "<<"Age: "<<getAge()<<"  "<<"GPA: "<<gpa<<endl;
        }
};
class Employee: public Person {
        int salary;
    public:
        Employee(string name, int age, int salary): Person(name, age) {
            this->salary=salary;
        }

        void displayInfo() {
            cout<<"name: "<<getName()<<"   "<<"Age: "<<getAge()<<"  "<<"Salary: "<<salary<<endl;
        }
};
```

```
int main() {
        Student s("fariba", 24, 3.3);
        Employee e("ali", 27, 50000);
        s.displayInfo();
        e.displayInfo();
}
```

Output:

```
name: fariba   Age: 24  GPA: 3.3

name: ali   Age: 27  Salary: 50000
```

The program defines a base class called Person, which has two private data members, name and age, and three public member functions, Person(), getName(), and getAge(). The Person() function is a constructor that initializes the name and age data members with the values passed to it. The getName() and getAge() functions are accessor functions that return the values of the name and age data members, respectively.

The program also defines two derived classes: Student and Employee. Both of these classes inherit from the Person class using the public access specifier. The Student class adds a new data member, gpa, and a new member function, displayInfo(), which overrides the base class member function with the same name. The Employee class adds a new data member, salary, and a new member function, displayInfo(), which also overrides the base class member function with the same name.

In the main() function, the program creates an instance of the Student class and an instance of the Employee class, passing in values for their respective data members. The program then calls the displayInfo() member function on each object, which prints out the name, age, and either GPA or salary, depending on which object is being displayed.

```
#include<iostream>
using namespace std;
class Person {
        string name;
        int age;
    public:
        Person(string name, int age) {
            this->name=name;
            this->age=age;
        }
        string getName() {
            return name;
        }
        int getAge() {
            return age;
        }
        void displayInfo() {
            cout<<"name: "<<name<<"   "<<"Age: "<<age<<"  ";

        }
};

class Student: public Person {
        float gpa;
    public:
        Student(string name, int age, float gpa): Person(name, age) {
            this->gpa=gpa;
        }

        void displayInfo() {

            Person::displayInfo();
            cout<<"GPA: "<<gpa<<endl;
        }
};
```

```
class Employee: public Person {
        int salary;
        public:
                Employee(string name, int age, int salary): Person(name, age) {
                        this->salary=salary;
                }

                void displayInfo() {
                        Person::displayInfo();
                        cout<<"Salary: "<<salary<<endl;
                }
};
```

```
int main() {
        Student s("fariba", 24, 3.3);
        Employee e("ali", 27, 50000);
        s.displayInfo();
        e.displayInfo();
}
```

The difference in the implementation of the displayInfo() member functions between the previous and current code is that the current code calls the base class implementation of the function and then adds more functionality in the derived class, while the previous code directly accessed the base class member functions to display the name and age of the person.