**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

**Икрамов Абдул Муним**

Проверил: Мусабаев Э.Б.

**Бишкек 2023**

# **Лабораторная работа №3**

**Постановка задачи:**

**Тема: Конструктор и деструктор**

**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 конструк-тора, 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 ти-ков, прошедших с момента запуска программы. Поделив это значение на макрос CLK\_TCK, получаем значение в секундах. Прототипы для clock() и CLK\_TCK содержатся в заголовоч-ном файле time.h.

**Исходный код программы на C++:**

*#include* <iostream>

*#include* <chrono>

*#include* <thread>

using namespace std;

class Timer {

private:

int seconds;

public:

*// Конструктор, время в секундах*

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

*// Конструктор, время в формате строки (чч:мм:сс)*

Timer(const string& *timeString*) {

int hours, minutes, secs;

sscanf(*timeString*.c\_str(), "%d:%d:%d", &hours, &minutes, &secs);

seconds = hours \* 3600 + minutes \* 60 + secs;

}

*// Конструктор, время в минутах и секундах*

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

*// Функция запуска таймера*

void run() {

*while* (seconds > 0) {

cout << "Осталось времени: " << seconds << " секунд\n";

cout << "Тик-так...\n";

cout << "------------------------\n";

*//добавление заддержки*

this\_thread::sleep\_for(chrono::seconds(1));

seconds--;

}

cout << "Время истекло! Звонок!\n";

}

};

int main() {

Timer timer1(10);

timer1.run();

Timer timer2("00:00:15");

timer2.run();

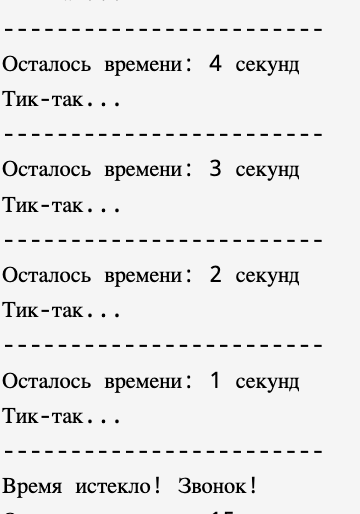
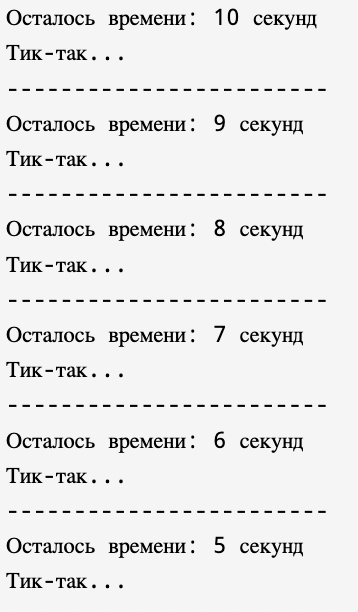
Timer timer3(0, 20);

timer3.run();

*return* 0;

}

Тест:



**2.** Создайте класс Nomenclature, описывающий товары на складе магазина.

Закрытыми элемен-тами класса будут: название товара, оптовая цена, розничная наценка и количество товаров на складе. Включите в класс открытые функции подсчета возможного чистого дохода при прода-же этого товара и вывода всех данных о товаре на экран. Для инициализации и удаления объек-тов класса используйте конструкторы и деструкторы.

**Исходный код программы на C++:**

*#include* <iostream>

*#include* <string>

using namespace std;

class Nomenclature {

private:

string itemName;

double wholesalePrice;

double retailMarkup;

int stockQuantity;

public:

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

: itemName(*name*), wholesalePrice(*wholesalePrice*), retailMarkup(*retailMarkup*), stockQuantity(*quantity*) {}

*// Деструктор класса*

~Nomenclature() {

cout << "Объект " << itemName << " удален.\n";

}

double calculateNetIncome() const {

*return* stockQuantity \* retailMarkup;

}

void displayProductInfo() const {

cout << "Название товара: " << itemName << endl;

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

cout << "Розничная наценка: $" << retailMarkup << endl;

cout << "Количество на складе: " << stockQuantity << " шт." << endl;

cout << "Возможный чистый доход: $" << calculateNetIncome() << endl;

}

};

int main() {

Nomenclature product1("Лаптоп", 800, 200, 10);

product1.displayProductInfo();

cout << "\n";

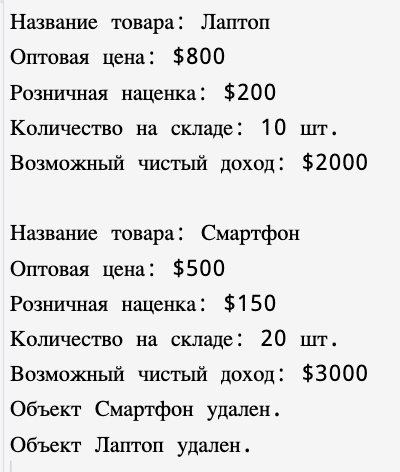
Nomenclature product2("Смартфон", 500, 150, 20);

product2.displayProductInfo();

*return* 0;

}

Тесты:



**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.

**Исходный код программы на C++:**

*#include* <iostream>

*#include* <string>

*#include* <chrono>

using namespace std;

class Soft {

private:

string programName;

string developer;

double diskSpace;

chrono::system\_clock::time\_point licenseExpirationDate;

public:

Soft(const string& *name*, const string& *dev*, double *space*, const chrono::system\_clock::time\_point& *expirationDate*)

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

*// Деструктор класса*

~Soft() {

cout << "Программа " << programName << " удалена.\n";

}

*// Функция подсчета количества дней до завершения лицензии*

int daysUntilExpiration() const {

auto now = chrono::system\_clock::now();

auto remainingTime = chrono::duration\_cast<chrono::hours>(licenseExpirationDate - now);

*return* remainingTime.count() / 24; *// Переводим часы в дни*

}

*// Функция вывода данных о программном обеспечении на экран*

void displaySoftwareInfo() const {

auto expirationDateT = chrono::system\_clock::to\_time\_t(licenseExpirationDate);

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

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

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

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

int daysLeft = daysUntilExpiration();

*if* (daysLeft <= 0) {

cout << "Лицензия истекла." << endl;

} *else* {

cout << "Осталось дней до завершения лицензии: " << daysLeft << " дней" << endl;

}

}

};

int main() {

auto expirationDate = chrono::system\_clock::now() + chrono::hours(0 \* 24);

Soft software1("Microsoft Office", "Microsoft", 500, expirationDate);

software1.displaySoftwareInfo();

cout << "\n";

auto expirationDate2 = chrono::system\_clock::now() + chrono::hours(10 \* 24);

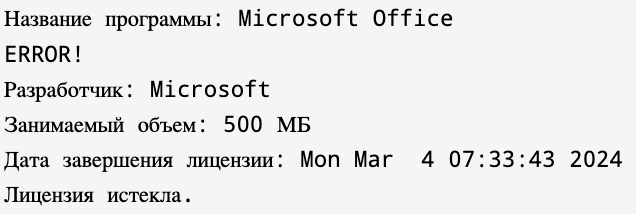
Soft software2("Adobe Photoshop", "Adobe", 1000, expirationDate2);

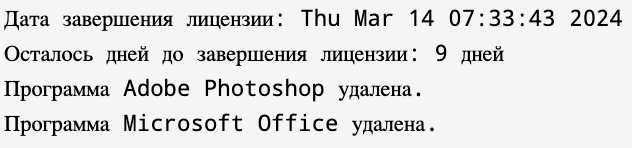
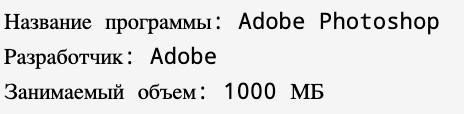
software2.displaySoftwareInfo();

*return* 0;

}

Тесты:

****

****

**Ответы на вопросы:**

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

Конструктор — это специальный метод класса, который инициализирует поля объектазначениями в момент создания объекта автоматически.

Всю необходимую работу по вызову конструктора выполняет компилятор.

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

1. Имя конструктора совпадает с именем класса.

2. У конструктора нет возвращаемого значения, даже результата типа void.  
Это объясняется тем, что конструктор автоматически вызывается системой, следовательно, не существует вызывающей функции, которой конструктор мог бывозвратить значение. Значит, указание возвращаемого типа в конструкторе не имеетсмысла.

3. Конструктор выполняется при создании объектов класса. Вернее, при создании каждогообъекта класса.

4. Конструктор не наследуется.

5. Конструктор может иметь аргументы, а может не иметь.

6. Конструктор можно перегружать.

7. Конструктор, отработав, запускает деструктор.

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

**Конструктор по умолчанию —** это конструктор, который может быть вызван безпередачи аргументов.

Для того, чтобы быть уверенным в том, что все поля объекта первоначальноинициализированы корректными значениями, используется первоначальная инициализацияполей всех объекта одинаковыми значениям автоматически в момент его создания. Чащевсего нулевыми значениями (для первоначальной очистки данных).

Использование этого конструктора удобно при наличии большого количества полей. В этом случае конструктор подстраховывает, чтобы не забыть инициализировать все поля. А потом можно присваивают другие значения, отличные от нуля, или всем или определенным полям объекта.

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

1. Инициализация полей объекта с помощью **списка инициализации:**

class Count {

   int count;

   Count() : count(0) {}

};

2. Инициализация полей объекта с помощью присвоения им значений в теле конструктора:

class Count {

   int count;

   Count() {

       count = 0;

   }

};

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

**Конструктор с аргументами** – это конструктор, имеющий хотя бы один аргумент, может инициализировать поля значениями, переданными ему в качестве аргументов. К тому же этот конструктор значительно упрощает код программы.

**6. Какие преимущества у конструктора с аргументами перед методом Set()?**

Уменьшение кода при одинаковых результатах.

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

Существует другой специальный метод класса, который автоматически вызывается при уничтожении объекта/ов, созданного/ых конструктором. Этот метод называется **деструктором**. Он работает в паре с конструктором.

Конструктор инициализирует поля объекта, а также отслеживает этот объект, пока он не выполнит свою задачу. В этот момент конструктор вызывает деструктор, который должен освободить памяти от отработанного объекта и весь оставшийся мусор от работы объекта.

Если память под объект отведена в динамической памяти с помощью спецификаторов newи delete, писать деструктор надо обязательно. Если же new не используется, писать деструктор все равно нужно для последующих проверок деструктором работы класса и удаления нединамических объектов.

**8. Особенности деструктор**

1. Деструктор имеет имя, совпадающее с именем конструктора, а, следовательно, и с именем класса.

2. Перед деструктором пишут символ тильда **~**.

3. Аргументов не имеет.

4. Возвращаемого значения не имеет.

5. Пишется следом за конструктором.

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

Конструкторы базовых классов вызываются в порядке их объявления, а деструкторы вызываются в обратном.

**10. Можно ли перегружать конструктор?**

Да, для перегрузки используются конструкторы с одним именем, но отличающиеся количеством аргументов.

**11. Что такое конструктор копирования?**

**Конструктор копирования** – это конструктор, с помощью которого возможно сделать третий способ инициализации полей объекта. При этом способе для инициализации полей объекта используются поля уже существующего объекта. Для этого не нужно самим создавать специальный конструктор, поскольку такой конструктор предоставляется компилятором для каждого создаваемого класса и называется конструктором копирования по умолчанию. Копирующий конструктор имеет единственный аргумент, являющийся объектом того же класса, что и конструктор.