# ***Name: Eman Mohamed Soliman Abd-Allah Rabie***

# ***ID: 20912022200595***

HR Management System

**Introduction**

The HR system is a software application that aims to automate the management of employee records, benefits, and payroll. The main objective of it is to provide an efficient and user-friendly way for HR personnel to manage employee data, benefits, and compensation.

**The classes**

1. Benefits (abstract class): This base class is to determine different types of benefits for employees. Mainly two types of benefits -> health benefit & dental benefit

Benefit

# m\_plantype : string

# m\_amount : double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ getInfo() : string

+ getDetails() : void

+ displayBenefit() : void

+ caculateBenefit() : double

DentalBenefit

- info : string

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ getInfo() : string

+ getDetails() : void

+ displayBenefit() : void

+ caculateBenefit() : double

HealthBenefit

- info : string

- coverage : string

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ getInfo() : string

+ getDetails() : void

+ displayBenefit() : void

+ caculateBenefit() : double

1. Department class: This class stores information about a department, including the department name, ID. Each employee should belong to the department

Depatrment

- m\_department : string

- m\_id : int

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ readDepartment( i : int , s : string ) : void

+ printDepartment() : void

1. Employee class: This base class stores the basic information of an employee, including their name, ID, job title, phone, email, and a list of benefits. There are four types of employees. The Hourly, Salaried, Commission and Manager is an Employee with different attributes and a new equation for payroll function.

Employee

- \*benefitList : Benefit

- benefitCount : int

- benefitSize : int

# employee\_id : int

# employee\_name : string

# employee\_phone : string

# employee\_email : string

# jobTitle : string

# totalBene : double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ setColor(color : int ) : void

+ storeBenefitList() : void

+ getBenefitList() : void

+deleteBenbfit( s : string ) : bool

+ editBenefit( s : string ) : bool

+ getEmployeeID() : int

+ dispalyDetails() : void

+ getEmployeeDetails() : void

+ getSalary() : double

+ caculatePay() : double

CommissionEmployee

- target : double

- rate : double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ getEmployeeDetails() : void

+ dispalyDetails() : void

+ caculatePay() : double

+ getSalary() : double

SalariedEmployee

+ salary:double

+ getSalary():double

+ getEmployeeDetails():void

+ dispalyDetails():void

+ caculatePay():double

ManagerEmployee

-bouns:double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+getSalary();double

+getEmployeeDetails():void

+dispalyDetails():void

+caculatePay():double

HourlyEmployee

- hoursWorked : double

- rate : double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ addMoreHours(int hours) : void

+ getEmployeeDetails() : void

+ dispalyDetails(): void

+ getSalary() : double

+ caculatePay() : double

1. HR System: This class is the main class that stores a list of employees and manages the operations of adding, removing, and searching for departments and employees. It also has a method to display information about a specific employee or the average salary of a department.

HRSystem

- EmployeeSize:int

- EmployeeCount:int

- \*EmployeeList :Employee

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ setColor(int color) :void

+ addEmployee():void

+ employeeSearch():void

+ deleteEmployee():void

+ editEmployee():void

+ addBPackage():void

+ deleteBPackage():void

+ editBPackage():void

+ salaryCaculation():void

+ displayAll():void

+ addDepart(int):void

The class diagram

Benefit

# m\_plantype : string

# m\_amount : double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ getInfo() : string

+ getDetails() : void

+ displayBenefit() : void

+ caculateBenefit() : double

DentalBenefit

- info : string

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ getInfo() : string

+ getDetails() : void

+ displayBenefit() : void

+ caculateBenefit() : double

HealthBenefit

- info : string

- coverage : string

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ getInfo() : string

+ getDetails() : void

+ displayBenefit() : void

+ caculateBenefit() : double

Depatrment

- m\_department : string

- m\_id : int

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ readDepartment( i : int , s : string ) : void

+ printDepartment() : void

Employee

- \*benefitList : Benefit

- benefitCount : int

- benefitSize : int

# employee\_id : int

# employee\_name : string

# employee\_phone : string

# employee\_email : string

# jobTitle : string

# totalBene : double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ setColor(color : int ) : void

+ storeBenefitList() : void

+ getBenefitList() : void

+deleteBenbfit( s : string ) : bool

+ editBenefit( s : string ) : bool

+ getEmployeeID() : int

+ dispalyDetails() : void

+ getEmployeeDetails() : void

+ getSalary() : double

+ caculatePay() : double

CommissionEmployee

- target : double

- rate : double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ getEmployeeDetails() : void

+ dispalyDetails() : void

+ caculatePay() : double

+ getSalary() : double

SalariedEmployee

+ salary:double

+ getSalary():double

+ getEmployeeDetails():void

+ dispalyDetails():void

+ caculatePay():double

ManagerEmployee

-bouns:double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+getSalary();double

+getEmployeeDetails():void

+dispalyDetails():void

+caculatePay():double

HourlyEmployee

- hoursWorked : double

- rate : double

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ addMoreHours(int hours) : void

+ getEmployeeDetails() : void

+ dispalyDetails(): void

+ getSalary() : double

+ caculatePay() : double

HRSystem

- EmployeeSize:int

- EmployeeCount:int

- \*EmployeeList :Employee

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

+ setColor(int color) :void

+ addEmployee():void

+ employeeSearch():void

+ deleteEmployee():void

+ editEmployee():void

+ addBPackage():void

+ deleteBPackage():void

+ editBPackage():void

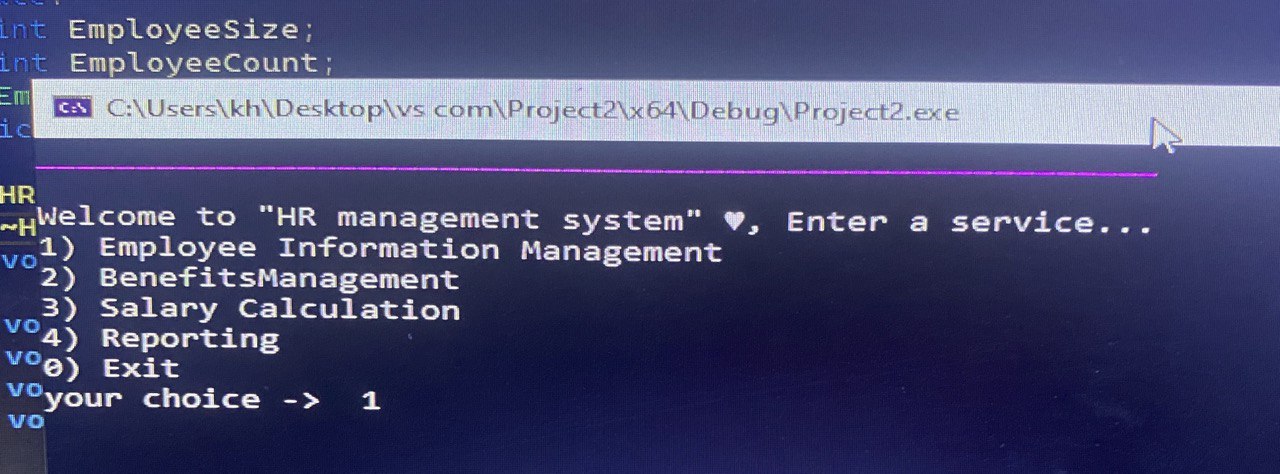
+ salaryCaculation():void

+ displayAll():void

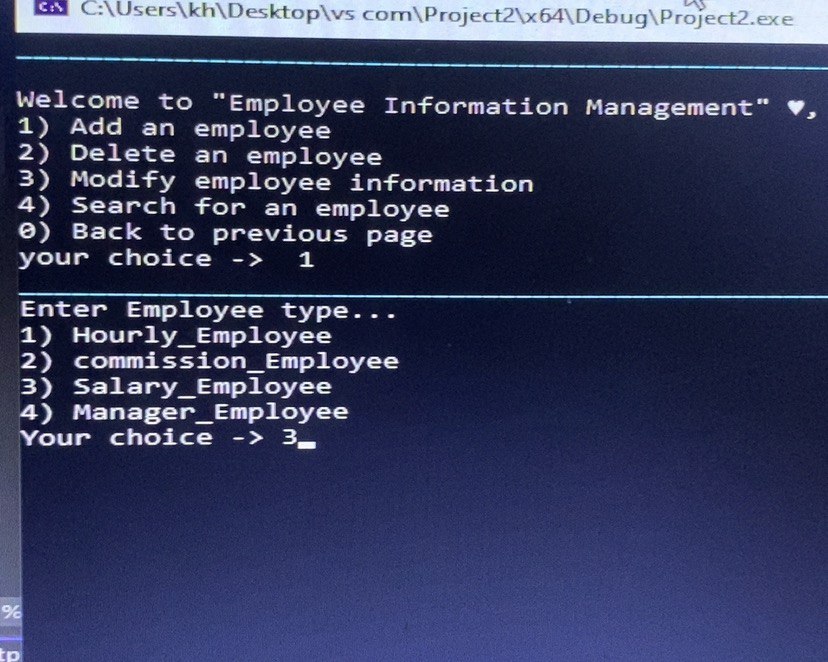
+ addDepart(int):void

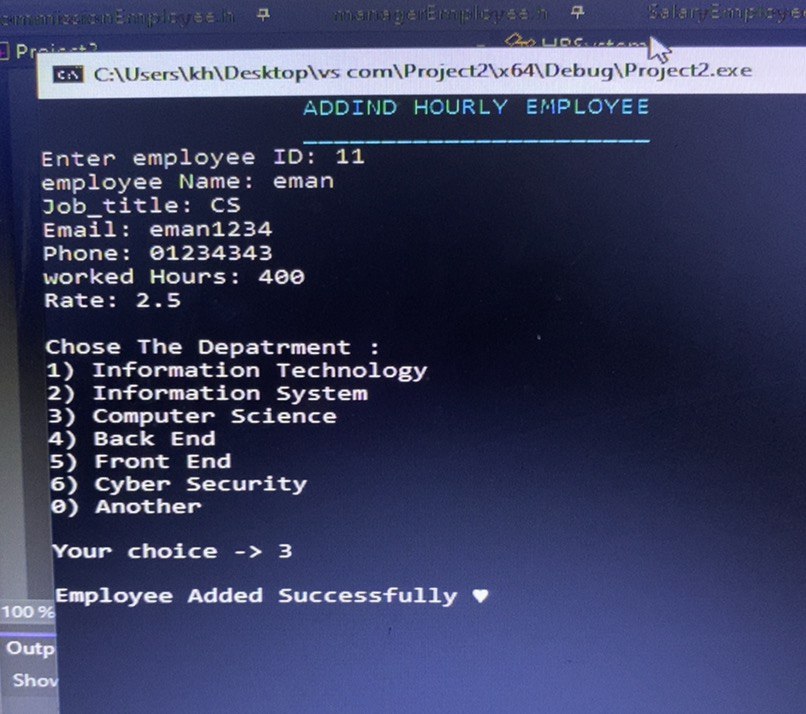
The output

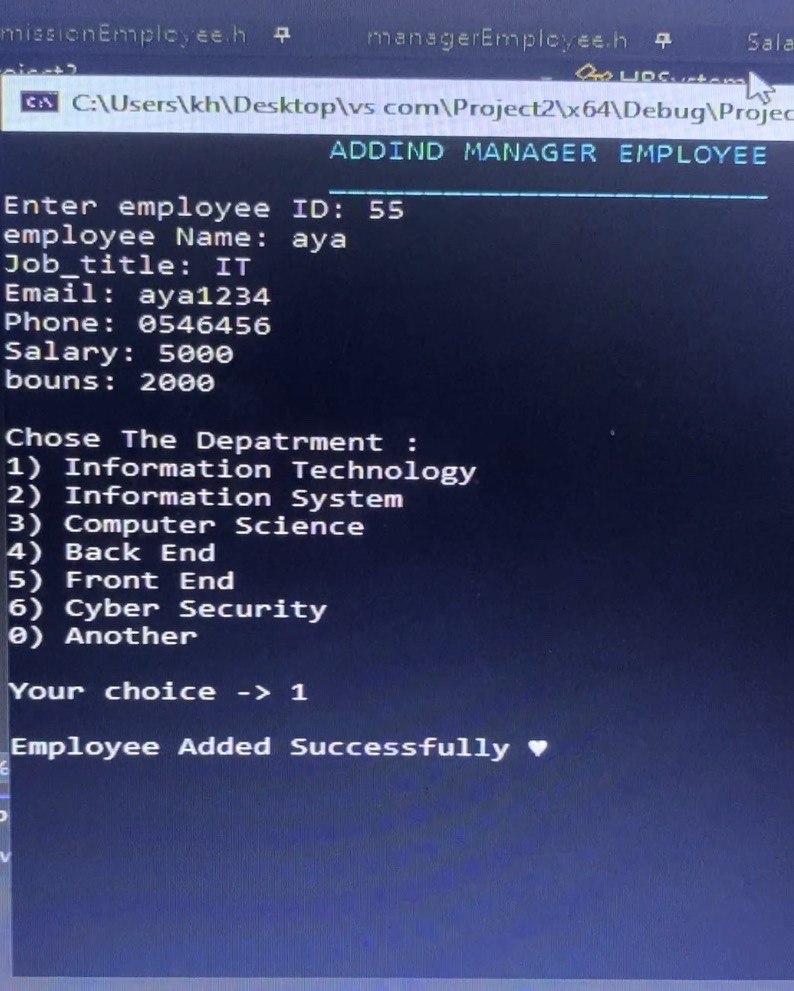
[1] Employee information management to Add, Edit, delete & search employees

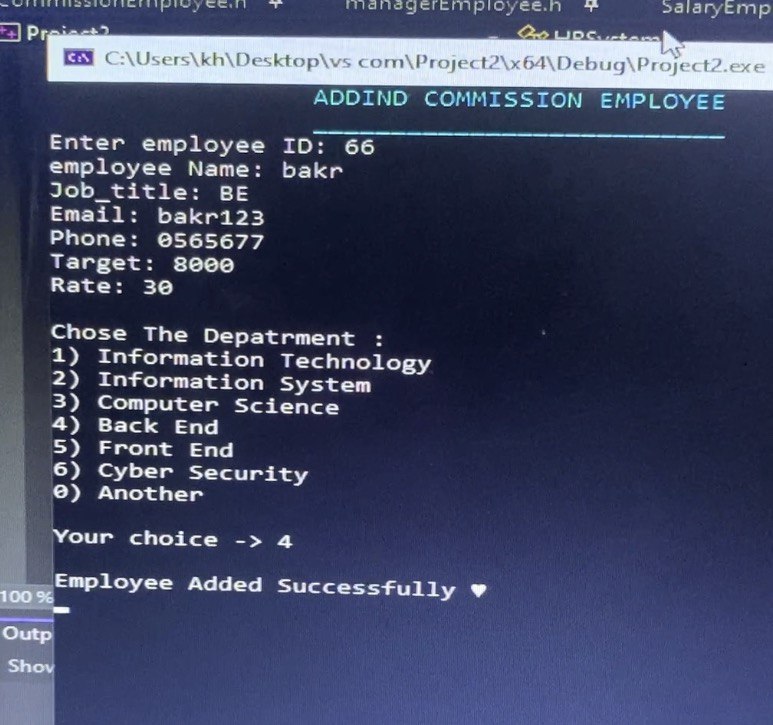


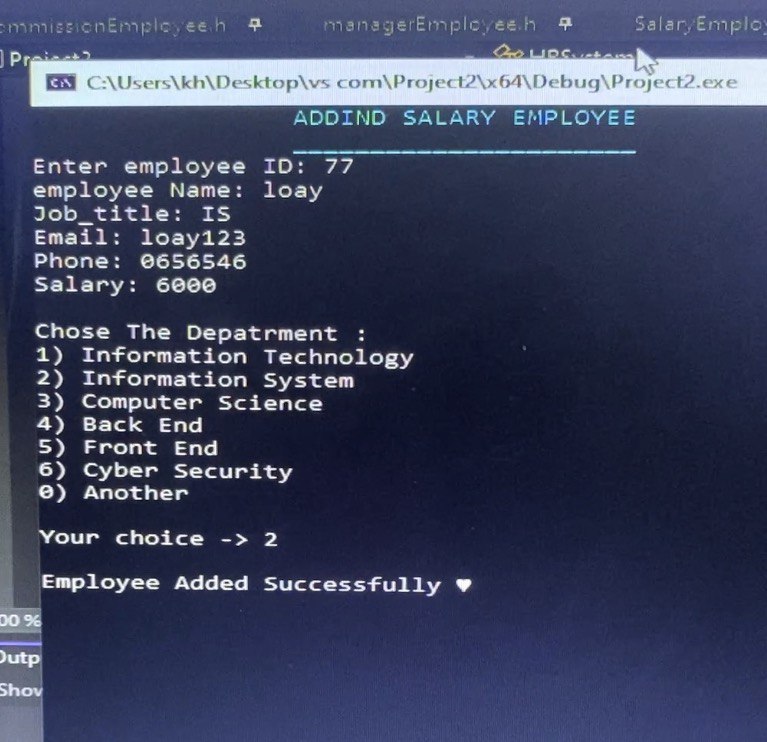
-> add (hourly, commission, salaried & manager) employees



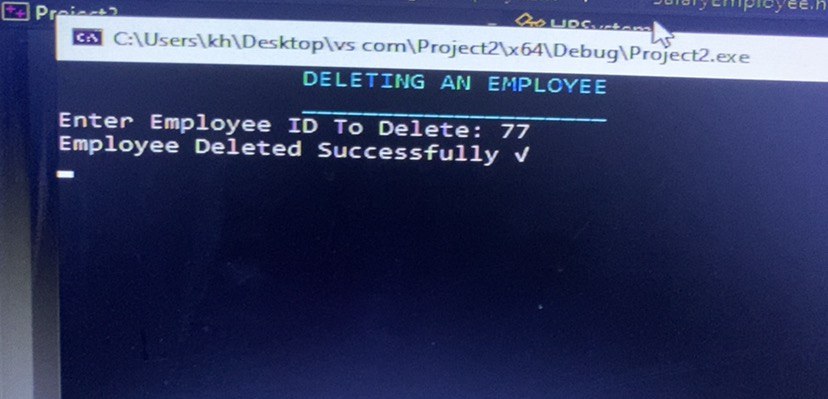




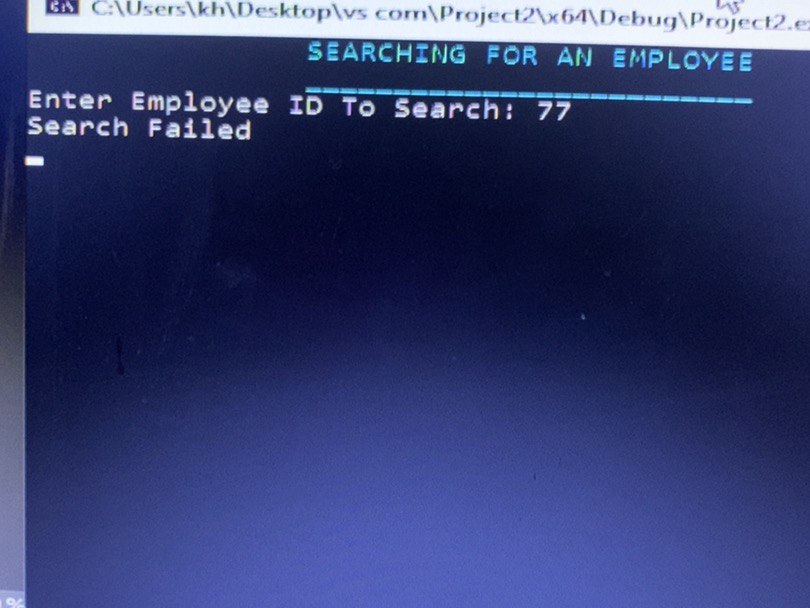




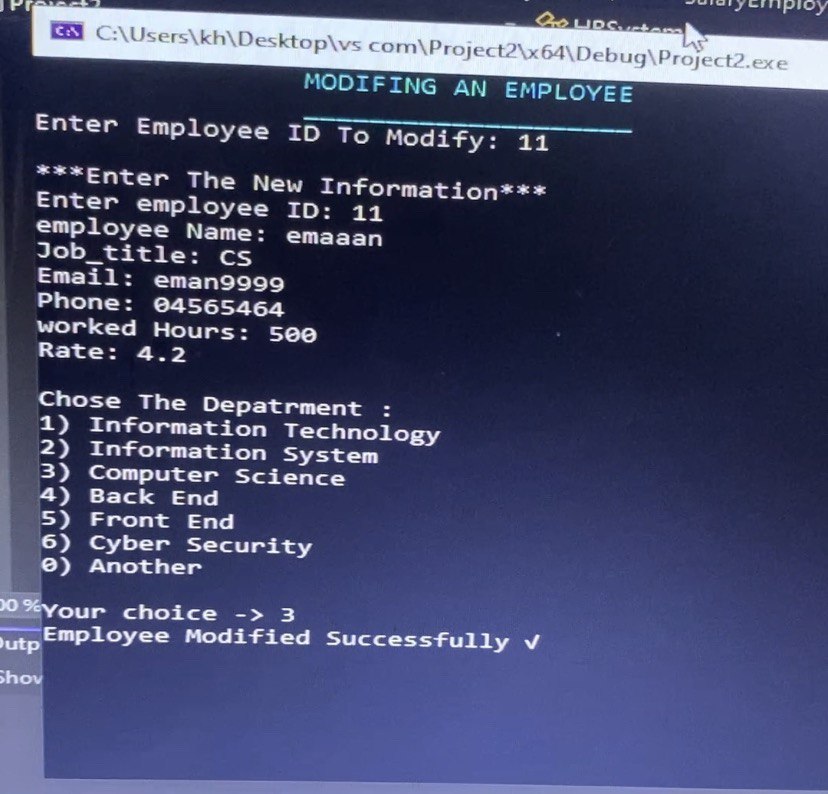
-> Delete an employee



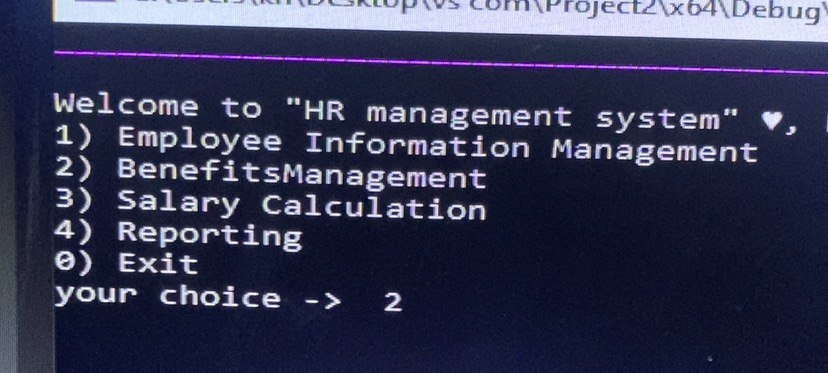
-> Search an employee



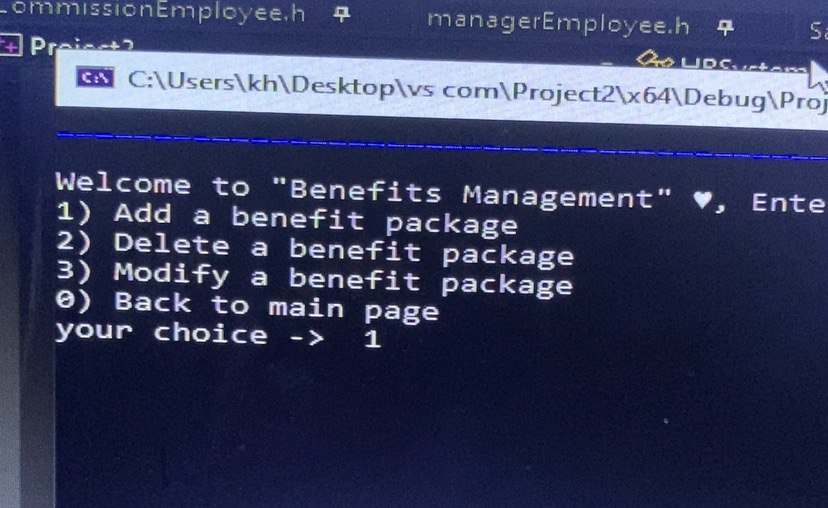
-> Edit an employee

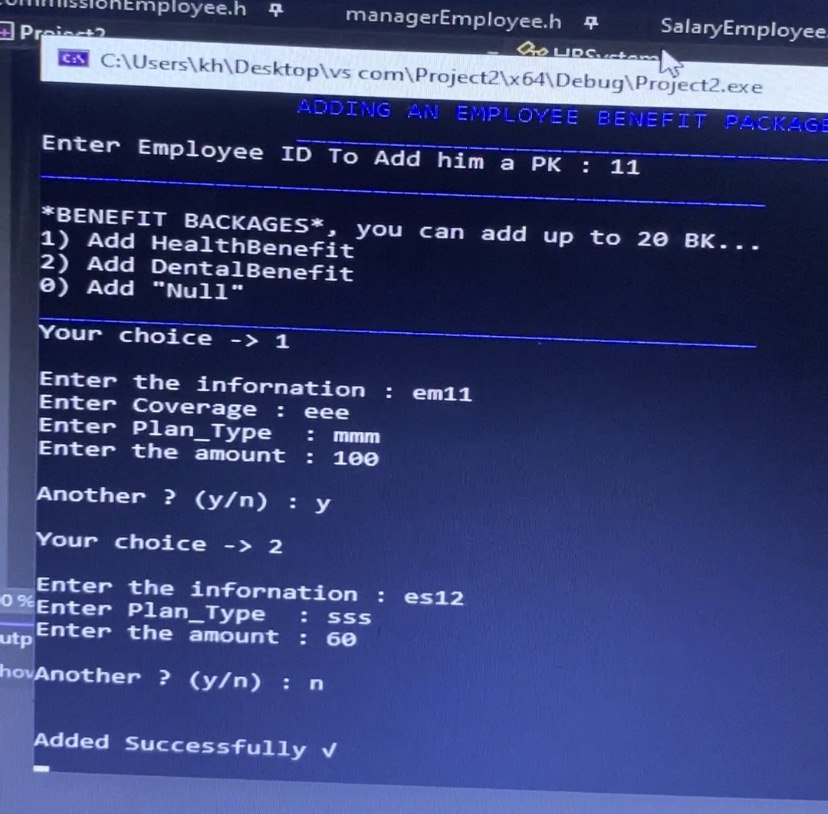


[2] Benefit Management: to add, edit & delete benefits

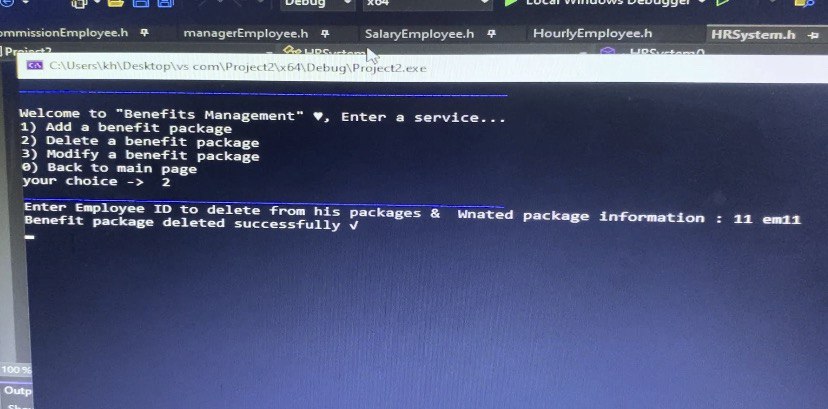


-> add

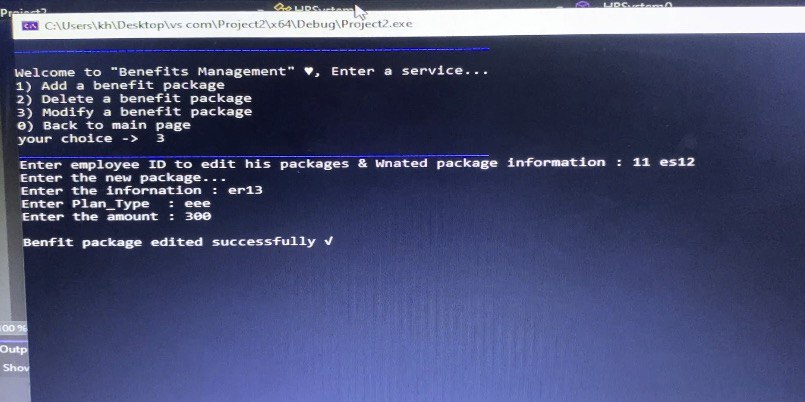




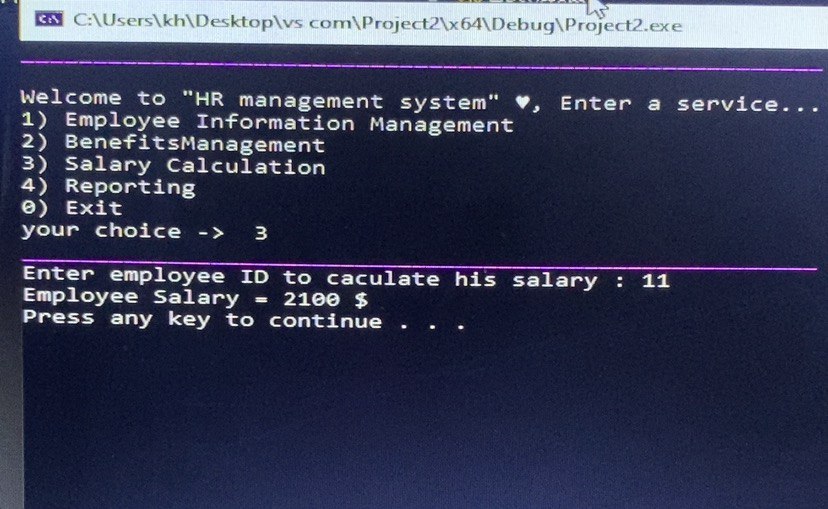
-> delete



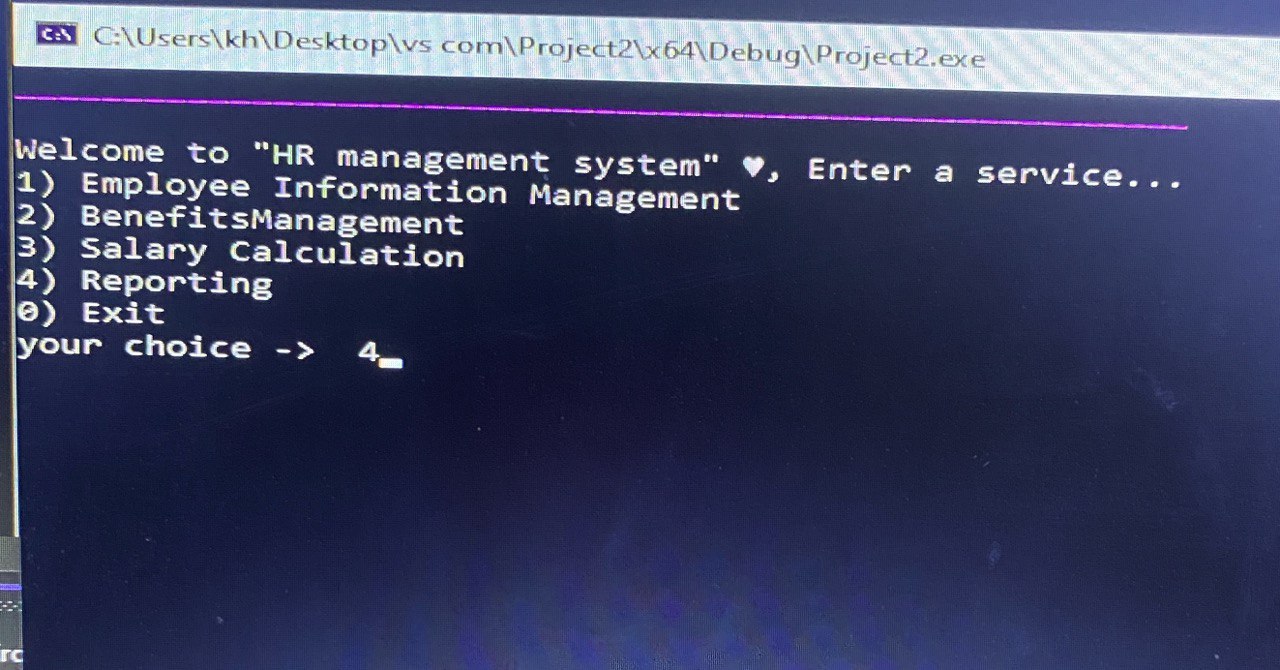
-> edit

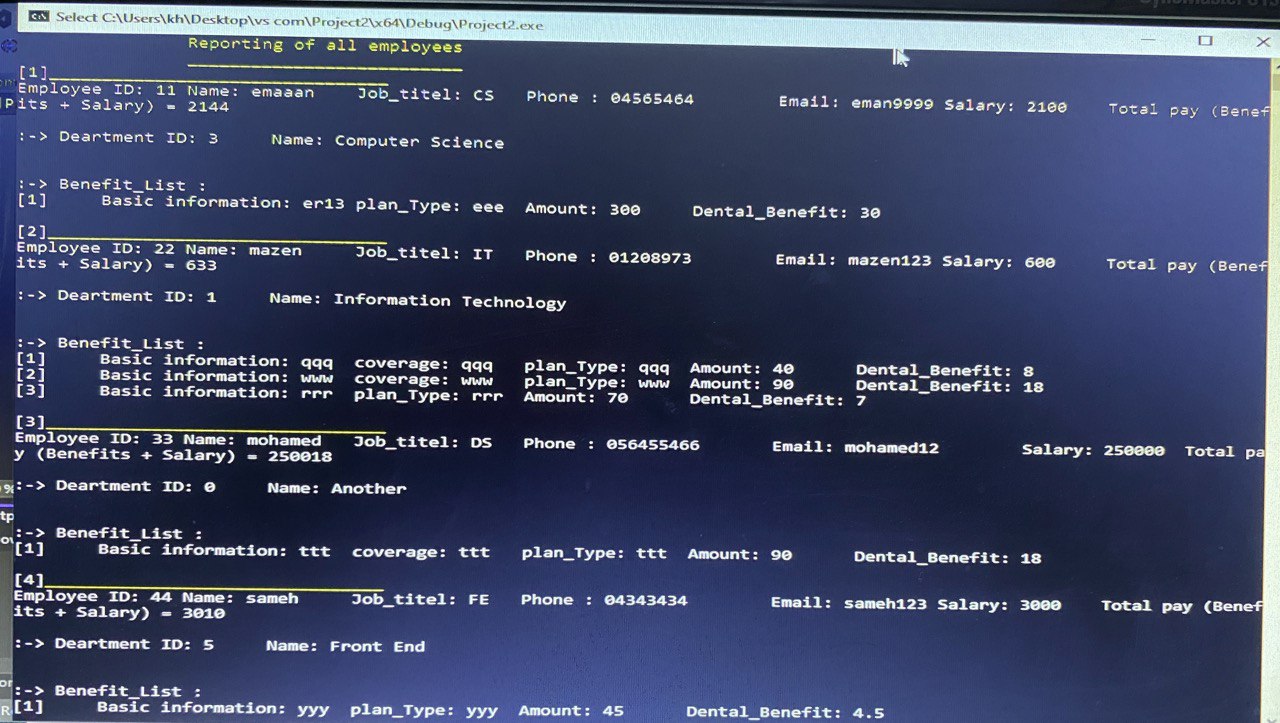


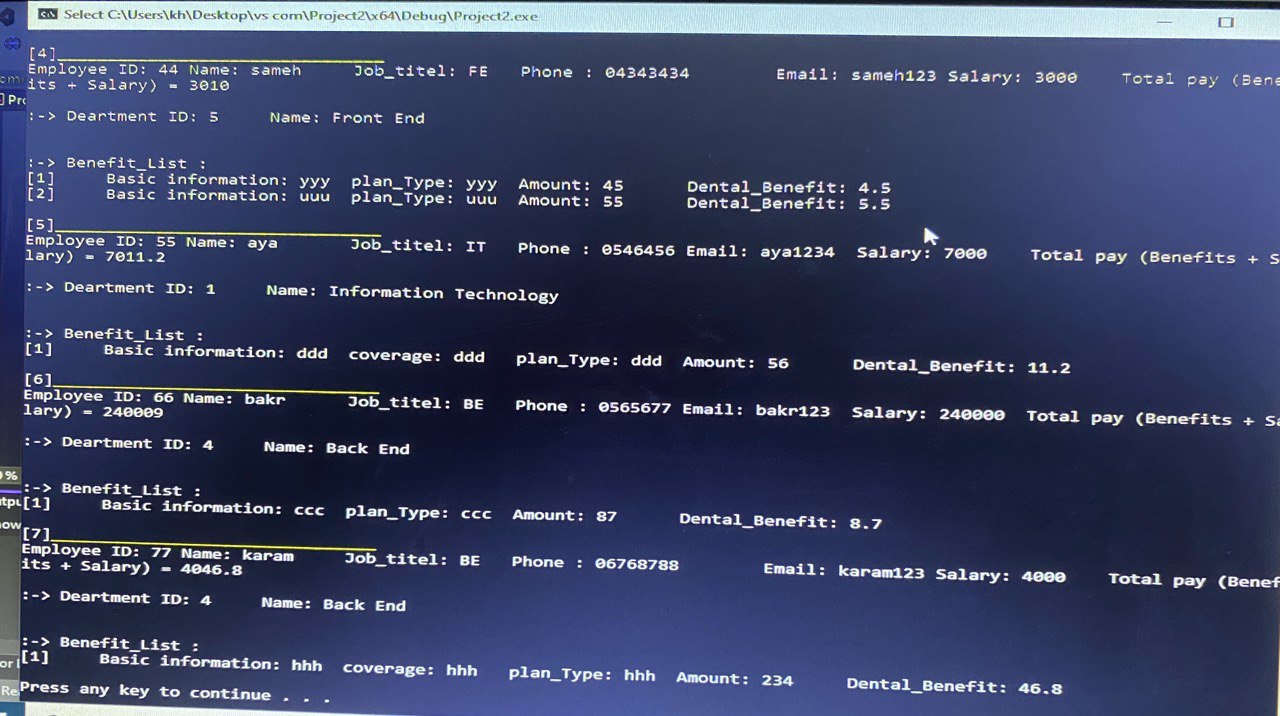
[3] Salary Calculation: to calculate wanted employee payroll

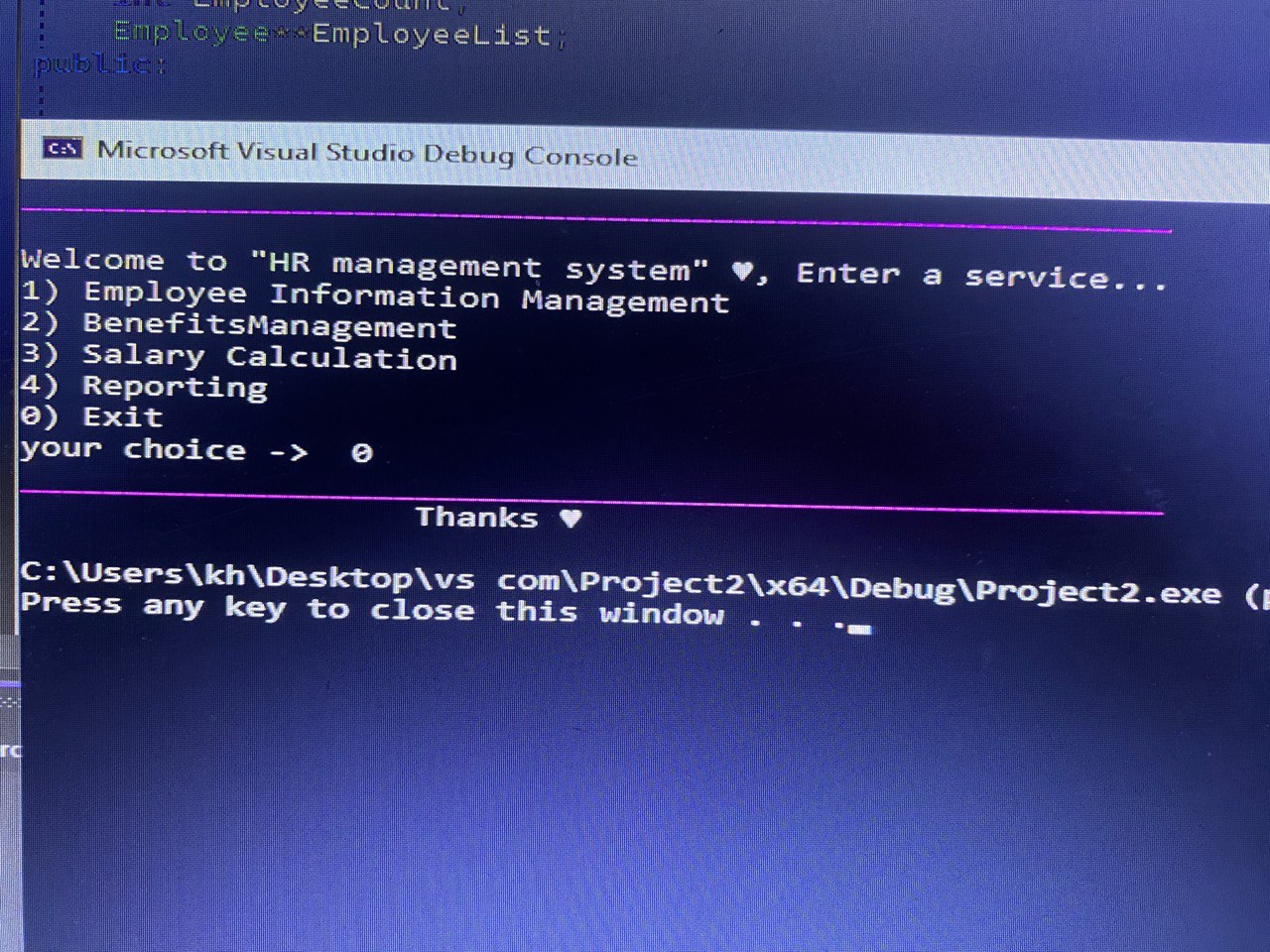


[4] Reporting: print all employees







[5] Finish

C++ code

1. Benefit

Header

#pragma once

#include <iostream>

#include <string>

using namespace std;

#ifndef BENEFIT\_H

#define BENEFIT\_H

class benefit

{

private:

protected:

string m\_planType;

double m\_amount;

public:

benefit();

virtual ~benefit();

virtual string getInfo()=0;

virtual void getDetails()=0;

virtual double caculateBenefit()=0;

virtual void displayBenefit()=0;

};

#endif // !BENEFIT\_H

Source

#include "benefit.h"

benefit::benefit()

{

}

benefit::~benefit()

{

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Health Benefit

Header

#pragma once

#include "benefit.h"

class HealthBenefit :public benefit

{

private:

string info;

string coverage;

public:

double caculateBenefit();

void displayBenefit();

void getDetails();

string getInfo();

};

Source

#include "HealthBenefit.h"

double HealthBenefit::caculateBenefit()

{

return .2\*m\_amount;

}

void HealthBenefit::displayBenefit()

{

cout << "Basic information: " << info <<"\tcoverage: "<<coverage << "\tplan\_Type: " << m\_planType << "\tAmount: " << m\_amount;

cout << "\tDental\_Benefit: " << caculateBenefit();

}

void HealthBenefit::getDetails()

{

cout << "Enter the infornation : "; cin >> info;

cout << "Enter Coverage : "; cin >> coverage;

cout << "Enter Plan\_Type : "; cin >> m\_planType;

cout << "Enter the amount : "; cin >> m\_amount;

}

string HealthBenefit::getInfo()

{

return info;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Dental Benefit

Header

#pragma once

#include "benefit.h"

class DentalBenefit :public benefit

{

private:

string info;

public:

double caculateBenefit();

void displayBenefit();

void getDetails();

string getInfo();

};

Source

#include "DentalBenefit.h"

double DentalBenefit::caculateBenefit()

{

return .1\*m\_amount ;

}

void DentalBenefit::displayBenefit()

{

cout << "Basic information: " << info << "\tplan\_Type: " << m\_planType << "\tAmount: " << m\_amount;

cout << "\tDental\_Benefit: " << caculateBenefit();

}

void DentalBenefit::getDetails()

{

cout << "Enter the infornation : "; cin >> info;

cout << "Enter Plan\_Type : "; cin >> m\_planType;

cout << "Enter the amount : "; cin >> m\_amount;

}

string DentalBenefit::getInfo()

{

return info;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Department

Header

#pragma once

#include <iostream>

#include <string>

using namespace std;

#ifndef DEPARTMENT\_H

#define DEPARTMENT\_H

class Department

{

private:

string m\_departName;

int m\_ID;

public:

Department();

~Department();

void readDepartment(int , string);

void printDepartment();

};

#endif // !DEPARTMENT\_H

Source

#include "Department.h"

Department::Department()

{

}

Department::~Department()

{}

void Department::readDepartment(int i , string n)

{

m\_ID = i;

m\_departName = n;

}

void Department::printDepartment()

{

cout << "\n:-> Deartment ID: " << m\_ID << "\tName: " << m\_departName << endl;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Employee

Header

#pragma once

#include <Windows.h>///sleep, color

#include <cstdlib>///system("cls")

#include "Department.h"

#include "benefit.h"

#include "HealthBenefit.h"

#include "Dentalbenefit.h"

class Employee : public Department

{

protected:

int employee\_id;

string employee\_name;

string employee\_phone;

string employee\_email;

string job\_title;

double totalBene=0 ;

private:

int benfitSize;

benefit\*\* benefitList;

int benefitCount;

public:

Employee();

~Employee();

void setColor(int color);

void storeBenefitList();

void getBenefitList();

bool deleteBenbfit(string );

bool editBenefit(string );

int getEmployeeID();

virtual void dispalyDetails();

virtual void getEmployeeDetails();

virtual double getSalary()=0;

virtual double caculatePay()=0;

};

Source

#include "Employee.h"

void Employee::setColor(int color)

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, color);

}

Employee::Employee()

{

benfitSize = 20;

benefitCount = 0;

//new.........|| بتاعه الاراي

benefitList = new benefit \* [benfitSize];

}

Employee::~Employee()

{

//delete elements -> ( pointers ) inside dynamic array

//(auto) returns ambiguous type automatically

for (auto i = 0; i < benefitCount; i++)

delete benefitList[i];

//delete dynamic array itself

delete[] benefitList;

}

int Employee::getEmployeeID()

{

return employee\_id;

}

void Employee::dispalyDetails()

{

cout << "Employee ID: " << employee\_id << "\tName: " << employee\_name

<< "\tJob\_titel: " << job\_title

<< "\tPhone : " << employee\_phone << "\tEmail: " << employee\_email ;

}

void Employee::getEmployeeDetails()

{

cout << "Enter employee ID: "; cin >> employee\_id;

cout << "employee Name: "; cin >> employee\_name;

cout << "Job\_title: "; cin >> job\_title;

cout << "Email: "; cin >> employee\_email;

cout << "Phone: "; cin >> employee\_phone;

}

void Employee::storeBenefitList()

{

if (benefitCount < benfitSize)

{

int c;

char ch;

setColor(9);

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

cout << "\n\*BENEFIT BACKAGES\*, you can add up to 20 BK... \n";

cout << "1) Add HealthBenefit \n"

<< "2) Add DentalBenefit \n"

<< "0) Add \"Null\" \n";

setColor(9);

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

do

{

cout << "Your choice -> ";

cin >> c;

cout << endl;

switch (c)

{

case 1:

//البوينتر هيتخزن فيه الاوبجيكت فلازم الاوبجيكت يتخزن ف ال هيب

//والا البوينتر بعد ال مين هيكون فاضى

//benefitList[0] (or any index) -> is a pointer

//new.......................|| بتاكه كل اوبيجيكت على حده deleted dy loop

benefitList[benefitCount] = new HealthBenefit;

benefitList[benefitCount]->getDetails();

totalBene += benefitList[benefitCount++]->caculateBenefit();

break;

case 2:

benefitList[benefitCount] = new DentalBenefit;

benefitList[benefitCount]->getDetails();

totalBene += benefitList[benefitCount++]->caculateBenefit();

break;

case 0:

return;

default:

cout << "invalid choice !\n";

break;

}

cout << "\nAnother ? (y/n) : ";

cin >> ch;

cout << endl;

} while (ch != 'n');

}

else

{

cout << "sorry, You can\'t add anymore " << endl;

}

}

void Employee::getBenefitList()

{

if (benefitCount == 0)

{

cout << ":-> There is\'t any benefit packages yet \n";

return;

}

cout << "\n\n:-> Benefit\_List : "<<endl;

for (int i = 0; i < benefitCount; i++)

{

cout << "[" << i + 1 << "]\t";

benefitList[i]->displayBenefit();

cout << endl;

}

}

bool Employee::deleteBenbfit(string inf)

{

if (benefitCount == 0)

{

return false;

}

bool found = false;

for (int i = 0; i < benefitCount; i++)

{

if (benefitList[i]->getInfo() == inf)

{

found = true;

benefitList[i] = benefitList[benefitCount - 1];

--benefitCount;

//delete old benefit from totalbene

totalBene -= benefitList[i]->caculateBenefit();

return true;

}

else

found = false;

}

if (found == false)

return false;

}

bool Employee::editBenefit(string inf)

{

if (benefitCount == 0)

{

return false;

}

bool found = false;

for (int j = 0; j < benefitCount; j++)

{

if (benefitList[j]->getInfo() == inf)

{

//delete old benefit from totalbene

totalBene -= benefitList[j]->caculateBenefit();

cout << "Enter the new package...\n";

benefitList[j]->getDetails();

//add the new one to totalbene

totalBene += benefitList[j]->caculateBenefit();

return true;

}

else

found = false;

}

if (found == false)

return false;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Hourly Employee

Header

#pragma once

#include "Employee.h"

class HourlyEmployee : public Employee

{

private:

double hoursWorked;

double rate;

public:

HourlyEmployee();

void addMoreHours(int hours);

void getEmployeeDetails();

void dispalyDetails();

double getSalary();

double caculatePay();

};

Source

#include "HourlyEmployee.h"

HourlyEmployee::HourlyEmployee()

{

}

void HourlyEmployee::addMoreHours(int hours)

{

hoursWorked += hours;

}

void HourlyEmployee::getEmployeeDetails()

{

Employee::getEmployeeDetails();

cout << "worked Hours: "; cin >> hoursWorked;

cout << "Rate: "; cin >> rate;

}

double HourlyEmployee::getSalary()

{

return rate \* hoursWorked;

}

double HourlyEmployee::caculatePay()

{

return getSalary() + totalBene;

}

void HourlyEmployee::dispalyDetails()

{

Employee::dispalyDetails();

cout << "\tSalary: " << getSalary();

cout << "\tTotal pay (Benefits + Salary) = "<<caculatePay();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Commission Employee

Header

#pragma once

#include "Employee.h"

class CommissionEmployee :public Employee

{

private:

double target;

double rate;

public:

CommissionEmployee();

void getEmployeeDetails();

void dispalyDetails();

double caculatePay();

double getSalary();

};

Source

#include "CommissionEmployee.h"

double CommissionEmployee::getSalary()

{

return rate\*target;

}

CommissionEmployee::CommissionEmployee()

{ }

void CommissionEmployee::getEmployeeDetails()

{

Employee::getEmployeeDetails();

cout << "Target: "; cin >> target;

cout << "Rate: "; cin >> rate;

}

double CommissionEmployee::caculatePay()

{

return getSalary() + totalBene;

}

void CommissionEmployee::dispalyDetails()

{

Employee::dispalyDetails();

cout << "\tSalary: " << getSalary();

cout << "\tTotal pay (Benefits + Salary) = " << caculatePay();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Salaried Employee

Header

#pragma once

#include "Employee.h"

class SalaryEmployee : public Employee

{

public:

double salary;

double getSalary();

SalaryEmployee();

void getEmployeeDetails();

void dispalyDetails();

double caculatePay();

};

Source

#include "SalaryEmployee.h"

double SalaryEmployee::getSalary()

{

return salary;

}

SalaryEmployee::SalaryEmployee()

{ }

void SalaryEmployee::getEmployeeDetails()

{

Employee::getEmployeeDetails();

cout << "Salary: "; cin >> salary;

}

double SalaryEmployee::caculatePay()

{

return getSalary() + totalBene;

}

void SalaryEmployee::dispalyDetails()

{

Employee::dispalyDetails();

cout << "\tSalary: " << getSalary();

cout << "\tTotal pay (Benefits + Salary) = " << caculatePay();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Manager Employee

Header

#pragma once

#include "SalaryEmployee.h"

class managerEmployee:public SalaryEmployee

{

private:

double bouns;

public:

managerEmployee();

double getSalary();

void getEmployeeDetails();

void dispalyDetails();

double caculatePay();

};

Source

#include "managerEmployee.h"

managerEmployee::managerEmployee()

{

}

double managerEmployee::getSalary()

{

return salary+bouns;

}

void managerEmployee::getEmployeeDetails()

{

SalaryEmployee::getEmployeeDetails();

cout << "bouns: "; cin >> bouns;

}

double managerEmployee::caculatePay()

{

return getSalary() + totalBene;

}

void managerEmployee::dispalyDetails()

{

Employee::dispalyDetails();

cout << "\tSalary: " << getSalary();

cout << "\tTotal pay (Benefits + Salary) = " << caculatePay();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. HR System

Header

#pragma once

#include "Employee.h"

#include "HourlyEmployee.h"

#include "SalaryEmployee.h"

#include "managerEmployee.h"

#include "CommissionEmployee.h"

class HRSystem

{

private:

int EmployeeSize;

int EmployeeCount;

Employee\*\*EmployeeList;

public:

HRSystem(int=100);

~HRSystem();

void setColor(int color);

void addEmployee();

void employeeSearch();

void deleteEmployee();

void editEmployee();

void addBPackage();

void deleteBPackage();

void editBPackage();

void salaryCaculation();

void displayAll();

void addDepart(int);

};

Source

#include "HRSystem.h"

void HRSystem::setColor(int color)

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole,color);

}

HRSystem::HRSystem(int Size)

{

EmployeeSize = Size;

EmployeeList = new Employee \* [EmployeeSize];

EmployeeCount = 0;

}

HRSystem::~HRSystem()

{

for (auto i = 0; i < EmployeeCount; i++)

delete EmployeeList[i];

delete[] EmployeeList;

}

void HRSystem::addBPackage()

{

if (EmployeeCount == 0)

{

cout << "There are no employees yet" << endl;

Sleep(2000);

return;

}

system("cls");

setColor(9);

cout << "\t\tADDING AN EMPLOYEE BENEFIT PACKAGE" << endl;

cout << "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

int id;

cout << "Enter Employee ID To Add him a PK : ";

cin >> id;

bool Found = false;

for (int i = 0; i < EmployeeCount; i++)

{

if (EmployeeList[i]->getEmployeeID() == id)

{

Found = true;

EmployeeList[i]->storeBenefitList();

cout << "\nAdded Successfully \xFB" << endl;

break;

}

else

Found = false;

}

if (Found == false)

cout << "Add Failed " << endl;

Sleep(2000);

}

void HRSystem::deleteBPackage()

{

if (EmployeeCount == 0)

{

cout << "There are no employees yet" << endl;

Sleep(2000);

return;

}

int id;

string info;

cout << "Enter Employee ID to delete from his packages & Wnated package information : ";

cin >>id>> info;

bool found = false;

for (int i = 0; i < EmployeeCount; i++)

{

if (EmployeeList[i]->getEmployeeID() == id)

{

if (EmployeeList[i]->deleteBenbfit(info))

{

cout << "Benefit package deleted successfully \xFB\n";

Sleep(2000);

return;

}

}

else

found = false;

}

if (found == false)

cout << "Delete faild " << endl;

Sleep(2000);

}

void HRSystem::editBPackage()

{

if (EmployeeCount == 0)

{

cout << "There are no employees yet" << endl;

Sleep(2000);

return;

}

int id;

string info;

cout << "Enter employee ID to edit his packages & Wnated package information : ";

cin>>id >> info;

bool found1 = false;

for (int i = 0; i < EmployeeCount; i++)

{

if (EmployeeList[i]->getEmployeeID() == id)

{

if (EmployeeList[i] ->editBenefit(info))

{

cout << "\nBenfit package edited successfully \xFB\n";

Sleep(2000);

return;

}

}

else

found1 = false;

}

if (found1 == false)

cout << "Edit faild " << endl;

Sleep(2000);

}

void HRSystem::addDepart(int x)

{

int c;

cout << endl;

cout << "Chose The Depatrment : \n" << "1) Information Technology\n" << "2) Information System\n"

<< "3) Computer Science\n" << "4) Back End\n" << "5) Front End\n" << "6) Cyber Security\n" << "0) Another\n" << endl;

cout << "Your choice -> ";

cin >> c;

switch (c)

{

case 1:

EmployeeList[x]->readDepartment(1, "Information Technology");

break;

case 2:

EmployeeList[x]->readDepartment(2, "Information System");

break;

case 3:

EmployeeList[x]->readDepartment(3, "Computer Science");

break;

case 4:

EmployeeList[x]->readDepartment(4, "Back End");

break;

case 5:

EmployeeList[x]->readDepartment(5, "Front End");

break;

case 6:

EmployeeList[x]->readDepartment(6, "Cyber Security");

break;

case 0:

EmployeeList[x]->readDepartment(0, "Another");

break;

default:

cout << "Invalid choice \n";

EmployeeList[x]->readDepartment(0, "Another");

break;

}

}

void HRSystem::addEmployee()

{

int c;

if (EmployeeCount < EmployeeSize)

{

cout << "Enter Employee type...\n"

<< "1) Hourly\_Employee \n"

<< "2) commission\_Employee \n"

<< "3) Salary\_Employee \n"

<< "4) Manager\_Employee \n"

<< "Your choice -> ";

cin >> c;

system("cls");

switch (c)

{

case 1:

setColor(11);

cout << "\t\t ADDIND HOURLY EMPLOYEE" << endl;

cout << "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

EmployeeList[EmployeeCount] = new HourlyEmployee;

EmployeeList[EmployeeCount]->getEmployeeDetails();

addDepart(EmployeeCount);

EmployeeCount++;

break;

case 2:

setColor(11);

cout << "\t\t ADDIND COMMISSION EMPLOYEE" << endl;

cout << "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

EmployeeList[EmployeeCount] = new CommissionEmployee;

EmployeeList[EmployeeCount]->getEmployeeDetails();

addDepart(EmployeeCount);

EmployeeCount++;

break;

case 3:

setColor(11);

cout << "\t\t ADDIND SALARY EMPLOYEE" << endl;

cout << "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

EmployeeList[EmployeeCount] = new SalaryEmployee;

EmployeeList[EmployeeCount]->getEmployeeDetails();

addDepart(EmployeeCount);

EmployeeCount++;

break;

case 4:

setColor(11);

cout << "\t\t ADDIND MANAGER EMPLOYEE" << endl;

cout << "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

EmployeeList[EmployeeCount] = new managerEmployee;

EmployeeList[EmployeeCount]->getEmployeeDetails();

addDepart(EmployeeCount);

EmployeeCount++;

break;

default:

cout << "Invalid choice \n";

Sleep(2000);

break;

}

cout << "\nEmployee Added Successfully \x03 " << endl;

Sleep(2000);

}

else

{

cout << "Sorry, you can\'t add any more.\n";

Sleep(2000);

}

}

void HRSystem::employeeSearch()

{

if(EmployeeCount==0)

{

cout << "There are no employees yet"<<endl;

Sleep(2000);

return;

}

system("cls");

setColor(11);

cout << "\t\tSEARCHING FOR AN EMPLOYEE" << endl;

cout << "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

int id;

cout << "Enter Employee ID To Search: ";

cin >> id;

bool Found = false;

for (int i = 0; i < EmployeeCount; i++)

{

if (EmployeeList[i]->getEmployeeID() == id)

{

Found = true;

cout << "Employee found successfully \xFB " << endl;

break;

}

else

Found = false;

}

if (Found == false)

cout << "Search Failed " << endl;

Sleep(2000);

}

void HRSystem::deleteEmployee()

{

if (EmployeeCount == 0)

{

cout << "There are no employees yet" << endl;

Sleep(2000);

return;

}

setColor(11);

system("cls");

cout << "\t\tDELETING AN EMPLOYEE" << endl;

cout << "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

int id;

cout << "Enter Employee ID To Delete: ";

cin >> id;

bool Found = false;

for (int i = 0; i < EmployeeCount; i++)

{

if (EmployeeList[i]->getEmployeeID() == id)

{

Found = true;

EmployeeList[i] = EmployeeList[EmployeeCount - 1];

EmployeeCount--;

cout << "Employee Deleted Successfully \xFB" << endl;

break;

}

else

Found = false;

}

if (Found == false)

cout << "Delete Failed " << endl;

Sleep(2000);

}

void HRSystem::editEmployee()

{

if (EmployeeCount == 0)

{

cout << "There are no employees yet" << endl;

Sleep(2000);

return;

}

system("cls");

setColor(11);

cout << "\t\tMODIFING AN EMPLOYEE" << endl;

cout << "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

int id;

cout << "Enter Employee ID To Modify: ";

cin >> id;

bool Found = false;

for (int i = 0; i < EmployeeCount; i++)

{

if (EmployeeList[i]->getEmployeeID() == id)

{

Found = true;

cout << "\n\*\*\*Enter The New Information\*\*\*\n";

EmployeeList[i] ->getEmployeeDetails();

addDepart(i);

cout << "Employee Modified Successfully \xFB" << endl;

break;

}

else

Found = false;

}

if (Found == false)

cout << "Modify Failed " << endl;

Sleep(2000);

}

void HRSystem::displayAll()

{

if (EmployeeCount == 0)

{

cout << "There are no employees yet" << endl;

Sleep(2000);

return;

}

system("cls");

setColor(14);

cout << "\t\tReporting of all employees \n";

cout << "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

setColor(15);

for (int i = 0; i < EmployeeCount; i++)

{

cout << "[" << i + 1 << "]";

setColor(14);

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

setColor(15);

EmployeeList[i]->dispalyDetails();

cout << endl;

EmployeeList[i]->printDepartment();

EmployeeList[i]->getBenefitList();

cout << endl;

}

system("pause");

}

void HRSystem::salaryCaculation()

{

if (EmployeeCount == 0)

{

cout << "There are no employees yet " << endl;

return;

}

int id;

cout << "Enter employee ID to caculate his salary : ";

cin >> id;

bool found = false;

for (int i = 0; i < EmployeeCount; i++)

{

if (EmployeeList[i]->getEmployeeID() == id)

{

found = true;

cout << "Employee Salary = ";

cout << EmployeeList[i]->getSalary() << " $" << endl;

system("pause");

}

else

found = false;

}

if (found == false)

cout << "There no employee with ID " << id << endl;

}

1. Main Code

#include "HRSystem.h"

using namespace std;

///color

HANDLE cout\_handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

struct mainDesign

{

HRSystem s1;

static inline void print(string s, bool newline = true)

{

///instead of cout

cout << s << (newline ? "\n" : "\t");

}

static void line(char c, int y, int x)

{

///separator

SetConsoleTextAttribute(cout\_handle, x);

for (int i = 0; i < y; i++)

cout << c;

cout << endl;

SetConsoleTextAttribute(cout\_handle, 15);

}

static void Back(int x)

{

cout << "Back to main page ";

for (int i = 0; i < x; i++)

{

cout << ".";

Sleep(1000);

}

cout << endl;

}

void HR()

{

system("cls");

line('\_', 55, 13);

print("\nWelcome to \"HR management system\" \x03, Enter a service...");

print("1) Employee Information Management ");

print("2) BenefitsManagement ");

print("3) Salary Calculation ");

print("4) Reporting ");

print("0) Exit ");

print("your choice ->", false);

}

void Employee\_Information\_Management()

{

int c;

while (true)

{

system("cls");

line('\_', 66, 11);

print("\nWelcome to \"Employee Information Management\" \x03, Enter a service...");

print("1) Add an employee ");

print("2) Delete an employee");

print("3) Modify employee information ");

print("4) Search for an employee ");

print("0) Back to previous page ");

print("your choice ->", false);

cin >> c;

line('\_', 66, 11);

switch (c)

{

case 1:

//add employee

s1.addEmployee();

break;

case 2:

//delete employee

s1.deleteEmployee();

break;

case 3:

//Modify employee information

s1.editEmployee();

break;

case 4:

//search employee

s1.employeeSearch();

break;

case 0:

Back(3);

return;

default:

cout << "Invalid choice,try again \n";

Sleep(2000);

}

}

}

void Benefits\_Management()

{

int c;

while (true)

{

system("cls");

line('\_', 54, 9);

print("\nWelcome to \"Benefits Management\" \x03, Enter a service...");

print("1) Add a benefit package ");

print("2) Delete a benefit package");

print("3) Modify a benefit package ");

print("0) Back to main page ");

print("your choice ->", false);

cin >> c;

line('\_', 54, 9);

switch (c)

{

case 1:

//add

s1.addBPackage();

break;

case 2:

//delete

s1.deleteBPackage();

break;

case 3:

//Modify

s1.editBPackage();

break;

case 0:

Back(3);

return;

default:

cout << "\nInvalid choice,try again \n";

Sleep(2000);

}

}

}

void SalaryCa()

{

s1.salaryCaculation();

}

void display()

{

s1.displayAll();

}

}E;

int main()

{

int c;

while (true)

{

E.HR();

cin >> c;

mainDesign::line('\_', 55,13);

Sleep(100);

switch (c)

{

case 1:

//Employee Information Management

E.Employee\_Information\_Management();

break;

case 2:

//Benefits Management

E.Benefits\_Management();

break;

case 3:

//Salary Calculation

E.SalaryCa();

break;

case 4:

//Reporting

E.display();

break;

case 0:

//Exit

mainDesign::print("\t\t Thanks \x03");

return 0;

default:

cout<<"\nInvalid choice, Try again \n";

Sleep(2000);

}

}

return 0;

}