ПНИПУ

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

Лабораторная работа №1.

Классы и объекты. Инкапсуляция

Выполнил студент группы РИС-23-3Б

Буковский Денис Владимирович

Проверила доцент кафедры ИТАС О.А. Полякова

2024

1. 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:

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

б. ввод с клавиатуры Read;

в. вывод на экран Show.

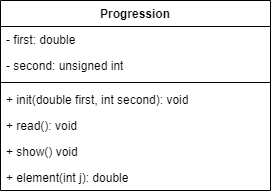
3. Реализовать внешнюю функцию make\_тип(), где тип — тип реализуемой структуры. Функция должна получать значения для полей структуры как параметры функции и возвращать структуру как результат. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

Элемент арифметической прогрессии aj вычисляется по формуле: аj = а0\*rj. Поле first — дробное число, первый элемент прогрессии a0, поле second — положительное целое число, постоянное отношение r. Реализовать метод element(int j) вычисление j-го элемента прогрессии.

2. Анализ задачи

Опишем класс Progression с полями типа double first и int second. Добавим методы init, read, show и element по требованию задания. Опишем метод создания объекта Progression makeprogression. Протестируем в функции main().

1. UML-диаграмма



1. Код

Progression.h

#pragma once

class Progression {

private:

double first;

unsigned int second;

public:

void init(double first, unsigned int second);

void read();

void show();

double element(int j);

};

Progression make\_progression(double first, unsigned int second);

Progression.cpp

#include <iostream>

#include "Progression.h"

using namespace std;

void Progression::init(double first, unsigned int second) {

this->first = first;

this->second = second;

}

void Progression::read() {

cout << "Enter first value (a0): ";

cin >> this->first;

cout << endl << "Enter second value (r): ";

cin >> this->second;

cout << endl;

}

void Progression::show() {

cout << "First parameter (a0): " << this->first << endl << "Second parameter (r): " << this->second << endl;

}

double Progression::element(int j) {

return (this->first \* pow(this->second, j));

}

Progression make\_progression(double first, unsigned int second) {

Progression new\_progression;

new\_progression.init(first, second);

return new\_progression;

}

main.cpp

#include <iostream>

#include "Progression.h"

using namespace std;

int main() {

Progression prog = make\_progression(1.563, 2);

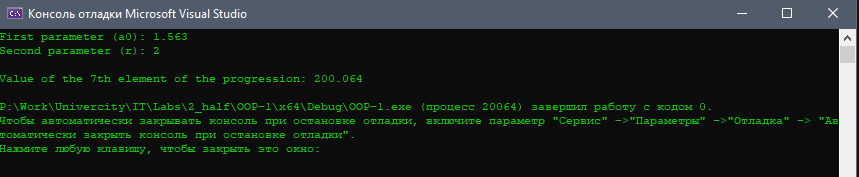
prog.show();

cout << endl <<"Value of the 7th element of the progression: " << prog.element(7) << endl;

return 0;

}

1. Вывод программы



1. Ответы на контрольные вопросы
2. Класс – это абстрактный тип данных, определяемый пользователем. Может содержать поля и методы. С помощью класса создаются объекты пользовательского типа.
3. Объект класса – это поименованная область памяти, содержащая поля и методы, присущие определённому классу.
4. Поля класса называются его атрибутами.
5. Функции класса называются его методами.
6. Спецификаторы доступа служат принципу инкапсуляции и позволяют защищать атрибуты и методы от несанкционированного доступа извне.
7. Спецификатор доступа public позволяет получать доступ к атрибуту или методу снаружи объекта класса, а также позволяет наследовать этот атрибут или метод.
8. Спецификатор protected защищает атрибут или метод от доступа извне, но всё ещё позволяет наследовать этот атрибут/метод.
9. Спецификатор доступа private защищает атрибут/метод от любого доступа, элемент принадлежит и будет принадлежать только классу, в котором он был описан.
10. При описании класса со спецификатором struct будет использоваться сугубо спецификатор Public.
11. При описании интерфейса класса должен быть использован спецификатор public. Только таким образом можно взаимодействовать с классом извне.
12. Если атрибуты имеют спецификатор доступа public – то напрямую. Если же они имеют другой спецификатор – нужно будет написать особые методы со спецификатором public – сеттеры.
13. Если атрибуты имеют спецификатор доступа public – то напрямую. Если же они имеют другой спецификатор – нужно будет написать особые методы со спецификатором public – геттеры.
14. S->name;
15. S.name;
16. S->name;
17. S.name;
18. S->name;