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:

Выполнил студент ИВТ-22-2б:

Шилов Валерий Михайлович

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

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.

**Описание класса**

Пользовательский класс КНИГА

Название – string

Автор – string

Год издания – int

**UML-диаграмма**

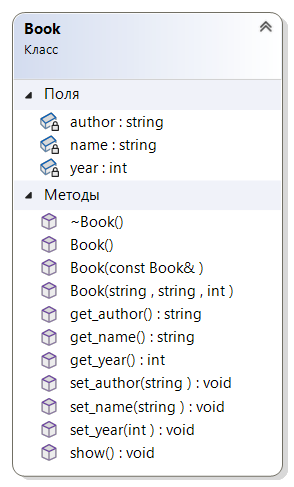


Рисунок 1 – UML-диаграмма

Описание функций

В функция make\_book() вводятся значения для полей экземпляра класса и экземпляр инициализируется этими значениями с помощью конструктора с параметрами. Функция print\_book() в качестве параметра принимает экземпляр класса и реализует вывод полей экземпляра с помощью метода show().

**Результат работы программы**

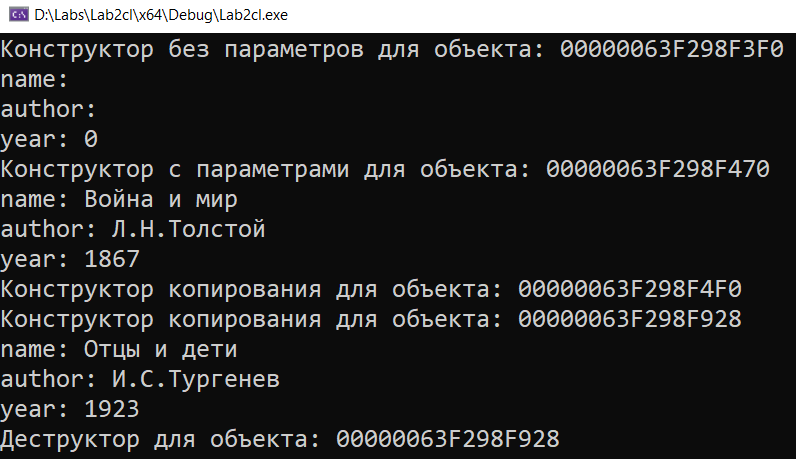


Рисунок 2 – результат работы программы

**Контрольные вопросы**

1. Для чего нужен конструктор?

Конструкторы - это методы, которые предназначены для инициализации объекта. Использование этих методов позволяет задать экземпляру класса необходимые значения для полей.

2. Сколько типов конструкторов существует в С++?

Существует три типа конструкторов:

- Конструктор с параметрами (объекту задаются вводимые значения)

- Конструктор без параметров (объекту задаются либо “пустые поля”, либо поля по умолчанию)

- Конструктор копирования (в качестве параметров подаются не отдельные поля, а объект того же класса, и конструируемому объекту задаются аналогичные поля)

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

Деструктор - это метод, которые освобождает выделенную под объект память. Деструктор вызывается автоматически при выходе из области видимости объекта. Описывать деструктор явным образом необходимо, когда объект содержит указатели на динамическую память, так как при вызове автоматического деструктора очистится память под сам объект, а память, на которую ссылались поля-указатели, будет всё ещё занята.

Пример деструктора:

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

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

Конструктор с параметрами используется, когда объекту требуется задать конкретные вводимые значения.

Пример конструктора с параметрами:

class Point {

int x, y;

public:

//конструктор с параметрами

Point(int X, int Y) {

x = X;

y = Y; }

void Print\_Point() {cout << “x: “ << x << “\ty: “ << y;}

};

int main() {

Point P(1, 2);

P.Print\_Point(); //X: 1 Y: 2

return 0;

}

Конструктор без параметров используется, когда объекту задаются либо “пустые поля”, либо поля по умолчанию.

Пример конструктора без параметров:

Point::Point() {

x = 0;

y = 0;

}

int main() {

Point P;

P.Print\_Point(); // X: 0 Y: 0

return 0;

}

Конструктор копирования используется, когда необходимо скопировать значения одного объекта другому. В качестве параметров передаются не отдельные поля, а объект того же класса, и конструируемому объекту задаются аналогичные поля.

Пример конструктора копирования:

Point::Point(const Point& p) {

x = p.x;

y = p.y;

}

int main() {

Point P(3, 4);

Point E = P;

E.Print\_Point(); //X: 3 Y: 4

return 0;

}

5. В каких случаях вызывается конструктор копирования?

Конструктор копирования вызывается в следующих случаях:

- Был описан конструктор копирования, и использован оператор присваивания (пример из вопроса (4))

- Объект передаётся в функцию (метод) в качестве параметра по значению. Тогда компилятор создает временный (локальный) объект, и использует конструктор копирования по умолчанию

void print(Point a) {a.Print\_Point();} //в функции создается временный объект, и метод Print\_Point применяется ко временному объекту, а не к тому, что передается в функцию.

int main() {

Point b(1, 2);

print(b); //x: 1 Y: 2

Объект Point a, который создаётся в функции, является копией объекта Point b

- Объект возвращается как значение функции

Point set\_info() {

Point s; //создаётся временный объект

s.x = 1;

s.y = 2;

return s; //объект возвращается как значение функции

}

int main() {

Point d = set\_info();

d.Print\_Point(); //X: 1 Y: 2

return 0;

}

Объект Point d является копией объекта Point s, который создаётся в функции.

6. Перечислить свойства конструкторов.

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

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

- Конструктор, вызываемый без параметров, называется конструктором по умолчанию.

- Параметры конструктора могут иметь любой тип, кроме этого же класса. Можно задавать значения параметров по умолчанию, но их может содержать только один из конструкторов.

- 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.

- Конструкторы не наследуются.

- Конструкторы нельзя описывать с модификаторами const, virtual и static.

- 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 (например, при передаче объекта из функции).

- Конструктор вызывается, если в программе встретилась какая-либо из синтаксических конструкций:

Конструктор вызывается, если в программе встретилась какая-либо из синтаксических 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:

Point p1(1, 2);//конструктор с параметрами

Point p2();//конструктор без параметров

Point p3 = p1;// конструктор копирования

Point p4 = Point (5, 20);//создается объект без имени и копируется

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

Point\*pp2 = new Point(4, 32);//указатель на объект

7. Перечислить свойства деструкторов.

- Деструктор вызывается автоматически, когда объект

удаляется из памяти: для локальных объектов это происходит при выходе из блока, в котором они объявлены (для глобальных — как часть процедуры выхода из main; для объектов, заданных через указатели, деструктор вызывается неявно при использовании операции delete)

- Имя деструктора начинается с тильды (~), непосредственно за которой следует имя класса.

- Не имеет аргументов и возвращаемого значения;

- Не наследуется;

- Не может быть объявлен как const или static;

- Может быть виртуальным.

8. К каким атрибутам имеют доступ методы класса?

Методы класса имеют доступ ко всем атрибутам, в том числе private, так как имеют одну область видимости с ними (то есть тоже доступны внутри класса).

9. Что представляет собой указатель this?

Указатель \*this - открытый константный для адреса указатель, содержащий адрес объекта, который вызывает метод класса.

Point::SetX(int x) {

this->x = x;

}

Таким образом, компилятор не путает поле класса и вводимый в конструктор параметр, имеющие одинаковое имя.

10. Какая разница между методами определенными внутри класса и вне класса?

Методы, указанные внутри класса, по умолчанию считаются встроенной функцией (inline), и не требуют указания принадлежности к классу. Методам, определенным вне функции, требуется указывать принадлежность классу.

class Point {

int x, y;

public: //метод, определенный внутри класса

void SetX(int x) {this->x = x;}

};

//метод, определенный вне класса

Person::SetX(int x) {this->x = x;}

11. Какое значение возвращает конструктор?

Конструктор никогда не возвращает значение. Всё, что делает конструктор - это задаёт передаваемому объекту нужные поля.

12. Какие методы создаются по умолчанию?

Конструктор (с “пустыми” полями) и деструктор (при выходе из области видимости).

13. Какое значение возвращает деструктор?

Деструктор не имеет возвращаемого значения

14. Дано описание класса

class Student {

string name;

int group;

public:

student(string, int);

student(const student&)

~student();

};

Какой метод отсутствует в описании класса?

В классе отсутствует конструктор по умолчанию student()

15. Какой метод будет вызван при выполнении следующих операторов:

student\*s; // создается указатель на объект

s = new student; //у объекта вызывается конструктор по умолчанию

16. Какой метод будет вызван при выполнении следующих операторов:

student s(“Ivanov”,20); // конструктор с параметрами

17. Какие методы будут вызваны при выполнении следующих операторов:

student s1(“Ivanov”,20); //конструктор с параметрами,

student s2=s1; //конструктор копирования

18. Какие методы будут вызваны при выполнении следующих операторов:

student s1(“Ivanov”,20); //конструктор с параметрами

student s2; //конструктор по умолчанию

s2=s1; // конструктор копирования

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

void print(student a) //конструктор копирования

{a.show();}

20. 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?

Так как у типа данных class по умолчанию private доступ, то по имени атрибута обратиться нельзя. Но в классе определены public методы (сеттеры), с помощью которых можно изменить значение атрибута следующим образом:

p.set\_name = “Валерий”;