Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: Лабораторная работа №2: «Классы и объекты. Использование конструкторов».

**Вариант 13**

Выполнил студент ИВТ-22-2б:

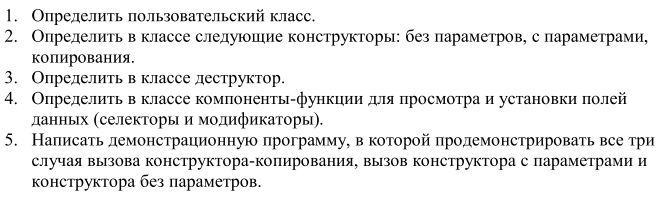
Коняев Александр Сергеевич

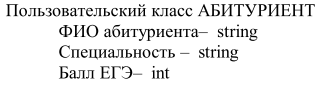
Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**





**Анализ задачи**

* Конструктор без параметров
* Конструктор с параметрами
* Конструктор копирования
* Метод get\_fio – получение фио
* Метод set\_fio – запись фио
* Метод get\_spec – получение специальности
* Метод set\_spec – запись специальности
* Метод get\_count – получение баллов ЕГЭ
* Метод set\_fio – запись баллов ЕГЭ
* Метод show – вывод данных об абитуриенте
* Функция make\_abiturient – создаёт абитуриента
* Функция print\_abiturient – выодит данные об абитуриенте

**UML – диаграмма**

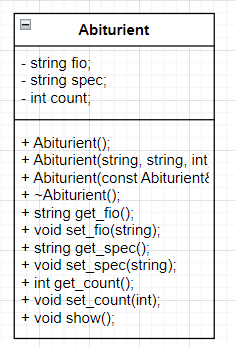


Рисунок 1 – UML-диаграмма класса «abiturient»

**Код программы.**

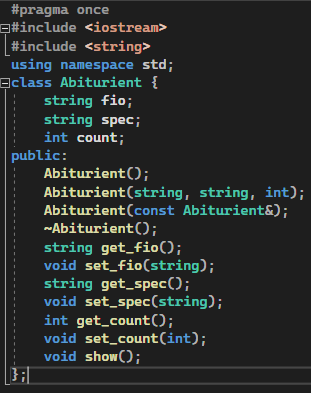


Рисунок 2 – Описание класса «abiturient»

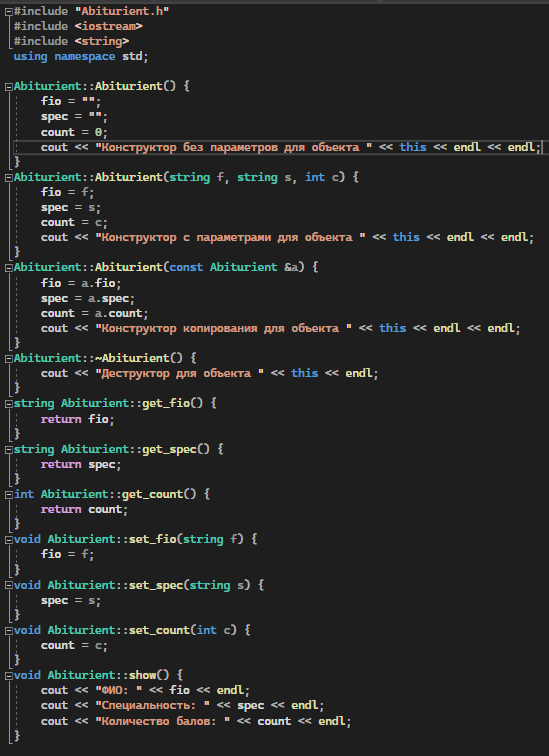


Рисунок 3 – Класс «abiturient»

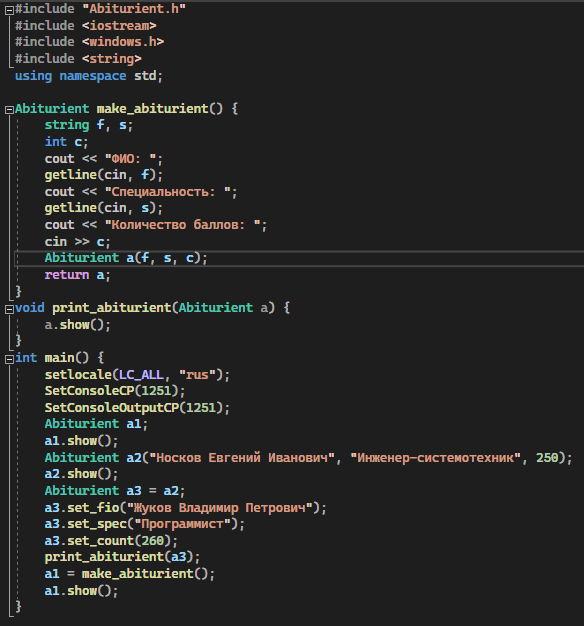


Рисунок 4 – Функция main

**Вывод программы.**

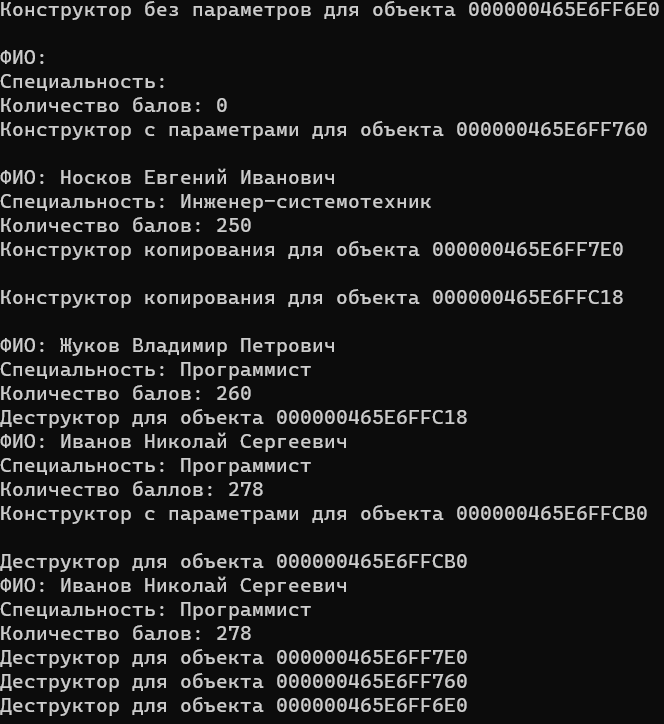
****

Рисунок 5 – Вывод программы

**Ответы на вопросы**

1. Конструктор в C++ используется для инициализации объектов класса. Он вызывается автоматически при создании объекта класса и позволяет задать начальные значения для его членов. Конструктор может принимать параметры, которые могут использоваться для инициализации членов класса. Кроме того, конструктор может выполнять дополнительные действия при создании объекта, например, выделять память или открывать файлы. Использование конструктора позволяет обеспечить правильную инициализацию объектов и упростить код программы.
2. 1. Конструктор по умолчанию (default constructor) - создает объект без параметров и инициализирует его значениями по умолчанию.

2. Конструктор с параметрами (parameterized constructor) - создает объект с заданными параметрами и инициализирует его значениями, переданными в конструктор.

3. Копирующий конструктор (copy constructor) - создает новый объект, инициализируя его значениями другого объекта того же класса. Копирующий конструктор используется, когда нужно создать копию объекта, например, при передаче объекта по значению в функцию или при возвращении объекта из функции.

1. Деструктор в C++ используется для освобождения ресурсов, которые были выделены объекту во время его жизни. Деструктор вызывается автоматически при уничтожении объекта, например, когда объект выходит из области видимости или, когда он удаляется явно с помощью оператора delete.

Деструктор может быть описан явно, если объект класса использует динамически выделенную память, открывает файлы или устанавливает соединения с базами данных. В таких случаях деструктор должен освободить выделенную память, закрыть файлы или разорвать соединения с базами данных, чтобы избежать утечек памяти или других проблем.

1. 1. Конструктор без параметров (default constructor) - используется для создания объекта без передачи параметров. Он инициализирует значения членов класса значениями по умолчанию. Конструктор без параметров может быть полезен, когда объект класса не требует специальной инициализации.

2. Конструктор с параметрами (parameterized constructor) - используется для создания объекта с передачей параметров. Он принимает значения параметров и инициализирует соответствующие члены класса. Конструктор с параметрами может быть полезен, когда объект класса требует специальной инициализации.

3. Копирующий конструктор (copy constructor) - используется для создания нового объекта на основе существующего объекта того же класса. Он копирует значения членов класса из одного объекта в другой. Копирующий конструктор может быть полезен, когда нужно создать копию объекта, например, при передаче объекта по значению в функцию или при возвращении объекта из функции.

1. Конструктор копирования в C++ вызывается в следующих случаях:

1. При передаче объекта по значению в функцию. Если объект передается в функцию по значению, то создается копия объекта, используя конструктор копирования.

2. При возвращении объекта из функции. Если функция возвращает объект, то создается копия объекта, используя конструктор копирования.

3. При инициализации одного объекта другим объектом того же класса. Например, если объявлен объект класса и затем создан новый объект, используя уже существующий объект в качестве инициализатора.

4. При использовании объекта в списке инициализации другого конструктора. Например, если создается объект класса, который содержит объект другого класса в качестве члена, то объект этого класса может быть инициализирован с помощью конструктора копирования.

1. Свойства конструкторов в C++:

1. Имя конструктора совпадает с именем класса.

2. Конструктор не имеет возвращаемого значения.

3. Конструктор может быть перегружен для поддержки различных типов параметров.

4. Конструктор вызывается автоматически при создании объекта класса.

5. Конструктор может использоваться для инициализации членов класса значениями по умолчанию или переданными параметрами.

6. Конструктор может быть определен явно или использоваться конструктор по умолчанию.

7. Конструктор может быть использован для выполнения дополнительных действий при создании объекта, например, для выделения памяти или открытия файлов.

8. Конструктор может быть использован для проверки корректности параметров, переданных в объект.

9. Конструктор копирования используется для создания копии объекта.

10. Деструктор используется для освобождения ресурсов, выделенных объекту во время его жизни.

1. Свойства деструкторов в C++:

1. Имя деструктора совпадает с именем класса, но начинается с символа "~".

2. Деструктор не имеет параметров и не имеет возвращаемого значения.

3. Деструктор вызывается автоматически при уничтожении объекта класса.

4. Деструктор используется для освобождения ресурсов, выделенных объекту во время его жизни, например, для освобождения динамически выделенной памяти, закрытия файлов или разрыва соединений с базами данных.

5. Деструктор может быть определен явно или использоваться деструктор по умолчанию.

6. Деструктор может быть использован для выполнения дополнительных действий при уничтожении объекта, например, для сохранения данных в файл или для отправки сообщений.

7. Деструктор может быть использован для проверки корректности состояния объекта перед его уничтожением.

8. Деструктор может быть виртуальным, если класс содержит виртуальные функции, чтобы гарантировать правильное уничтожение объекта при использовании наследования.

9. Если класс содержит динамически выделенную память, то деструктор должен освободить эту память, чтобы избежать утечек памяти.

1. Методы класса имеют доступ ко всем атрибутам класса, включая закрытые (private), защищенные (protected) и открытые (public) члены класса.
2. Указатель this в C++ представляет собой указатель на текущий объект класса, для которого вызывается метод. Он используется для доступа к членам объекта и для передачи объекта в качестве параметра в другие методы.
3. Методы, определенные в классе и вне класса, имеют различия в доступе к членам класса и в использовании ключевых слов.

Методы, определенные в классе, называются методами-членами (member functions) и имеют доступ к закрытым (private) и защищенным (protected) членам класса. Они могут быть вызваны только для объектов класса и могут использовать ключевое слово this для доступа к членам объекта. Методы-члены также могут быть перегружены и виртуальными.

Методы, определенные вне класса, называются методами-нечленами (non-member functions) и не имеют доступа к закрытым и защищенным членам класса. Они могут быть вызваны для объектов класса и для других объектов, но не могут использовать ключевое слово this. Методы-нечлены могут быть перегружены и могут быть объявлены как дружественные (friend) классу, чтобы иметь доступ к закрытым и защищенным членам.

1. Конструктор в C++ не возвращает значение, так как его основная задача - инициализировать объект класса. Конструктор вызывается автоматически при создании объекта, и его задача заключается в том, чтобы инициализировать члены объекта в соответствии с заданными параметрами или значениями по умолчанию. После того, как объект был успешно создан и инициализирован, конструктор завершает свою работу.
2. В C++ создаются следующие методы по умолчанию:

1. Конструктор по умолчанию - создается автоматически, если не определен ни один конструктор.

2. Конструктор копирования - создается автоматически, если не определен ни один конструктор копирования.

3. Оператор присваивания - создается автоматически, если не определен ни один оператор присваивания.

4. Деструктор - создается автоматически, если не определен ни один деструктор.

1. Деструктор в C++ не возвращает значения, так как его основное назначение - это освобождение ресурсов, выделенных объекту во время его жизни.
2. Конструктор без параметров.
3. В данном случае будет вызван конструктор по умолчанию класса student, так как при создании объекта с помощью оператора new выделяется память под объект и вызывается его конструктор.
4. Конструктор с параметрами.
5. Конструктор с параметрами и копирования.
6. Конструктор с параметрами, без параметров и копирования.
7. Без параметров.
8. С помощью метода set\_name.