**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И. Раззакова**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Дисциплина «**Объектно-Ориентированное Программирование**»

Отчет

Лабораторная работа №3

Выполнил: студент группы ПИ-2-21

Сатыбалдыев Нурислам

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

////1 задание

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <iomanip>

using namespace std;

class Timer {

private:

int seconds;

public:

// Конструкторы

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

Timer(const string& timeString) {

parseTimeString(timeString);

}

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

// Функция для запуска таймера

void run() {

while (seconds > 0) {

displayTime();

sleep(1); // Задержка на 1 секунду

--seconds;

}

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

}

private:

// Функция для отображения времени

void displayTime() const {

int minutes = seconds / 60;

int remainingSeconds = seconds % 60;

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

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

}

// Функция для парсинга времени из строки в формате "mm:ss"

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;

}

}

}

// Обработка ошибки

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

exit(EXIT\_FAILURE);

}

// Функция для создания задержки

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(10); // 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;

}

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

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.

// 2 задание

#include <iostream>

#include <string>

using namespace std;

class Nomenclature {

private:

string productName;

double wholesalePrice;

double retailMarkup;

int quantityInStock;

public:

// Конструктор по умолчанию

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

// Конструктор с параметрами

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

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

// Деструктор

~Nomenclature() {

// Дополнительные действия по очистке, если необходимо

}

// Функция подсчета возможного чистого дохода при продаже товара

double calculateNetIncome(int quantitySold) const {

double retailPrice = calculateRetailPrice();

return quantitySold \* (retailPrice - wholesalePrice);

}

// Функция вывода всех данных о товаре на экран

void displayProductInfo() const {

cout << "Product Name: " << productName << endl;

cout << "Wholesale Price: $" << wholesalePrice << endl;

cout << "Retail Markup: " << retailMarkup << "%" << endl;

cout << "Quantity in Stock: " << quantityInStock << endl;

cout << "Retail Price: $" << calculateRetailPrice() << endl;

}

private:

// Функция подсчета розничной цены

double calculateRetailPrice() const {

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

}

};

int main() {

// Пример использования класса

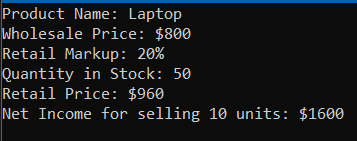
Nomenclature product1("Laptop", 800.0, 20.0, 50);

product1.displayProductInfo();

cout << "Net Income for selling 10 units: $" << product1.calculateNetIncome(10) << endl;

return 0;

}



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.

// 3 задание

#include <iostream>

#include <string>

#include <ctime>

class Soft {

private:

std::string programName;

std::string developer;

double diskSpace;

time\_t licenseExpirationDate;

public:

// Конструктор по умолчанию

Soft() : programName(""), developer(""), diskSpace(0.0), licenseExpirationDate(0) {}

// Конструктор с параметрами

Soft(const std::string& name, const std::string& dev, double space, time\_t expirationDate)

: programName(name), developer(dev), diskSpace(space), licenseExpirationDate(expirationDate) {}

// Деструктор

~Soft() {

// Здесь можно добавить код для освобождения ресурсов, если это необходимо

}

// Функция для подсчета количества дней до завершения лицензии

int daysUntilLicenseExpires() const {

time\_t currentTime;

time(&currentTime);

double secondsRemaining = difftime(licenseExpirationDate, currentTime);

return static\_cast<int>(secondsRemaining / (60 \* 60 \* 24));

}

// Функция для вывода всех данных о программном обеспечении на экран

void displaySoftwareInfo() const {

std::cout << "Program Name: " << programName << std::endl;

std::cout << "Developer: " << developer << std::endl;

std::cout << "Disk Space: " << diskSpace << " GB" << std::endl;

std::cout << "Days until License Expiration: " << daysUntilLicenseExpires() << " days" << std::endl;

}

};

int main() {

// Пример использования класса Soft

time\_t expirationDate;

time(&expirationDate);

expirationDate += 30 \* 24 \* 60 \* 60; // Добавим 30 дней к текущей дате

Soft software1("MyApp", "MyCompany", 2.5, expirationDate);

// Вывод информации о программном обеспечении

std::cout << "Software 1 Information:" << std::endl;

software1.displaySoftwareInfo();

return 0;

}

