Министерство образования и науки Российской Федерации

Федеральное государственное образовательное автономное учреждение высшего образования

"Пермский национальный исследовательский политехнический университет"

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Дисциплина: Основы алгоритмизации и программирования

Тема: Классы и объекты. Инкапсуляция.

|  |  |
| --- | --- |
| Выполнил работу | |
| Студент группы РИС-22-1б | |
| Деревнин И.В. | |
|  | |
| Проверил работу | |
| Доцент кафедры ИТАС | |
| Полякова О.А. | |
|  | |

Пермь – 2023

**Анализ предметной области**

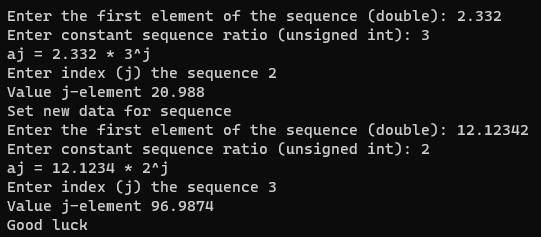
**Постановка задачи**

1. Реализовать определение нового класса. Для демонстрации работы с объектами написать главную функцию. Продемонстрировать разные способы создания объектов и массивов объектов.
2. Структуры-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:
   1. Метод инициализации Init (метод должен контролировать значения аргументов на корректность);
   2. Ввод с клавиатуры Read;
   3. 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 работы.

Вариант 15: Элемент арифметической прогрессии аj вычисляется по формуле: . Поле first дробное число, первый элемент прогрессии а0, поле second - положительное целое число, постоянное отношение г. Реализовать метод element(int j) -вычисление j-го элемента прогрессии.

**Анализ задачи**

Класс MyClass представляет из себя класс, для вычисления элемента в арифметической прогрессии. Он хранит 2 поля first – число с плавающей точкой, которое отвечает за aj в формуле арифметической прогрессии и second – r – которое является степенью в которое возводится число в арифметической прогрессии. Также для взаимодействия с классом есть методы: element – метод, который отвечает за вывод ответа, Read – чтение данных с консоли, Init – инициализация данных, Show – вывод данных на консоль. Также есть конструкторы и деструктор. Для создания класса определена функция make\_myclass, которая создает класс с нужными параметрами изначально, а не заменяет их как метод Init.

****Тестирование программы**

*Рис. 1 – Тестирование программы с классом MyClass..*

**Заключение**

Была разработана программа, которая считает j-элемент арифметической прогрессии по заданной формуле. Для реализации данной программы был описан класс, который хранит в себе 2 поля для вычисления j-го элемента прогрессии, а также методы, благодаря которым можно менять значения полей класса.

1. Что такое класс?

Класс - абстрактный тип данных, который создает пользователь, состоящий из данных и функций. Позволяет представить часть кода в виде одного объекта. В объектно-ориентированном программировании программа представляется в виде множества объектов, что позволяет регулировать отношения между ними.

1. Что такое объект (экземпляр) класса?

Объект (экземпляр) - отдельный переменная класса, которая характеризуется данными его полей.

Пример объекта:

Point A;

1. Как называются поля класса?

Атрибуты, в примере из вопроса (1) это поля x, y

1. Как называются функции класса?

Методы, в примере из вопроса (1) это функция GetX()

1. Для чего используются спецификаторы доступа?

Спецификаторы используются для регулирования области видимости и времени жизни переменной. Действие любого спецификатора распространяется до следующего спецификатора или до конца класса.

1. Для чего используется спецификатор public?

Спецификатор public задаёт последующим переменным глобальную область видимости (доступны в коде вне класса) и время жизни до конца выполнения программы.

1. Для чего используется спецификатор private?

Спецификатор private задаёт последующим переменным локальную область видимости (доступны только в классе) и время жизни - во время компиляции кода описания класса.

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

Для типа данных class по умолчанию действует спецификатор private.

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

Для типа данных struct по умолчанию действует спецификатор public.

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

Для описания класса должен использоваться спецификатор public для того, чтобы его поля/методы были доступны для использования в других частях кода, и класс не был изолированной структурой данных.

1. Каким образом можно изменить значение атрибутов экземпляра класса?

Если атрибуты имеют public доступ, то изменить поля можно, обратившись по имени, по указателю или по ссылке.

class Point {

public:

int x, y;

};

int main() {

Point a; //доступ по имени

a.x = 20;

Point\* b = new Point; //доступ по указателю

b->x = 19;

Point& c = a; //доступ по ссылке

c.x = 18;

return 0;

}

Если атрибуты имеют private доступ, то изменить поля можно, обратившись к public методам-модификаторам.

class Point{

int x;

public:

void SetX(int n) { x= n; }

};

int main() {

Point a;

//a.x = 5; невозможно из-за private доступа поля

a.SetX(5);

return 0;

}

1. Каким образом можно получить значения атрибутов экземпляра класса?

Если атрибуты имеют public доступ, то получить доступ к полям можно, обратившись по имени, по указателю или по ссылке.

class Point {

public:

int x;

};

int main() {

Point a;

cout << a.x; //доступ по имени

Point\* b = new Point;

cout << b->x; //доступ по указателю

Point& c = a;

cout << c.x; //доступ по ссылке

return 0;

}

Если атрибуты имеют private доступ, то получить доступ к полям можно, обратившись к public методам-модификаторам.

class Point{

int x;

public:

int GetX() { return x; }

};

int main() {

Student a;

//cout << a.x; невозможно из-за private доступа поля

cout << a.GetX();

return 0;

}

1. Класс определен 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?

Так как у типа данных struct по умолчанию public доступ, то к полям можно обратиться непосредственно. Так как экземпляр класса задан через указатель, то к нему можно обратиться s->name, что равносильно записи (\*s).name.

1. Класс определен следующим образом

struct Student {

string name;

int group;

};

Объект класса определен следующим образом

Student s;

Как можно обратиться к полю name объекта s?

Так как у типа данных struct по умолчанию public доступ, то к полям можно обратиться непосредственно. Так как экземпляр класса задан по имени, то к нему можно обратиться s.name.

1. 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?

Так как у типа данных class по умолчанию private доступ, то к полям нельзя обратиться непосредственно, и требуются специальные методы с public доступом, которые называются геттеры или селекторы.

1. 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?

Так как у типа данных class по умолчанию private доступ, то к полям нельзя обратиться непосредственно, и требуются специальные методы с public доступом, которые называются геттеры или селекторы. Допустим, в public секции определен некоторый метод

void GetName() { return name; }

Тогда к полю можно обратиться следующим образом

s.GetName();

1. Класс описан следующим образом

class Student {

public:

char\* name;

int group;

};

Объект класса определен следующим образом

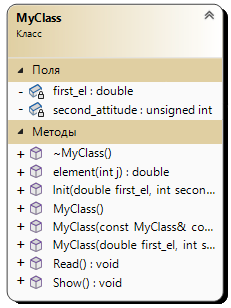
Student \*s = new Student;

Как можно обратиться к полю name объекта s?

Так как у данного класса указан public доступ, то к полям можно обратиться непосредственно. Так как экземпляр объявлен через указатель, то обратиться к нему можно s->name или (\*s).name.

**Приложения**

Приложение UML-диаграмма



Приложение Б – код программы

Main.cpp:

#include "class.h"

MyClass Make\_MyClass(double first, int second)

{

if (second < 0)

exit(-1);

MyClass temp(first, second);

return temp;

}

int main()

{

int b; double a;

cout << "Enter the first element of the sequence (double): ";

cin >> a;

cout << "Enter constant sequence ratio (unsigned int): ";

cin >> b;

MyClass mc = Make\_MyClass(a, b);

mc.Show();

int j;

cout << "Enter index (j) the sequence "; cin >> j;

cout << "Value j-element " << mc.element(j) << endl;

cout << "Set new data for sequence" << endl;

mc.Read();

mc.Show();

cout << "Enter index (j) the sequence "; cin >> j;

cout << "Value j-element " << mc.element(j) << endl;

cout << "Good luck" << endl;

return 0;

}

class.h:

#pragma once

#include <iostream>

#include <cmath>

using namespace std;

class MyClass

{

private:

double first\_el; //первое поле

unsigned int second\_attitude; //второе поле

public:

MyClass() { first\_el = second\_attitude = 0; } //конструктор по умолчанию

MyClass(const MyClass& copy); //конструктор копирования

MyClass(double first\_el, int second\_attitude); //конструктор с параметрами

~MyClass() {} //деструктор класса

double element(int j) { return first\_el \* pow(second\_attitude, j); } //метод отвечающий за вывод ответа

void Read(); //чтение с консоли

void Init(double first\_el, int second\_attitude); //инициализация

void Show(); //вывод на консоль

};

class.cpp:

#include "class.h"

MyClass::MyClass(double first\_el, int second\_attitude)

{

this->first\_el = first\_el;

this->second\_attitude = second\_attitude;

}

MyClass::MyClass(const MyClass& copy)

{

this->first\_el = copy.first\_el;

this->second\_attitude = copy.second\_attitude;

}

void MyClass::Read()

{

cout << "Enter the first element of the sequence (double): ";

cin >> first\_el;

cout << "Enter constant sequence ratio (unsigned int): ";

cin >> second\_attitude;

}

void MyClass::Init(double first\_el, int second\_attitude)

{

this->first\_el = first\_el;

this->second\_attitude = second\_attitude;

}

void MyClass::Show()

{

cout << "aj = " << first\_el << " \* " << second\_attitude << "^j" << endl;

}