**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И. РАЗЗАКОВА**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**ОТЧЕТ**

**по лабораторной работе №3**

**Выполнила:** студент группы ПИ-3-21

Джанышбекова Акмарал

**Проверила:** Мусабаев Э. Б.

**Бишкек 2024**

**Задание №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.

**Код программы:**

#include <iostream>

#include <ctime>

#include <string>

#include <thread>

#include <chrono>

#include <stdio.h>

#include <windows.h>

#include <mmsystem.h>

#pragma comment(lib, "Winmm.lib") // Чтобы связать библиотеку winmm.lib

using namespace std;

class Timer {

private:

int sec; //секунды

public:

// Конструктор 1: время указано в секундах (целое число)

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

// Конструктор 2: время указано в секундах (строка)

Timer(const string& s) {

sec = stoi(s); //преобразование строки в целое число

}

// Конструктор 3: время указано в минутах и секундах (два целых числа)

Timer(int m, int s) : sec(m \* 60 + s) {}

void run() {

cout << "Таймер начал работу\n";

// Получаем начальное время

clock\_t start\_time = clock();

while (sec > 0) {

// Время, прошедшее с начала отсчета

double elapsed\_time = (clock() - start\_time) / (double)CLOCKS\_PER\_SEC;

// Если прошла одна секунда, уменьшаем время таймера

if (elapsed\_time >= 1) {

sec--;

start\_time = clock();

cout << sec << " секунды\n";

}

this\_thread::sleep\_for(chrono::milliseconds(50)); // небольшая задержка, чтобы не загружать процессор

}

Sleep(1000); system("cls");

PlaySound(L"qwerty", NULL, SND\_FILENAME | SND\_ASYNC); Sleep(2000);

cout << "Звонок! Звонок! Звонок!\n";

}

};

int main() {

setlocale(LC\_ALL, "ru");

int choice;

int minutes, seconds;

string sec\_str;

while (true) {

cout << "/\*---------\* Главное Меню \*---------\*/\n";

cout << "1. Задать таймер \*в секундах - целое число\*\n";

cout << "2. Задать таймер \*в секундах - строка\*\n";

cout << "3. Задать таймер \*в минутах и секундах - два целых числа\*\n";

cout << "4. 1Выход\n";

cout << "Выберите действие: ";

cin >> choice; // выбор пользователя

switch (choice) {

case 1:

cout << "Введите количество секунд: ";

cin >> seconds;

Timer(seconds).run(); //выполнения метода run() с передачей значения seconds

break;

case 2:

cout << "Введите количество секунд (строкой): ";

cin >> sec\_str;

Timer(sec\_str).run(); //выполнения метода run() с передачей значения sec\_str

break;

case 3:

cout << "Введите количество минут: ";

cin >> minutes;

cout << "Введите количество секунд: ";

cin >> seconds;

Timer(minutes, seconds).run();//выполнения метода run() с передачей значений minutes и seconds

break;

case 4:

cout << "Выход из программы. До свидания!\n";

choice = 0;

break;

default:

cout << "Неверный выбор. Попробуйте снова.\n";

break;

}

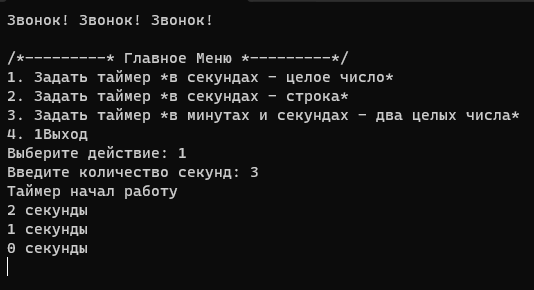
cout << "\n"; // Для лучшей читаемости выводим пустую строку между итерациями

}

return 0;

}

**Результат работы программы:**



**Задание №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 программы:**

#include <iostream>

#include <string>

class Nomenclature {

private:

std::string name;

double wholesalePrice;

double retailMarkup;

int quantity;

public:

Nomenclature(const std::string& n, double wPrice, double rMarkup, int q) :

name(n), wholesalePrice(wPrice), retailMarkup(rMarkup), quantity(q) {}

double potentialProfit() {

return quantity \* (retailMarkup - wholesalePrice);

}

void displayInfo() {

std::cout << "Название товара: " << name << "\n";

std::cout << "Оптовая цена: " << wholesalePrice << "\n";

std::cout << "Розничная наценка: " << retailMarkup << "\n";

std::cout << "Количество на складе: " << quantity << "\n";

std::cout << "Потенциальный чистый доход: " << potentialProfit() << "\n";

}

~Nomenclature() {

std::cout << "Товар " << name << " был удален.\n";

}

};

int main() {

setlocale(LC\_ALL, "ru");

int numProducts;

std::cout << "Введите количество видов товара: ";

std::cin >> numProducts;

std::cin.ignore(); // Clear the newline character from the buffer

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

std::string name;

double wholesalePrice, retailMarkup;

int quantity;

std::cout << "Введите название товара " << i + 1 << ": ";

std::cin >> name;

std::cout << "Введите оптовую цену для " << name << ": ";

std::cin >> wholesalePrice;

std::cout << "Введите розничную наценку для " << name << ": ";

std::cin >> retailMarkup;

std::cout << "Введите количество товаров на складе для " << name << ": ";

std::cin >> quantity;

Nomenclature product(name, wholesalePrice, retailMarkup, quantity);

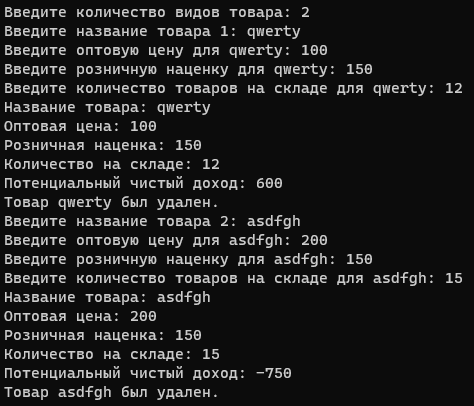
product.displayInfo();

}

return 0;

}

**Результат работы программы:**

****

**Задание №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 программы:**

#include <iostream>

#include <string>

#include <chrono>

#include <ctime>

#define \_CRT\_SECURE\_NO\_WARNINGS

using namespace std;

class Soft {

private:

string programName;

string developer;

double occupiedVolume;

tm expirationDate;

// Вспомогательная функция для вычисления разницы между двумя датами

int daysBetween(const tm& date1, const tm& date2) const {

chrono::system\_clock::time\_point tp1 = chrono::system\_clock::from\_time\_t(mktime(const\_cast<tm\*>(&date1)));

chrono::system\_clock::time\_point tp2 = chrono::system\_clock::from\_time\_t(mktime(const\_cast<tm\*>(&date2)));

return chrono::duration\_cast<chrono::duration<int, ratio<24 \* 60 \* 60>>>(tp2 - tp1).count();

}

public:

// Конструктор

Soft(const string& programName, const string& developer, double occupiedVolume, const tm& expirationDate)

: programName(programName), developer(developer), occupiedVolume(occupiedVolume), expirationDate(expirationDate) {}

// Деструктор

~Soft() {}

// Функция подсчета количества дней до завершения лицензии

int daysToLicenseEnd() const {

time\_t now = time(nullptr);

tm currentDate;

if (localtime\_s(&currentDate, &now) != 0) {

// Обработка ошибки

}

return daysBetween(currentDate, expirationDate);

}

// Функция вывода информации о программе на экран

void displayInfo() const {

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

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

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

cout << "Дата завершения лицензии: " << expirationDate.tm\_mday << "." << expirationDate.tm\_mon + 1 << "." << expirationDate.tm\_year + 1900 << endl;

}

};

int main() {

Soft\* software = nullptr;

setlocale(LC\_ALL, "ru");

while (true) {

cout << "Меню:" << endl;

cout << "1. Создать новую запись о программе" << endl;

cout << "2. Показать информацию о программе" << endl;

cout << "3. Показать количество дней до завершения лицензии" << endl;

cout << "4. Выход" << endl;

int choice;

cin >> choice;

switch (choice) {

case 1: {

cin.ignore();

string programName, developer;

double occupiedVolume;

tm date = {};

cout << "Введите название программы: ";

getline(cin, programName);

cout << "Введите разработчика: ";

getline(cin, developer);

cout << "Введите занимаемый объем (MB): ";

cin >> occupiedVolume;

cout << "Введите дату завершения лицензии (ДД ММ ГГГГ): ";

cin >> date.tm\_mday >> date.tm\_mon >> date.tm\_year;

date.tm\_mon -= 1;

date.tm\_year -= 1900;

delete software;

software = new Soft(programName, developer, occupiedVolume, date);

cout << "Запись создана!" << endl;

break;

}

case 2:

if (software) {

software->displayInfo();

}

else {

cout << "Сначала создайте запись о программе." << endl;

}

break;

case 3:

if (software) {

cout << "Дней до завершения лицензии: " << software->daysToLicenseEnd() << endl;

}

else {

cout << "Сначала создайте запись о программе." << endl;

}

break;

case 4:

delete software;

return 0;

default:

cout << "Неверный выбор. Повторите ввод." << endl;

break;

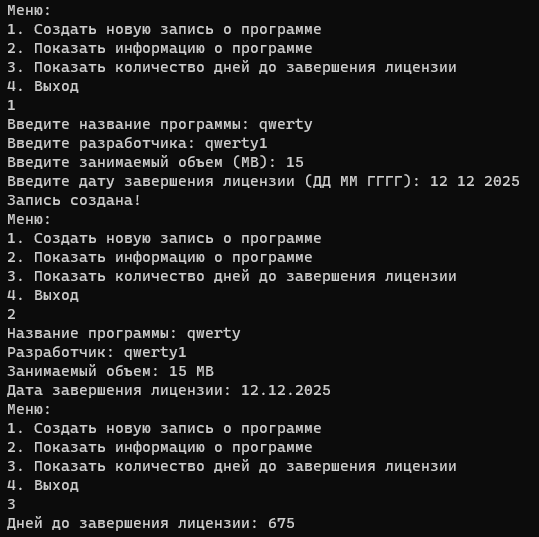
}

}

return 0;

}

**Результат работы программы:**

****

**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ №3**

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

Конструктор - это специальный метод в классе, который используется для инициализации объектов этого класса. Он вызывается автоматически при создании объекта. Конструктор имеет тот же имя, что и класс, и не возвращает значения.

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

- Основные отличия конструктора от других методов класса

- Конструктор имеет то же имя, что и класс.

- Он вызывается при создании объекта класса.

- Не имеет возвращаемого значения (нет типа возвращаемого значения).

- Конструкторы могут быть перегружены, т.е., класс может иметь несколько конструкторов с разными параметрами.

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

Конструктор по умолчанию - это конструктор, который не принимает аргументов. Он создает объект с начальными значениями по умолчанию. Его следует использовать, если вы хотите создавать объекты без передачи каких-либо параметров.

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

В конструкторе по умолчанию можно использовать инициализацию членов класса в списке инициализации или в теле конструктора. Предпочтительнее использовать список инициализации, так как это более эффективный способ инициализации и может предотвратить ненужные операции присваивания.

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

Конструктор с аргументами - это конструктор, который принимает один или более аргументов для инициализации объекта класса. Он используется, когда требуется инициализировать объект с определенными значениями при его создании.

1. *Какие преимущества у конструктор с аргументами перед методом Set()?*

Конструктор с аргументами позволяет инициализировать объект при его создании, что делает код более надежным и предсказуемым. В отличие от метода `Set()`, конструктор с аргументами не требует дополнительных вызовов и обеспечивает более прямой и чистый интерфейс.

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

Деструктор - это специальный метод класса, который вызывается при уничтожении объекта. Он используется для освобождения ресурсов, таких как динамически выделенная память или закрытие файлов, и для выполнения других операций, которые могут потребоваться при завершении объекта

1. *Особенности деструктор*

-Деструктор имеет тот же имя, что и класс, но с символом `~` перед ним.

- Он не принимает аргументов и не имеет возвращаемого значения.

- Вызывается автоматически при уничтожении объекта.

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

Конструкторы выполняются в порядке создания объектов, а деструкторы выполняются в обратной последовательности, при уничтожении объектов. То есть, первым создается конструктор верхнего уровня, а первым выполняется его деструктор.

1. *Можно ли перегружать конструктор?*

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

1. *Что такое конструктор копирования?*

Давайте рассмотрим ваши вопросы по порядку:

Конструктор копирования - это специальный конструктор, который создает новый объект на основе существующего объекта, копируя его значения и состояние. Этот конструктор обычно используется для создания глубоких копий объектов и предотвращения совместного использования данных.