**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. И. РАЗЗАКОВА**

**Кафедра «Программное обеспечение компьютерных систем»**

**Отчет**

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ С++**

**Лабораторная работа №3**

**Выполнила: Тойчубекова Айчурок ПИ-2-21**

**Проверил: Мусабаев Э.Б.**

**Бишкек 2024**

**Задания:**

1. Разработайте программу с классом **Timer**, в которой таймер действует как таймер обратного отсчета. При создании объекта типа **timer** ему присваивается начальное значение времени. В результате вызова функции **run()** таймер начинает отсчет в сторону уменьшающихся значений, пока не достигнет значения 0, после чего зазвонит звонок. **Создайте 3 варианта конструктора, т.е. конструктор перегрузите 3 раза** для того, чтобы можно было указывать время в секундах с помощью целого числа или строки, или в минутах и секундах, если указываются два целых числа.

Эта программа использует библиотечную функцию **clock(),** возвращающую число тиков, прошедших с момента запуска программы. Поделив это значение на макрос **CLK\_TCK**, получаем значение в секундах. Прототипы для **clock**() и **CLK\_TCK** содержатся в заголовочном файле time.h.

**Код:**

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <iomanip>

using namespace std;

class Timer {

private:

int seconds;

public:

Timer(int initialSeconds) : seconds(initialSeconds) {}

Timer(const string& timeString) {

parseTimeString(timeString);

}

Timer(int minutes, int seconds) : seconds(minutes \* 60 + seconds) {}

void run() {

while (seconds > 0) {

displayTime();

sleep(1); // Задержка на 1 секунду

--seconds;

}

cout << "\aTime's up! Ding ding ding!" << endl; // Звонок

}

private:

void displayTime() const {

int minutes = seconds / 60;

int remainingSeconds = seconds % 60;

cout << setw(2) << setfill('0') << minutes << ":"

<< setw(2) << setfill('0') << remainingSeconds << "\r" << flush;

}

void parseTimeString(const string& timeString) {

size\_t colonPos = timeString.find(':');

if (colonPos != string::npos) {

char\* end;

long minutes = strtol(timeString.c\_str(), &end, 10);

if (\*end == ':' && end != timeString.c\_str()) {

// Преобразование минут успешно, продолжаем с секундами

long seconds = strtol(end + 1, &end, 10);

if (\*end == '\0') {

// Преобразование секунд успешно, устанавливаем значение таймера

this->seconds = minutes \* 60 + seconds;

return;

}

}

}

cerr << "Invalid time string format. Use \"mm:ss\"." << endl;

exit(EXIT\_FAILURE);

}

void sleep(int seconds) const {

clock\_t delay = seconds \* CLOCKS\_PER\_SEC;

clock\_t start = clock();

while ((clock() - start) < delay) {}

}

};

int main() {

// Примеры использования конструкторов

Timer timer1(180); // Таймер на 3 минуты

Timer timer2("05:30"); // Таймер на 5 минут 30 секунд

Timer timer3(2, 30); // Таймер на 2 минуты 30 секунд

// Запуск таймеров

cout << "Timer 1: ";

timer1.run();

cout << "Timer 2: ";

timer2.run();

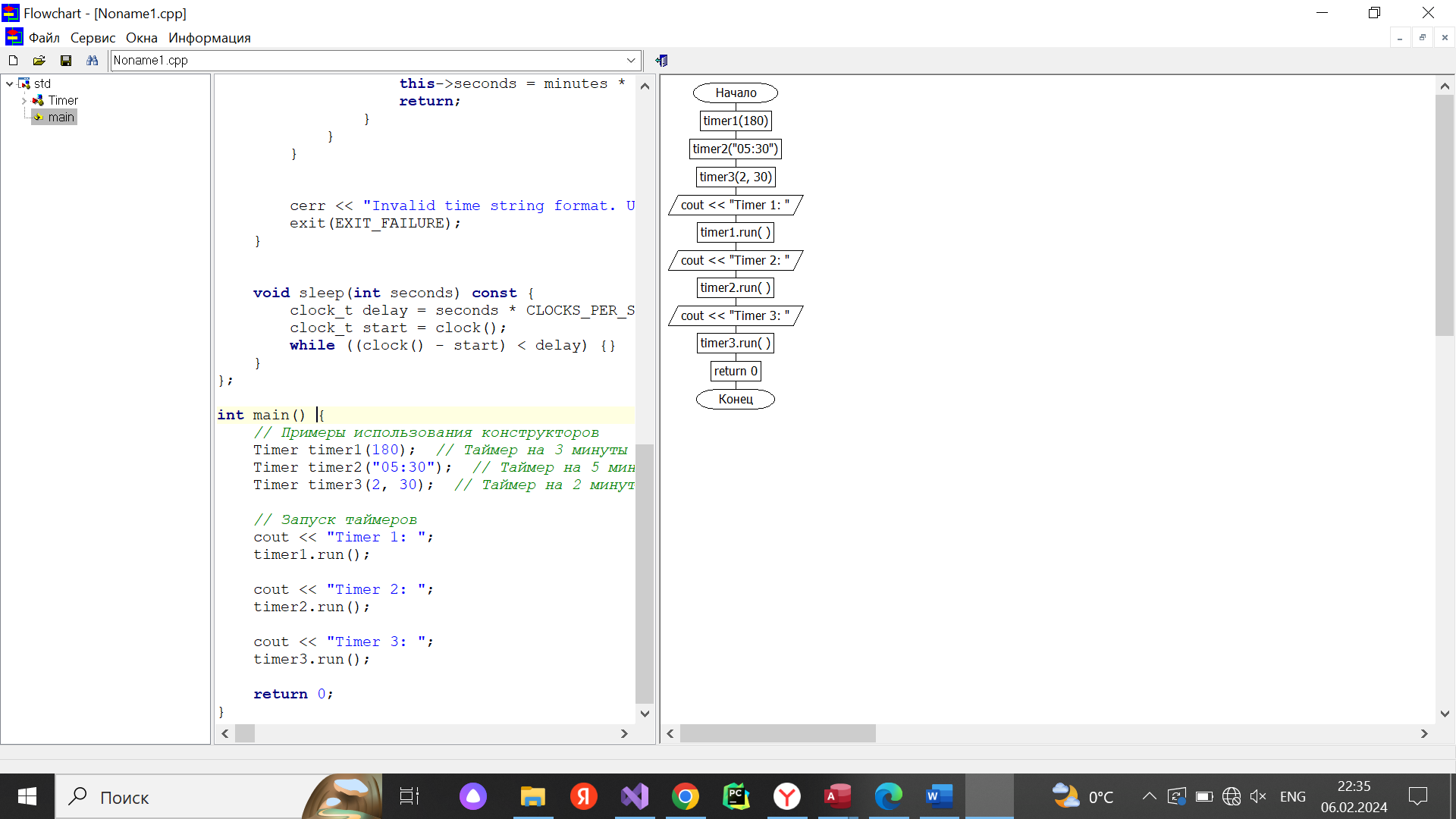
cout << "Timer 3: ";

timer3.run();

return 0;

}

**Блок-схема:**



2. Создайте класс **Nomenclature**, описывающий товары на складе магазина. Закрытыми элементами класса будут: название товара, оптовая цена, розничная наценка и количество товаров на складе. Включите в класс открытые функции подсчета возможного чистого дохода при продаже этого товара и вывода всех данных о товаре на экран. Для инициализации и удаления объектов класса используйте конструкторы и деструкторы.

**Код:**

#include <iostream>

#include <string>

using namespace std;

class Nomenclature {

private:

string productName;

double wholesalePrice;

double retailMarkup;

int quantityInStock;

public:

Nomenclature() : productName(""), wholesalePrice(0.0), retailMarkup(0.0), quantityInStock(0) {}

Nomenclature(const string& name, double wholesale, double markup, int quantity)

: productName(name), wholesalePrice(wholesale), retailMarkup(markup), quantityInStock(quantity) {}

~Nomenclature() {

}

double calculateNetIncome(int quantitySold) const {

double retailPrice = calculateRetailPrice();

return quantitySold \* (retailPrice - wholesalePrice);

}

void displayProductInfo() const {

cout << "Название: " << productName << endl;

cout << "Оптовая цена: $" << wholesalePrice << endl;

cout << "Розничная наценка: " << retailMarkup << "%" << endl;

cout << "Количество на складе: " << quantityInStock << endl;

cout << "Розничная цена: $" << calculateRetailPrice() << endl;

}

private:

double calculateRetailPrice() const {

return wholesalePrice \* (1 + retailMarkup / 100);

}

};

int main() {

setlocale(LC\_ALL, "Russian");

Nomenclature product1("Фотоаппарат", 800.0, 20.0, 50);

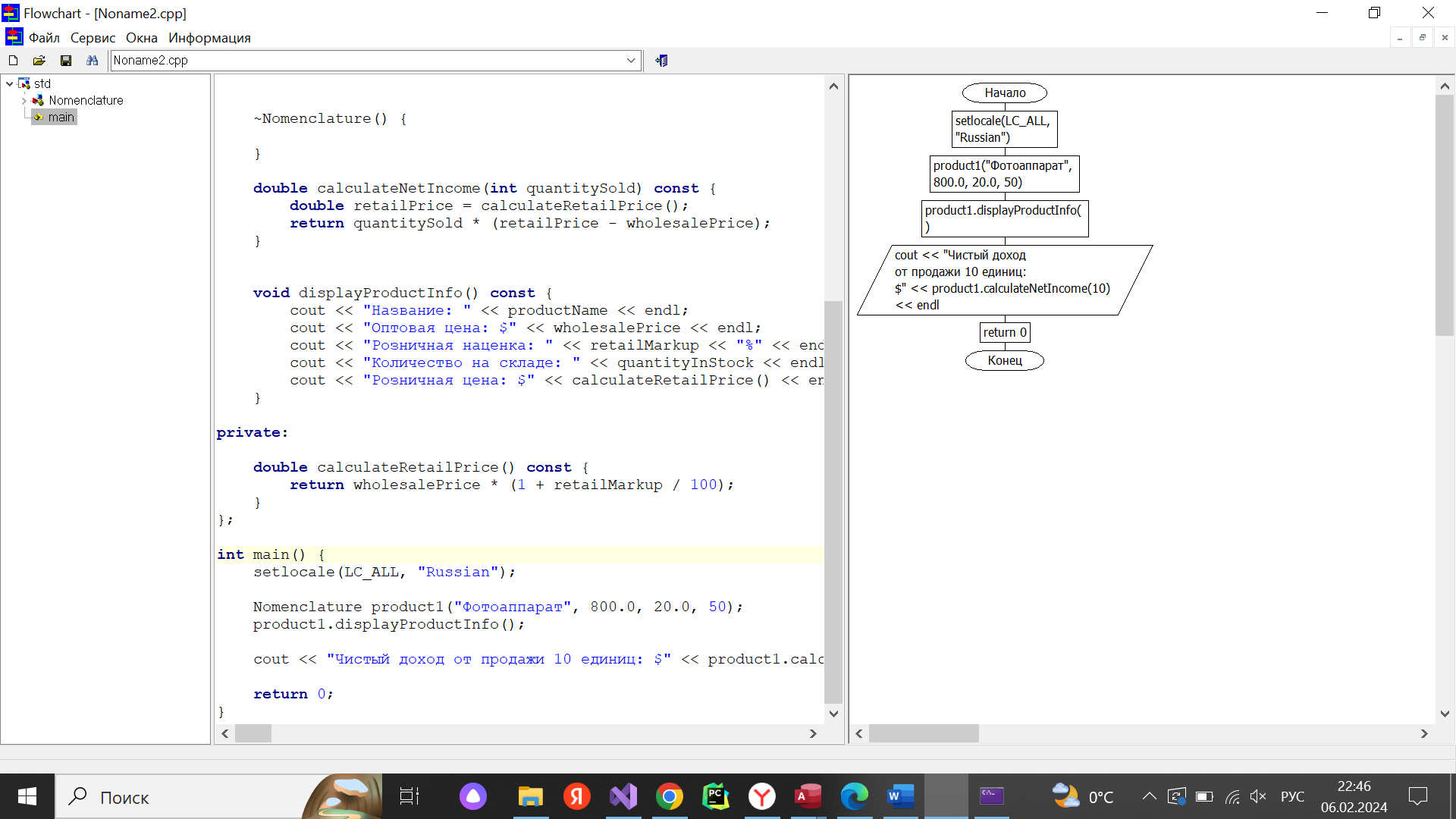
product1.displayProductInfo();

cout << "Чистый доход от продажи 10 единиц: $" << product1.calculateNetIncome(10) << endl;

return 0;

}

**Блок-схема:**



3. Создайте класс **Soft**, который содержит информацию об установленном программном обеспечении. Закрытыми элементами класса будут: название программы, разработчик, занимаемый объем, дата завершения лицензии. Включите в класс открытые функции подсчета количества дней до завершения лицензии и вывода всех данных об установленном программном обеспечении на экран. Для инициализации и удаления объектов класса используйте конструкторы и деструкторы.

**Код:**

#include <iostream>

#include <string>

#include <ctime>

using namespace std;

class Soft {

private:

string programName;

string developer;

double occupiedSpace;

time\_t licenseExpiration;

public:

Soft(const string& name, const string& dev, double space, time\_t expiration)

: programName(name), developer(dev), occupiedSpace(space), licenseExpiration(expiration) {}

~Soft() {

cout << "Программа " << programName << " удалена." << endl;

}

int daysUntilLicenseExpiration() const {

time\_t currentTime;

time(&currentTime);

int secondsRemaining;

if (difftime(licenseExpiration, currentTime) > 0) {

secondsRemaining = static\_cast<int>(difftime(licenseExpiration, currentTime));

}

else {

secondsRemaining = 0;

}

int daysRemaining = secondsRemaining / (60 \* 60 \* 24);

return daysRemaining;

}

void displaySoftwareInfo() const {

cout << "Название программы: " << programName << endl;

cout << "Разработчик: " << developer << endl;

cout << "Занимаемый объем: " << occupiedSpace << " MB" << endl;

char buffer[26];

ctime\_s(buffer, sizeof(buffer), &licenseExpiration);

cout << "Дата завершения лицензии: " << buffer;

}

};

int main() {

setlocale(LC\_ALL, "Russian");

time\_t expirationTime;

time(&expirationTime);

expirationTime += 30 \* 24 \* 60 \* 60; // Лицензия на 30 дней от текущего момента

Soft software1("PyCharm", "Тойчубекова Айчурок", 160.5, expirationTime);

cout << "Информация о программном обеспечении:" << endl;

software1.displaySoftwareInfo();

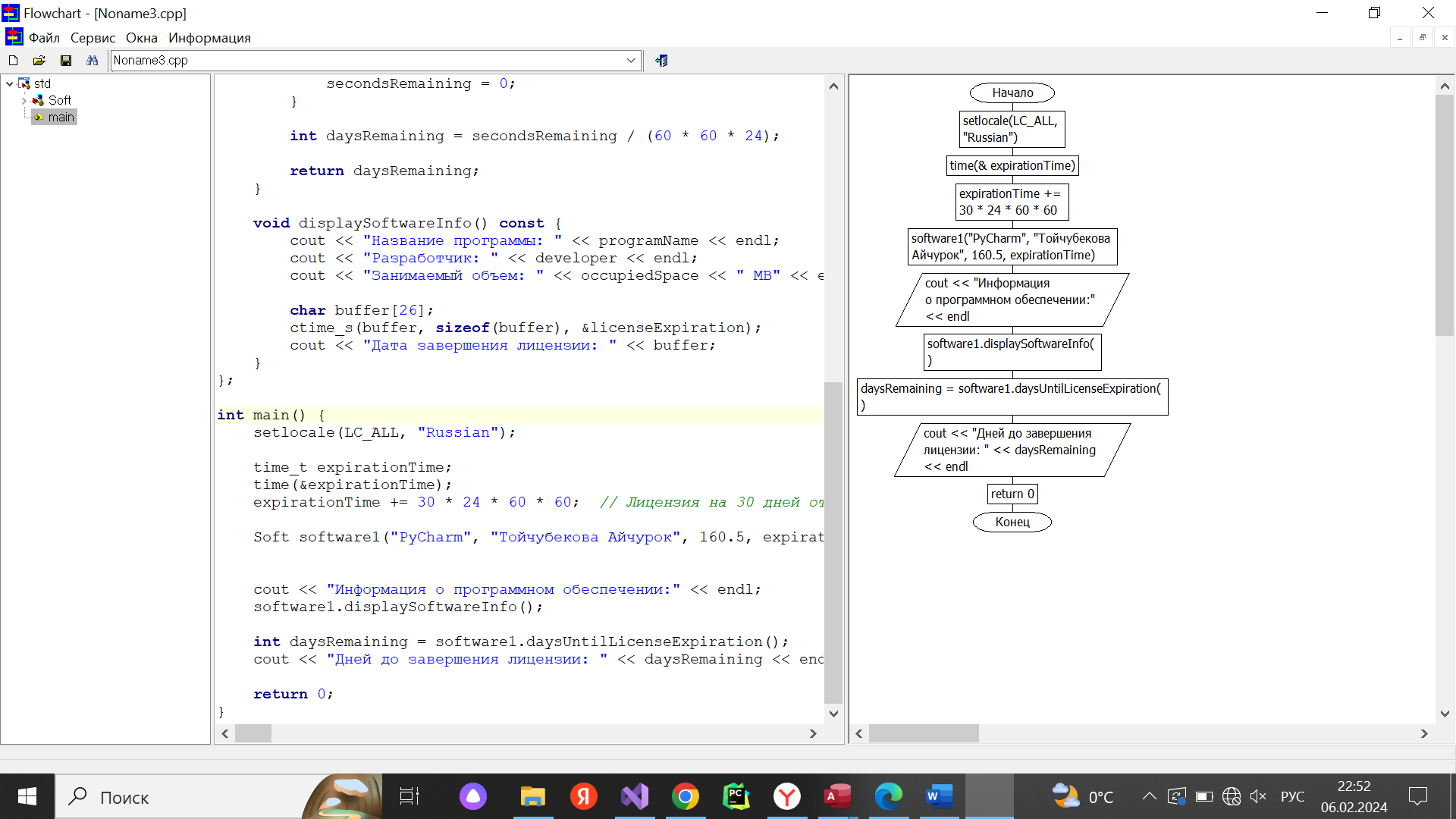
int daysRemaining = software1.daysUntilLicenseExpiration();

cout << "Дней до завершения лицензии: " << daysRemaining << endl;

return 0;

}

**Блок-схема:**

****

**ВОПРОСЫ:**

**1. Что такое конструктор? Как он вызывается на выполнение?**

Конструктор - это специальный метод в классе C++, который выполняется при создании объекта этого класса. Конструктор имеет тот же имя, что и класс, и не возвращает значения. Вызывается автоматически при создании объекта.

**2. Особенности конструктора, отличающие его от других методов класса?**

Конструктор имеет тот же имя, что и класс, и вызывается автоматически при создании объекта. Он не имеет возвращаемого типа, и его целью является инициализация объекта.

**3. Что такое конструктор по умолчанию? В каком случае следует его использовать?**

Конструктор по умолчанию - это конструктор без параметров. Он инициализирует объект значениями по умолчанию. Используется, когда объект создается без явного указания начальных значений.

**4.Способы инициализации в конструкторе по умолчанию? Какой способ инициализации предпочтительнее?**

Инициализация в конструкторе по умолчанию может быть выполнена через инициализацию членов класса в теле конструктора или с использованием инициализационного списка. Использование инициализационного списка предпочтительнее, так как позволяет эффективнее управлять инициализацией.

**5. Что такое конструктор с аргументами? Когда используют конструктор с аргументами?**

Конструктор с аргументами - это конструктор, который принимает параметры при создании объекта. Используется, когда требуется передать начальные значения для инициализации объекта.

**6. Какие преимущества у конструктора с аргументами перед методом Set()?**

Конструктор с аргументами обеспечивает инициализацию объекта при его создании, что делает код более надежным и уменьшает возможность создания объекта в неполностью инициализированном состоянии.

**7. Что такое деструктор? Для чего его используют?**

Деструктор - это метод класса, который вызывается при уничтожении объекта. Он используется для выполнения операций по очистке ресурсов, выделенных объекту во время его жизни.

**8. Особенности деструктора**

Деструктор имеет тот же синтаксис, что и конструктор, но начинается с символа "~". Он не принимает аргументов и не возвращает значения.

**9. В какой последовательности выполняется конструкторы, в какой деструкторы классов?**

При создании объекта вызываются конструкторы классов в порядке от базового класса к производному. При уничтожении объекта вызываются деструкторы в обратном порядке - от производного к базовому.

**10. Можно ли перегружать конструктор?**

Да, конструкторы могут быть перегружены, т.е. класс может иметь несколько конструкторов с разными параметрами.

**11. Что такое конструктор копирования?**

Конструктор копирования - это специальный конструктор, который создает новый объект как копию существующего объекта. Он используется при передаче объектов по значению, возврате объектов из функций и других ситуациях, когда требуется создать копию объекта.