

**employee.h**

```
// Definition of class Employee      - employee.h
#ifndef employee_h
#define employee_h
#include <string>
using namespace std;

class Employee      // employee class
{
public :
    /**
     *      constructor
     */
    Employee(const string& name=string(),unsigned int id=0);
    /**
     *      copy constructor
     */
    Employee(const Employee& emp);
    /**
     *      destructor
     */
    virtual ~Employee();
    /**
     *      update info - prompt user to update info
     */
    virtual void Update();
    /**
     *      display info
     */
    virtual void Print() const = 0 ;
protected :
    /**
     *      employee name
     */
    string stfname;
    /**
     *      employee id.
     */
    unsigned int stfid;
};

#endif
```

**employee.cpp**

```
#include "employee.h"
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

const char Yes = 'y';
const char No = 'n';

// constructor
Employee::Employee(const string& name,unsigned int id)
:stfname(name),stfid(id) {}

// copy constructor
Employee::Employee(const Employee& emp) : stfname(emp.stfname),
stfid(emp.stfid) {}

// destructor
Employee::~~Employee() {}

//      cout << "-Employee" << endl;
}

// update info
void Employee::Update() {

    char ans = 'Y';
    cout << " update name ? (y/n) "; cin >> ans;
    if (tolower(ans) == Yes) {
        cout << "\tname? ";
        cin >> stfname;
    }
    cout << " update id. ? (y/n) "; cin >> ans;
    if (tolower(ans) == Yes) {
        cout << "\tid? ";
        cin >> stfid;
    }
}

// print info
void Employee::Print() const {
    cout << stfname << " " << setw(5) << setfill('0') << stfid ;
}
```

**worker.h**

```
// Definition of class Worker          - worker.h
#ifndef worker_h
#define worker_h
#include "employee.h"
#include <string>
using namespace std;

// worker class, a derived class of Employee
class Worker : public Employee
{
public:
    // type of term of employment
    enum Status { permanent, temporary, TotStatus };
    /**
     *      constructor
     */
    Worker(const string& name=string(), unsigned int id=0, Status status=permanent);
    /**
     *      copy constructor
     */
    Worker(const Worker& worker);
    /**
     *      destructor
     */
    ~Worker();
    /**
     *      update info
     */
    void Update();
    /**
     *      display infor
     */
    void Print() const;
private:
    Status stfterm;          // term of employment
};

#endif
```

**worker.cpp**

```
// Function definition of class Worker          - worker.cpp
#include "employee.h"
#include "worker.h"
#include <iostream>
using namespace std;

const int SLen= 10;          // length of status name
const char StatusName[Worker::TotStatus][SLen] = { "permanent", "temporary" };

// constructor
Worker::Worker(const string& name, unsigned int id, Status status) : Employee(name, id),
stfterm(status) {}

// copy constructor
Worker::Worker(const Worker& worker) : Employee(*(Employee*)&worker),
stfterm(worker.stfterm) {}

// destructor
Worker::~~Worker() {}

// update info
void Worker::Update() {
    Employee::Update();
    unsigned int ans = unsigned int(stfterm);
    cout << " status: permanent(0) or temporary(1)? ";
    cin >> ans;
    if (ans < Worker::TotStatus)
        stfterm = Status(ans);
}

// print data
void Worker::Print() const {
    Employee::Print();
    cout << " " << StatusName[stfterm];
}

}
```

## testpoly.h

```
// Test polymorphism      - testpoly.cpp
```

```
#include "employee.h"
```

```
#include "worker.h"
```

```
#include <iostream>
```

```
using namespace std;
```

```
// selection
```

```
const char WorkerStf = 'w';
```

```
//const char ManagerStf = 'm';
```

```
//const char ScientistStf = 's';
```

```
const char Exit = 'x';
```

```
const char Update = 'u';
```

```
const char Print = 'p';
```

```
const char Complete = 'c';
```

```
// see effect of polymorphism
```

```
void seevirtual(Employee*& emptr) {
```

```
    char op;
```

```
    do {
```

```
        cout << "\n(U)pdate, (P)rint, (C)omplete ? ";
```

```
        cin >> op;
```

```
        switch (tolower(op)) {
```

```
            case Update :
```

```
                emptr->Update();
```

```
                break;
```

```
            case Print :
```

```
                cout << "\t";
```

```
                emptr->Print();
```

```
                cout << endl;
```

```
                break;
```

```
            case Complete :
```

```
                break;
```

```
            default :
```

```
                break;
```

```
        }
```

```
    } while (op != Complete);
```

```
}
```

```
// try polymorphism
```

```
void tryvirtual() {
```

```
    Employee* emptr = NULL;
```

```
    char choice;
```

```
    bool exit = false;
```

```
    string name = "unknown";
```

```
    unsigned int id=0;
```

```
    do {
```

```
        cout << "\nType of Staff : (W)orker, (M)anager, (S)cientist, e(X)it? ";
```

```
        cin >> choice;
```

```
        if (tolower(choice) != Exit) {
```

```
            cout << "\n  name? "; cin >> name;
```

```
            cout << "  id? "; cin >> id;
```

```
        }
```

```
        switch (tolower(choice)) {
```

```
            case WorkerStf :
```

```
                emptr = new Worker(name,id);
```

```
                break;
```

```
        //
```

```
        //
```

```
            fill in here      for instantiating a Manager object
```

```
        //
```

```
            :
```

```
        //
```

```
            fill in here      for instantiating a Scientist object
```

```
        //
```

```
            :
```

```
        //
```

```
            case Exit :
```

```
                exit = true;
```

```
                break;
```

```
            default :
```

```
                break;
```

```
        }
```

```
        if (emptr) {
```

```
            seevirtual(emptr);
```

```
            delete emptr;
```

```
            emptr = NULL;
```

```
        }
```

```
    } while (!exit);
```

```
}
```

```
int main()
```

```
{
```

```
    tryvirtual();
```

```
    return 0;
```

```
}
```