**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: Эмилбекова Элиза ПИ-2-21**

**Проверил: Мусабаев Э.Б.**

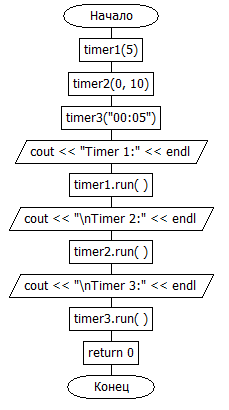
**Бишкек 2024**

**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.

**Блок-схема:**



**Код:**

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <string>

using namespace std;

class Timer {

private:

int seconds;

public:

// Конструктор для времени в секундах

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

// Конструктор для времени в минутах и секундах

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

// Конструктор для времени в виде строки "мм:сс"

Timer(const string& timeStr) {

size\_t found = timeStr.find(":");

if (found != string::npos) {

int min = stoi(timeStr.substr(0, found));

int sec = stoi(timeStr.substr(found + 1));

seconds = min \* 60 + sec;

}

else {

cerr << "Invalid time format. Please use MM:SS." << endl;

exit(EXIT\_FAILURE);

}

}

// Функция запуска таймера

void run() {

clock\_t start = clock();

while (seconds > 0) {

if ((clock() - start) / CLOCKS\_PER\_SEC == 1) {

seconds--;

start = clock();

}

}

cout << "\aTimer has finished!" << endl;

}

};

int main() {

// Примеры использования конструкторов

Timer timer1(5); // Время в секундах

Timer timer2(0, 10); // Время в минутах и секундах

Timer timer3("00:05"); // Время в виде строки "мм:сс"

// Запуск таймеров

cout << "Timer 1:" << endl;

timer1.run();

cout << "\nTimer 2:" << endl;

timer2.run();

cout << "\nTimer 3:" << endl;

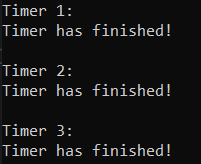
timer3.run();

return 0;

}

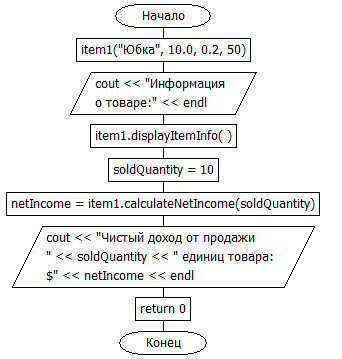
**Работа программы:**

**Выводится сообщение когда заданное время равняется 0 .**



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.

**Блок-схема:**



**Код:**

#include <iostream>

#include <string>

using namespace std;

class Nomenclature {

private:

string itemName;

double wholesalePrice;

double retailMarkup;

int quantityInStock;

public:

// Конструктор с параметрами

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

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

// Деструктор

~Nomenclature() {

// Возможные дополнительные операции при удалении объекта

cout << "Объект " << itemName << " удален." << endl;

}

// Функция подсчета возможного чистого дохода при продаже товара

double calculateNetIncome(int soldQuantity) {

double retailPrice = wholesalePrice \* (1 + retailMarkup);

double grossIncome = retailPrice \* soldQuantity;

double netIncome = grossIncome - (wholesalePrice \* soldQuantity);

return netIncome;

}

// Функция вывода данных о товаре на экран

void displayItemInfo() { cout << "Наименование товара: " << itemName << endl;

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

cout << "Наценка на розницу: " << (retailMarkup \* 100) << "%" << endl;

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

}

};

int main() {

setlocale(LC\_ALL, "Russian");

Nomenclature item1("Юбка", 10.0, 0.2, 50);

cout << "Информация о товаре:" << endl;

item1.displayItemInfo();

int soldQuantity = 10;

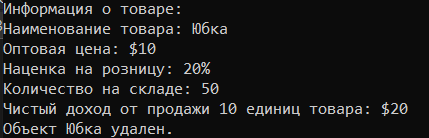
double netIncome = item1.calculateNetIncome(soldQuantity);

cout << "Чистый доход от продажи " << soldQuantity << " единиц товара: $" << netIncome << endl;

return 0;

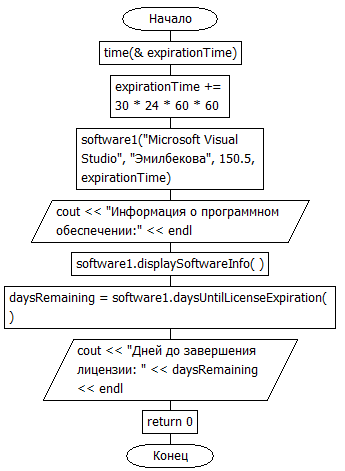
}

**Работа программы:**



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.

**Блок-схема:**

****

**Код:**

#include <iostream>

#include <string>

#include <ctime>

using namespace std;

class Soft {

private:

string programName;

string developer;

double occupiedSpace;

time\_t licenseExpiration;

public:

// Конструктор с параметрами

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

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

// Деструктор

~Soft() {

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

}

// Функция подсчета количества дней до завершения лицензии

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("Microsoft Visual Studio", "Эмилбекова", 150.5, expirationTime);

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

software1.displaySoftwareInfo();

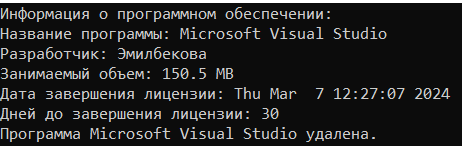
int daysRemaining = software1.daysUntilLicenseExpiration();

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

return 0;

}

**Работа программы:**

****

**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.