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 №2 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 №2**

**Створення статичних і динамічних бібліотек для POSIX сумісних (сертифікованих) операційних систем**

Мета роботи полягає у оволодінні практичними навичками розробки ПО на системному рівні з використанням бібліотек мови С (C++) для POSIX OS, на прикладі ОС Linux та автоматизації розробки з використанням Makefile статичних і динамічних бібліотек.

Другою метою є ознайомлення з варіантами реалізації алгоритмів у процедурному і ООП парадигмах з використанням мови С (C++).

**Завдання:**

1. Розробити консольний додаток у парадигмі ООП, що виконує наступні функції:

- додавання “Робітника” до інформаційній системи або іншу інформаційну сутність за спрямованістю, що вказана у ЛР1 і за результатами ЛР1, далі просто “Робітника”;

- видалення “Робітника” зі штату, при цьому він має залишатися у інформаційній системи, як минулий;

- збільшення (зменшення) зарплатні “Робітника”, або іншу аналогічну інформаційну сутність за тематичною за спрямованістю, що вказана у ЛР1 відповідно до варіанту;

- зображення інформації про “Робітника”, або іншу аналогічну інформаційну сутність за тематичною за спрямованістю, що вказана у ЛР1 відповідно до варіанту.

Для розробки скористатися вихідним кодом та інформацією, що вказана у [5].

Реалізація вихідного коду показана у додатку 2.

2. Створити два репозіторія, користуючись інформацією [1]. У одному програма має бути розділена на основний коді і статичні бібліотеки. У іншому репозиторії розробити динамічні бібліотеки.

3. За бажанням можна реалізувати функціональність п.1 у чистому стилі ООП поза шаблоном [5] самостийно. Або реалізувати у процедурному варіанті. Результатом роботи мають бути 2 репозиторія. У першому 5 файлів вихідного коду відповідно додатку 1 і Makefile для створення динамічних бібліотек. У другому репозиторії 5 файлів вихідного коду відповідно додатку 1 і Makefile для створення статичних бібліотек.

**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 actualise();

void deactualise();

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();

bool getIsActual();

private:

string pCode;

string pName;

int pSalary;

string pDuties;

string pRequirements;

int pNumber;

bool fActual;

};

}

**Position.cpp**

#include <iostream>

#include <string>

#include "Position.h"

using namespace std;

namespace Records {

Position::Position()

{

pCode = "";

pName = "";

pSalary = 0;

pDuties = "";

pRequirements = "";

fActual = false;

}

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::actualise()

{

fActual = true;

}

void Position::deactualise()

{

fActual = false;

}

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 << (fActual ? "It is actual" : "It is not actual") << 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;

}

bool Position::getIsActual()

{

return fActual;

}

}

**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();

void displayActual();

void displayUnactual();

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++);

thePosition.actualise();

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();

}

}

void Database::displayActual()

{

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

if (mPosition[i].getIsActual()) {

mPosition[i].display();

}

}

}

void Database::displayUnactual()

{

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

if (!mPosition[i].getIsActual()) {

mPosition[i].display();

}

}

}

}

**RadioCon.cpp**

#include <iostream>

#include <stdexcept>

#include <string>

#include "Database.h"

using namespace std;

using namespace Records;

int displayMenu();

void doAdd(Database& inDB);

void doDeactualise(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:

doDeactualise(\*positionDB);

break;

case 3:

doRaise(\*positionDB);

break;

case 4:

doLower(\*positionDB);

break;

case 5:

positionDB->displayAll();

break;

case 6:

positionDB->displayActual();

break;

case 7:

positionDB->displayUnactual();

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:

doDeactualise(positionDB);

break;

case 3:

doRaise(positionDB);

break;

case 4:

doLower(positionDB);

break;

case 5:

positionDB.displayAll();

break;

case 6:

positionDB.displayActual();

break;

case 7:

positionDB.displayUnactual();

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) Deactualise the position" << endl;

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

cout << "4) Lower salary" << endl;

cout << "5) List all positions" << endl;

cout << "6) List all actual positions" << endl;

cout << "7) List all unactual 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 doDeactualise(Database& inDB)

{

int positionNumber;

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);

emp.deactualise();

}

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 для програми із статичними бібліотеками**

CC = g++

CFLAGS = -Wall -std=c++11

SOURCES = Position.cpp Database.cpp RadioCon.cpp

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

LIBS = libDatabase.a libPosition.a

TARGET = labtwostat

all: $(TARGET)

$(TARGET): $(OBJS) $(LIBS)

$(CC) $(CFLAGS) $^ -o $@

libDatabase.a: Database.o

ar rcs $@ $^

libPosition.a: Position.o

ar rcs $@ $^

%.o: %.cpp

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

clean:

rm -f $(TARGET) $(LIBS) $(OBJS)

**Makefile для програми із динамічними бібліотеками**

CC = g++

CFLAGS = -Wall -std=c++11 -fPIC

SOURCES = Position.cpp Database.cpp RadioCon.cpp

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

LIBS = libDatabase.so libPosition.so

TARGET = labtwodyn

all: $(TARGET)

$(TARGET): $(OBJS) $(LIBS)

$(CC) $(CFLAGS) $^ -o $@

libDatabase.so: Database.o

$(CC) -shared -o $@ $^

libPosition.so: Position.o

$(CC) -shared -o $@ $^

%.o: %.cpp

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

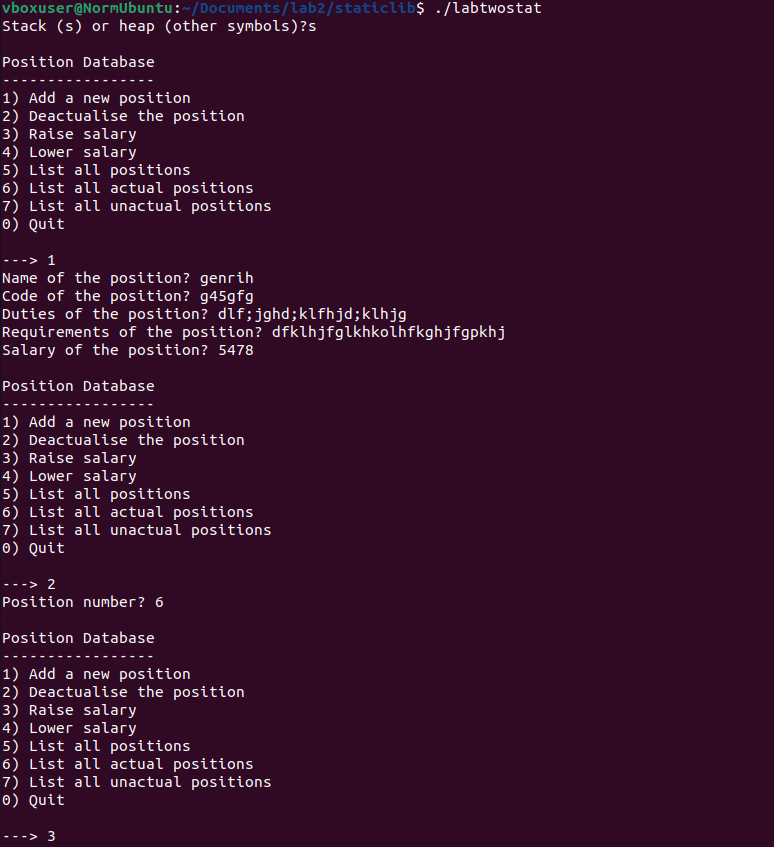
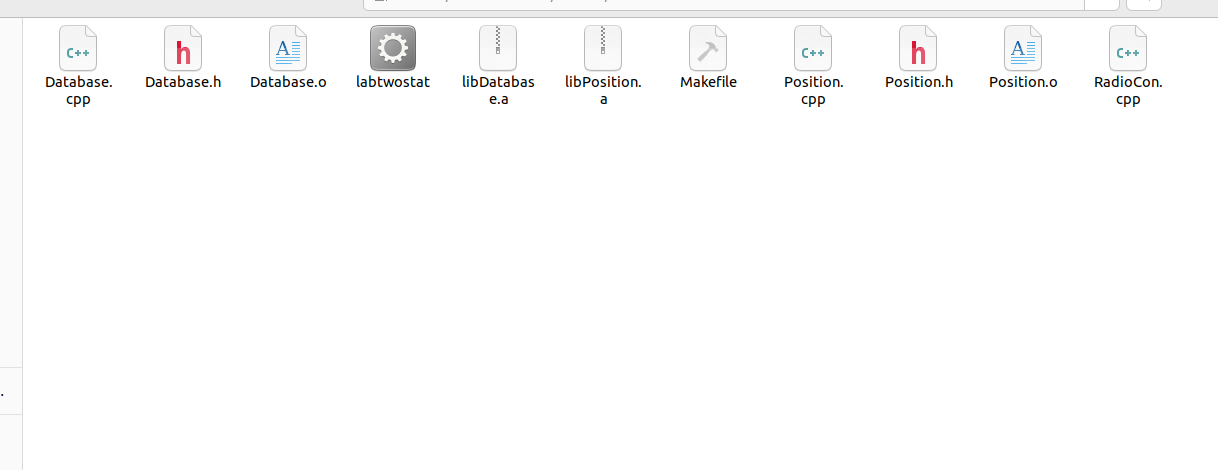
clean:

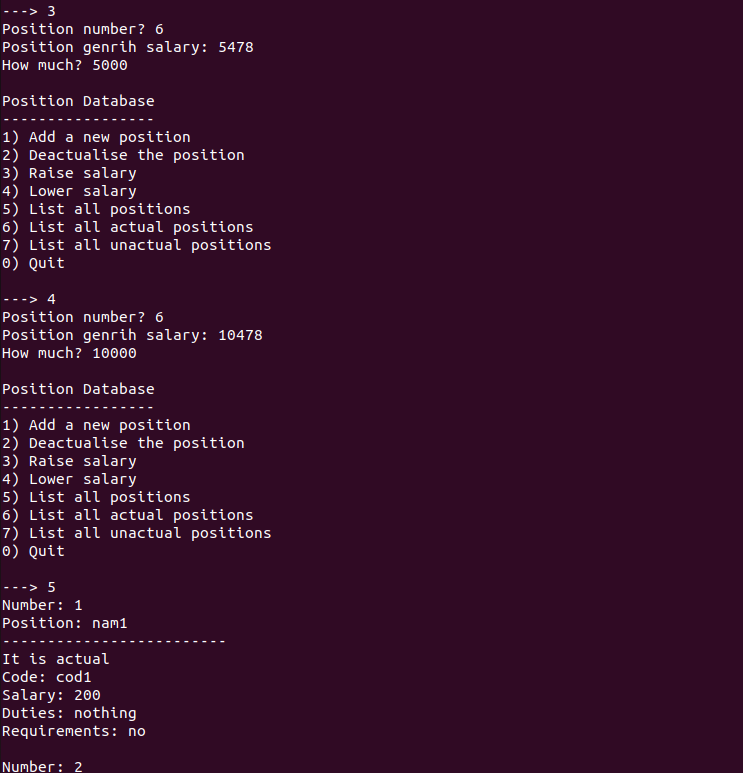
rm -f $(TARGET) $(LIBS) $(OBJS)

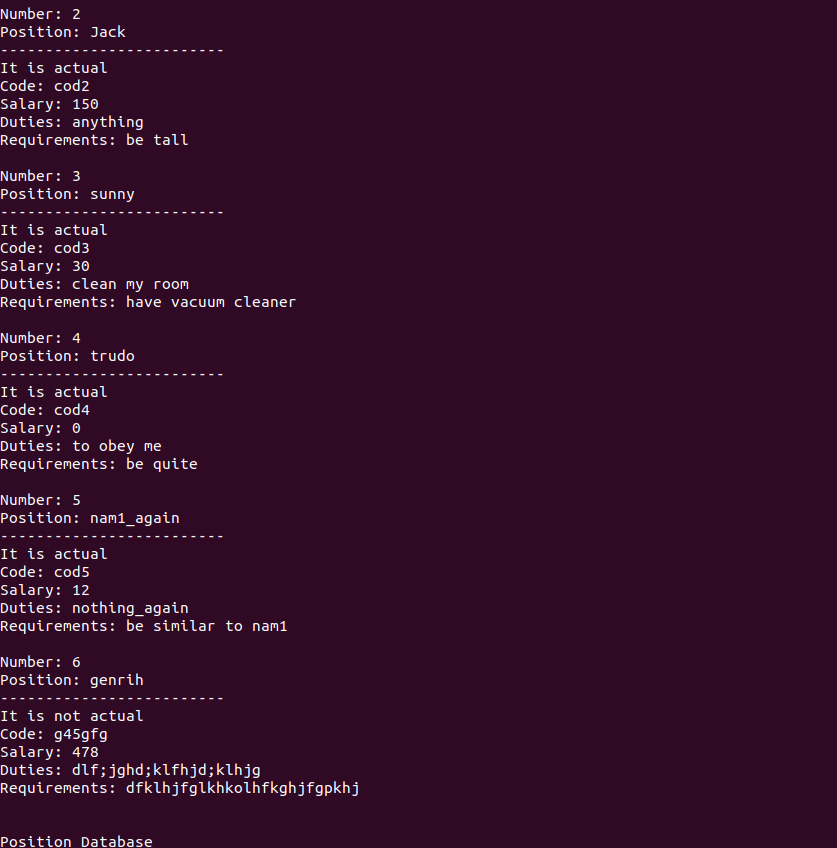
**Скріншоти виконання:**

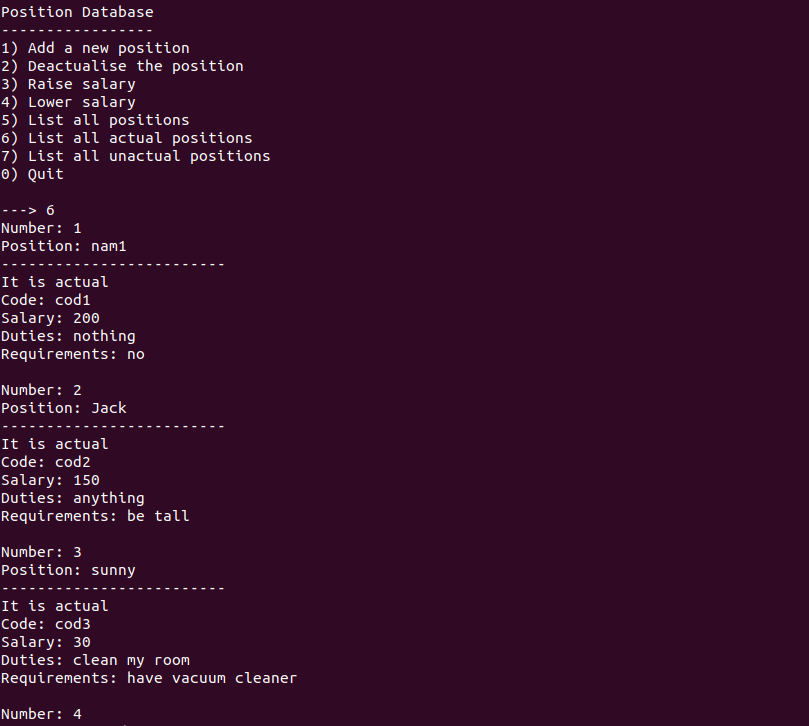
Клас на стеку, статичні бібліотеки:

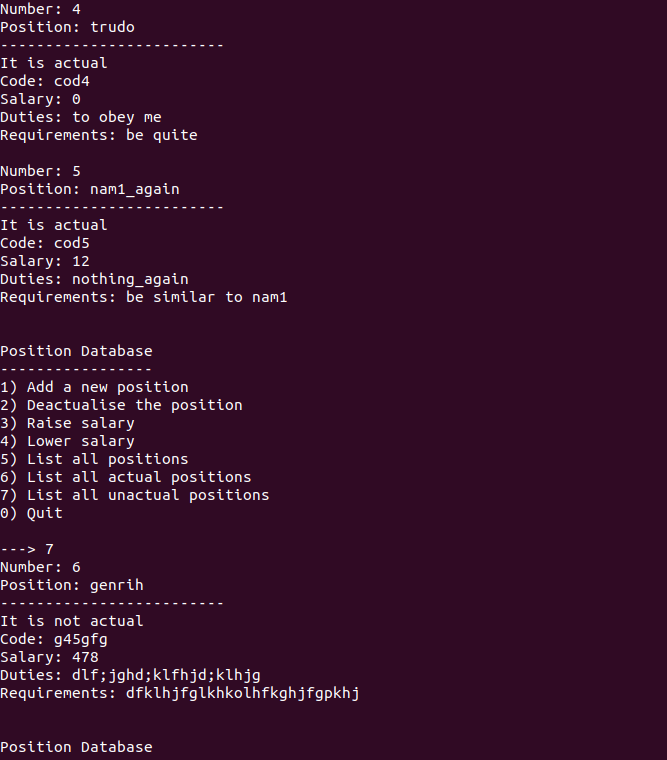
Ітогові файли:

Результат виконання:





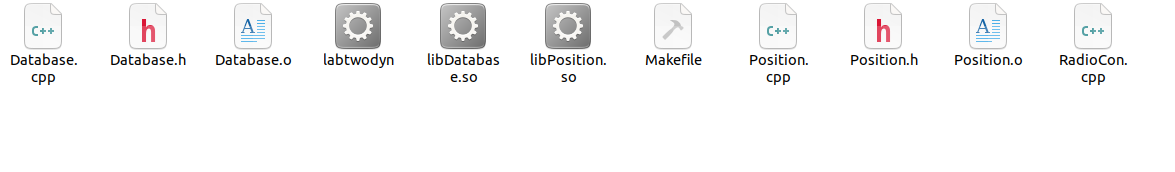




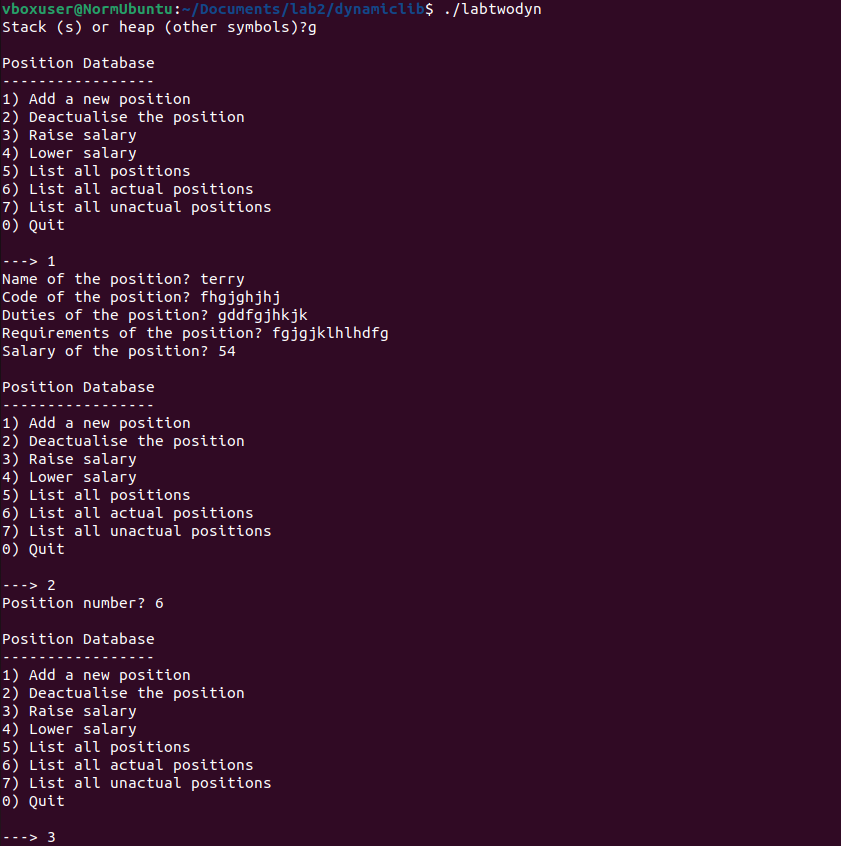


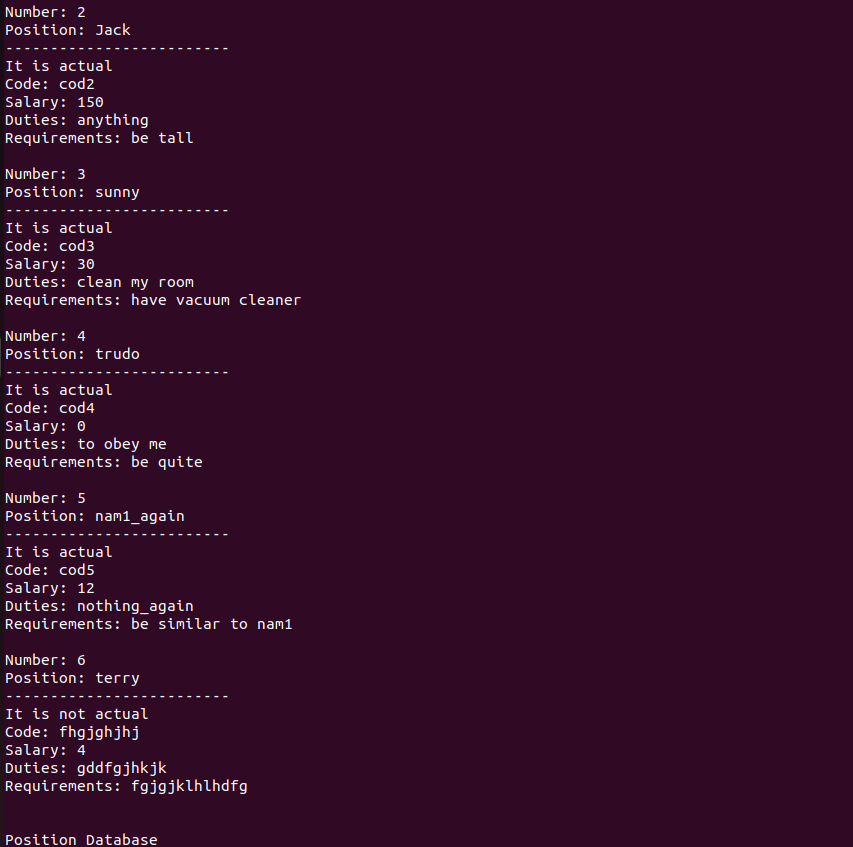
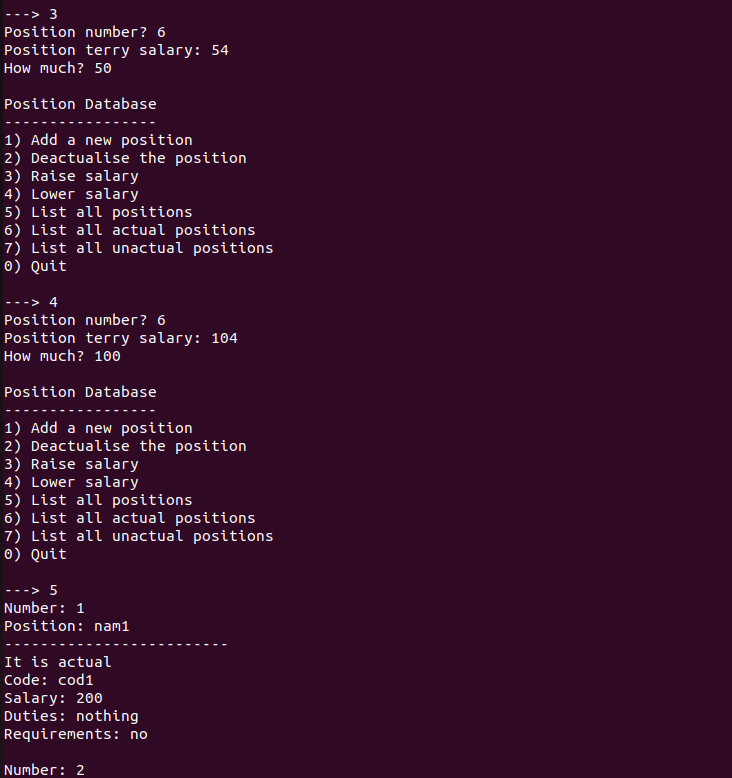
Клас на купі, динамічні бібліотеки:

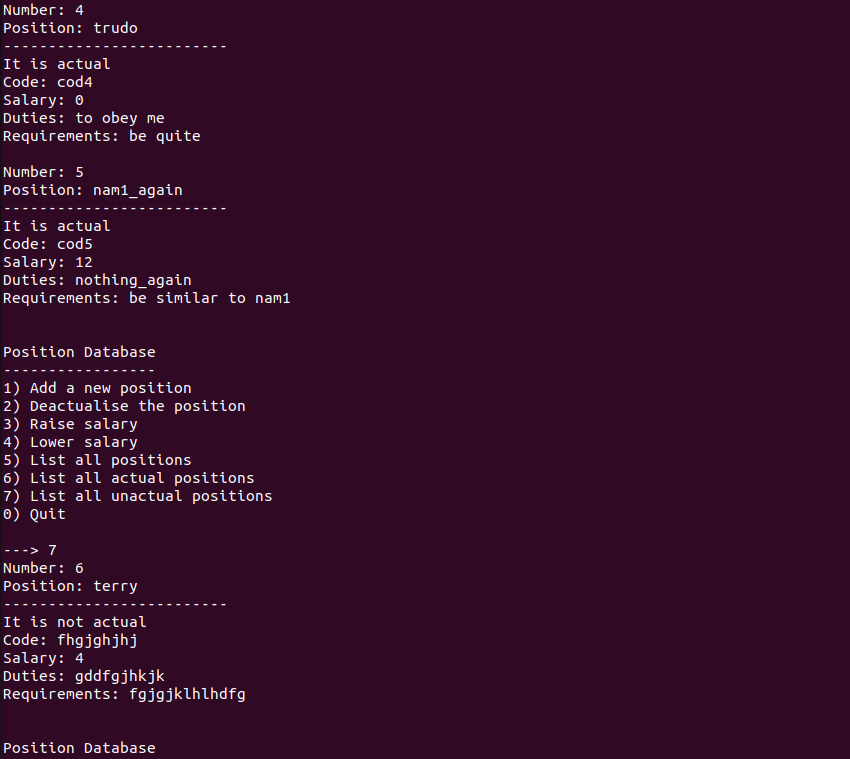
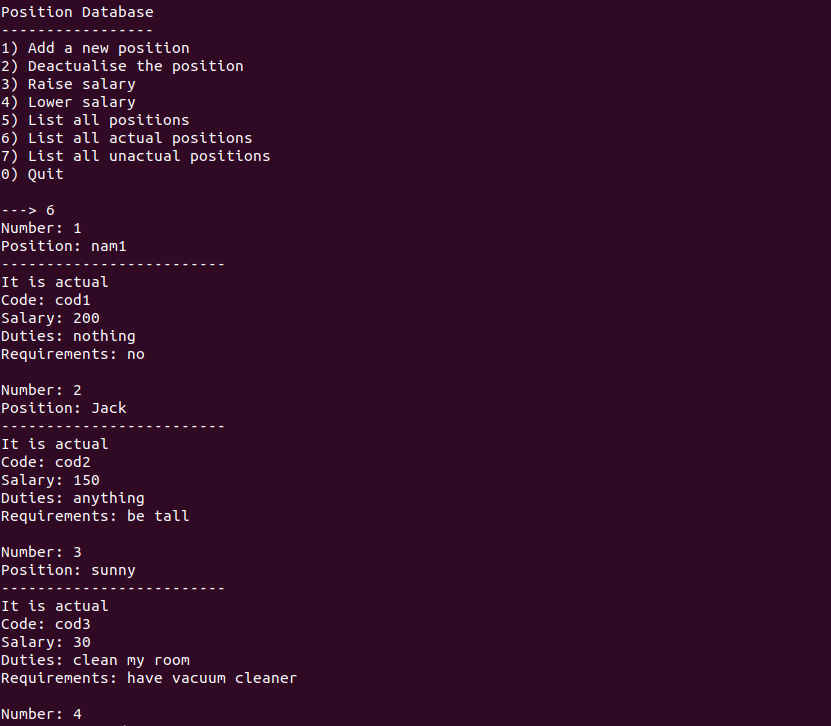
Ітогові файли:



Результат виконання:









**Висновок:** Під час виконання лабораторної роботи, я набув навичок та знань з створення статичних та динамічних бібліотек за допомогою Makefile. Було створено код для класів тематичної спрямованості та екземпляри класів, що були протестовані на купі пам’яті та стеку; статичні та динамічні бібліотеки. Для компіляції було застосовано утиліту Makefile.