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 4

Выполнил:

студент группы ИВТ-20-2Б Тедеев А.З.

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. Структура-пара – структура с двумя полями, которые обычно имеют имена first и second.



2

**Анализ задачи**

**1.** Для решения задачи необходимо:

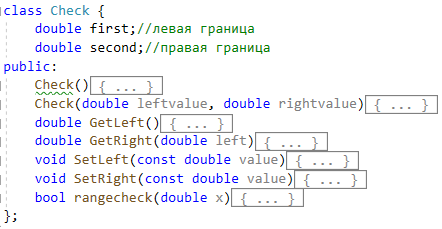
**1.1.** Организовать класс Check для хранения левой и правой границы отрезка через соответствующие поля.

**1.2.** Организовать необходимые методы для ввода данных в поля first и second: гетторы, сетторы, конструкторы.

**1.3.** В функции main() использовать switch() для выбора пользователем, как он(-а) хочет ввести данные.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Класс Check для хранения левой и правой границы отрезка через соответствующие поля.



**2.2.** Переменные типа Check: One, Second – объекты класса Check.

Check One;

Check Second(left, right);

**2.3.** Переменные типа double: left, right, x, method, где left – левая граница отрезка, right – правая граница отрезка, x – переменная, которую необходимо проверить на входимость и method – переменная для выбора пользователем, как он хочет выполнить задачу.

**2.4.** Массивы типа Check: Arr[], Arr1[n] – массивы объектов класса.

Check Arr[1];

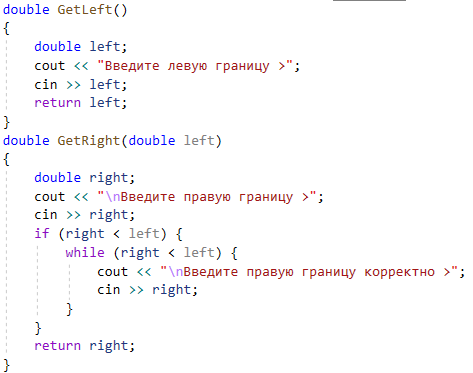
int n = 1;

Check\* Arr1 = new Check[n];

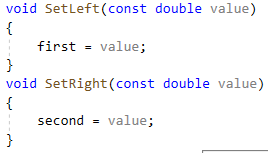
3

**3.** Для решения задачи данные были представлены в следующем виде:

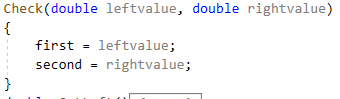
**3.1.** Данные вводятся через объекты класса Check через гетторы.



**3.2.** Присваиваются через методы SetLeft, SetRight.

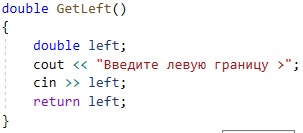


**3.3.** Данные также вводятся через объекты класса Check через конструктор.

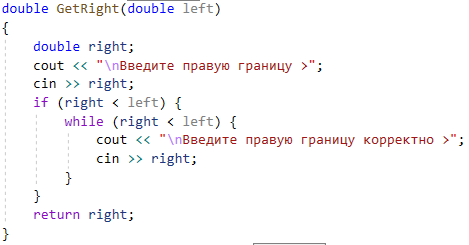


**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Для ввода правой и левой границы используются гетторы (методы, описанные в классе для ввода), в которых используется функция cin.

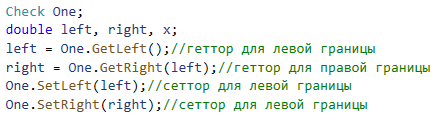


4

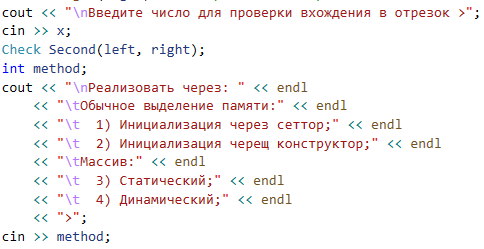


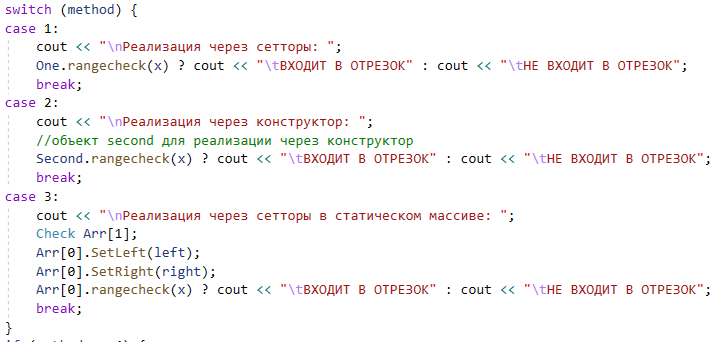
**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** В функции main() вызываются методы ввода для границ отрезка.

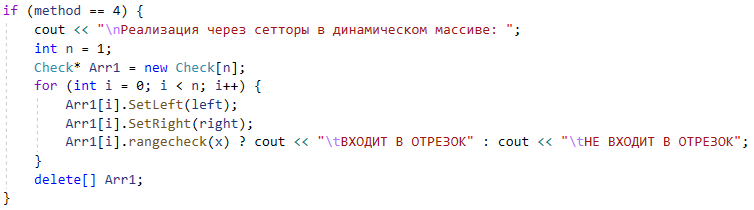


**5.2.** Вводится число, которое необходимо проверить на входимость в отрезок и пользователю предлагается выбрать, в каком типе данных необходимо хранить данные: объект или массив объектов, через switch().





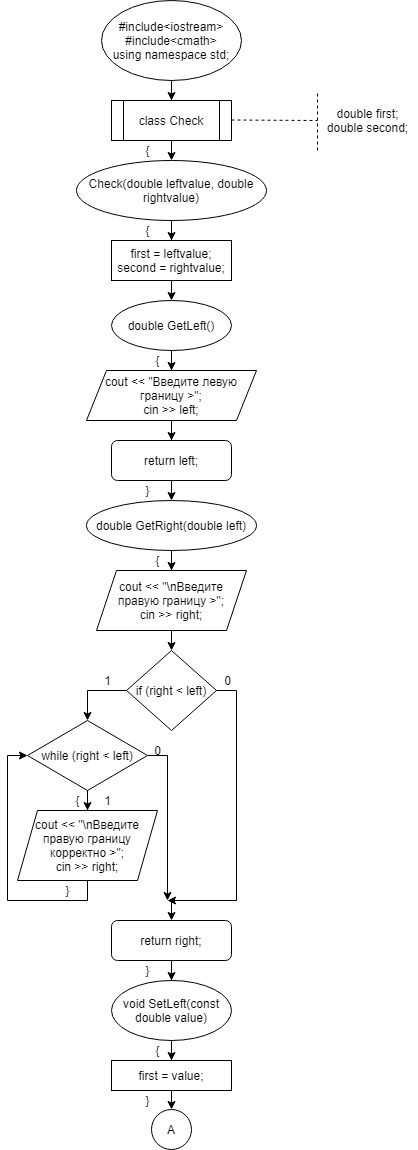
5



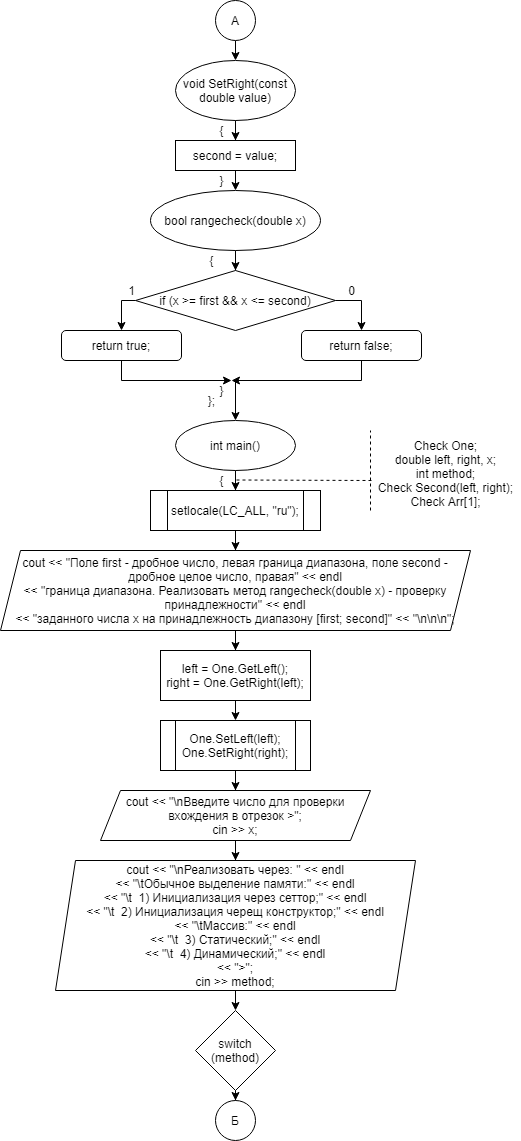
**5.3.** В классе Check также существует метод rangecheck(), который возвращает true или false в зависимости от того, входит ли введённое число в отрезок или нет, в фукцию main() и с помощью тернарного оператора уже осуществляется вывод результата на консоль.

6

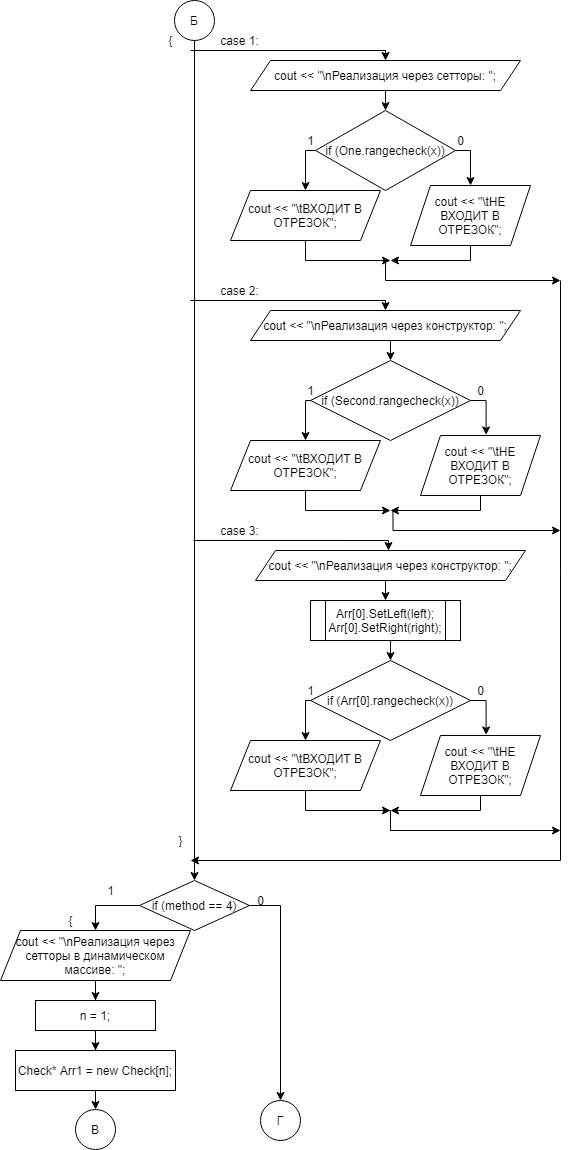
**Блок-схема**



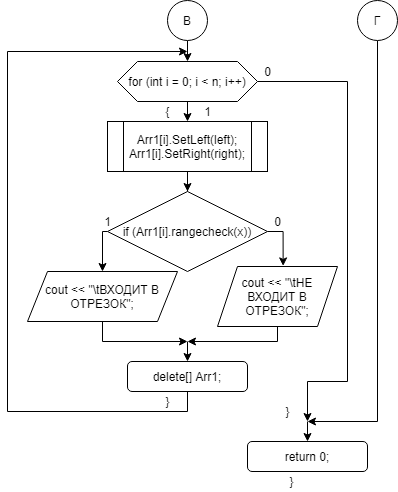
7



8



9



11

**Код**

#include<iostream>

#include<cmath>

using namespace std;

class Check {

double first;//левая граница

double second;//правая граница

public:

Check() //конструктур для объекта без параметров

{

}

Check(double leftvalue, double rightvalue)

{

first = leftvalue;

second = rightvalue;

}

double GetLeft()

{

double left;

cout << "Введите левую границу >";

cin >> left;

return left;

}

double GetRight(double left)

{

double right;

cout << "\nВведите правую границу >";

cin >> right;

if (right < left) {

while (right < left) {

cout << "\nВведите правую границу корректно >";

cin >> right;

}

}

return right;

}

void SetLeft(const double value)

{

first = value;

}

void SetRight(const double value)

{

second = value;

}

bool rangecheck(double x)

{

if (x >= first && x <= second) {

return true;

}

else

return false;

}

};

int main()

{

setlocale(LC\_ALL, "ru");

cout << "Поле first - дробное число, левая граница диапазона, поле second - дробное целое число, правая" << endl

<< "граница диапазона. Реализовать метод rangecheck(double x) - проверку принадлежности" << endl

<< "заданного числа x на принадлежность диапазону [first; second]" << "\n\n\n";

12

Check One;

double left, right, x;

left = One.GetLeft();//геттор для левой границы

right = One.GetRight(left);//геттор для правой границы

One.SetLeft(left);//сеттор для левой границы

One.SetRight(right);//сеттор для левой границы

cout << "\nВведите число для проверки вхождения в отрезок >";

cin >> x;

Check Second(left, right);

int method;

cout << "\nРеализовать через: " << endl

<< "\tОбычное выделение памяти:" << endl

<< "\t 1) Инициализация через сеттор;" << endl

<< "\t 2) Инициализация черещ конструктор;" << endl

<< "\tМассив:" << endl

<< "\t 3) Статический;" << endl

<< "\t 4) Динамический;" << endl

<< ">";

cin >> method;

switch (method) {

case 1:

cout << "\nРеализация через сетторы: ";

One.rangecheck(x) ? cout << "\tВХОДИТ В ОТРЕЗОК" : cout << "\tНЕ ВХОДИТ В ОТРЕЗОК";

break;

case 2:

cout << "\nРеализация через конструктор: ";

//объект second для реализации через конструктор

Second.rangecheck(x) ? cout << "\tВХОДИТ В ОТРЕЗОК" : cout << "\tНЕ ВХОДИТ В ОТРЕЗОК";

break;

case 3:

cout << "\nРеализация через сетторы в статическом массиве: ";

Check Arr[1];

Arr[0].SetLeft(left);

Arr[0].SetRight(right);

Arr[0].rangecheck(x) ? cout << "\tВХОДИТ В ОТРЕЗОК" : cout << "\tНЕ ВХОДИТ В ОТРЕЗОК";

break;

}

if (method == 4) {

cout << "\nРеализация через сетторы в динамическом массиве: ";

int n = 1;

Check\* Arr1 = new Check[n];

for (int i = 0; i < n; i++) {

Arr1[i].SetLeft(left);

Arr1[i].SetRight(right);

Arr1[i].rangecheck(x) ? cout << "\tВХОДИТ В ОТРЕЗОК" : cout << "\tНЕ ВХОДИТ В ОТРЕЗОК";

}

delete[] Arr1;

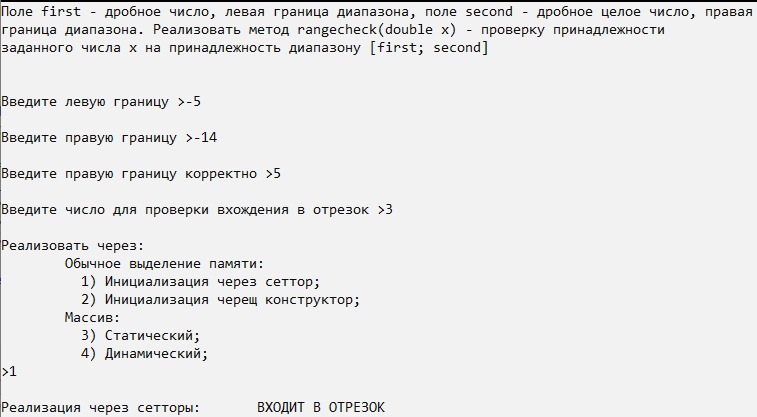
}

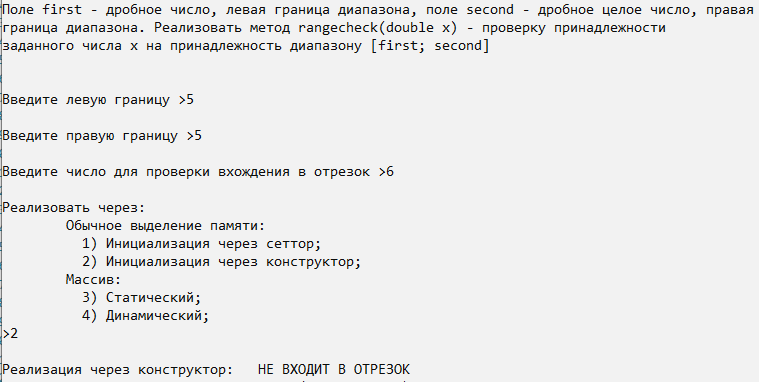
return 0;

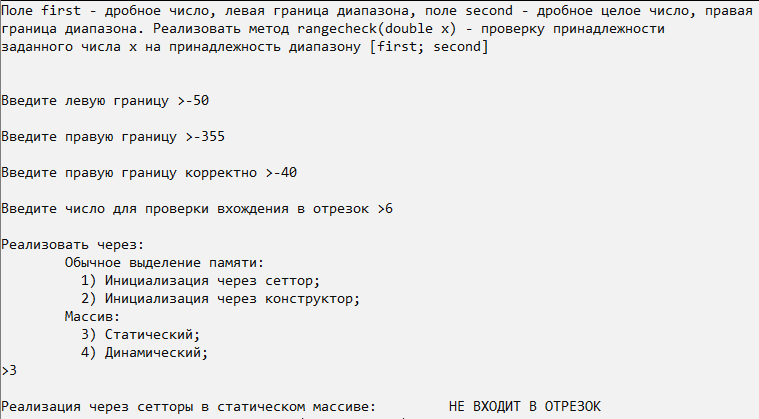
}

12

**Скриншоты**







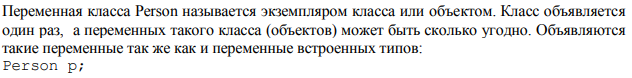
13

**Контрольные вопросы**



1. 



2.



3. Полями/атрибутами называются данные класса.



4. Функции класса называются методами.



5. Спецификаторы доступа управляют видимостью элементов класса.



6. Спецификатор public используется, чтобы задать общедоступную видимость.



7. Спецификатор private используется, чтобы задать видимость только внутри класса для полей или методов под этим спецификатором.



8. Для спецификатора class по умолчанию используется private.



9. Для спецификатора struct по умолчанию используется public.



10. Интерфейс класса описывается после спецификатора public, чтобы к методам класса можно было обратится из других функций.

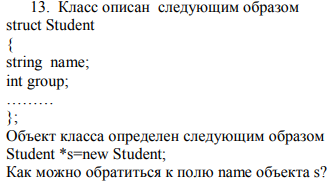
14



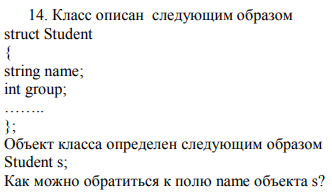
11. Значения атрибутов экземпляра класса можно изменить с помощью методов класса: модификаторов.



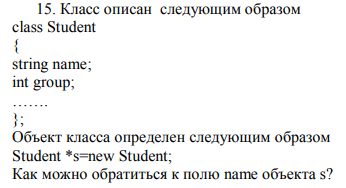
12. Значения атрибутов экземпляра класса можно получить с помощью методов класса: селекторов.



13. Через «->» можно обратиться к полю name, так как оно имеет открытый доступ public.

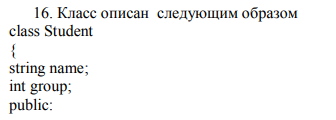


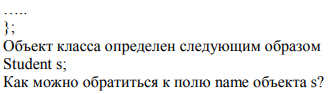
14. Через «.» можно обратиться к полю name, так как оно имеет открытый доступ public.



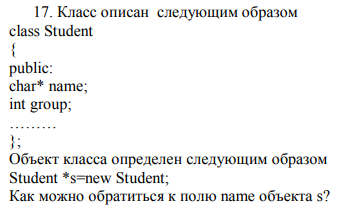
15. Необходимо прописать модификатор для класса Student.

15





16. Необходимо прописать модификатор для класса Student.



17. 16. Необходимо прописать селектор и модификатор для класса Student.

16