МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ

УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Лабораторна робота №2

з дисципліни: «Системне програмування»

ТЕХНОЛОГІЯ СТВОРЕННЯ СТАТИЧНИХ І ДИНАМІЧНИХ БІБЛІОТЕК

Варіант 1

СПЕЦІАЛЬНОСТІ

121 – Інженерія програмного забезпечення

СПЕЦІАЛІЗАЦІЇ

Інженерія програмного забезпечення комп’ютерних систем

Виконали:

Балукова В. Кушніренко Ю. Курдій Д.

Кількість балів:\_\_\_\_\_\_\_ Оцінка \_\_\_\_\_\_

Група: ІТ-93

Викладач: Катін П.Ю.

Підпис:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2021

**Мета лабораторної роботи**

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.

**Завдання лабораторної роботи**

1. Реалізувати лабораторну роботу з використанням Makefile, GCC для POSIX OS. У якості вихідних даних використовуються результати попередньої роботи. Програма будується у вигляді інформаційної системи.

2. Створити два репозиторії. У одному програма має бути розділена на основний коді статичні бібліотеки. У іншому репозиторії розробити динамічні бібліотеки.

3. Користуючись результатами ЛР1 створити репозиторій, що працює зі статичними бібліотеками, з відповідним Makefile, показати його функціональність.

4. Користуючись результатами ЛР2 створити репозиторій, що працює з динамічними бібліотеками, з відповідним Makefile, показати його функціональність.

5. Реалізувати функціональність запису і читання інформаційної системи до файлу.

**Етапи виконання лабораторної роботи**

Makefile

CCX = g++  
MAIN := main\_db  
MAIN\_DB := main\_db  
MAIN\_NAME := UI  
MAIN\_NAME\_DB := UI  
BUILD\_DIR := ./buildDir  
STATIC\_LIB := lib\_static  
DYNAMIC\_LIB := lib\_dynamic  
  
SRC := Worker.cpp DataBase.cpp  
OBJS := $(SRC:%=$(BUILD\_DIR)/%.o)  
  
collect\_static : $(BUILD\_DIR)/$(MAIN)\_s  
  
# collecting main and static library  
$(BUILD\_DIR)/$(MAIN)\_s : $(BUILD\_DIR)/$(MAIN\_NAME).o $(BUILD\_DIR)/$(STATIC\_LIB).a  
 @$(CCX) -o $@ $< -L$(BUILD\_DIR) -l\_static  
  
# compiling main file  
$(BUILD\_DIR)/$(MAIN\_NAME).o: $(MAIN\_NAME).cpp  
 @$(CXX) -c $< -o $@  
  
# creating a static library:  
$(BUILD\_DIR)/$(STATIC\_LIB).a : $(OBJS)  
 @ar crs $@ $^  
  
collect\_dynamic : $(BUILD\_DIR)/$(MAIN)\_d  
# collecting main and dynamic library  
$(BUILD\_DIR)/$(MAIN)\_d : $(MAIN\_NAME).cpp $(BUILD\_DIR)/$(DYNAMIC\_LIB).so  
 @$(CCX) $< -o $@ -L $(BUILD\_DIR) -l\_dynamic  
  
# creating a dynamic library:  
$(BUILD\_DIR)/$(DYNAMIC\_LIB).so : $(OBJS)  
 @$(CCX) $^ -shared -o $@ -lc  
  
# compile files  
$(BUILD\_DIR)/%.cpp.o : %.cpp  
 @$(CXX) -c -fPIC $< -o $@  
  
  
clean:  
 @rm -f $(BUILD\_DIR)/\*.o $(BUILD\_DIR)/\*.a $(BUILD\_DIR)/\*.so $(BUILD\_DIR)/$(MAIN)  
 @rm $(BUILD\_DIR)/$(MAIN)\_s  
 @rm $(BUILD\_DIR)/$(MAIN)\_d  
  
clean\_src:  
 @rm -f \*.o \*.a \*.so  
  
run\_static:  
 @./$(BUILD\_DIR)/$(MAIN)\_s  
  
run\_dynamic:  
 @./$(BUILD\_DIR)/$(MAIN)\_d

Worker.cpp

#include <iostream>  
#include "Worker.h"  
  
using namespace std;  
  
Worker::Worker()  
{  
 // initialize data members  
 mPIB = "Unknown Worker";  
 mAge = 0;  
 mWorkerCode = 0;  
 mSex = "Unknown";  
 mPassport = "Unknown";  
 mPositionCode = 0;  
 mAddress = " Unknown";  
  
}  
  
Worker::~Worker()  
{  
 // nothing much to do in terms of cleanup  
}  
  
int Worker::calculateSalaryInDollars()  
{  
 if (getPositionCode() == 0) {  
  
 return 700;  
 }  
   
   
 return static\_cast<int>((getPositionCode() \* 500));  
}  
  
int Worker::getWorkerCode()  
{  
 return mWorkerCode;  
}  
  
void Worker::setWorkerCode(int inCode)  
{  
 mWorkerCode = inCode;  
}  
int Worker::getPositionCode()  
{  
 return mPositionCode;  
}  
  
void Worker::setPositionCode(int inCode)  
{  
 mPositionCode = inCode;  
}  
  
string Worker::getPIB()  
{  
 return mPIB;  
}  
  
void Worker::setPIB(string inPIB)  
{  
 mPIB = inPIB;  
}  
string Worker::getSex()  
{  
 return mSex;  
}  
  
void Worker::setSex(string inSex)  
{  
 mSex = inSex;  
}  
  
  
int Worker::getAge()  
{  
 return mAge;  
}  
  
void Worker::setAge(int inAge)  
{  
 mAge = inAge;  
}  
  
string Worker::getAddress()  
{  
 return mAddress;  
}  
  
void Worker::setAddress(string inAddress)  
{  
 mAddress = inAddress;  
}  
  
  
string Worker::getPassport()  
{  
 return mPassport;  
}  
  
void Worker::setPassport(string inPassport)  
{  
 mPassport = inPassport;  
}  
  
void Worker::show() {  
 cout << "Worker Info" << endl  
 <<"Worker code: " << getWorkerCode() << endl  
 <<"PIB: " << getPIB() << endl  
 <<"SEX: " << getSex() << endl  
 <<"Age: " << getAge() << endl  
 <<"Position Code: " << getPositionCode() << endl  
 <<"Address: " << getAddress() << endl  
 <<"Passport: " << getPassport() << endl  
 <<"Salary in dollars: " << calculateSalaryInDollars()   
 << endl;  
  
}

Worker.h

#include <string>  
class Worker  
{  
 public:  
 Worker();  
 ~Worker();  
  
 int calculateSalaryInDollars();  
  
 int getWorkerCode();  
 void setWorkerCode(int code);  
 std::string getPIB();  
 void setPIB(std::string PIB);  
 std::string getSex();  
 void setSex(std::string sex);  
 int getAge();  
 void setAge(int age);  
 int getPositionCode();  
 void setPositionCode(int code);  
 std::string getAddress();  
 void setAddress(std::string address);  
 std::string getPassport();  
 void setPassport(std::string passport);  
 void show();  
 private:  
 int mWorkerCode;  
 std::string mPIB;  
 std::string mSex;  
 int mAge;  
 int mPositionCode;  
 std::string mAddress;  
 std::string mPassport;  
};

DataBase.cpp

#include <iostream>  
#include <stdexcept>  
#include "DataBase.h"  
#include <fstream>  
using namespace std;  
DataBase::DataBase() {  
 Worker worker;  
 worker.setPIB("Worker1");  
 worker.setAge(20);  
 worker.setAddress("Address1");  
 worker.setPIB("Passport1");  
 worker.setPositionCode(0);  
 worker.setSex("Male");  
 worker.setWorkerCode(this->nextWorkerCode++);  
 Workers.push\_back(worker);  
  
 Worker worker2;  
 worker2.setPIB("Worker2");  
 worker2.setAge(20);  
 worker2.setAddress("Address2");  
 worker2.setPIB("Passport2");  
 worker2.setPositionCode(1);  
 worker2.setSex("Female");  
 worker2.setWorkerCode(this->nextWorkerCode++);  
 Workers.push\_back(worker2);  
  
 Worker worker3;  
 worker3.setPIB("Worker3");  
 worker3.setAge(18);  
 worker3.setAddress("Address3");  
 worker3.setPIB("Passport3");  
 worker3.setPositionCode(0);  
 worker3.setSex("Male");  
 worker3.setWorkerCode(this->nextWorkerCode++);  
 Workers.push\_back(worker3);  
};  
DataBase::~DataBase() {};  
  
Worker& DataBase::addWorker(const string& pib, const int& positionCode){  
 Worker\* worker = new Worker();  
 worker->setPIB(pib);  
 worker->setPositionCode(positionCode);  
 worker->setWorkerCode(this->nextWorkerCode++);  
 Workers.push\_back(\*worker);  
 return \*worker;  
}  
 Worker& DataBase::getWorker(const string& pib, const int& positionCode) {  
 for(Worker& Worker : this->Workers) {  
 if(Worker.getPIB() == pib && Worker.getPositionCode() == positionCode) return Worker;  
 }  
  
}  
 Worker& DataBase::getWorker(const int& WorkerCode) {  
 for(Worker& Worker : this->Workers) {  
 if(Worker.getWorkerCode() == WorkerCode) return Worker;  
 }  
}  
  
void DataBase::displayAll() const {  
 for(Worker Worker : this->Workers) {  
 Worker.show();  
 }  
}  
void DataBase::displayWithPositionCode(const int& positionCode) const {  
 for(Worker Worker : this->Workers) {  
 if(Worker.getPositionCode() == positionCode)   
 Worker.show();  
 }  
}

DataBase.h

#pragma once  
#include "Worker.h"  
#include <string>  
#include <vector>  
  
using namespace std;  
class DataBase {  
 public:  
 DataBase();  
  
 ~DataBase();  
 Worker& addWorker(const string& pib, const int& positionCode);  
 Worker& getWorker(const int& WorkerCode) ;  
 Worker& getWorker(const string& pib, const int& positionCode) ;  
 void displayAll() const;  
 void displayWithPositionCode(const int& positionCode) const;  
  
 private:  
 int nextWorkerCode = 0;  
 vector<Worker> Workers;  
};

DataBaseTest.cpp

#include "DataBase.h"  
#include <iostream>  
  
using namespace std;  
  
int main() {  
 DataBase\* my\_db = new DataBase();  
 cout<<"Test Start"<<endl;  
 my\_db->displayAll();  
 cout<<"Test End"<<endl;  
  
}

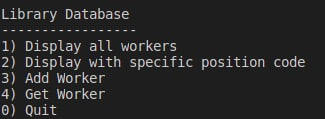
LibraryTest.cpp

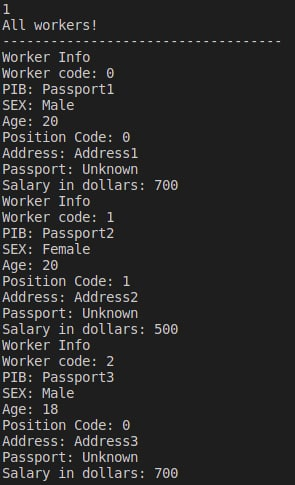
#include <iostream>  
#include "Worker.h"  
  
using namespace std;  
  
int main(int argc, char\*\* argv)  
{  
 Worker WorkerStack;  
 Worker\* WorkerHeap;  
 WorkerHeap = new Worker();  
  
 WorkerStack.setAge(20);  
 WorkerStack.setPIB("John Brown");  
 WorkerStack.setSex("male");  
 WorkerStack.setAddress("Kyiv");  
 WorkerStack.setPositionCode(1);  
  
 WorkerHeap->setAge(29);  
 WorkerHeap->setPIB("Alice White");  
 WorkerHeap->setSex("female");  
 WorkerHeap->setAddress("Kyiv");  
  
 std::cout << WorkerStack.getPIB() << " "  
 << WorkerStack.getAge() << " "  
 << WorkerStack.getSex() << " "  
 << WorkerStack.getAddress() << " "  
 << WorkerStack.getPositionCode() << " "  
 << WorkerStack.calculateSalaryInDollars()  
 << std::endl;  
  
  
 std::cout << WorkerHeap->getPIB() << " "  
 << WorkerHeap->getAge() << " "  
 << WorkerHeap->getSex() << " "  
 << WorkerHeap->getAddress() << " "  
 << WorkerStack.getPositionCode() << " "  
 << WorkerHeap->calculateSalaryInDollars()   
 << std::endl;  
  
 delete WorkerHeap;  
  
 return 0;  
}

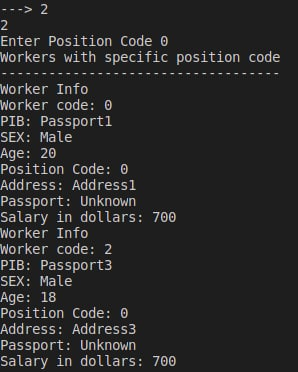
UI.cpp

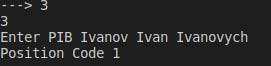
#include <iostream>  
#include <stdexcept>  
#include <exception>  
#include "DataBase.h"  
  
using namespace std;  
  
int displayMenu();  
void displayAll(DataBase\* db);  
void displayWithPositionCode(DataBase\* db);  
void addWorker(DataBase\* db);  
void getWorker(DataBase\* db);  
  
int main()  
{  
 DataBase\* marriageDB = new DataBase();  
 bool done = false;  
 while (!done) {  
 int selection = displayMenu();  
 cout<<selection<<endl;  
 switch (selection) {  
 case 0:  
 done = true;  
 break;  
 case 1:  
 displayAll(marriageDB);  
 break;  
 case 2:  
 displayWithPositionCode(marriageDB);  
 break;  
 case 3:  
 addWorker(marriageDB);  
 break;  
 case 4:  
 getWorker(marriageDB);  
 break;  
 default:  
 cerr << "Unknown command." << endl;  
 break;  
 }  
 }  
 return 0;  
}  
  
int displayMenu()  
{  
 int selection;  
 cout << endl;  
 cout << "Library Database" << endl;  
 cout << "-----------------" << endl;  
 cout << "1) Display all workers" << endl;  
 cout << "2) Display with specific position code " << endl;  
 cout << "3) Add Worker" << endl;  
 cout << "4) Get Worker" << endl;  
 cout << "0) Quit" << endl;  
 cout << endl;  
 cout << "---> ";  
 cin >> selection;  
 return selection;  
}  
  
void displayAll(DataBase\* db)  
{  
 cout << "All workers! "<<endl;  
 cout << "-----------------------------------"<<endl;  
 db->displayAll();  
}  
  
void displayWithPositionCode(DataBase\* db)  
{  
 int positionCode = 0;  
 cout<<"Enter Position Code ";  
 cin>>positionCode;  
 cout << "Workers with specific position code "<<endl;  
 cout << "-----------------------------------"<<endl;  
 db->displayWithPositionCode(positionCode);  
}  
  
  
void addWorker(DataBase\* db)  
{  
 string pib;  
 int positionCode = 0;  
 cout << "Enter PIB ";  
 getline(cin, pib);  
 getline(cin, pib);  
 cout << "Position Code ";  
 cin >> positionCode;  
 db->addWorker(pib, positionCode);  
}  
  
void getWorker(DataBase\* db)  
{  
 int WorkerCode;  
 cout << "Enter Worker Code ";  
 cin >> WorkerCode;  
 db->getWorker(WorkerCode).show();  
}

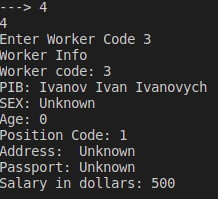
Виконаємо команди make collect\_static та make run\_static:

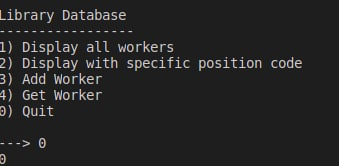












**Висновок**

На цій лабораторній роботі ми навчилися створювати та використовувати статичні (.a) та динамічні (.so) бібліотеки. Це допомогло розвинути навики роботи з Makefile та gcc (g++). Створили застосунок, який зберігає дані у файлі, і може їх зчитувати, тобто, стан зберігається між запусками. Розробили консольний інтерфейс користувача для програми.