**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

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

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №3**

Выполнил: студент группы ПИ-3-21

Урманбетов Султан

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1**

**Постановка задачи:**

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.

**Исходные данные:**

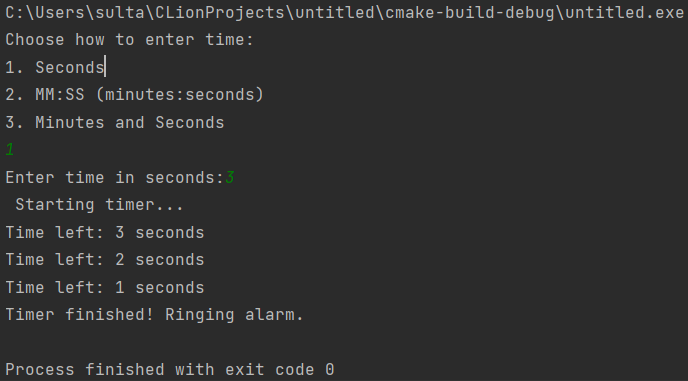
время

**Исходный код программы на C++:**

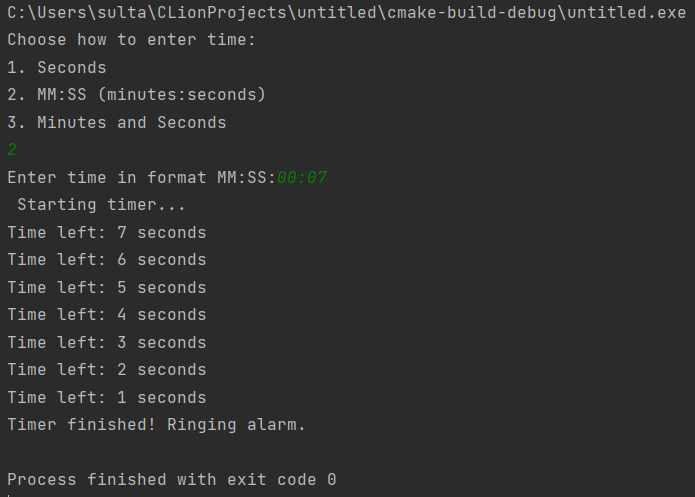
#include <bits/stdc++.h>  
using namespace std;  
class Timer {  
private:  
 int seconds;  
  
public:  
 explicit Timer(int sec) : seconds(sec) {}  
 explicit Timer(const string& time\_str) {  
 int minutes = 0;   
 sscanf(time\_str.c\_str(), "%d:%d", &minutes, &seconds);  
 this->seconds = minutes \* 60 + seconds;  
 }  
 Timer(int minutes, int seconds) : seconds(minutes \* 60 + seconds) {}  
  
 void run() {  
 cout<<"Starting timer..."<<endl;  
 while (seconds > 0) {  
 cout<<"Time left: "<<seconds<<" seconds"<<endl;  
 this\_thread::sleep\_for(chrono::seconds(1));  
 seconds--;  
 }  
 cout<<"Timer finished! Ringing alarm."<<endl;  
 }  
};  
  
int main() {  
 cout<<"Choose how to enter time:"<<endl;  
 cout<<"1. Seconds"<<endl;  
 cout<<"2. MM:SS (minutes:seconds)"<<endl;  
 cout<<"3. Minutes and Seconds"<<endl;  
 int choice;  
 cin>>choice;  
  
 if (choice == 1) {  
 cout<<"Enter time in seconds: ";  
 int sec;  
 cin>>sec;  
 Timer timer(sec);  
 timer.run();  
 } else if (choice == 2) {  
 cout<<"Enter time in format MM:SS: ";  
 string time\_str;  
 cin>>time\_str;  
 Timer timer(time\_str);  
 timer.run();  
 } else if (choice == 3) {  
 cout<<"Enter time in minutes and seconds: ";  
 int min, sec2;  
 cin>>min>>sec2;  
 Timer timer(min, sec2);  
 timer.run();  
 } else {  
 cout<<"Invalid choice."<<endl;  
 }  
 return 0;  
}

**Тесты:**

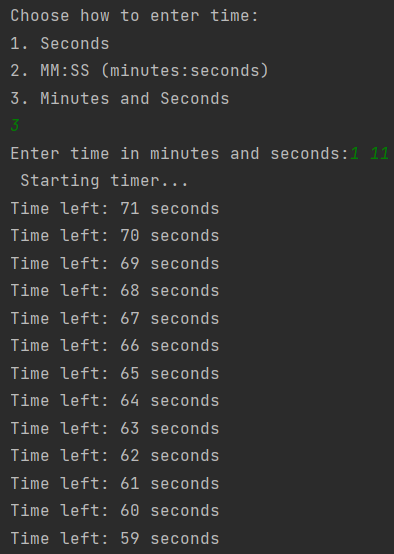
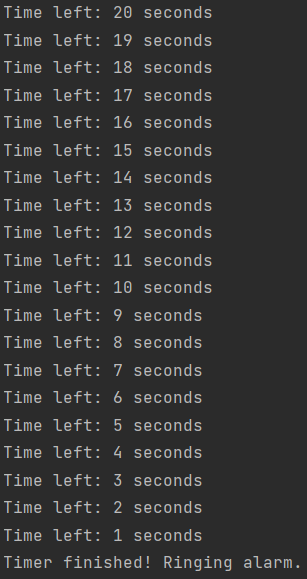
**Тест№1: Ввод секундами**

****

**Тест№2: Ввод в формате MM:SS**

****

**Тест№3: Ввод минутами и секундами**

**** ****

# **Задание №2**

**Постановка задачи: Опишите класс «студенческая группа».**

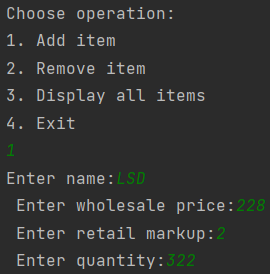
Создайте класс **Nomenclature**, описывающий товары на складе магазина. Закрытыми элементами класса будут: название товара, оптовая цена, розничная наценка и количество товаров на складе. Включите в класс открытые функции подсчета возможного чистого дохода при продаже этого товара и вывода всех данных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

**Код программы на C++:**

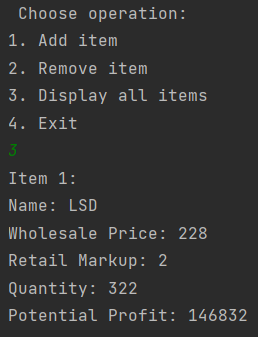
#include <bits/stdc++.h>  
  
#include <utility>  
using namespace std;  
const bool rakNaGoreNeSwistnet=true;  
class Nomenclature {  
private:  
 string name;  
 double wholesalePrice;  
 double retailMarkup;  
 int quantity;  
  
public:  
 explicit Nomenclature(string n = "", double wPrice = 0.0, double rMarkup = 0.0, int q = 0)  
 : name(std::move(n)), wholesalePrice(wPrice), retailMarkup(rMarkup), quantity(q) {}  
  
 [[nodiscard]] double calculateProfit() const {  
 double retailPrice = wholesalePrice \* (1 + retailMarkup);  
 return retailPrice \* quantity - wholesalePrice \* quantity;  
 }  
  
 void display() const {  
 cout<<"Name: "<<name<<endl;  
 cout<<"Wholesale Price: "<<wholesalePrice<<endl;  
 cout<<"Retail Markup: "<<retailMarkup<<endl;  
 cout<<"Quantity: "<<quantity<<endl;  
 cout<<"Potential Profit: "<<calculateProfit()<<endl;  
 }  
  
 void input() {  
 cout<<"Enter name: ";  
 cin>>name;  
 cout<<"Enter wholesale price: ";  
 cin>>wholesalePrice;  
 cout<<"Enter retail markup: ";  
 cin>>retailMarkup;  
 cout<<"Enter quantity: ";  
 cin>>quantity;  
 }  
};  
  
int main() {  
 vector<Nomenclature> store;  
  
 while (rakNaGoreNeSwistnet) {  
 cout<<"Choose operation:"<<endl;  
 cout<<"1. Add item"<<endl;  
 cout<<"2. Remove item"<<endl;  
 cout<<"3. Display all items"<<endl;  
 cout<<"4. Exit"<<endl;  
  
 int choice;  
 cin>>choice;  
  
 switch (choice) {  
 case 1: {  
 Nomenclature item;  
 item.input();  
 store.push\_back(item);  
 break;  
 }  
 case 2: {  
 if (!store.empty()) {  
 cout<<"Enter index of item to remove: ";  
 int index;  
 cin>>index;  
 index--;  
 if (index >= 0 && index < store.size()) {  
 store.erase(store.begin() + index);  
 cout<<"Item removed successfully."<<endl;  
 } else {  
 cout<<"Invalid index."<<endl;  
 }  
 } else {  
 cout<<"Store is empty."<<endl;  
 }  
 break;  
 }  
 case 3: {  
 if (!store.empty()) {  
 for (size\_t i = 0; i < store.size(); ++i) {  
 cout<<"Item "<<i+1<<":"<<endl;  
 store[i].display();  
 cout<<endl;  
 }  
 } else {  
 cout<<"Store is empty."<<endl;  
 }  
 break;  
 }  
 case 4:  
 cout<<"Exiting program."<<endl;  
 return 0;  
 default:  
 cout<<"Invalid choice. Try again."<<endl;  
 }  
 }  
}

**Тесты:**

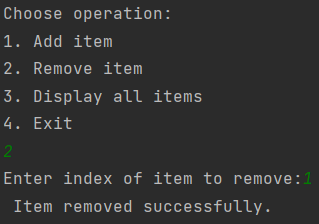
**Тест№1: Add Item**

****

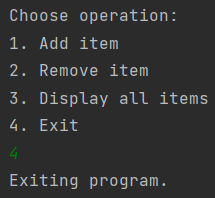
**Teст№2: Display all item**

****

**Teст№3: Remove item**

****

**Teст№4: Exit**

****

# **Задание №3**

**Постановка задачи:**

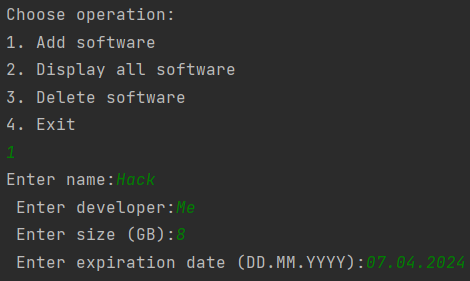
Создайте класс **Soft**, который содержит информацию об установленном программном обеспечении. Закрытыми элементами класса будут: название программы, разработчик, занимаемый объем, дата завершения лицензии. Включите в класс открытые функции подсчета количества дней до завершения лицензии и вывода всех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

**Код программы на C++:**

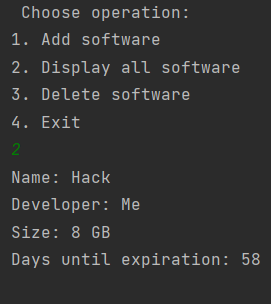
#include <bits/stdc++.h>  
using namespace std;  
  
class Soft {  
private:  
 string name;  
 string developer;  
 double size;  
 chrono::system\_clock::time\_point expirationDate;  
  
public:  
 explicit Soft(string n = "", string dev = "", double sz = 0.0)  
 : name(std::move(n)), developer(std::move(dev)), size(sz) {}  
  
 ~Soft() = default;  
  
 [[nodiscard]] int daysUntilExpiration() const {  
 auto now = chrono::system\_clock::now();  
 auto diff = expirationDate - now;  
 return chrono::duration\_cast<chrono::hours>(diff).count() / 24;  
 }  
  
 void display() const {  
 cout<<"Name: "<<name<<endl;  
 cout<<"Developer: "<<developer<<endl;  
 cout<<"Size: "<<size<<" GB"<<endl;  
 cout<<"Days until expiration: "<<daysUntilExpiration()<<endl;  
 }  
  
 void input() {  
 cout<<"Enter name: ";  
 cin.ignore();  
 getline(cin, name);  
 cout<<"Enter developer: ";  
 getline(cin, developer);  
 cout<<"Enter size (GB): ";  
 cin>>size;  
  
 cout<<"Enter expiration date (DD.MM.YYYY): ";  
 int day, month, year;  
 char discard;  
 cin>>day>>discard>>month>>discard>>year;  
  
 tm expiration\_tm = {};  
 expiration\_tm.tm\_year = year - 1900;  
 expiration\_tm.tm\_mon = month - 1;  
 expiration\_tm.tm\_mday = day;  
  
 expirationDate = chrono::system\_clock::from\_time\_t(mktime(&expiration\_tm));  
 }  
  
 [[nodiscard]] string getName() const {  
 return name;  
 }  
};  
  
int main() {  
 vector<Soft> softwareList;  
  
 while (true) {  
 cout<<"Choose operation:"<<endl;  
 cout<<"1. Add software"<<endl;  
 cout<<"2. Display all software"<<endl;  
 cout<<"3. Delete software"<<endl;  
 cout<<"4. Exit"<<endl;  
  
 int choice;  
 cin>>choice;  
  
 switch (choice) {  
 case 1: {  
 Soft software;  
 software.input();  
 softwareList.push\_back(software);  
 break;  
 }  
 case 2: {  
 if (!softwareList.empty()) {  
 for (const auto& software : softwareList) {  
 software.display();  
 cout<<endl;  
 }  
 } else {  
 cout<<"No software installed."<<endl;  
 }  
 break;  
 }  
 case 3: {  
 cout<<"Enter the name of the software to delete: ";  
 string nameToDelete;  
 cin.ignore();  
 getline(cin, nameToDelete);  
  
 bool found = false;  
 for (auto it = softwareList.begin(); it != softwareList.end(); ++it) {  
 if ((\*it).getName() == nameToDelete) {  
 softwareList.erase(it);  
 found = true;  
 break;  
 }  
 }  
  
 if (!found) {  
 cout<<"Software not found."<<endl;  
 } else {  
 cout<<"Software deleted successfully."<<endl;  
 }  
 break;  
 }  
 case 4:  
 cout<<"Exiting program."<<endl;  
 return 0;  
 default:  
 cout<<"Invalid choice. Try again."<<endl;  
 }  
 }  
}

**Тесты:**

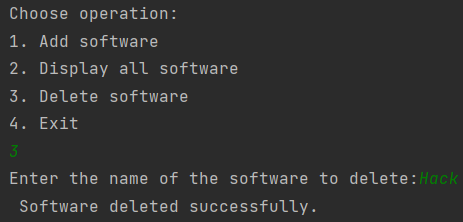
**Тест№1: Add software**

****

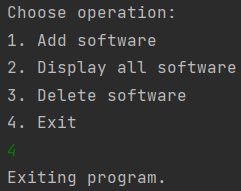
**Тест№2: Display all software**

****

**Тест№3: Delete software**



**Тест№4: Exit**



**Ответы на вопросы:**

1. **Что такое конструктор? Как он вызывается на выполнение?**
2. **Особенности конструктора, отличающие его от других методов класса?**
3. **Что такое конструктор по умолчанию? В каком случае следует его использовать?**
4. **Способы инициализации в конструкторе по умолчанию? Какой способ инициализации предпочтительнее?**
5. **Что такое конструктор с аргументами? Когда используют конструктор с аргументами?**
6. **Какие преимущества у конструктора с аргументами перед методом Set()?**
7. **Что такое деструктор? Для чего его используют?**
8. **Особенности деструктор**
9. **В какой последовательности выполняется конструкторы, в какой деструкторы классов?**
10. **Можно ли перегружать конструктор?**
11. **Что такое конструктор копирования?**