Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 - «Программная инженерия»

**«Лабораторная работа №1**

**"Классы и объекты. Инкапсуляция"»**

Выполнил студент гр. РИС-24-3б Носков Егор Михайлович

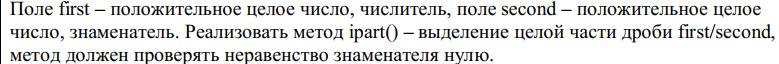
Проверил: Доцент кафедры ИТАС Ольга Андреевна Полякова

|  |  |
| --- | --- |
|  | |
| (оценка) (подпись) | |
|  |
| (дата) |

Г. Пермь, 2024

**Постановка задачи**

**Вариант 1:**

****

**Программа на языке C++**

#include <iostream>

#include <cmath>

using namespace std;

class Division{

    int First;

    int Second;

    public:

    void Show(){

        cout<<"Первый элемент: "<<First<<endl;

        cout<<"Второй элемент: "<<Second<<endl;

    }

    void Read(){

    }

    void Init(double F, int S){

        First=F;

        Second=S;

    }

    int Ipart(){

        if(Second!=0) return First/Second;

        else{

            cout<<"Знаменатель равен нулю"<<endl;

            return 0;

        }

    }

};

Division Make\_Division(int F,int S){

    Division t;

    t.Init(F,S);

    return t;

}

int main(){

    Division A, B;

    cout<<"1 способ с объектами:"<<endl;

    int First1=4,Second1=2;

    A.Init(First1,Second1);

    A.Show();

    cout<<"Результат деления: "<<endl<<A.Ipart()<<endl;

    cout<<"2 способ с объектами:"<<endl;

    B.Read();

    B.Show();

    cout<<"Результат деления: "<<endl<<B.Ipart()<<endl;

    cout<<"3 способ с объектами:"<<endl;

    Division\* C = new Division;

    C->Init(100,5);

    C->Show();

    cout<<"Результат деления: "<<endl<<C->Ipart()<<endl;

    Division arr[3];

    cout<<"Статический массив"<<endl;

    for(int i=0;i<3;i++){

        cout<<i+1<<endl;

        arr[i].Read();

    }

    cout<<endl;

    for(int i=0;i<3;i++){

        arr[i].Show();

    }

    cout<<endl;

    for(int i=0;i<3;i++){

        cout<<"Результат деления: "<<arr[i].Ipart()<<endl;

    }

    Division\* arr1=new Division[3];

    cout<<"Динамический массив"<<endl;

    for(int i=0;i<3;i++){

        cout<<i+1<<endl;

        arr1[i].Read();

    }

    cout<<endl;

    for(int i=0;i<3;i++){

        arr1[i].Show();

    }

    cout<<endl;

    for(int i=0;i<3;i++){

        cout<<"Результат деления: "<<arr1[i].Ipart()<<endl;

    }

    delete[] arr1;

    int First1,Second1;

    cout<<"Функция Make\_Division"<<endl;

    cout<<"Введите первый элемент: "<<endl;

    cin>>First1;

    cout<<"Введите второй элемент: "<<endl;

    cin>>Second1;

    Division division=Make\_Division(First1, Second1);

    division.Show();

    cout<<"Результат деления: "<<division.Ipart()<<endl;

    return 0;

}

**Контрольные вопросы**  
  
**1.** Класс - это определяемый пользователем тип данных, который служит чертежом (шаблоном) для создания объектов. Он определяет атрибуты (данные) и методы (действия), которые могут выполнять объекты этого класса.  
  
**2.** Объект - это конкретный экземпляр класса. Это переменная, созданная на основе шаблона, предоставленного классом, с выделенной для неё памятью. Каждый объект имеет собственные значения атрибутов.  
  
**3.** Поля класса обычно называются *атрибутами*, *членами данных*, или *переменными-членами*.  
  
**4.** Функции класса называются *методами* или *функциями-членами*.  
  
**5.** Спецификаторы доступа (public, private, protected) определяют область видимости и уровень доступа к членам класса (атрибутам и методам) извне класса. Они реализуют *инкапсуляцию*.  
  
**6.** public означает, что члены класса (атрибуты и методы) доступны из любой части программы, как внутри класса, так и снаружи.  
  
**7.** private означает, что члены класса доступны только внутри самого класса (его методами). Прямой доступ извне класса запрещен. Это позволяет скрыть внутреннюю реализацию класса и контролировать доступ к его данным.  
  
**8.** По умолчанию для class используется спецификатор доступа private.  
  
**9.** По умолчанию для struct используется спецификатор доступа public.  
  
**10.** Для описания интерфейса класса должен использоваться спецификатор доступа public. Интерфейс - это набор методов, которые другие части программы будут использовать для взаимодействия с объектом класса. Эти методы должны быть общедоступными, чтобы их можно было вызывать извне класса.  
  
**11.** • Если атрибуты объявлены как public, то их можно изменить напрямую через оператор .: object.attribute = newValue;.  
• Если атрибуты объявлены как private или protected, то для их изменения используются public методы класса (сеттеры): object.set\_attribute(newValue);.  
  
**12.** • Если атрибуты объявлены как public, то их можно получить напрямую через оператор .: value = object.attribute;.  
• Если атрибуты объявлены как private или protected, то для их получения используются public методы класса (геттеры): value = object.get\_attribute();.  
  
**Практические вопросы (13-15):**  
  
**13.**

```c\+\+  
struct Student {  
string name;  
int group;  
};  
  
Student \*s = new Student;  
  
// Как можно обратиться к полю name объекта s?  
s->name = "Новое имя"; // Присвоение значения полю name  
cout << s->name << endl; // Получение значения поля name  
```

Поскольку s - это *указатель* на объект Student, для доступа к членам объекта используется оператор -> (стрелка).  
  
**14.**

```c\+\+  
struct Student {  
string name;  
int group;  
};  
  
Student s;  
  
// Как можно обратиться к полю name объекта s?  
s.name = "Новое имя"; // Присвоение значения полю name  
cout << s.name << endl; // Получение значения поля name  
```

Поскольку s - это *объект* Student, для доступа к членам объекта используется оператор . (точка).  
  
**15.**

```c\+\+  
class Student {  
string name;  
int group;  
};  
  
Student \*s = new Student;  
  
// Как можно обратиться к полю name объекта s?  
// НЕЛЬЗЯ получить доступ напрямую, необходимо использовать методы (геттеры/сеттеры)  
// т.к. по умолчанию атрибуты класса имеют спецификатор доступа private.  
  
//Предположим, что в классе Student определены методы setName и getName:  
  
class Student {  
private:  
string name;  
int group;  
  
public:  
void setName(const string& newName) { name = newName; }  
string getName() const { return name; }  
};  
Student \*s = new Student;  
s->setName("Новое имя");  
cout << s->getName() << endl;  
```  
В классе Student (не struct!) атрибуты по умолчанию являются private. Это означает, что к ним нельзя получить прямой доступ извне класса. Чтобы получить или изменить значения атрибутов, необходимо использовать public методы класса (геттеры и сеттеры). Поскольку s - это указатель, используем оператор -> для вызова методов: s->setName("Новое имя"); и cout << s->getName() << endl;  
  
Внимание: Этот ответ предполагает, что в классе Student определены методы setName и getName, так как иначе доступа к полю name не будет.

**Ссылка на репозиторий с отчетом Github**

****