

Федеральное государственное автономное образовательное учреждение
высшего образования «Пермский национальный исследовательский
политехнический университет»

Лабораторная работа №1
«Классы и объекты. Инкапсуляция»

Выполнил:

студент первого курса

ЭТФ группы РИС-23-36

Акбашева Софья Руслановна

Проверила:

Доцент кафедры ИТАС О. А. Полякова

Пермь, 2024

Классы и объекты. Инкапсуляция

Цель задания

- 1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
- 2) Использование классов и объектов в ОО программе.

Постановка задачи

1. Реализовать определение нового класса. Для демонстрации работы с объектами написать главную функцию. Продемонстрировать разные способы создания объектов и массивов объектов.

2. Структура-пара – структура с двумя полями, которые обычно имеют имена `first` и `second`. Требуется реализовать тип данных с помощью такой структуры. Во всех заданиях должны присутствовать :

a. метод инициализации `Init` (метод должен контролировать значения аргументов на корректность);

b. ввод с клавиатуры `Read`;

c. вывод на экран `Show`.

3. Реализовать внешнюю функцию `make_тип()`, где `тип` – тип реализуемой структуры. Функция должна получать значения для полей структуры как параметры функции и возвращать структуру как результат. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

Задание

Элемент арифметической прогрессии a_j вычисляется по формуле: $a_j = a_0 \cdot r^j$. Поле `first` – дробное число, первый элемент прогрессии a_0 , поле `second` – положительное целое число, постоянное отношение r . Реализовать метод `element(int j)` – вычисление j -го элемента прогрессии.

Анализ задачи

- 1) Вводятся значения для первого элемента прогрессии (first), коэффициента прогрессии (second) и номера элемента (j_elem). Вводятся эти значения с помощью функций cin и cout.
- 2) После ввода значений создается объект класса Progression с параметрами 1 и 1. Затем значения first и second устанавливаются в объекте prog с помощью методов setFirst и setSecond.
- 3) Наконец, вычисляется значение элемента прогрессии с помощью метода element объекта prog и выводится на экран с помощью функции cout.
- 4) конструктор класса Progression. Конструктор принимает два параметра - first и second, которые присваиваются соответствующим полям объекта.
- 5) Затем объявляется метод element, который вычисляет j-ый элемент прогрессии по формуле $a_j = a_0 * r^j$.
- 6) Далее объявляются методы setFirst и setSecond, которые устанавливают значения полей first и second соответственно.
- 7) директива #pragma once, которая предотвращает повторную загрузку заголовочного файла, если он уже был включен
- 8) класс Progression. Класс содержит два private поля - first и second, которые являются double значениями.
- 9) Также объявляются четыре public метода - конструктор класса Progression, который принимает два double параметра, метод element, который принимает int параметр и возвращает double значение, и два метода setFirst и setSecond, которые принимают double параметры.

Блок схема

Progression
- first: double - second: double
+ Progression(double, double) + element(int): double + setFirst(double): void + setSecond(double): void

Код

Файл Class_1.cpp

```
#include "progression.h" //объявления классов

int main() {
    system("chcp 1251 > Null"); //кодировка
    double first;
```

```

int second, j_elem;
cout << "Последовательность  $a_j = a_0 * r^j$ " << endl << endl;

cout << "Введите дробное число - a(0): ";
cin >> first;

do {
    cout << "Введите целое положительное число - r: ";
    cin >> second;
} while (second < 0);

do {
    cout << "Введите номер элемента геометрической прогрессии - j: ";
    cin >> j_elem;
} while (j_elem < 1);
cout << endl;

Progression prog(2, 2); //объект класса Progression с параметрами 1 и 1.
prog.setFirst(first); //устанавливает значение private поля first объекта prog на 3
prog.setSecond(second); //устанавливает значение private поля second объекта prog на 2
cout << "Значение функции = " << prog.element(j_elem) << endl;
return 0; //завершение программы
}

```

Файл progression.cpp

```

#pragma once //предотвращает повторную загрузку заголовочного файла, если он уже был включен.
#include "progression.h" //объявления классов
#include <cmath> //для возведения в степень

Progression::Progression(double first = 1, double second = 1) { //объявление конструктора класса Progression
    cout << "Вызов конструктора класса Progression("<<first<< ", "<<second<<")" << endl;
    this->first = first; //присваивает значение параметра first конструктора полю first объекта
    this->second = second; //присваивает значение параметра second конструктора полю second объекта
};

double Progression::element(int j) { //вычисляю  $a_j = a_0 * r^j$  (j-ый элемент прогрессии)
    cout << "Вызов метода element("<<j<<")" << endl;
    return first * pow(second, j);
}

void Progression::setFirst(double a0) { //устанавливаю число a0
    cout << "Private поле first(a0) = "<< a0 << endl;
    this->first = a0;
}

void Progression::setSecond(double r) { //устанавливаю число r
    cout << "Private поле second(r) = "<< r << endl;
    this->second = r;
}

```

Файл progression.h

```

#pragma once //предотвращает повторную загрузку заголовочного файла, если он уже был включен
#include <iostream> //стандартные потоки ввода и вывода
using namespace std;

class Progression { //объявление класса Progression
private: //объявление private полей класса Progression
    double first; //объявляет private поле first типа double
    int second; //объявляет private поле second типа double
public: //объявление public методов класса Progression.

```

```

Progression(double, double); //объявляет конструктор класса Progression, который принимает два double
параметра.
double element(int); //объявляет метод element(), который принимает int параметр и возвращает double
значение.
void setFirst(double); //объявляет метод setFirst(), который принимает double параметр.
void setSecond(double); //объявляет метод setSecond(), который принимает double параметр.
};

```

Результат работы

```

Последовательность  $a_j = a_0 * r^j$ 
Введите дробное число -  $a(0)$ : 5.1
Введите целое положительное число - r: 3
Введите номер элемента геометрической прогрессии - j: 7

Вызов конструктора класса Progression(2, 2)
Private поле first( $a_0$ ) = 5.1
Private поле second(r) = 3
Вызов метода element(7)
Значение функции = 11153.7

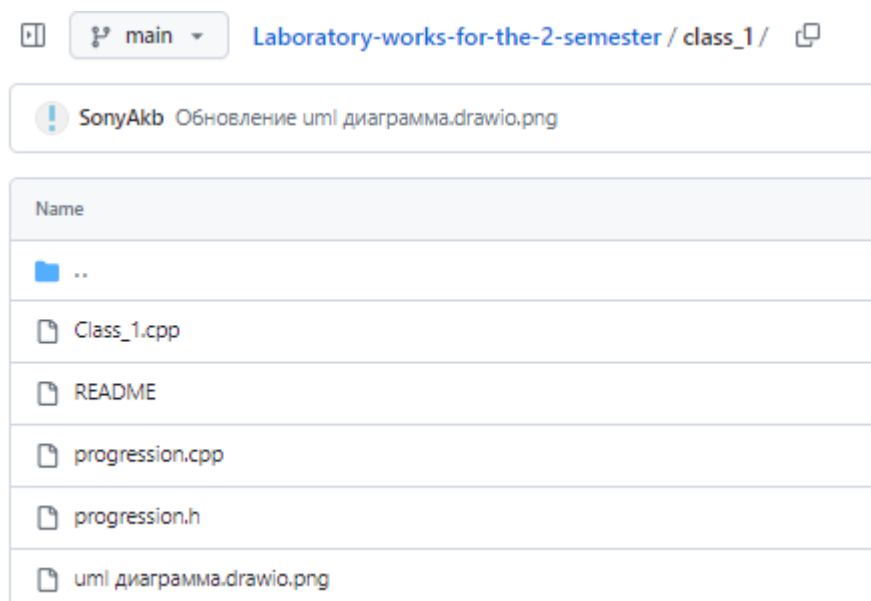
```

Вывод

В ходе работы я применила знания о работе с классами. Я реализовала определение нового класса. Для демонстрации работы с объектами написала главную функцию. Продемонстрировала разные способы создания объектов и массивов объектов.

GitHub

Ссылка: https://github.com/SonyAkb/Laboratory-works-for-the-2-semester/tree/main/class_1



Контрольные вопросы

1. Что такое класс?

Класс – это абстрактный тип данных, определяемый пользователем. Представляет собой модель реального объекта в виде данных и функций для работы с ними. Он содержит данные (поля) и функции (методы), которые определяют поведение объекта.

2. Что такое объект (экземпляр) класса?

Объект (экземпляр) класса - это переменная типа класса. Их может быть бесконечное количество.

```
class Progression
private: //объявление
    double first;
    int second; //объявление
```

3. Как называются поля класса?

Поля - переменные внутри класса, также называются атрибутами класса. Они хранят информацию, специфичную для каждого объекта класса.

```
class Progression { //объявление класса Progression
private: //объявление private полей класса Progression
    double first; //объявление private поля first
    int second; //объявление private поля second
public: //объявление public методов класса Progression
    Progression(double, double); //объявление конструктора
    double element(int); //объявление метода element
    void setFirst(double); //объявление метода setFirst
    void setSecond(double); //объявление метода setSecond
};
```

ПОЛЯ

МЕТОДЫ

4. Как называются функции класса?

Функции класса обычно называются методами. Они определяют поведение объекта и могут работать с его полями.

5. Для чего используются спецификаторы доступа?

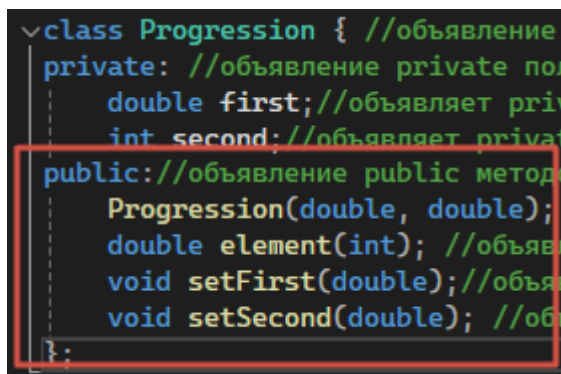
Спецификаторы доступа используются для управления доступом к членам класса. Они определяют, какие части класса могут быть доступны

извне и внутри самого класса. Всего есть 3 спецификатора public, private, protected.

В классе (class) все поля и методы по умолчанию имеют модификатор доступа private.

6. Для чего используется спецификатор public?

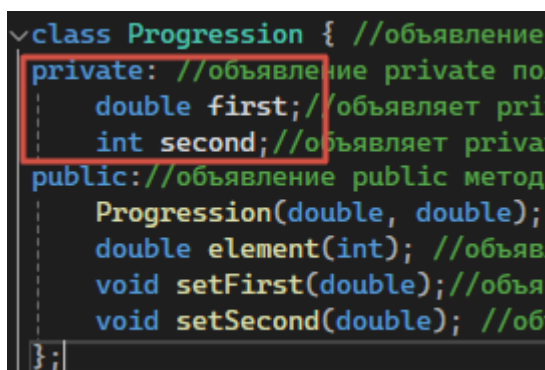
Спецификатор public используется для определения членов класса, которые доступны другим объектам и функциям вне класса. Это позволяет другим частям программы использовать и изменять эти члены.



```
class Progression { //объявление
private: //объявление private по
    double first; //объявляет priv
    int second; //объявляет privat
public: //объявление public метод
    Progression(double, double);
    double element(int); //объяв
    void setFirst(double); //объя
    void setSecond(double); //об
};
```

7. Для чего используется спецификатор private?

Спецификатор private используется для определения членов класса, которые доступны только внутри самого класса. Это предотвращает несанкционированный доступ к этим членам извне класса и обеспечивает инкапсуляцию, что помогает поддерживать целостность и безопасность данных класса. Функции друзья могут использовать заприваченные поля и методы.



```
class Progression { //объявление
private: //объявление private по
    double first; //объявляет priv
    int second; //объявляет privat
public: //объявление public метод
    Progression(double, double);
    double element(int); //объяв
    void setFirst(double); //объя
    void setSecond(double); //об
};
```

8. Если описание класса начинается со спецификатора class, то какой спецификатор доступа будет использоваться по умолчанию?

Если описание класса начинается со спецификатора class, то по умолчанию используется спецификатор доступа private. Это означает, что все члены класса будут закрытыми, если явно не указать другой спецификатор доступа.

9. Если описание класса начинается со спецификатора struct, то какой спецификатор доступа будет использоваться по умолчанию?

Если описание класса начинается со спецификатора struct, то по умолчанию используется спецификатор доступа public. Все члены класса будут общедоступными, если явно не указать другой спецификатор доступа.

10. Какой спецификатор доступа должен использоваться при описании интерфейса класса? Почему?

При описании интерфейса класса следует использовать спецификатор доступа public. Интерфейс класса определяет публичные методы и свойства, которые должны быть доступны для использования другими частями программы. Использование private ограничило бы доступ к этим членам, что противоречит цели интерфейса.

11. Каким образом можно изменить значения атрибутов экземпляра класса?

Напрямую, через ссылку, через указатель, через методы классы (сеттеры)

12. Каким образом можно получить значения атрибутов экземпляра класса?

Напрямую, через указатель, через методы классы (геттеры)

13. Класс описан следующим образом struct Student

```
{  
string name; int group;  
.....
```


};

Объект класса определен следующим образом Student *s=new Student;

Как можно обратиться к полю name объекта s?

s->name = "Olga"

14. Класс описан следующим образом struct Student

{

string name; int group;

.....

};

Объект класса определен следующим образом Student s;

Как можно обратиться к полю name объекта s?

s.name = "Olga"

15. Класс описан следующим образом class Student

{

string name; int group;

.....

};

Объект класса определен следующим образом Student *s=new Student;

Как можно обратиться к полю name объекта s?

s->name = "Olga"

```

struct Student
{
    string name;
    int group;

public:
    void show_name() {
        cout << name;
    }
    void set_name() {
        cin >> name;
    }
};

int main()
{
    Student* s = new Student;
    s->set_name();
    s->show_name();
}

```

16. Класс описан следующим образом class Student

```

{
string name; int group; public:
.....
};

```

Объект класса определен следующим образом Student s;

Как можно обратиться к полю name объекта s?

s.name = “Olga”

```

struct Student
{
    string name;
    int group;

public:
    void show_name() {
        cout << name;
    }
    void set_name() {
        cin >> name;
    }
};

int main()
{
    Student s;
    s.set_name();
    s.show_name();
}

```

17. Класс описан следующим образом class Student

```

{
public:
char* name; int group;

```

.....

};

Объект класса определен следующим образом Student *s=new Student;

Как можно обратиться к полю name объекта s?

s->name = "Olga"