МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №1 з дисципліни

«Системне програмування С і С++»

Варіант 2

|  |  |
| --- | --- |
| Перевірив:  Сокульський О.Є. | Виконав:  студент 3 курсу  групи ІП-11 ФІОТ  Прищепа В.С. |

Київ-2023

**Лабораторна робота №1**

**Технологія розробки системних програм для POSIX сумісних (сертифікованих) операційних систем**

Мета лабораторної роботи полягає у набутті знань, умінь та навичок з технології розроблення системного програмного забезпечення (ПЗ) з використанням мови С і С++ для POSIX сумісних (сертифікованих) операційних систем.

**Завдання:**

1. Відповідно до завдання і тематичної спрямованості, що надано у додатку 1, користуючись вихідним кодом додатку 2, створити класи, що реалізують тематичну спрямованість додатку 1.

2 Під час розробки навчальної програми для кожного класу створити заголовочний файл і файл реалізації.

3 Створити файл main.cpp для екземплярів класу на купі пам'яті і на стеку. Провести тестування класів на етапі виконання. Для цього скористатися утилітою Makefile, текстовим редактором або середовищем розробки. Запуск реалізувати у POSIX сумісній ОС.

**Тематична спрямованість для створення класів ЛР 1. Додаток 1**

No2: Інформаційна система Радіостанції.

Клас

Посади (Код посади, Найменування посади, Оклад, Обов'язки, Вимоги) [5 записів].

Передбачити функціональність додавання інформації, вивід інформації на консоль.

Передбачити функціональність поведінки класу, в залежності від значення певних членів класу.

Під час тестування на етапі виконання створити екземпляри класу на купі пам'яті і на стеку.

**Код програми:**

**Position.h**

#include <iostream>

using namespace std;

namespace Records {

class Position

{

public:

Position();

Position(string inCode, string inName, int inSalary, string inDuties, string inRequirements);

void sRaise(int num);

void sLower(int num);

void display();

// Accessors and setters

void setCode(string inCode);

string getCode();

void setName(string inName);

string getName();

void setSalary(int inSalary);

int getSalary();

void setDuties(string inDuties);

string getDuties();

void setRequirements(string inRequirements);

string getRequirements();

void setNumber(int inNumber);

int getNumber();

private:

string pCode;

string pName;

int pSalary;

string pDuties;

string pRequirements;

int pNumber;

};

}

**Position.cpp**

#include <iostream>

#include <string>

#include "Position.h"

using namespace std;

namespace Records {

Position::Position()

{

pCode = "";

pName = "";

pSalary = 0;

pDuties = "";

pRequirements = "";

}

Position::Position(string inCode, string inName, int inSalary, string inDuties, string inRequirements)

{

this->pCode = inCode;

this->pName = inName;

this->pSalary = inSalary;

this->pDuties = inDuties;

this->pRequirements = inRequirements;

}

void Position::sRaise(int num)

{

if (pSalary + num >= 0) {

pSalary += num;

}

else {

cout << "Wrong input!\n";

}

}

void Position::sLower(int num)

{

if (pSalary - num >= 0) {

pSalary -= num;

}

else {

cout << "Wrong input!\n";

}

}

void Position::display()

{

cout << "Number: " << getNumber() << endl;

cout << "Position: " << getName() << endl;

cout << "-------------------------" << endl;

cout << "Code: " << getCode() << endl;

cout << "Salary: " << getSalary() << endl;

cout << "Duties: " << getDuties() << endl;

cout << "Requirements: " << getRequirements() << endl;

cout << endl;

}

// Accessors and setters

void Position::setCode(string inCode)

{

pCode = inCode;

}

string Position::getCode()

{

return pCode;

}

void Position::setName(string inName)

{

pName = inName;

}

string Position::getName()

{

return pName;

}

void Position::setSalary(int inSalary)

{

if (inSalary >= 0) {

pSalary = inSalary;

}

else {

cout << "Wrong input!\n";

}

}

int Position::getSalary()

{

return pSalary;

}

void Position::setDuties(string inDuties)

{

pDuties = inDuties;

}

string Position::getDuties()

{

return pDuties;

}

void Position::setRequirements(string inRequirements)

{

pRequirements = inRequirements;

}

string Position::getRequirements()

{

return pRequirements;

}

void Position::setNumber(int inNumber)

{

pNumber = inNumber;

}

int Position::getNumber()

{

return pNumber;

}

}

**Database.h**

#include <iostream>

#include "Position.h"

using namespace std;

namespace Records {

const int kMaxPositions = 100;

const int kFirstPositionNumber = 1;

class Database

{

public:

Database();

~Database();

Position& addPosition(string inCode, string inName, int inSalary, string inDuties, string inRequirements);

Position& getPosition(int inPositionNumber);

Position& getPosition(string inName);

void displayAll();

protected:

Position mPosition[kMaxPositions];

int mNextSlot;

int mNextPositionNumber;

};

}

**Database.cpp**

#include <iostream>

#include <stdexcept>

#include <string>

#include "Database.h"

using namespace std;

namespace Records {

Database::Database()

{

mNextSlot = 0;

mNextPositionNumber = kFirstPositionNumber;

}

Database::~Database()

{

}

Position& Database::addPosition(string inCode, string inName, int inSalary, string inDuties, string inRequirements)

{

if (mNextSlot >= kMaxPositions) {

cerr << "There is no more room to add the new position!" << endl;

throw exception();

}

Position& thePosition = mPosition[mNextSlot++];

thePosition.setCode(inCode);

thePosition.setName(inName);

thePosition.setSalary(inSalary);

thePosition.setDuties(inDuties);

thePosition.setRequirements(inRequirements);

thePosition.setNumber(mNextPositionNumber++);

return thePosition;

}

Position& Database::getPosition(int inPositionNumber)

{

for (int i = 0; i < mNextSlot; i++) {

if (mPosition[i].getNumber() == inPositionNumber) {

return mPosition[i];

}

}

cerr << "No position with number " << inPositionNumber << endl;

throw exception();

}

Position& Database::getPosition(string inName)

{

for (int i = 0; i < mNextSlot; i++) {

if (mPosition[i].getName() == inName) {

return mPosition[i];

}

}

cerr << "No match with name " << inName << endl;

throw exception();

}

void Database::displayAll()

{

for (int i = 0; i < mNextSlot; i++) {

mPosition[i].display();

}

}

}

**Radio.cpp**

#include <iostream>

#include <stdexcept>

#include <string>

#include "Database.h"

using namespace std;

using namespace Records;

int displayMenu();

void doAdd(Database& inDB);

void doRaise(Database& inDB);

void doLower(Database& inDB);

int main(int argc, char\*\* argv)

{

char typ;

cout << "Stack (s) or heap (other symbols)?";

cin >> typ;

if (typ == 's') {

Database\* positionDB = new Database;

positionDB->addPosition("cod1", "nam1", 200, "nothing", "no");

positionDB->addPosition("cod2", "Jack", 150, "anything", "be tall");

positionDB->addPosition("cod3", "sunny", 30, "clean my room", "have vacuum cleaner");

positionDB->addPosition("cod4", "trudo", 0, "to obey me", "be quite");

positionDB->addPosition("cod5", "nam1\_again", 12, "nothing\_again", "be similar to nam1");

bool done = false;

while (!done) {

int selection = displayMenu();

switch (selection) {

case 1:

doAdd(\*positionDB);

break;

case 2:

doRaise(\*positionDB);

break;

case 3:

doLower(\*positionDB);

break;

case 4:

positionDB->displayAll();

break;

case 0:

done = true;

break;

default:

cerr << "Unknown command." << endl;

}

}

delete positionDB;

}

else {

Database positionDB;

positionDB.addPosition("cod1", "nam1", 200, "nothing", "no");

positionDB.addPosition("cod2", "Jack", 150, "anything", "be tall");

positionDB.addPosition("cod3", "sunny", 30, "clean my room", "have vacuum cleaner");

positionDB.addPosition("cod4", "trudo", 0, "to obey me", "be quite");

positionDB.addPosition("cod5", "nam1\_again", 12, "nothing\_again", "be similar to nam1");

bool done = false;

while (!done) {

int selection = displayMenu();

switch (selection) {

case 1:

doAdd(positionDB);

break;

case 2:

doRaise(positionDB);

break;

case 3:

doLower(positionDB);

break;

case 4:

positionDB.displayAll();

break;

case 0:

done = true;

break;

default:

cerr << "Unknown command." << endl;

}

}

}

}

int displayMenu()

{

int selection;

cout << endl;

cout << "Position Database" << endl;

cout << "-----------------" << endl;

cout << "1) Add a new position" << endl;

cout << "2) Raise salary" << endl;

cout << "3) Lower salary" << endl;

cout << "4) List all positions" << endl;

cout << "0) Quit" << endl;

cout << endl;

cout << "---> ";

cin >> selection;

return selection;

}

void doAdd(Database& inDB)

{

string Code, Name, Duties, Requirements;

int Salary;

cout << "Name of the position? ";

cin >> Name;

cout << "Code of the position? ";

cin >> Code;

cout << "Duties of the position? ";

cin >> Duties;

cout << "Requirements of the position? ";

cin >> Requirements;

cout << "Salary of the position? ";

cin >> Salary;

if (Salary < 0) {

cout << "Wromg input!\n";

return;

}

try {

inDB.addPosition(Code, Name, Salary, Duties, Requirements);

}

catch (std::exception ex) {

cerr << "Unable to add new position!" << endl;

}

}

void doRaise(Database& inDB)

{

int positionNumber, raiseAmount;

cout << "Position number? ";

cin >> positionNumber;

try {

Position& emp = inDB.getPosition(positionNumber);

}

catch (std::exception ex) {

cerr << "Unable to find position!" << endl;

return;

}

Position& emp = inDB.getPosition(positionNumber);

cout << "Position " << emp.getName() << " salary: " << emp.getSalary() << endl;

cout << "How much? ";

cin >> raiseAmount;

emp.sRaise(raiseAmount);

}

void doLower(Database& inDB)

{

int positionNumber, lowerAmount;

cout << "Position number? ";

cin >> positionNumber;

try {

Position& emp = inDB.getPosition(positionNumber);

}

catch (std::exception ex) {

cerr << "Unable to find position!" << endl;

return;

}

Position& emp = inDB.getPosition(positionNumber);

cout << "Position " << emp.getName() << " salary: " << emp.getSalary() << endl;

cout << "How much? ";

cin >> lowerAmount;

emp.sLower(lowerAmount);

}

**Makefile**

CC = g++

CCFLAGS = -std=c++11 -Wall

SOURCES = Position.cpp Database.cpp Radio.cpp

OBJECTS = $(SOURCES:.cpp=.o)

EXECUTABLE = PositionDB

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)

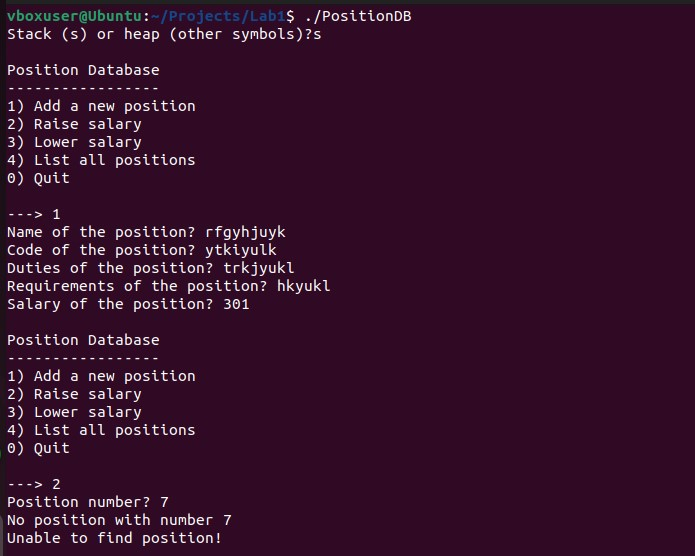
$(CC) $(CCFLAGS) $(OBJECTS) -o $@

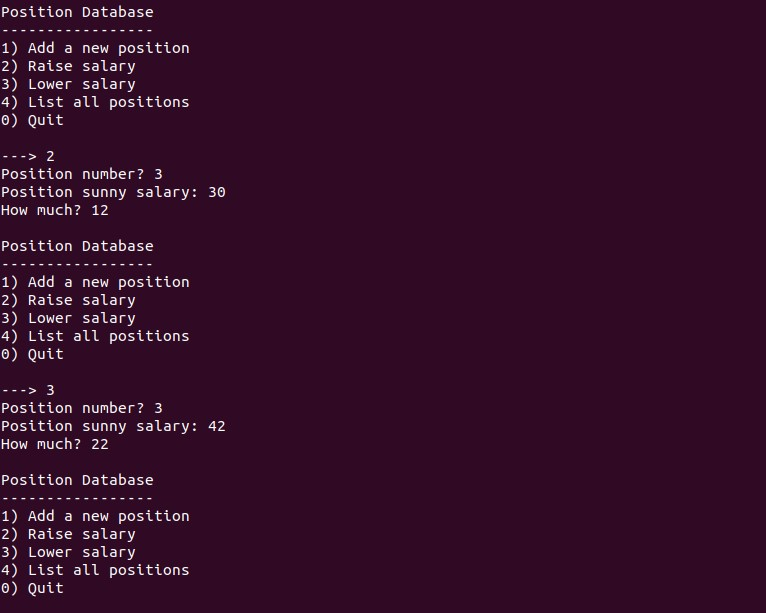
%.o: %.cpp

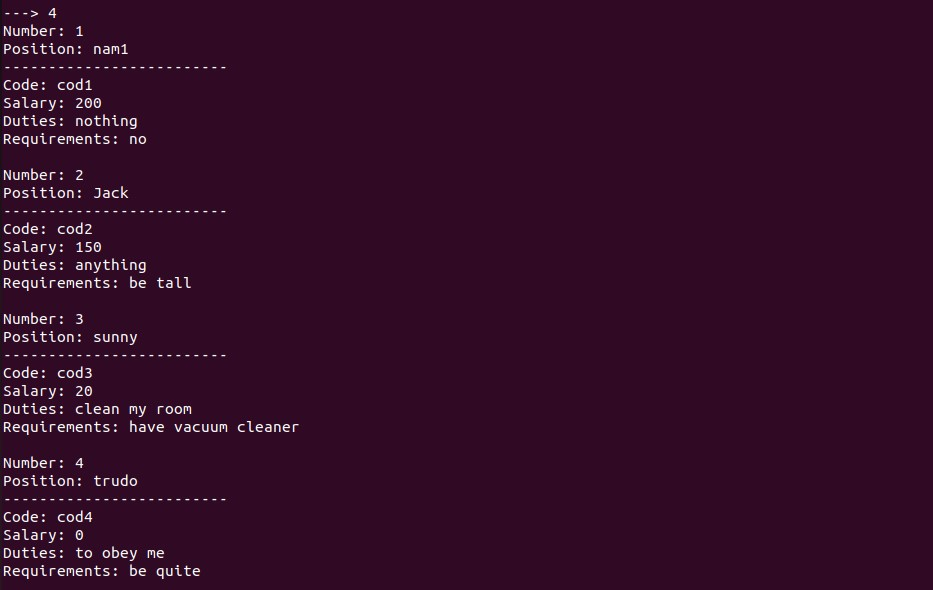
$(CC) $(CCFLAGS) -c $< -o $@

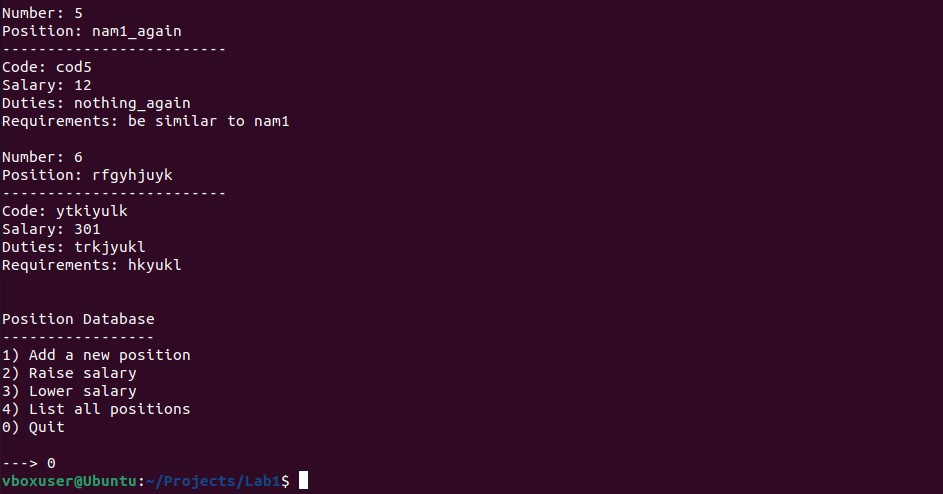
**Скріншоти виконання:**

Клас на стеку:

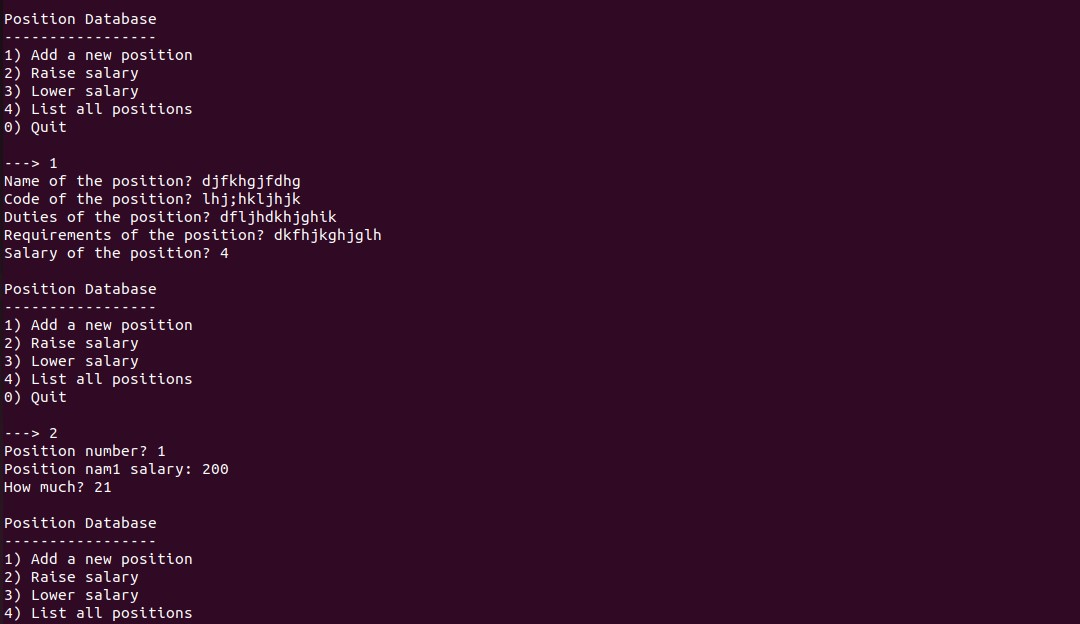
Рис.1 — Інтерфейс, додавання нової посади та невдала спроба підвищити зарплатню посаді, якої нема в БД.

Рис.2 — Підвищення та зменшення заробітних плат існуючим посадам.

Рис.3 — Виведення списку посад після внесених змін.

Рис.4 — Продовження списку посад та завершення виконання.

Клас на купі:

Рис.5 - Інтерфейс, додавання нової посади та підвищення зарплатні.

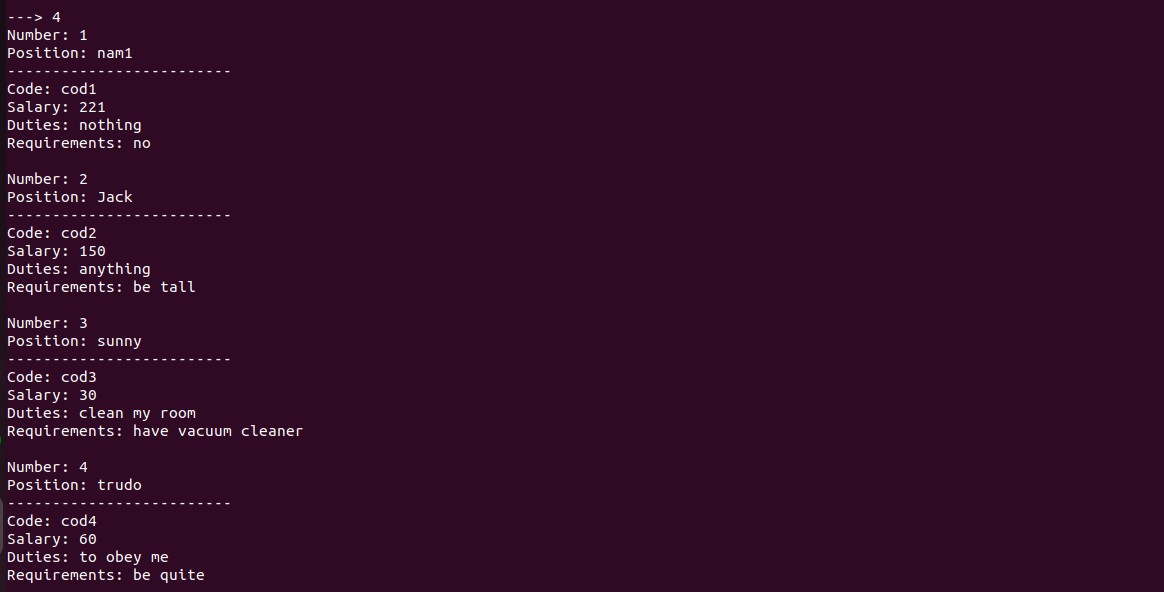
Рис.6 — Пониження зарплатні посаді.

Рис.7 - Виведення списку посад після внесених змін.

Рис.8 — Продовження списку посад та завершення виконання програми.

**Висновок:** Під час виконання лабораторної роботи, я набув навичок та знань з технології розробки ПЗ з використанням мови С++ для POSIX сумісних операційних систем. Було створено код для класів тематичної спрямованості та екземпляри класів, що були протестовані на купі пам’яті та стеку. Для компіляції було застосовано утиліту Makefile.