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 №3 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 №3**

**Розробка графічного інтерфейсу з використанням бібліотеки Qt длястандартних потоків вводу-виводу**

Мета роботи полягає у оволодінні методикою розробки графічного інтерфейсу і вдосконалення роботи з файлами і потоками введення-виведення. Ознайомитися з різними способами роботи з файлами. Навчитися програмно керувати доступом до файлів (читати та записувати дані). Ознайомитися з файловим діалогом. За бажанням додати його. До існуючого проекту додати можливість працювати з файлами.

**Завдання:**

1. Встановити та налаштувати Qt Creator. Створення вихідного коду. Зберегти його у робочому каталозі, що містить програмну. Доробити вихідний код відповідно до поданого завдання.

2. Створити новий проект з графічним інтерфейсом. У разі появи помилок внести зміни до вихідного коду.

3. Після успішного виконання п. 1, 2 запустити програму на виконання.

4. Реалізувати графічний інтерфейс і розробити інформаційну систему відповідно до варіанту. Після покрокового виконання програми зробити скриншоти екранів і створити звіт, додаючи до звіту скриншоти екранів.

5. Зробити висновки щодо особливостей виконання лабораторної роботи, наявності специфічних особливостей роботи програми.

**Код програми:**

**main.cpp:**

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

**mainwindow.h:**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QLineEdit>

#include <QTextEdit>

#include <QLabel>

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void addButton();

void retireButton();

void raiseButton();

void lowerButton();

void allButton();

void actualButton();

void unactualButton();

private:

QTextEdit \*outputField;

QLineEdit \*inputCode;

QLineEdit \*inputName;

QLineEdit \*inputSalary;

QLineEdit \*inputDuties;

QLineEdit \*inputRequirements;

QLineEdit \*inputPosition;

QLineEdit \*inputAmount;

};

#endif // MAINWINDOW\_H

**mainwindow.cpp:**

#include "mainwindow.h"

#include <QPushButton>

#include <QFile>

#include <QTextStream>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

{

setWindowTitle("App 1");

setGeometry(100, 100, 500, 570);

setMinimumSize(500, 570);

outputField = new QTextEdit("", this);

outputField->setGeometry(180, 10, 310, 270);

outputField->setReadOnly(true);

outputField->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOn);

QPushButton \*addButton = new QPushButton("Add new position", this);

addButton->setGeometry(10, 10, 160, 30);

connect(addButton, SIGNAL (released()), this, SLOT (addButton()));

QPushButton \*retireButton = new QPushButton("Unactualise position", this);

retireButton->setGeometry(10, 50, 160, 30);

connect(retireButton, SIGNAL (released()), this, SLOT (retireButton()));

QPushButton \*raiseButton = new QPushButton("Raise salary", this);

raiseButton->setGeometry(10, 90, 160, 30);

connect(raiseButton, SIGNAL (released()), this, SLOT (raiseButton()));

QPushButton \*lowerButton = new QPushButton("Lower salary", this);

lowerButton->setGeometry(10, 130, 160, 30);

connect(lowerButton, SIGNAL (released()), this, SLOT (lowerButton()));

QPushButton \*allButton = new QPushButton("Show all", this);

allButton->setGeometry(10, 170, 160, 30);

connect(allButton, SIGNAL (released()), this, SLOT (allButton()));

QPushButton \*actualButton = new QPushButton("Show actual", this);

actualButton->setGeometry(10, 210, 160, 30);

connect(actualButton, SIGNAL (released()), this, SLOT (actualButton()));

QPushButton \*unactualButton = new QPushButton("Show unactual", this);

unactualButton->setGeometry(10, 250, 160, 30);

connect(unactualButton, SIGNAL (released()), this, SLOT (unactualButton()));

inputCode = new QLineEdit(this);

inputCode->setGeometry(100, 290, 390, 30);

inputName = new QLineEdit(this);

inputName->setGeometry(100, 330, 390, 30);

inputSalary = new QLineEdit(this);

inputSalary->setGeometry(100, 370, 390, 30);

inputDuties = new QLineEdit(this);

inputDuties->setGeometry(100, 410, 390, 30);

inputRequirements = new QLineEdit(this);

inputRequirements->setGeometry(100, 450, 390, 30);

inputPosition = new QLineEdit(this);

inputPosition->setGeometry(100, 490, 390, 30);

inputAmount = new QLineEdit(this);

inputAmount->setGeometry(100, 530, 390, 30);

QLabel \*codeLabel = new QLabel("Position code", this);

codeLabel->setGeometry(10, 290, 80, 30);

QLabel \*nameLabel = new QLabel("Position name", this);

nameLabel->setGeometry(10, 330, 80, 30);

QLabel \*salaryLabel = new QLabel("Position salary", this);

salaryLabel->setGeometry(10, 370, 80, 30);

QLabel \*dutiesLabel = new QLabel("Position duties", this);

dutiesLabel->setGeometry(10, 410, 80, 30);

QLabel \*requirementsLabel = new QLabel("Requirements", this);

requirementsLabel->setGeometry(10, 450, 80, 30);

QLabel \*positionLabel = new QLabel("Number", this);

positionLabel->setGeometry(10, 490, 80, 30);

QLabel \*amountLabel = new QLabel("Amount", this);

amountLabel->setGeometry(10, 530, 80, 30);

}

void MainWindow::addButton()

{

outputField->clear();

QFile file("E:/Install/Projects/System programming/lab3/lab3/records.txt");

int recCount = 0;

if (file.open(QIODevice::ReadOnly | QIODevice::Text)){

QTextStream in(&file);

while(!in.atEnd()){

QString line = in.readLine();

recCount++;

}

file.close();

recCount /= 7;

}

else{

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream out(&file);

file.close();

}

if(recCount >=5){

outputField->setText("Limit of records is reached!");

}

else{

file.open(QIODevice::Append | QIODevice::Text);

QTextStream out(&file);

recCount++;

out << QString::number(recCount);

out << "\n";

out << inputCode->text();

out << "\n";

out << inputName->text();

out << "\n";

bool isDouble;

double number = inputSalary->text().toDouble(&isDouble);

if (isDouble){

if(number < 0){

out << "0";

}

else{

out << inputSalary->text();

}

}

else{

out << "0";

}

out << "\n";

out << inputDuties->text();

out << "\n";

out << inputRequirements->text();

out << "\n";

out << "true";

out << "\n";

file.close();

}

}

void MainWindow::retireButton()

{

outputField->clear();

bool isNum;

int pos = inputPosition->text().toInt(&isNum);

if(!isNum){

pos = 0;

}

QFile file("E:/Install/Projects/System programming/lab3/lab3/records.txt");

QFile helpfile("E:/Install/Projects/System programming/lab3/lab3/help.txt");

file.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in(&file);

helpfile.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream out(&helpfile);

QString line;

bool isFirst = true;

bool isThat = false;

bool isExec = false;

while(!in.atEnd()){

line = in.readLine();

if(line.toInt() == pos){

isThat = true;

}

else{

isThat = false;

}

if(!isFirst){

out << "\n";

}

isFirst = false;

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

if (isThat){

out << "false";

isExec = true;

}

else{

out << line;

}

}

file.remove();

file.close();

helpfile.rename("E:/Install/Projects/System programming/lab3/lab3/records.txt");

helpfile.close();

if(!isExec){

outputField -> setText("Record is not found!");

}

}

void MainWindow::raiseButton()

{

outputField->clear();

bool isNum;

int pos = inputPosition->text().toInt(&isNum);

if(!isNum){

pos = 0;

}

double amou = inputAmount->text().toDouble(&isNum);

if(!isNum){

amou = 0;

}

QFile file("E:/Install/Projects/System programming/lab3/lab3/records.txt");

QFile helpfile("E:/Install/Projects/System programming/lab3/lab3/help.txt");

file.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in(&file);

helpfile.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream out(&helpfile);

QString line;

bool isFirst = true;

bool isThat = false;

bool isExec = false;

double helpval = 0;

while(!in.atEnd()){

line = in.readLine();

if(line.toInt() == pos){

isThat = true;

}

else{

isThat = false;

}

if(!isFirst){

out << "\n";

}

isFirst = false;

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

if (isThat){

helpval = line.toDouble();

helpval += amou;

if(helpval < 0){

helpval = 0;

}

out << QString::number(helpval) << "\n";

isExec = true;

}

else{

out << line << "\n";

}

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line;

}

file.remove();

file.close();

helpfile.rename("E:/Install/Projects/System programming/lab3/lab3/records.txt");

helpfile.close();

if(!isExec){

outputField -> setText("Record is not found!");

}

}

void MainWindow::lowerButton()

{

outputField->clear();

bool isNum;

int pos = inputPosition->text().toInt(&isNum);

if(!isNum){

pos = 0;

}

double amou = inputAmount->text().toDouble(&isNum);

if(!isNum){

amou = 0;

}

QFile file("E:/Install/Projects/System programming/lab3/lab3/records.txt");

QFile helpfile("E:/Install/Projects/System programming/lab3/lab3/help.txt");

file.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in(&file);

helpfile.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream out(&helpfile);

QString line;

bool isFirst = true;

bool isThat = false;

bool isExec = false;

double helpval = 0;

while(!in.atEnd()){

line = in.readLine();

if(line.toInt() == pos){

isThat = true;

}

else{

isThat = false;

}

if(!isFirst){

out << "\n";

}

isFirst = false;

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

if (isThat){

helpval = line.toDouble();

helpval -= amou;

if(helpval < 0){

helpval = 0;

}

out << QString::number(helpval) << "\n";

isExec = true;

}

else{

out << line << "\n";

}

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line << "\n";

line = in.readLine();

out << line;

}

file.remove();

file.close();

helpfile.rename("E:/Install/Projects/System programming/lab3/lab3/records.txt");

helpfile.close();

if(!isExec){

outputField -> setText("Record is not found!");

}

}

void MainWindow::allButton()

{

outputField->clear();

QFile file("E:/Install/Projects/System programming/lab3/lab3/records.txt");

file.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in(&file);

QString allText;

bool isFirst = true;

QString line;

while(!in.atEnd()){

line = in.readLine();

if(!isFirst){

allText.append("\n\n");

}

isFirst = false;

allText.append("Position: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Code: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Name: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Salary: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Duties: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Requirements: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Position is ");

if (line.toStdString() == "true"){

allText.append("actual!");

}

else{

allText.append("unactual!");

}

}

file.close();

outputField -> setText(allText);

}

void MainWindow::actualButton()

{

outputField->clear();

QFile file("E:/Install/Projects/System programming/lab3/lab3/records.txt");

file.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in(&file);

QString allText;

QString totalText;

bool isFirst = true;

QString line;

while(!in.atEnd()){

allText.clear();

line = in.readLine();

allText.append("Position: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Code: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Name: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Salary: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Duties: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Requirements: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Position is ");

if (line.toStdString() == "true"){

allText.append("actual!");

if(!isFirst){

totalText.append("\n\n");

}

isFirst = false;

totalText.append(allText);

}

else{

allText.append("unactual!");

}

}

file.close();

outputField -> setText(totalText);

}

void MainWindow::unactualButton()

{

outputField->clear();

QFile file("E:/Install/Projects/System programming/lab3/lab3/records.txt");

file.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in(&file);

QString allText;

QString totalText;

bool isFirst = true;

QString line;

while(!in.atEnd()){

allText.clear();

line = in.readLine();

allText.append("Position: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Code: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Name: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Salary: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Duties: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Requirements: ");

allText.append(line);

allText.append("\n");

line = in.readLine();

allText.append("Position is ");

if (line.toStdString() == "false"){

allText.append("unactual!");

if(!isFirst){

totalText.append("\n\n");

}

isFirst = false;

totalText.append(allText);

}

else{

allText.append("actual!");

}

}

file.close();

outputField -> setText(totalText);

}

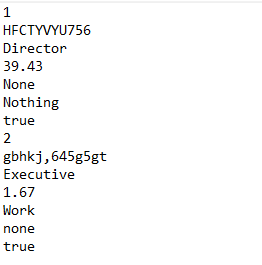
MainWindow::~MainWindow()

{

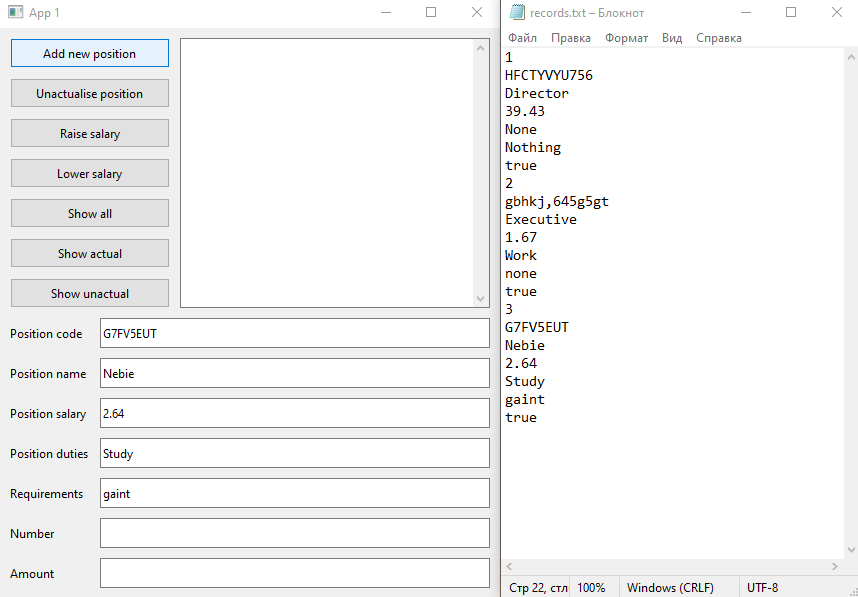
}

**Виконання:**

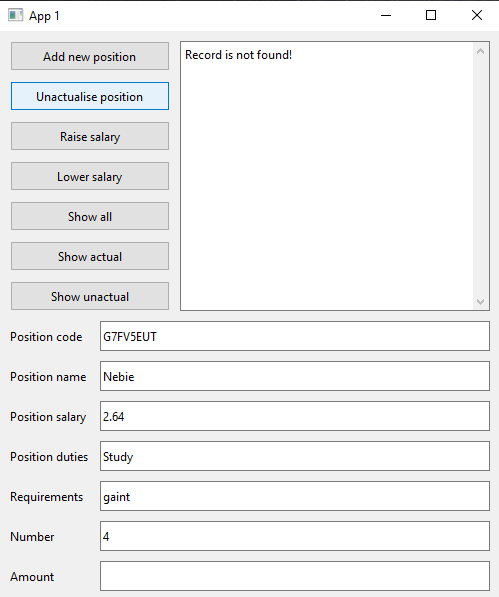
Нехай маємо пару записів у файлі, в якому зберігаються дані:



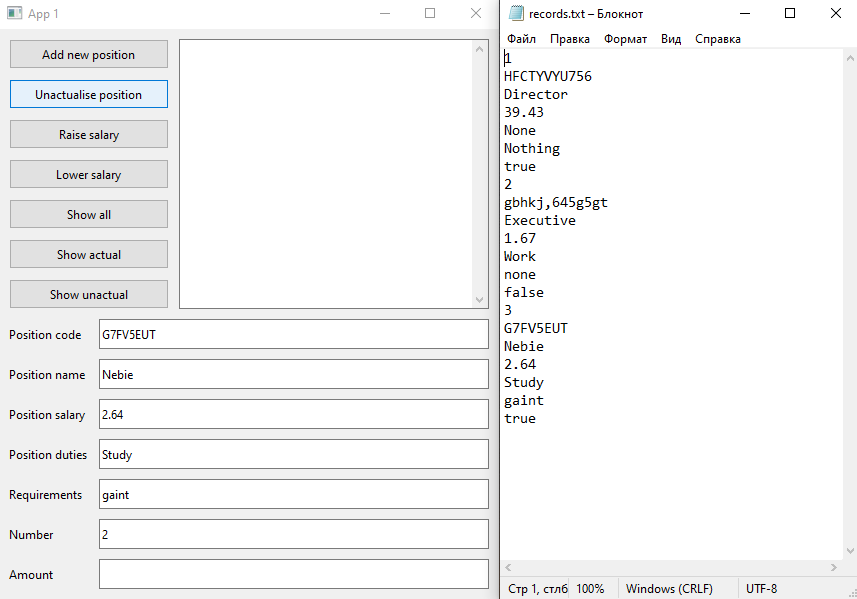
Додамо новий запис:



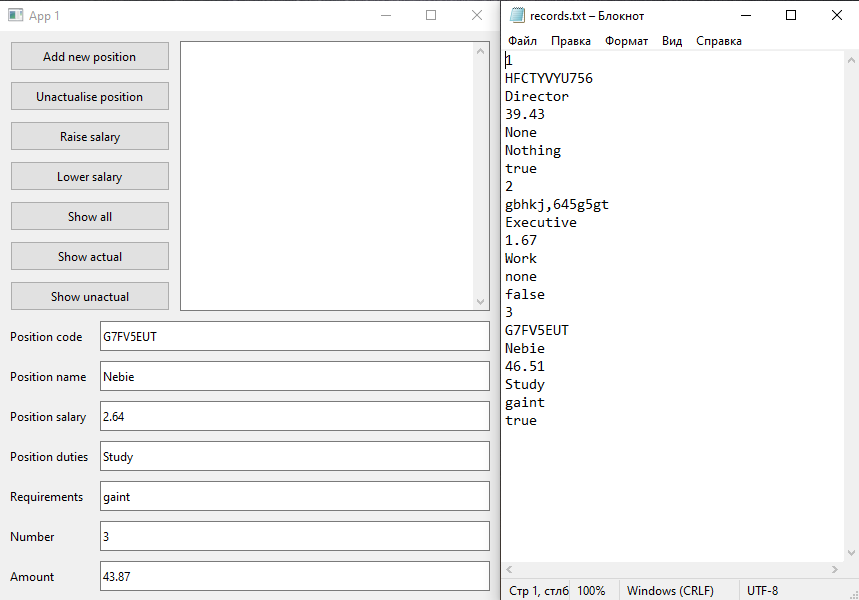
Спробуємо змінити неіснуючий запис:



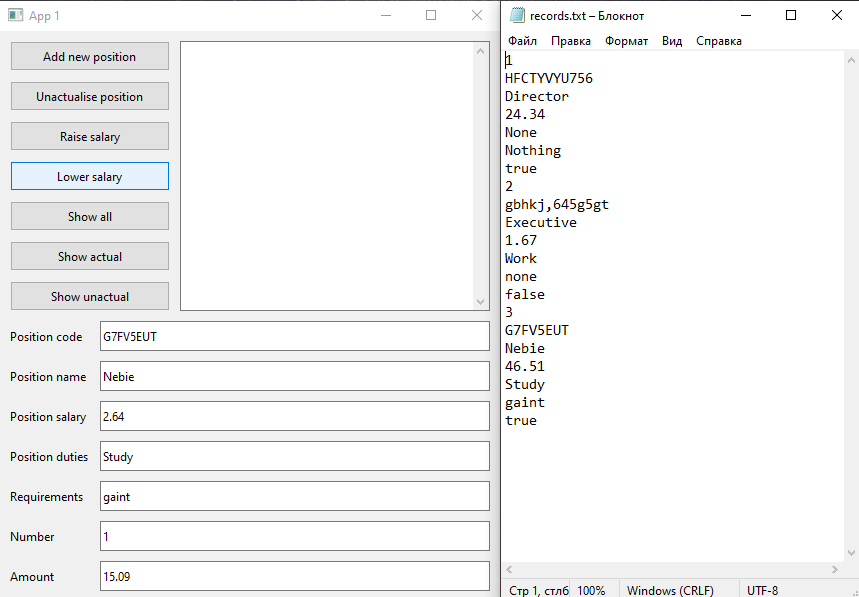
Звільнимо другу позицію:



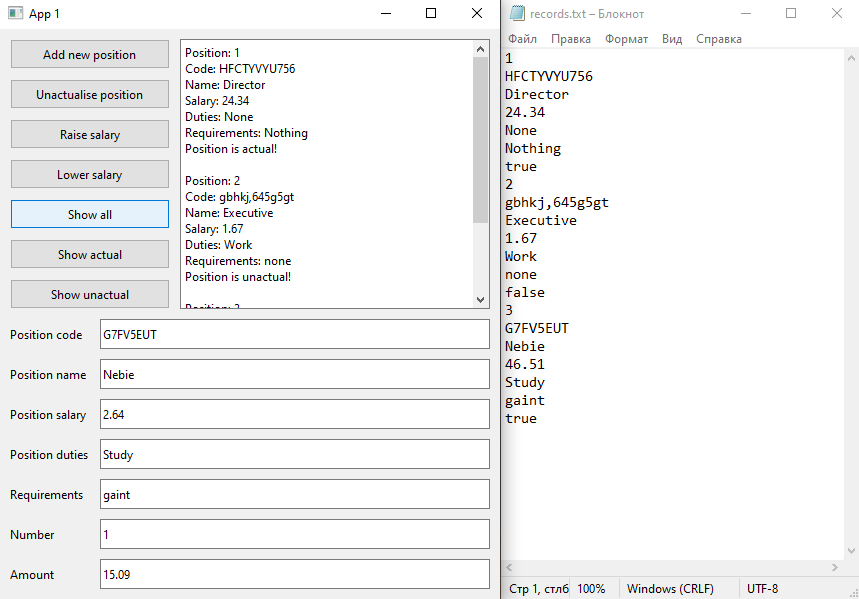
Підвищимо зарплатню третій позиції:



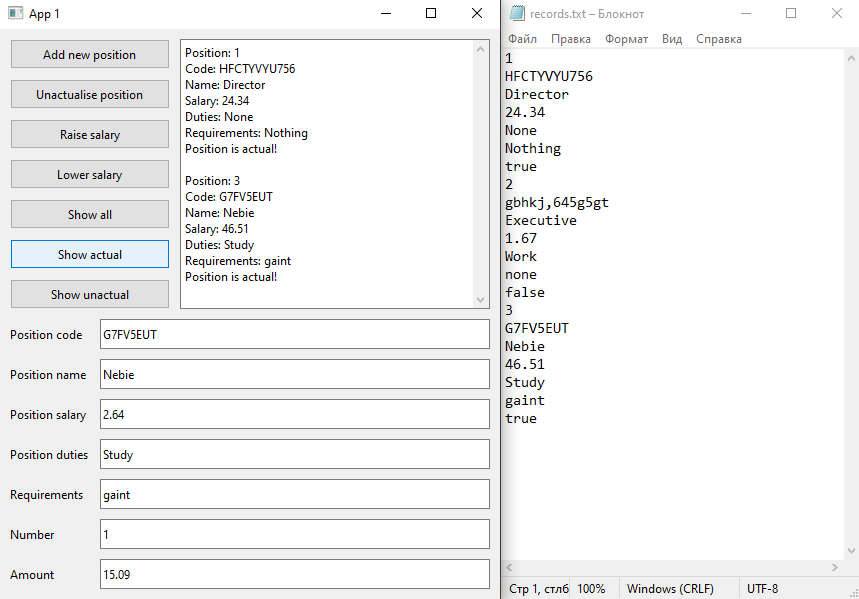
Зменшимо зарплатню першій позиції:



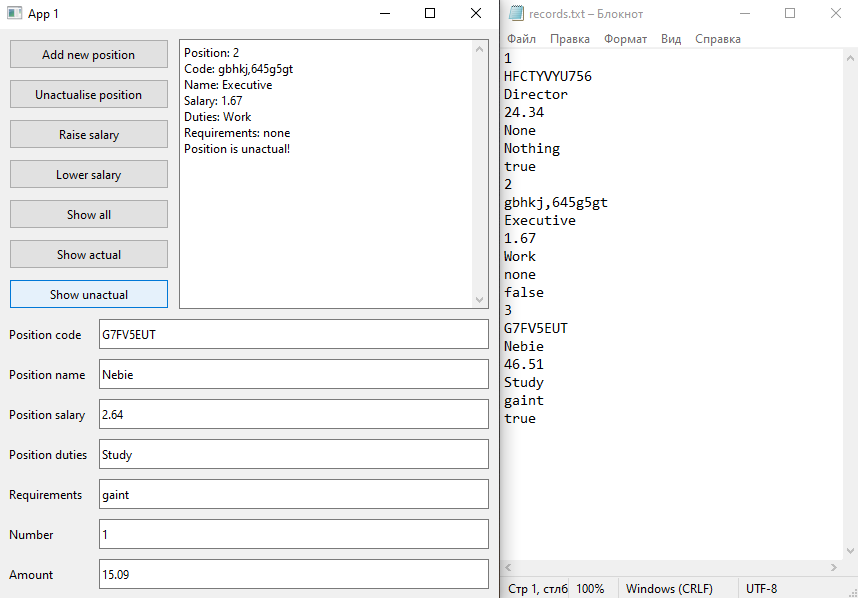
Виведемо всі записи у поле виведення:



Виведемо всі актуальні позиції:



Виведемо всі неактуалні позиції:



**Висновок:** під час виконання лабораторної роботи я реалізував функціонал 2 лабораторної роботи у вигляді десктопного застосунку із графічним інтерфейсом. Так, як всі дані зберігаються у файлі, то мені вдалося обійтися без масиву для об’єктів класу, що були реалізовані у лабораторних роботах 1 і 2, замість цього програма правильно записує й зчитує записи із текстового файлу. Для введення даних використовуються стрічки вводу по одній на кожен параметр, що так чи інакше треба було б ввести. З результатів видно, що самі по собі ці стрічки введення не очищуються, адже це зроблено для того, щоб не доводилося щоразу вводити одне і те саме щоразу, за необхідності.