VNU – HCMC UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

PROJECT REPORT 2:

EMPLOYEES SALARY

Instructor guide: Tran Duy Quang

Students:

- **Nguyen Nhat Truong – 20120229**

Class: OOP 20_3

PROJECT 2:EMPLOYEES SALARY

I/ THÔNG TIN CHUNG:

1. THÔNG TIN SINH VIÊN:

Họ và tên	MSSV
Nguyễn Nhật Trường	20120229

II/ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH:

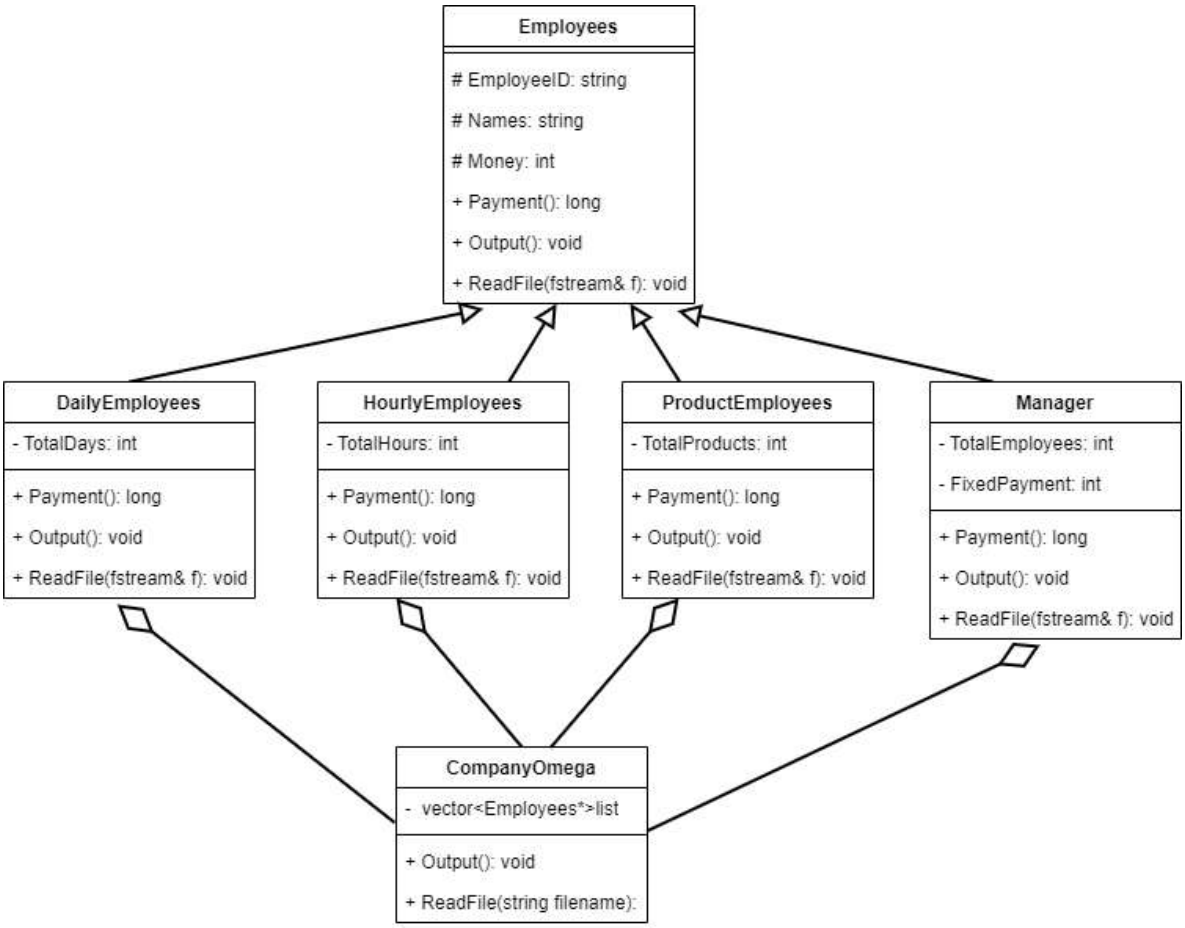
Hoàn thành 100% yêu cầu của project.

III/ PROJECT:

1. Mục tiêu chính:

- Dựa vào các dữ liệu trong file November2021.txt để tạo ra các lớp nhằm đọc và lưu dữ liệu.
- Sử dụng fstream để đọc file dữ liệu
- Sử dụng kế thừa và đa hình để áp dụng vào các lớp trong bài
- Tìm hiểu về kiểu factory design pattern để áp dụng vào bài

2. Class diagram:



3. Các bước làm Project:

Khi đọc đề bài và đọc file cpp, tôi nghĩ mình nên tạo ra các lớp để lưu thông tin các loại nhân viên (do có nhiều loại), vì các loại nhân viên đều có vài thuộc tính giống nhau nên tôi tạo ra thêm lớp Employees để làm lớp cha dùng kế thừa, sau đó tôi tạo ra lớp công ty OmegaCompany để đọc file tổng, phân loại các nhân viên và quản lý tất cả nhân viên, dùng vecto và đa hình trong lớp công ty để lưu lại tất cả thông tin của nhân viên

a/ Header files:

- **Employees.h:**

+ Khai báo các hàm setter, getter, constructor, and destructor. Khai báo các thuộc tính chung của tất cả các loại nhân viên như EmployeeID, Names, Money

```
//Tạo lớp Employees để làm lớp cha cho các nhân viên
class Employees
{
// Khai báo các thuộc tính chung có ở tất cả các lớp
protected:
    string EmployeeID; // Loại nhân viên
    string Names; //Tên nhân viên
    int Money; //Số tiền lương để nhân cho thuộc tính riêng của mỗi loại nhân viên
public:
    //Constructor and Destructor
    Employees();
    ~Employees();
    // Viết các hàm chức năng, dùng kế thừa để kế thừa các hàm của các loại nhân viên
    virtual long Payment() = 0; // hàm tính lương
    virtual void Output(); //hàm xuất
    virtual void ReadFile(fstream& f) = 0; // hàm đọc file
public: //getter, setter
    string Getter_EmployeeID();
    string Getter_Names();
    int Getter_Money();
};
```

+ Viết các hàm chức năng trong lớp (có sử dụng virtual để đa hình từ các lớp của những loại nhân viên)

```
virtual long Payment() = 0; // hàm tính lương
virtual void Output(); //hàm xuất
virtual void ReadFile(fstream& f) = 0; // hàm đọc file
```

- **DailyEmployees.h:**

+ Viết lớp này nhằm lưu thông tin của loại nhân viên DailyEmployees trong file, lớp này được kế thừa public từ lớp cha Employees

+ Khai báo private thuộc tính riêng của lớp này là TotalDays và các constructor, destructor, các hàm chức năng như tính lương, đọc file và xuất ra màn hình

```
class DailyEmployees:public Employees
{
private:
    int TotalDays; //thuộc tính TotalDays riêng của loại nhân viên DailyEmployees
public:
    //Constructor and Destructor
    DailyEmployees();
    ~DailyEmployees();
    long Payment();//Hàm tính lương
    void ReadFile(fstream& f); //Hàm đọc file
    void Output();// Hàm xuất ra màn hình
};
```

- **HourlyEmployees.h, ProductEmployees.h, Manager.h:** Cài đặt tương tự như DailyEmpoyees, chỉ khác ở các thuộc tính riêng của mỗi loại nhân viên (xem source trong bài)

- **CompanyOmega.h:**

+ Tạo lớp này nhằm đọc file, phân loại và tạo danh sách nhân viên để lưu lại tất cả thông tin về nhân viên.

+ Tạo vecto Employees nhằm lưu thông tin nhân viên:

```
private:
    vector<Employees*>list;//Dùng vecto để tạo list lưu thông tin
```

+ Khai báo các hàm constructor, destructor và các hàm chức năng như đọc file và xuất ra màn hình

```
public:
    //Constructor and Destructor
    CompanyOmega();
    ~CompanyOmega();
    void ReadFile(string filename);//Hàm đọc file
    void Output();// Hàm xuất ra màn hình
```

b/ In Source file:

- **Employees.cpp:**

Viết các hàm constructor, destructor, getter, setter cơ bản.

Do tất cả các nhân viên đều các tên, nên ở hàm Output ta chỉ xuất tên, còn các thông tin khác ta sẽ đa hình đến các nhân viên để xuất:

```
void Employees::Output()
{
    cout << "Name: " << this->Names << endl;
}
```

- **DailyEmployees.cpp:**

+ Cài đặt các hàm cơ bản như constructor và destructor

```
DailyEmployees::DailyEmployees()
{
    this->TotalDays = 0;
}
DailyEmployees::~DailyEmployees()
{}
```

+ Dựa vào cấu trúc trong file November2021.txt để viết hàm đọc file

```
DailyEmployee: John Walker
   DailyPayment=100$; TotalDays=28
DailyEmployee: Lyndsea Carlie
   DailyPayment=120$; TotalDays=10
```

Ta thấy rằng cứ mỗi 2 dòng sẽ là 1 nhân viên, từ đầu đến dấu 2 chấm sẽ là loại nhân viên, ta đọc loại này ở trong hàm đọc file ở class CompanyOmega để phân biệt acsc loại nhân viên với nhau.

Vậy nên trong các hàm đọc file ở các lớp nhân viên, ta sẽ tiến hành đọc sau dấu ":". Xem giải thích rõ hơn trong hình source code dưới đây:

```
void DailyEmployees::ReadFile(fstream &f)
{
    string temp = "";
    getline(f, temp);
    this->Names = temp.substr(1);// Xóa dấu cách ở phần đầu của tên
    getline(f, temp, '=');
    string Str_Money = "";
    getline(f, Str_Money, '$');//Lấy được số tiền nhưng ở string
    this->Money = stoi(Str_Money);//Chuyển từ string về kiểu số nguyên
    getline(f, temp, '=');
    f >> this->TotalDays;//Lấy các ký tự sau dấu bằng để lưu vào TotalsDay
    getline(f, temp);
}
```

+ Viết các hàm chức năng như tính lương và xuất dữ liệu dựa vào đề

```
//Hàm tính lương của DailyEmployees
long DailyEmployees::Payment()
{
    return TotalDays * Money;
}

//Hàm xuất của DailyEmployees
void DailyEmployees::Output()
{
    Employees::Output();
    cout << "Payment: " << this->Payment() << endl;
}
```

- **HourlyEmployees.cpp, ProductEmployees.cpp:**

  + Cài đặt các hàm cơ bản như constructor và destructor và hàm đọc file tương tự như trong file DailyEmployees.cpp

  + Chỉ có hàm tính lương sẽ hơi khác nhau, vì nó phải lấy số tiền nhân với thuộc tính riêng của mỗi loại:

```
//Hàm tính lương
long HourlyEmployees::Payment()
{
    return TotalHours * Money;
}
```

```
//Hàm tính lương
long ProductEmployees::Payment()
{
    return TotalProducts * Money;
}
```

- **Manager.cpp:**

- + Cài đặt các hàm cơ bản như constructor và destructor và hàm đọc file tương tự như trong file DailyEmployees.cpp

  + Cài đặt hàm đọc file sẽ khác do kết cấu các thuộc tính của Manager trong file khác với 3 loại nhân viên còn lại, bên cạnh đó còn có thêm tiền phụ cấp. Xem chi tiếp cách đọc file trong hình dưới đây:

```
//Dựa vào file các thông tin về Manager để thực hiện đọc file
void Manager::ReadFile(fstream& f)
{
    string temp = "";
    getline(f, temp);
    this->Names = temp.substr(1); //Xóa ký tự khoảng trắng sau dấu ':' để lấy được tên

    getline(f, temp, '=');
    string Str_Fixed = "";
    getline(f, Str_Fixed, '$');//Tiếp tục xuống đọc đến dấu bằng sau đó lấy ký tự đến '$'
                               //để lấy ra FixedPayment, nhưng nó đang ở kiểu string
    this->FixedPayment = stoi(Str_Fixed); //Dùng hàm stoi để chuyển sang số nguyên và lưu lại

    getline(f, temp, '=');
    string Str_Employees = "";
    getline(f, Str_Employees, ';'); //Tiếp tục đọc đến dấu bằng sau đó lấy ký tự
                                    //đến ';' để lấy ra TotalEmployees, nhưng nó đang ở kiểu string
    this->TotalEmployees = stoi(Str_Employees); //Tương tự như trên, dùng hàm stoi để chuyển sang số nguyên và lưu lại

    getline(f, temp, '=');
    string Str_Money = "";
    getline(f, Str_Money, '$');//Tiếp tục đọc đến dấu bằng sau đó lấy ký tự đến '$; để lấy raMoney, nhưng nó đang ở kiểu string
    this->Money = stoi(Str_Money);//Tương tự như trên, dùng hàm stoi để chuyển sang số nguyên và lưu lại
}
```

+ Cài đặt các hàm chức năng như tính lương và xuất theo đề:

```
//Hàm tính lương
long Manager::Payment()
{
    return TotalEmployees * Money + FixedPayment;
}

//Hàm xuất
void Manager::Output()
{
    Employees::Output();
    cout << "Payment: " << this->Payment() << endl;
}
```

- **CompanyOmega.cpp:**

+ Viết các hàm constructor, destructor

+ Vì đây là hàm đọc file chính, nên ý tưởng ban đầu sẽ là: Cho đọc các ký tự đầu của mỗi 2 dòng cho đến dấu ':' đến khi hết file để lưu lại các loại nhân viên, sau đó so sánh với các chuỗi khai báo sẵn để xác định đó là loại nhân viên nào, sau đó tạo mới lớp của loại nhân viên đó rồi trỏ đến hàm đọc file của lớp đó để lưu thông tin. Để chi tiết hơn, xem giải thích từng dòng ở source code bên dưới:

```
//Hàm đọc file chính nó sẽ ở trong công ty, sẽ đọc và phân loại từng nhân viên
void CompanyOmega::ReadFile(string filename)
{
    fstream f;
    f.open(filename, ios::in);
    //Sử dụng các chuỗi tương ứng với loại nhân viên để so sánh trong lúc đọc file, nhằm biết nhân viên đó thuộc loại nào
    string DEID = "DailyEmployee";
    string HEID = "HourlyEmployee";
    string PEID = "ProductEmployee";
    string MAID = "Manager";
    Employees* E; //Khởi tạo con trỏ lớp Employees
    if (f.is_open())
    {
        while (!f.eof())
        {
            string temp = "";
            getline(f, temp, ':');//Đọc từ đầu đòng đến dấu ':' sau đó lưu chuỗi này vào temp
            if (temp == DEID) //So sánh xem nó là loại nhân viên nào
            {
                //Factory Design Pattern
                E = new DailyEmployees; //Nếu chuỗi đó là loại DailyEmployees thì tạo mới, sau đó đọc file
                E->ReadFile(f);
            }
```

```
            //Tương tự với các loại còn lại
            else if (temp == HEID)
            {
                //Factory Design Pattern
                E = new HourlyEmployees;
                E->ReadFile(f);
            }
            else if (temp == PEID)
            {
                //Factory Design Pattern
                E = new ProductEmployees;
                E->ReadFile(f);
            }
            else if(temp == MAID)
            {
                //Factory Design Pattern
                E = new Manager;
                E->ReadFile(f);
            }

            //Lưu nó vào list
            list.push_back(E);
        }
    }
}
```

+ Viết hàm xuất file từ list danh sách đã tạo vecto và lưu thông tin:

```
//Hàm xuất tất cả thông tin
void CompanyOmega::Output()
{
    for (int i = 0; i < list.size(); i++)
    {
        list[i]->Output();
    }
}
```

IV. CÁC ĐIỀU HỌC ĐƯỢC KHI HOÀN THÀNH ĐỒ ÁN:

- Biết được cách nhận biết các thông tin trong 1 file và sau đó đọc file để lưu lại các thông tin trong một vector.
- Hiểu hơn về tính kế thừa, đa xạ, đa hình giữa các lớp với nhau trong OOP
- Tìm hiểu về các kiểu thiết kế: Factory, Prototype và Singleton
- Học được cách viết file readme.md trên github

Cảm ơn thầy rất nhiều vì đã cho em học được những điều hay ho và tuyệt với trên qua đồ án này.

THE END