Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им. И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление: 710400 «Программная инженерия»

Дисциплина: «Объектно – ориентированное программирование»

**ОТЧЕТ**

По лабораторной работе №3.

Тема: «Конструктор и деструктор»

                                                                       Выполнила: студент группы

ПИ(б)-2-21 Абдылдаева Аэлита

Проверил: Мусабаев Э. Б.

**Бишкек – 2024**

**Тема: Конструктор и деструктор**

***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.

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 <string>

#include <locale>

using namespace std;

class Timer {

private:

int seconds;

public:

Timer() : seconds(0) {}

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

Timer(const string& timeStr);

Timer(int min, int sec);

void run();

};

Timer::Timer(const string& timeStr) {

int colonPos = timeStr.find(':');

if (colonPos != string::npos) {

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

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

seconds = min \* 60 + sec;

}

else {

seconds = 0;

}

}

Timer::Timer(int min, int sec) {

seconds = min \* 60 + sec;

}

void Timer::run() {

cout << "Таймер запущен!" << endl;

while (seconds > 0) {

cout << "Оставшееся время: " << seconds << " секунд" << endl;

clock\_t start = clock();

while ((clock() - start) / (double)CLOCKS\_PER\_SEC < 1.0);

seconds--;

}

cout << "Время вышло! Звонок!" << endl;

}

int main() {

setlocale(LC\_ALL, "rus");

while (true) {

cout << "Введите начальное время таймера в одном из следующих форматов:" << endl;

cout << "1. В секундах" << endl;

cout << "2. В формате 'минуты:секунды'" << endl;

cout << "3. В минутах и секундах (два числа)" << endl;

int option;

cin >> option;

Timer timer;

switch (option) {

case 1: {

int sec;

cout << "Введите количество секунд: ";

cin >> sec;

timer = Timer(sec);

break;

}

case 2: {

string timeStr;

cout << "Введите время в формате 'минуты:секунды': ";

cin >> timeStr;

timer = Timer(timeStr);

break;

}

case 3: {

int min, sec;

cout << "Введите количество минут: ";

cin >> min;

cout << "Введите количество секунд: ";

cin >> sec;

timer = Timer(min, sec);

break;

}

default:

cout << "Некорректный ввод." << endl;

return 1;

}

timer.run();

cout << "Хотите запустить еще один таймер? (y/n): ";

char choice;

cin >> choice;

if (choice != 'y' && choice != 'Y') {

break;

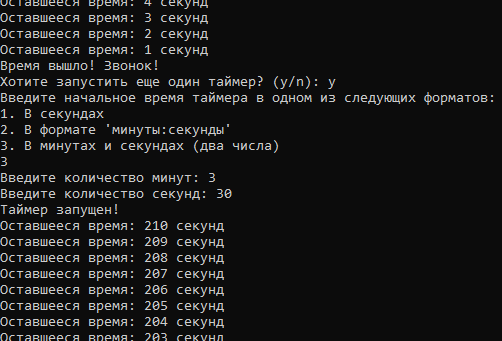
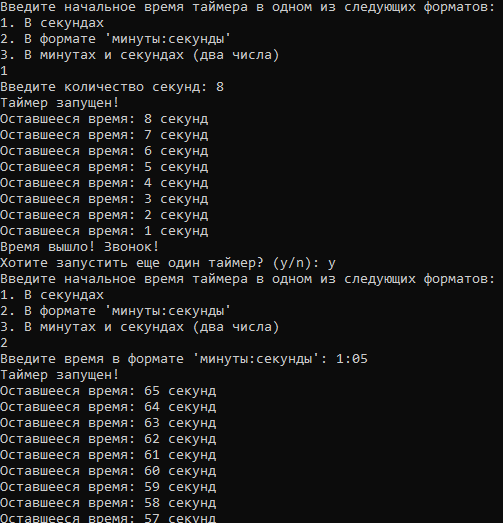
}

}

return 0;

}

***Результат:***

******

***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:***

#include <iostream>

#include <string>

#include <iomanip>

#include <vector>

#include <windows.h>

using namespace std;

class Nomenclature {

private:

string itemName;

double wholesalePrice;

double markup;

int quantity;

public:

Nomenclature() : itemName(""), wholesalePrice(0.0), markup(0.0), quantity(0) {}

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

: itemName(name), wholesalePrice(wholesalePrice), markup(markup), quantity(quantity) {}

~Nomenclature() {}

string getName() const {

return itemName;

}

double calculateWholesaleProfit() const {

return (wholesalePrice \* quantity);

}

double calculateRetailProfit() const {

double retailPrice = wholesalePrice \* (1 + markup / 100);

return retailPrice \* quantity;

}

void printDetails() const {

cout << setw(20) << left << itemName

<< setw(15) << right << wholesalePrice

<< setw(15) << right << markup

<< setw(15) << right << quantity << endl;

}

};

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<Nomenclature> products;

while (true) {

cout << "Меню:" << endl;

cout << "1. Добавить товар" << endl;

cout << "2. Вывести список всех товаров" << endl;

cout << "3. Посчитать доход" << endl;

cout << "4. Выход" << endl;

cout << "Выберите опцию: ";

int choice;

cin >> choice;

switch (choice) {

case 1: {

string name;

double wholesalePrice, markup;

int quantity;

cout << "Введите название товара: ";

cin.ignore();

getline(cin, name);

cout << "Введите оптовую цену: ";

cin >> wholesalePrice;

cout << "Введите розничную наценку (в процентах): ";

cin >> markup;

cout << "Введите количество товаров на складе: ";

cin >> quantity;

products.push\_back(Nomenclature(name, wholesalePrice, markup, quantity));

break;

}

case 2: {

if (products.empty()) {

cout << "Список товаров пуст." << endl;

}

else {

cout << setw(20) << left << "Название товара"

<< setw(15) << right << "Оптовая цена"

<< setw(15) << right << "Наценка, %"

<< setw(15) << right << "Количество" << endl;

for (const auto& product : products) {

product.printDetails();

}

}

break;

}

case 3: {

double totalWholesaleProfit = 0;

double totalRetailProfit = 0;

for (const auto& product : products) {

double wholesaleProfit = product.calculateWholesaleProfit();

double retailProfit = product.calculateRetailProfit();

cout << "Доход от товара '" << product.getName() << "' по оптовой цене: " << wholesaleProfit << endl;

cout << "Доход от товара '" << product.getName() << "' по розничной цене: " << retailProfit << endl;

totalWholesaleProfit += wholesaleProfit;

totalRetailProfit += retailProfit;

}

cout << "Общий доход по оптовым ценам: " << totalWholesaleProfit << endl;

cout << "Общий доход по розничным ценам: " << totalRetailProfit << endl;

break;

}

case 4: {

cout << "Выход из программы." << endl;

return 0;

}

default:

cout << "Некорректный ввод. Пожалуйста, попробуйте снова." << endl;

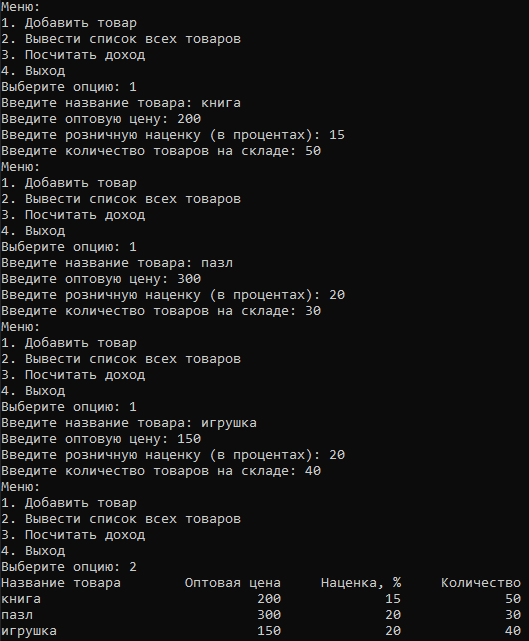
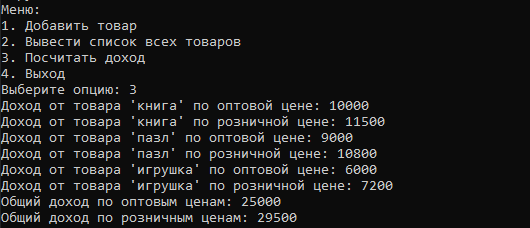
}

}

return 0;

}

***Результат:***

****** ******

***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:***

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <ctime>

#include <vector>

#include <sstream>

#include <windows.h>

using namespace std;

class Soft {

private:

string name;

string developer;

double size;

time\_t expirationDate;

public:

Soft() : name(""), developer(""), size(0.0), expirationDate(0) {}

Soft(const string& name, const string& developer, double size, time\_t expirationDate)

: name(name), developer(developer), size(size), expirationDate(expirationDate) {}

~Soft() {}

long long daysUntilExpiration() const {

time\_t now = time(nullptr);

long long secondsPerDay = 24LL \* 60 \* 60;

return (expirationDate - now) / secondsPerDay;

}

void printDetails() const {

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

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

cout << "Занимаемый объем: " << size << " МБ" << endl;

cout << "Дата завершения лицензии: " << ctime(&expirationDate);

cout << "Дней до завершения лицензии: " << daysUntilExpiration() << endl;

}

};

time\_t parseDate(const string& date) {

tm time = {};

stringstream ss(date);

ss >> time.tm\_mday;

ss.ignore();

ss >> time.tm\_mon;

time.tm\_mon--;

ss.ignore();

ss >> time.tm\_year;

time.tm\_year -= 1900;

time.tm\_hour = 0;

time.tm\_min = 0;

time.tm\_sec = 0;

return mktime(&time);

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<Soft> software;

while (true) {

cout << "Меню:" << endl;

cout << "1. Добавить программное обеспечение" << endl;

cout << "2. Посчитать количество дней до окончания лицензии" << endl;

cout << "3. Выход" << endl;

cout << "Выберите опцию: ";

int choice;

cin >> choice;

switch (choice) {

case 1: {

string name, developer;

double size;

string expirationDateStr;

cout << "Введите название программы: ";

cin.ignore();

getline(cin, name);

cout << "Введите разработчика: ";

getline(cin, developer);

cout << "Введите занимаемый объем в МБ: ";

cin >> size;

cout << "Введите дату завершения лицензии (в формате \"день, месяц, год\"): ";

cin.ignore();

getline(cin, expirationDateStr);

time\_t expirationDate = parseDate(expirationDateStr);

software.push\_back(Soft(name, developer, size, expirationDate));

break;

}

case 2: {

cout << "Список программного обеспечения:" << endl;

for (const Soft& item : software) {

item.printDetails();

cout << endl;

}

break;

}

case 3:

cout << "Выход из программы." << endl;

return 0;

default:

cout << "Некорректный ввод. Пожалуйста, попробуйте снова." << endl;

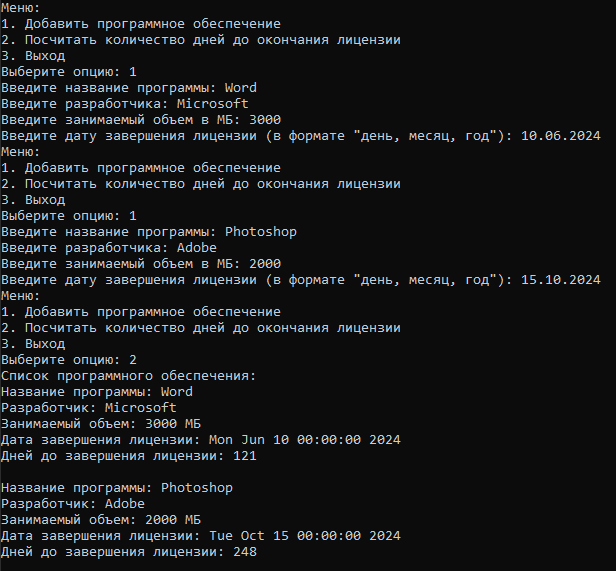
}

}

return 0;

}

***Результат:***

******