Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.1**

Дисциплина: основы алгоритмизации и программирования

Тема: “Классы и объекты. Инкапсуляция”

Вариант 15

Выполнил:

студент группы ИВТ-20-2Б

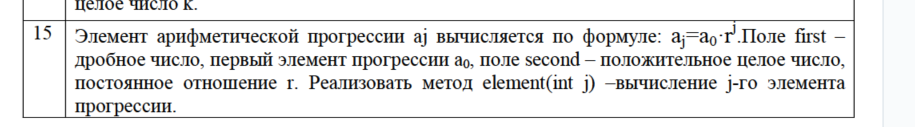
Чувашев Максим

Проверила: доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**

1. Реализовать определение нового класса. Продемонстрировать разные способы создания объектов и массивов объектов.
2. 

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Организовать класс fraction для хранения первое и второе число
   2. Организовать геттеры и сеттеры для данных полей, а так же реализовать конструкторы.
2. В ходе работы были использованы следующие типы данных:

Класс fraction для хранения первого и второго числа в соответствующих полях.

class fraction

{

private:

double first;

int second;

public:

double getFirst()

{

return first;

}

int getSecond()

{

return second;

}

void setFirst(double firstENT)

{

first = firstENT;

}

void setSecond(int secondENT)

{

second = secondENT;

}

void Init(double, int); // метод для инициализации полей

void Read(); // метод для чтения значений полей

void Show(); // метод для вывода значений полей

double element(int); // вычисление степени

int EnterJ(); // ввод индекса желаемого элемента

};

1. Для решения задачи данные были представлены в следующем виде:
   1. Данные вводятся через объекты класса fraction через гетторы.

double getFirst()

{

return first;

}

int getSecond()

{

return second;

}

* 1. Присваиваются через сеттеры

void setFirst(double firstENT)

{

first = firstENT;

}

void setSecond(int secondENT)

{

second = secondENT;

}

* 1. Данные также вводятся конструктор класса.

void fraction::Init(double F, int S)

{

first = F;

second = S;

}

1. Поставленные задачи будут решены следующими действиями:
   1. В функции main происходит вызов конструктора для объекта fruction, далее производится вывод полей в консоль. Далее идет вызов функции element, которая вернет тот элемент арифметической прогрессии, который запросит пользователь.

int main()

{

system("chcp 1251>nul");

setlocale(LC\_ALL, "Ru");

//определение переменных А и В

fraction A;

fraction B;

A.Init(2.0, 4);//инициализация переменной А

B.Read();//ввод полей переменных В

A.Show();//вывод значений полей переменной А

B.Show();//вывод значений полей переменной В

fraction\* X = new fraction;//выделение памяти под динамическую переменную

int j = X->EnterJ();

//вывод значения степени, вычисленного с помощью функции element(j)

cout << "A.element(" << A.getFirst() << "," << A.getSecond() << ")=" << A.element(j) << endl;

cout << "B.element(" << B.getFirst() << "," << B.getSecond() << ")=" << B.element(j) << endl;

//указатели

X->Init(2.0, 5);//инициализация

X->Show();//вывод значений полей

X->element(j);//вычисление j элемента

cout << "X.element(" << X->getFirst() << "," << X->getSecond() << ")=" << X->element(j) << endl;

//массивы

fraction mas[3];//определение массива

for (int i = 0; i < 3; i++)

mas[i].Read();//чтение значений полей for (i=0;i<3; i++)

for (int i = 0; i < 3; i++)

mas[i].Show(); //вывод значений полей

for (int i = 0; i < 3; i++)

{

mas[i].element(j); //вычисление степени

cout << "mas[" << i << "].element(" << mas[i].getFirst() << "," << mas[i].getSecond() << ")=";

cout << mas[i].element(j) << endl;

}

//динамические масивы

fraction\* p\_mas = new fraction[3];//выделение памяти

for (int i = 0; i < 3; i++)

p\_mas[i].Read();//чтение значений полей

for (int i = 0; i < 3; i++)

p\_mas[i].Show();//вывод значений полей

for (int i = 0; i < 3; i++)

{

p\_mas[i].element(j);//вычисление степени

cout << "p\_mas[" << i << "].element(" << p\_mas[i].getFirst() << "," << p\_mas[i].getSecond();

cout << ")=" << p\_mas[i].element(j) << endl;

}

//вызов функции make\_fraction()

double y;

int z;

cout << "first?"; cin >> y;

cout << "second?"; cin >> z;

//переменная F формируется с помощью функции make\_fraction()

fraction F = make\_fraction(y, z);

F.Show();

return 0;

}

* 1. Функция element, которая подсчитывает и возвращаем в мейн тот элемент арифметической прогрессии, который требуется

double fraction::element(int j)

{

return first \* pow(second, j);

}

* 1. В функции EnterJ() производится ввод номера элемента который хочет пользователь

int fraction::EnterJ()

{

int j;

cout << "Какой элемент прогрессии вы хотите вывести?" << endl;

cin >> j;

while (j < 0)

{

cout << "Какой элемент прогрессии вы хотите вывести?" << endl;

cin >> j;

}

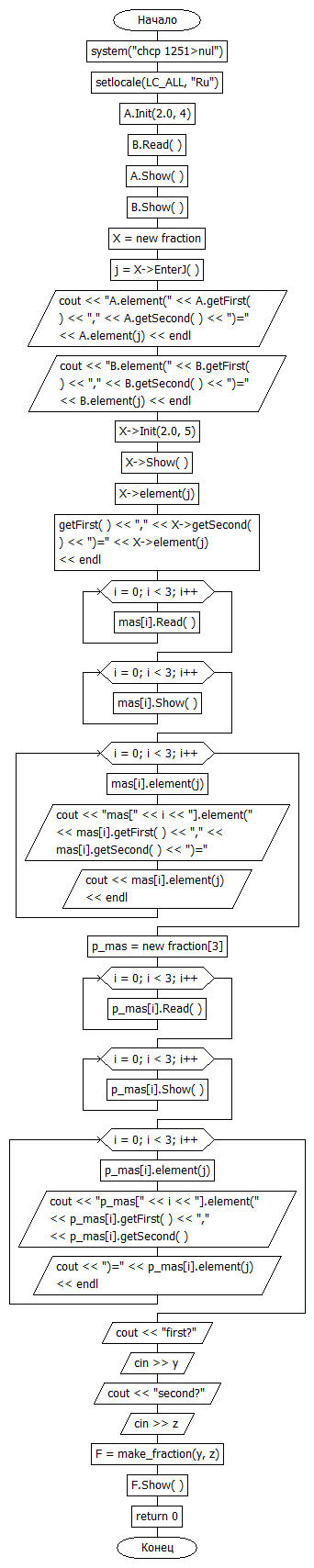
return j;

}

**Блок-схема**

**Sourse.cpp**





**Fraction.cpp**



**Fruction.h**



**Код**

#include <iostream>

#include "fruction.h"

using namespace std;

fraction make\_fraction(double F, int S)

{

fraction t;//создали временную переменную

t.Init(F, S);//инициализировали поля переменной t с помощью параметров функции

return t;//вернули значение переменной t

}

int main()

{

system("chcp 1251>nul");

setlocale(LC\_ALL, "Ru");

//определение переменных А и В

fraction A;

fraction B;

A.Init(2.0, 4);//инициализация переменной А

B.Read();//ввод полей переменных В

A.Show();//вывод значений полей переменной А

B.Show();//вывод значений полей переменной В

fraction\* X = new fraction;//выделение памяти под динамическую переменную

int j = X->EnterJ();

//вывод значения степени, вычисленного с помощью функции element(j)

cout << "A.element(" << A.getFirst() << "," << A.getSecond() << ")=" << A.element(j) << endl;

cout << "B.element(" << B.getFirst() << "," << B.getSecond() << ")=" << B.element(j) << endl;

//указатели

X->Init(2.0, 5);//инициализация

X->Show();//вывод значений полей

X->element(j);//вычисление j элемента

cout << "X.element(" << X->getFirst() << "," << X->getSecond() << ")=" << X->element(j) << endl;

//массивы

fraction mas[3];//определение массива

for (int i = 0; i < 3; i++)

mas[i].Read();//чтение значений полей for (i=0;i<3; i++)

for (int i = 0; i < 3; i++)

mas[i].Show(); //вывод значений полей

for (int i = 0; i < 3; i++)

{

mas[i].element(j); //вычисление степени

cout << "mas[" << i << "].element(" << mas[i].getFirst() << "," << mas[i].getSecond() << ")=";

cout << mas[i].element(j) << endl;

}

//динамические масивы

fraction\* p\_mas = new fraction[3];//выделение памяти

for (int i = 0; i < 3; i++)

p\_mas[i].Read();//чтение значений полей

for (int i = 0; i < 3; i++)

p\_mas[i].Show();//вывод значений полей

for (int i = 0; i < 3; i++)

{

p\_mas[i].element(j);//вычисление степени

cout << "p\_mas[" << i << "].element(" << p\_mas[i].getFirst() << "," << p\_mas[i].getSecond();

cout << ")=" << p\_mas[i].element(j) << endl;

}

//вызов функции make\_fraction()

double y;

int z;

cout << "first?"; cin >> y;

cout << "second?"; cin >> z;

//переменная F формируется с помощью функции make\_fraction()

fraction F = make\_fraction(y, z);

F.Show();

return 0;

}

#include<iostream>

#include"fruction.h"

// реализация метода для инициализации полей структуры

using namespace std;

void fraction::Init(double F, int S)

{

first = F;

second = S;

}

//реализация метода для чтения значений полй структуры

void fraction::Read()

{

cout << "\nfirst?";

cin >> first;

cout << "\nsecond?";

cin >> second;

}

// реализация метода для вывода значений полей структуры

void fraction::Show()

{

cout << "\nfirst = " << first;

cout << "\nsecond = " << second;

cout << endl;

}

// метод для возведения в степень

double fraction::element(int j)

{

return first \* pow(second, j);

}

// ввод индекса желаемого элемента

int fraction::EnterJ()

{

int j;

cout << "Какой элемент прогрессии вы хотите вывести?" << endl;

cin >> j;

while (j < 0)

{

cout << "Какой элемент прогрессии вы хотите вывести?" << endl;

cin >> j;

}

return j;

}

#pragma once

class fraction

{

private:

double first;

int second;

public:

double getFirst()

{

return first;

}

int getSecond()

{

return second;

}

void setFirst(double firstENT)

{

first = firstENT;

}

void setSecond(int secondENT)

{

second = secondENT;

}

void Init(double, int); // метод для инициализации полей

void Read(); // метод для чтения значений полей

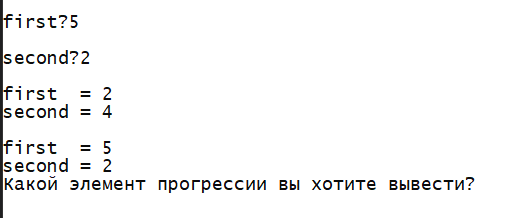
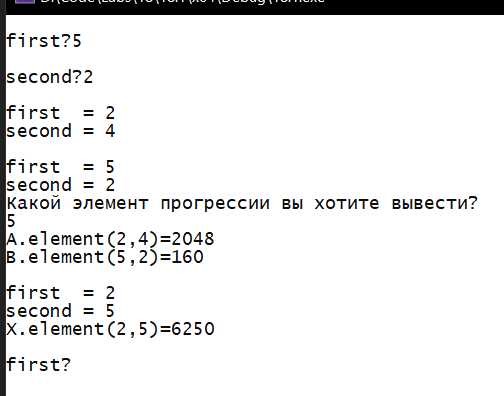
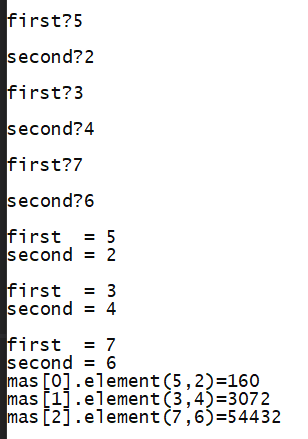
void Show(); // метод для вывода значений полей

double element(int); // вычисление степени

int EnterJ(); // ввод индекса желаемого элемента

};

**Скриншоты**

* 1. 
  2. 
  3. 
  4. 