HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF ELECTRONICS AND TELECOMMUNICATIONS

PROJECT REPORT

Project 9: Management of programming company employee

Name of students: Nguyen Tien Dat

Nguyen Duc Phuc Hoang

Phung Mac Quan

Ma Khoa Hoc

Course name : C/C++ Progamming Techniques

Course ID : ET2031E

Name of instructor: Ms Do Thi Ngoc Diep

Hanoi, 11th July 2022

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Table of work resume

No	Name	Student Id	Email	Work assigment	Completion	Note
01	Nguyen Tien Dat	20213569	dat.nt213569@sis.hust.edu.vn	- Coding for attributes, EnterList,PrintStaff, RemoveStaff, EditStaff method of class Menu and main function - Research for system design,class design and algorithms - Fix bugs	100%	
02	Nguyen Duc Phuc Hoang	20210385	hoang.ndp210385@sis.hust.edu.vn	- Report for the project - Coding for class Programmer , CalculateSalary, StatisticSalary method of class Menu - Support for algorithms researching	100%	
03	Ma Khoa Hoc	20210388	hoc.mk210388@sis.hust.edu.vn	- Coding for class Staff and main function - Support for system design researching - Write comment for code - Fix bugs	100%	
04	Phung Mac Quan	20213584	quan.pm213584@sis.hust.edu.vn	- Coding for class Tester, getStaffListSize ,SearchStaff method - Support for class design and design class UML diagrams - Testing for bug - Make interface look better	100%	

Table of Contents

I. Introduction

1.1. Problem's introduction

Management of programming company employees:A software company has two types of employees, programmers and testers. The employee's salary is calculated as follows: for programmers, the monthly salary is calculated as basic salary + overtime hours * 200,000 VND. Tester's salary is calculated by basic salary + number of detected errors * 20,000 VND.

1.2 System requirements

Building an employee management system including the following functions:

-Add, edit, delete an employee

-Search for employees by name, ID, phone number, salary, start time at the company.

-Calculate the average salary of the employees in the company at a ime entered.

-Statistics of employees whose monthly salary is lower than the average salary of employees in the company.

II. Analysis processing

2.1. System analysis

2.1.1. Specify objects in systems

- The same point:
  - Object Programmer and Object Tester have the same access specifiers as Staff Object such as: Fullname, StaffID, PhoneNumber, YearStart, BasicSalary.
- The different points:

- The only difference between Object Programmer and Object Tester is Object Programmer has a private attribute: OvertimeHour and Object Tester has a private attribute: Bugfixed.

2.1.2. Inheritance properties of system

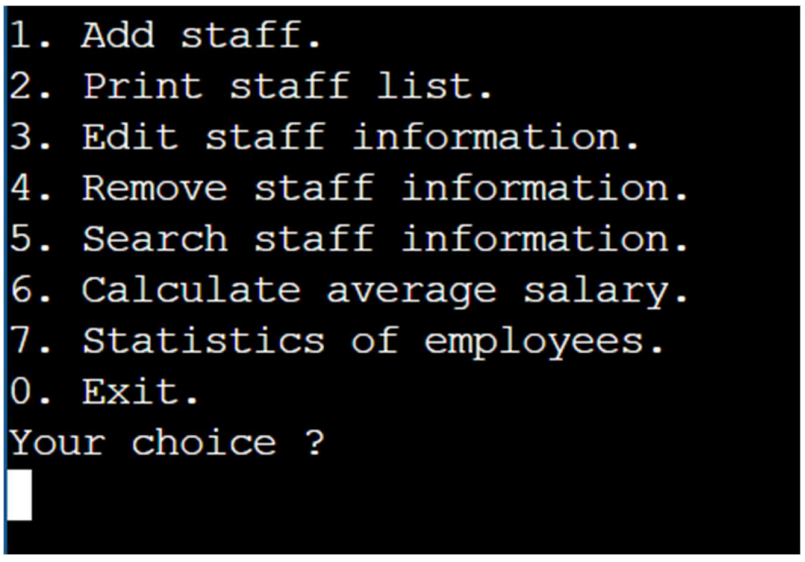
- Programmer Class and Tester Class (child) inherits the attributes and methods from the Staff class (parent) including 5 attributes: FullName, StaffID, PhoneNumber, YearStart, BasicSalary and 4 methods: getInformation, printfInformation, FinalSalary, editInformation.
- The access scope of members remains the same, except the private is not accessible.

2.1.3. Polymorphism properties of system

- Methods in Staff Class are declared by prepending the 'virtual' keyword so Programmer Class and Tester Class (Child) inherited from (Parent) can be overloaded methods in Staff Class such as: getInformation, printfInformation, FinalSalary, editInformation.

2.2. Activity descriptions

The main menu of the program:



You can enter any number from 1 to 7 so as to use the one of seven functions

a. Add, edit, delete a staff:

- Add a staff: Enter 1 to add a staff.

```
1. Programmer.
2. Tester.
0. Exit.
Your choice?
```

- Enter the type of the staff
- Enter Full name, Staff ID, Phone Number, Year Start Working, Basic Salary and Overtime Hours for the Programmer.
- Enter Full name, Staff ID, Phone Number, Year Start Working, Basic Salary and Number of Bug Fixed for the Tester.

- Edit a staff information: Enter 3 to edit information.

```
What position of employee do you want to edit from the lis
```

- Choose the position of employee that you want to edit from the list (In the main menu enter 2 to print staff list).

Tester

```
What do you want to edit?
0. Exit.
1. Full name.
2. Staff ID.
3. Phone number.
4. Year start working.
5. Basic salary.
6. Bug fixed.
```

Programmer

```
What do you want to edit?
0. Exit.
1. Full name.
2. Staff ID.
3. Phone number.
4. Year start working.
5. Basic salary.
6. Overtime hours.
```

Enter any number from 1 to 6 in order to edit one of six entered data

- Delete a staff: Enter 4 to delete a staff.

```
What position of employee do you want to remove from the lis
```

Choose the position of employee that you want to delete from the list

b. Search for staffs by informations: Enter 5 to search for staffs.

```
1. Search by name.
2. Search by staff id.
3. Search by phone number.
4. Search by starting year.
0. Exit.
Your choice?
```

- Enter any number from 1 to 4 in order to search the staff by Name/ ID / Phone Number / Starting Year

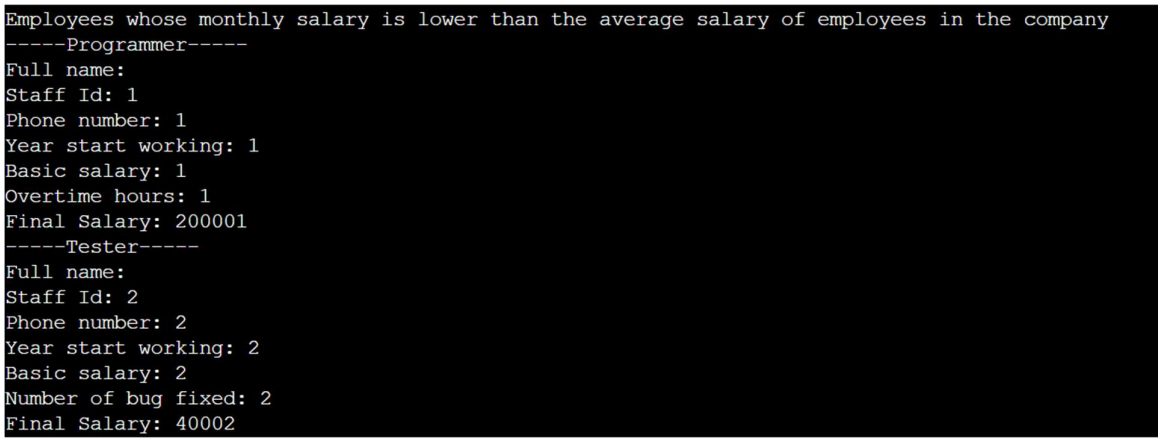
c. Caculate average salary:

- Enter 6 to calculate avg. salary of all staffs.

```
Average salary : 213335
```

d. Statistics of staffs whose monthly salary is lower than the avg. salary of staffs in the company:

Enter 7 to print statistics.



2.3. Class descriptions

The program includes:

- A class named “Staff” containing all the information of the staff.
- 2 classes inherited from class “Staff” to work with 2 purposes :

+ A class named “Programmer” is created to work with programmers’ information .

+ A class named “Tester” is created to work with testers’ information

- A class named “Menu” is created to choose the object that is a programmer or tester to work and work with the properties of that object
- The methods for the functions of the program are included in the class “programmer” , “tester” and “menu”

III. Design

3.1. System design

First of all, we declare a class named “Staff” containing all the information of the staff. Then, creating some methods and attributes for class “Staff”.

```
class Staff
{
private:
    string FullName;
    string StaffID;
    string PhoneNumber;
    string YearStart;
    int BasicSalary;
public:
    Staff(){
        FullName = "";
        StaffID = "";
        PhoneNumber = "";
        YearStart = "";
        BasicSalary= 0;
    }
    virtual void getInformation();
    virtual void printInformation();
    virtual int FinalSalary(){};
    virtual void editInformation();
    friend class Programmer;
    friend class Tester;
    friend class Menu;
};
```

The next steps, we implement 3 methods in class “Staff”

1. getInformation

```
void Staff :: getInformation(){
    fflush(stdin);
    cout << "Full name: ";
    getline(cin , FullName);
    cout << "Staff ID: ";
    cin >> StaffID;
    cout << "Phone number: ";
    cin >> PhoneNumber;
    cout << "Year start working: ";
    cin >> YearStart;
    cout << "Basic salary: " ;
    cin >> BasicSalary;
}
```

```
void Staff :: printInformation(){
    cout << "Full name: " << FullName << endl;
    cout << "Staff Id: " << StaffID<< endl;
    cout << "Phone number: " << PhoneNumber << endl;
    cout << "Year start working: "<<YearStart << endl;
    cout << "Basic salary: " << BasicSalary << endl;
}
```

3) editInformation

```
void Staff :: editInformation(){
        system("cls");
        cout << "What do you want to edit? " << endl;
        cout << "0. Exit. " << endl;
        cout << "1. Full name. " <<endl;
        cout << "2. Staff ID. " << endl;
        cout << "3. Phone number. " << endl;
        cout << "4. Year start working. " << endl;
        cout << "5. Basic salary. " << endl;
}
```

Then, because of the requirement to two types: programmers and testers, we have to create 2 classes inheriting from class “Staff”

I) Programmer

```
class Programmer : public Staff
{
private:
    int OvertimeHour;
public :
    Programmer(){
        OvertimeHour=0;
    }
    void getInformation();
    void printInformation();
    int FinalSalary();
    void editInformation();
};
```

In class “ Programmer” , we declare new attributes is: OvertimeHour and we declare and implement 3 methods:

1. getInformation

```
void Programmer :: getInformation(){
    cout << "Programmer" << endl;
    Staff :: getInformation();
    cout << "Overtime hours: ";
    cin >> OvertimeHour;
}
```

2. printfInformation

```
void Programmer :: printInformation(){
    cout << "-----Programmer-----" << endl;
    Staff :: printInformation();
    cout << "Overtime hours: " << OvertimeHour << endl;
    cout << "Final Salary: " << FinalSalary() << endl;
}
```

3. FinalSalary

```
int Programmer :: FinalSalary(){
    int FinalSalaryPro;
    FinalSalaryPro = BasicSalary + OvertimeHour * 200000;
    return FinalSalaryPro;
}
```

4. editInformation

```
void Programmer :: editInformation(){
    int editchoice;
    do
    {  fflush(stdin);
        Staff :: editInformation();
        cout << "6. Overtime hours. " << endl;
        cin >> editchoice;
        switch (editchoice)
        {
        case 1:
            fflush(stdin);
            cout << "New name : ";
            getline(cin,FullName);
            break;
        case 2:
            cout << "New Staff ID : ";
            cin >> StaffID;
            break;
        case 3:
            cout << "New phone number : " ;
            cin >> PhoneNumber;
            break;
```

```
        case 4:
            cout << "New starting year : ";
            cin >> YearStart;
            break;
        case 5:
            cout << "New basic salary : ";
            cin >> BasicSalary;
            break;
        case 6:
            cout << "New overtime hour : ";
            cin >> OvertimeHour;
            break;
        case 0:
            break;
        default:
        cout << "Wrong choice, again !";
        system("pause");
        }
    } while (editchoice != 0);
}
```

II) Tester

In Tester class, we declare a new attribute is : BugFixed and the rest is the same as Programmer Class.

```
class Tester : public Staff
{
private:
    int BugFixed;
public:
    Tester(){
        BugFixed =0;
    };
    void getInformation();
    void printInformation();
    int FinalSalary();
    void editInformation();
};
    void Tester :: getInformation(){ ···
    void Tester :: printInformation(){ ···
    int Tester :: FinalSalary(){ ···
    void Tester :: editInformation(){ ···
```

Continuously,we declare a class named ”Menu” .It contains some methods and attributes

```
class Menu
{
private:
    vector<Staff*> StaffList;
public:
    void EnterList();
    void PrintList();
    void EditStaff();
    void RemoveStaff();
    void SearchStaff();
    long CalculateAverageSalary();
    void StatisticSalary();
    int getStaffListSize();
};
```

1.EnterList()

To store the address into the pointers in the vector, we use an intermediate Staff pointer to point to the memory cell containing the information of the object just entered. Then, the intermediate pointer will be returned to the vector and can be used to store the information of the next object. The data memory location of the first object stored in the pointer has just been returned to vector

```
//The method selects the object to be entered, enters the information, and stores the information in memory
void Menu :: EnterList(){
    Staff* stf;
    int choice;
    do
    {
        system("cls");
        cout << "1. Programmer. " << endl;
        cout << "2. Tester. " << endl;
        cout << "0. Exit. " << endl;
        cout << "Your choice? " << endl;
        cin >> choice;
        switch (choice)
        {
        case 1:
            system("cls");
            stf = new Programmer();
            stf->getInformation();
            StaffList.push_back(stf);
            break;
        case 2:
            system("cls");
            stf = new Tester();
            stf->getInformation();
            StaffList.push_back(stf);
            break;
        case 0 :
            break;
        default:
        cout << "Wrong choice, again !";
        system("pause");
        }
    } while (choice != 0 );
}
```

2.PrintList()

We use a for () loop to call method printInformation of all object saved in memory that be pointed by Staff* vector

```
//The method that prints the information of all employees on the screen
void Menu :: PrintList(){
    for (int i=0;i< StaffList.size();i++)
    {
        StaffList[i] ->printInformation();
        cout << "========" << i+1 << "/" << StaffList.size() <<"========" << endl;
    }
}
```

3.EditStaff()

Just call method editInformation of selected objects

```
//The method to enter the position to edit and edit employee information
void Menu :: EditStaff(){
        int editpos;
        cout << "What position of employee do you want to edit from the list ? " << endl;
        cin >> editpos;
        while (editpos > getStaffListSize())
        {
            cout << " Re-enter position: " << endl;
            cin >> editpos;
        }
        StaffList[editpos -1]->editInformation();
}
```

4.RemoveStaff()

The easiest way is use method erase() for vector class that was declared in STL

```
//The method to enter the position to be removed and remove the employee's information in that position
void Menu :: RemoveStaff(){
    int removepos;
    cout << "What position of employee do you want to remove from the list ? " << endl;
    cin >> removepos;
    while ( removepos > getStaffListSize())
    {
        cout << "Re-enter position: " << endl;
        cin >> removepos;
    }

   /* Another way
    for (int  i = removepos-1; i < StaffList.size() ; i++)
    {
        StaffList[i] = StaffList[i+1];
    }
    StaffList.resize(StaffList.size()-1); */
    StaffList.erase(StaffList.begin() + removepos - 1);
}
```

5.SearchStaff()

To find a staff based on informations, we use find() method of string class.

With syntax string1.find(string2), it will return first positions that string2 appear in string1 or if string2 don't appear,it will return -1.

So, in case of return value is different from -1, we will print their informations

```
//The method that searches and prints out all the information of the employees by using a known information
void Menu :: SearchStaff(){
    int searchchoice;

    do
    {   system("cls");
        cout << "1. Search by name. " << endl;
        cout << "2. Search by staff id. " << endl;
        cout << "3. Search by phone number. " <<endl;
        cout << "4. Search by starting year. " << endl;
        cout << "0. Exit. " << endl;
        cout << "Your choice? " << endl;
        cin >> searchchoice;
        switch (searchchoice)
        {
        //search by name
        case 1:
        {
            string searchname;
            fflush(stdin);
            cout << "Enter name: " ;
            getline(cin,searchname);
            int count=0;
            system("cls");
            for (int i=0; i<StaffList.size();i++)
            {
                if ((StaffList[i]->FullName).find(searchname) != -1 )
                {
                    StaffList[i]->printInformation();
                    cout << "Position : " << ++count << endl ;
                }
            }
            cout <<"===============" << endl << "Total : " << count << endl;
            system("pause");
            break;
        }
```

Case 2,3,4 are similar to case 1

```
    } while (searchchoice != 0);

}
```

End of the method.

6. CalculateAverageSalary

For this feature, we need to calculate the total salary of all declared staffs then divide its by the number of employees

```
//The method that calculates the average salary of all employees
long Menu :: CalculateAverageSalary(){
    long AverageSalary=0;
    for (int i=0;i <StaffList.size();i++)
    {
        AverageSalary += StaffList[i]->FinalSalary();
    }
    AverageSalary /= StaffList.size();
    return AverageSalary;
}
```

7. StatisticSalary

With CalculateAverageSalary method, we can use its to check whose salary is lower than the average salary and print their informations out.

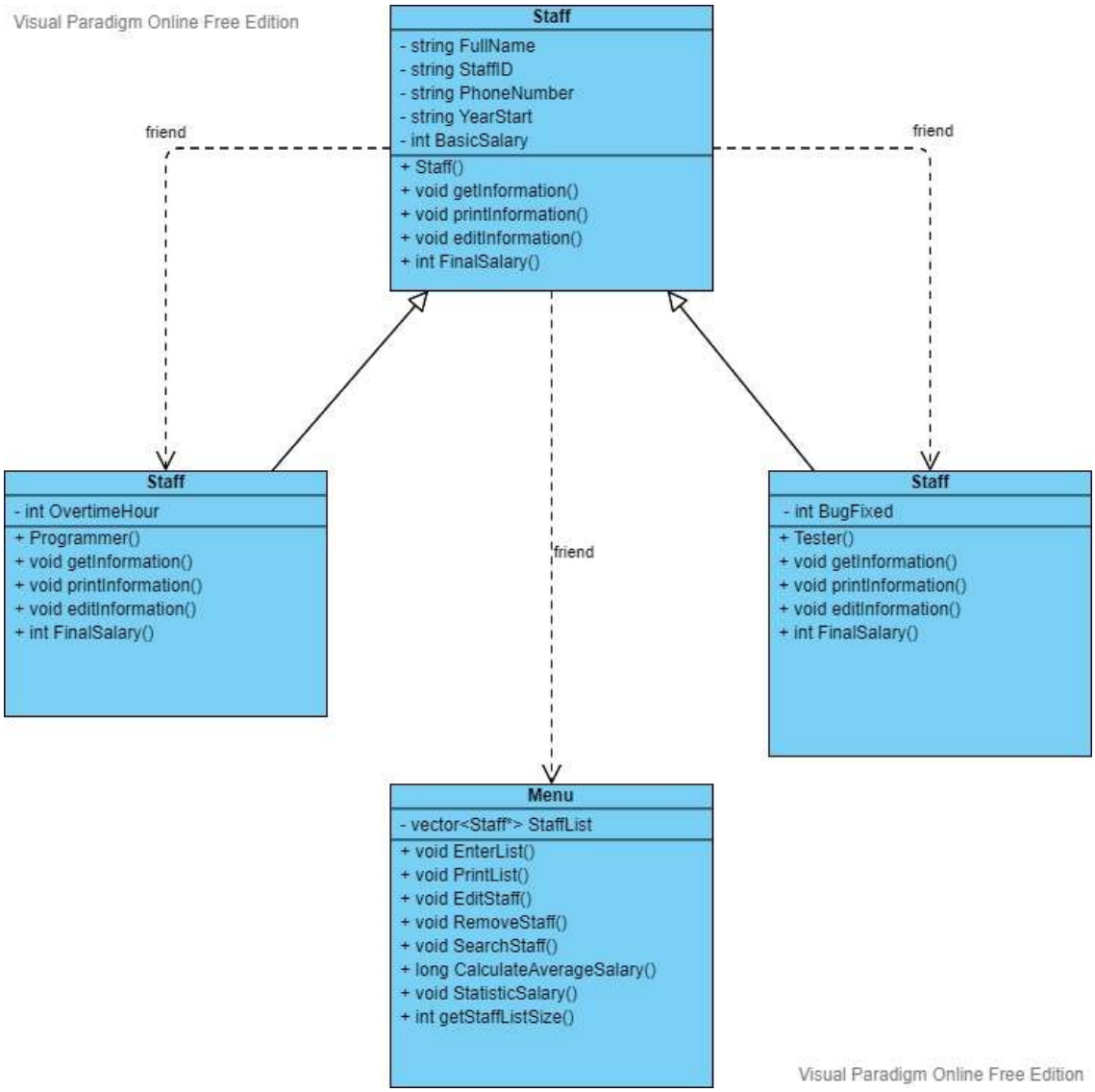
```
//The method that prints the information of the employees whose salary is less than the average salary
void Menu :: StatisticSalary(){
    cout << "Employees whose monthly salary is lower than the average salary of employees in the company" << endl;
    for (int i=0;i< StaffList.size();i++)
    {
        if (StaffList[i]->FinalSalary() <= CalculateAverageSalary() )
        {
            StaffList[i]->printInformation();
        }
    }
    system("pause");
}
```

8. getStaffListSize

A getter to access size of vector

```
//The method to get the number of employees have been saved
int Menu :: getStaffListSize(){
    return StaffList.size();
}
```

3.2. Class design



IV. Overview

4.1. Self-assessment of results.

- In working process, our team have completed all functions required. Everybody in group also finish their work on time.
- The system works stably most of the time.
- Time to implement features is fast and also has no delay.
- Some functions has not been optimized yet.

4.2. Limitations of systems.

- Some bugs because of buffer often appear during use.

  Example : When user enter an unvailid type of variable, the system will be in an infinity loop or be crash.

- The user interface is still not very nice.

V. References

- Library used : bits/stdc++.h
- https://cplusplus.com/reference/vector/vector/
- https://cplusplus.com/reference/string/string/