Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Лабораторная работа №2

Классы и объекты. Использование конструкторов.

Выполнил

Студент группы РИС-22-2б

Вековшинин Д. А.

Проверила

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2023

Постановка задачи

1. Определить пользовательский класс. 2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.

3. Определить в классе деструктор.

4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы). 5. Написать демонстрационную программу, в которой продемонстрировать все три случая вызова конструктора-копирования, вызов конструктора с параметрами и конструктора без параметров

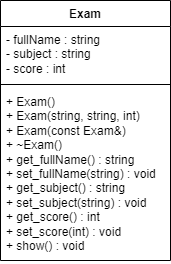
Вариант 10: Пользовательский класс ЭКЗАМЕН

ФИО студента – string

Предмет – string

Оценка – int

UML-диаграмма



Определение компонентных функций

//конструктор без параметров

Exam::Exam()

{

fullName = "";

subject = "";

score = 0;

cout << "Конструктор без параметров для объекта " << this << endl;

}

//конструктор с параметрами

Exam::Exam(string n, string su, int sc)

{

fullName = n;

subject = su;

score = sc;

cout << "Конструктор с параметрами для объекта " << this << endl;

}

//конструктор копирования

Exam::Exam(const Exam& e)

{

fullName = e.fullName;

subject = e.subject;

score = e.score;

cout << "Конструктор копирования для объекта " << this << endl;

}

//деструктор

Exam::~Exam()

{

cout << "Деструктор для объекта " << this << endl;

}

//селекторы

string Exam::get\_fullName()

{

return fullName;

}

string Exam::get\_subject()

{

return subject;

}

int Exam::get\_score()

{

return score;

}

//модификаторы

void Exam::set\_fullName(string n)

{

fullName = n;

}

void Exam::set\_subject(string su)

{

subject = su;

}

void Exam::set\_score(int sc)

{

score = sc;

}

//метод для просмотра атрибутов

void Exam::show()

{

cout << "ФИО: " << fullName << endl;

cout << "Предмет: " << subject << endl;

cout << "Оценка: " << score << endl;

}

Определение функций make\_() и print\_()

Exam make\_exam()

{

string surname;

string name;

string patronymic;

string fullName;

string subject;

int score;

cout << "Введите ФИО: ";

cin >> surname >> name >> patronymic;

fullName = surname + ' ' + name + ' ' + patronymic;

cout << "Введите название предмета: ";

cin >> subject;

cout << "Введите оценку: ";

cin >> score;

Exam e(fullName, subject, score);

return e;

}

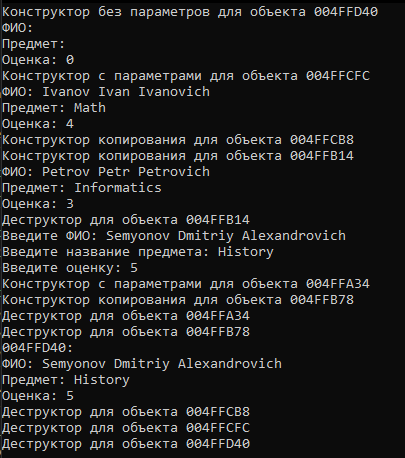
void print\_exam(Exam t)

{

t.show();

}

Результаты работы программы



Ответы на контрольные вопросы

1. Для чего нужен конструктор?

Для инициализации объекта.

Exam::Exam(string n, string su, int sc)

{

fullName = n;

subject = su;

score = sc;

}

1. Сколько типов конструкторов существует в С++?

Существует три типа конструкторов:

1) конструктор с параметрами.

2) конструктор без параметров.

3) конструктор копирования.

//конструктор без параметров

Exam::Exam()

{

fullName = "";

subject = "";

score = 0;

}

//конструктор с параметрами

Exam::Exam(string n, string su, int sc)

{

fullName = n;

subject = su;

score = sc;

}

//конструктор копирования

Exam::Exam(const Exam& e)

{

fullName = e.fullName;

subject = e.subject;

score = e.score;

}

1. Для чего используется деструктор? В каких случаях деструктор описывается явно?

Для освобождения ресурсов, выделенных конструктором объекту.

Описывать в классе деструктор явным образом требуется в случае, когда объект содержит указатели на память, выделяемую динамически — иначе при уничтожении объекта память, на которую ссылались его поля-указатели, не будет помечена как свободная.

Деструктор для класса Person, в котором поле name реализуется как динамическая

строка будет выглядеть так:

Person::~Person() {delete [] name;}

1. Для чего используется конструктор без параметров? Конструктор с параметрами? Конструктор копирования?

Конструктор с параметрами используется для инициализации объекта требуемыми значениями;

Конструктор без параметров используется для создания «пустого» объекта;

Конструктор копирования используется для создания объекта, аналогичного тому, который уже существует.

//конструктор без параметров

Exam::Exam()

{

fullName = "";

subject = "";

score = 0;

}

//конструктор с параметрами

Exam::Exam(string n, string su, int sc)

{

fullName = n;

subject = su;

score = sc;

}

//конструктор копирования

Exam::Exam(const Exam& e)

{

fullName = e.fullName;

subject = e.subject;

score = e.score;

}

1. В каких случаях вызывается конструктор копирования?

Этот конструктор вызывается в тех случаях, когда новый объект создается путем

копирования существующего:

- при описании нового объекта с инициализацией другим объектом;

- при передаче объекта в функцию по значению;

- при возврате объекта из функции

//конструктор копирования

Exam::Exam(const Exam& e)

{

fullName = e.fullName;

subject = e.subject;

score = e.score;

}

1. Перечислить свойства конструкторов.

1) Конструктор не возвращает значение, даже типа void. Нельзя получить указатель на конструктор.

2) Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации (при этом используется механизм перегрузки).

3) Конструктор, вызываемый без параметров, называется конструктором по умолчанию.

4) Параметры конструктора могут иметь любой тип, кроме этого же класса.

Можно задавать значения параметров по умолчанию, но их может содержать

только один из конструкторов.

5) Если программист не указал ни одного конструктора, компилятор создает его автоматически. Такой конструктор вызывает конструкторы по умолчанию для полей класса. В случае, когда класс содержит константы или ссылки, при попытке создания объекта класса будет выдана ошибка, поскольку их необходимо инициализировать конкретными значениями, а конструктор по умолчанию этого делать не умеет.

6) Конструкторы не наследуются.

7) Конструкторы нельзя описывать с модификаторами const, virtual и static.

8) Конструкторы глобальных объектов вызываются до вызова функции main.

Локальные объекты создаются, как только становится активной область их

действия. Конструктор запускается и при создании временного объекта (на

пример, при передаче объекта из функции).

9) Конструктор вызывается, если в программе встретилась какая-либо из син- таксических конструкций:

имя\_класса имя\_объекта [(список параметров)];// Список параметров

не должен быть пустым

имя класса (список параметров);// Создается объект без имени

(список может быть пустым)

имя\_класса имя\_объекта = выражение;// Создается объект без имени и

копируется

Примеры:

Person p1(”Ivanov”,23);//конструктор с параметрами

Person p2();//конструктор без параметров

Person p3=p1;// конструктор копирования

Person p4=Person (”Sidorov”,20);//создается объект без имени и

копируется

Person\* pp1=new(Person);//указатель на пустой объект

Person\*pp2=new Person(”Petrov”,32);//указатель на объект

1. Перечислить свойства деструкторов.

1) не имеет аргументов и возвращаемого значения;

2) не наследуется;

3) не может быть объявлен как const или static;

4) может быть виртуальным.

Если деструктор явным образом не определен, компилятор автоматически создает пустой деструктор.

1. К каким атрибутам имеют доступ методы класса?

Методы класса имеют неограниченный доступ ко всем элементам класса,

независимо от спецификаторов доступа и порядка объявления методов в классе.

class Exam

{

//атрибуты

string fullName;

string subject;

int score;

public:

Exam();//конструктор без параметров

Exam(string, string, int);//конструктор с параметрами

Exam(const Exam&);//конструктор копирования

~Exam();//деструктор

};

1. Что представляет собой указатель this?

Метод (кроме статических методов) неявно получает в качестве аргумента указатель на тот объект, для которого он вызван. Этот указатель обозначается ключевым словом this и может быть использован в теле метода. В явном виде этот указатель применяется в основном для возвращения из метода указателя (return this;) или ссылки (return \*this;) на вызвавший объект.

Пример:

Person&Old(Person&P)

{

if(P.GetAge()>60) return \*this;

}

Запись \*this представляет собой значение текущего объекта.

1. Какая разница между методами, определенными внутри класса и вне класса?

Определение метода внутри класса ничем не отличается от определения обычной функции. По умолчанию такой метод считается встроенной функцией (inline). Если метод определяется вне функции, то принадлежность

метода классу указывается с помощью имени класса: Имя\_класса::Имя\_метода. В классе присутствует только прототип.

class Exam

{

Exam(string, string, int);//конструктор с параметрами

};

//конструктор с параметрами

Exam::Exam(string n, string su, int sc)

{

fullName = n;

subject = su;

score = sc;

cout << "Конструктор с параметрами для объекта " << this << endl;

}

1. Какое значение возвращает конструктор?

Конструктор не возвращает значение, даже типа void.

//конструктор без параметров

Exam::Exam() {};

1. Какие методы создаются по умолчанию?

Конструктор без параметров и конструктор копирования

1. Какое значение возвращает деструктор?

Не имеет возвращаемого значения

1. Дано описание класса

class Student

{

string name;

int group;

public:

student(string, int);

student(const student&)

~student();

};

Какой метод отсутствует в описании класса?

student();//конструктор без параметров

1. Какой метод будет вызван при выполнении следующих операторов:

student\*s;

s=new student;

student();//конструктор без параметров

1. Какой метод будет вызван при выполнении следующих операторов:

student s(“Ivanov”,20);

student(string, int);//конструктор с параметрами

1. Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20);

student s2=s1;

student(string, int);//конструктор с параметрами

student(const student&);//конструктор копирования

1. Какие методы будут вызваны при выполнении следующих операторов:

student s1(“Ivanov”,20);

student s2;

s2=s1;

student();//конструктор без параметров

student(string, int);//конструктор с параметрами

student(const student&);//конструктор копирования

1. Какой конструктор будет использоваться при передаче параметра в функцию print():

void print(student a)

{a.show();}

student(const student&);//конструктор копирования

1. Класс описан следующим образом:

class Student

{

string name;

int age;

public:

void set\_name(string);

void set\_age(int );

…..

};

Student p;

Каким образом можно присвоить новое значение атрибуту name объекта р?

