Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа № 15**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**(Семестр** 2)

Выполнил студент гр. ИВТ-21-1б

Гребнев Алексей Дмитриевич

Проверил:

Яруллин Денис Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г.Пермь -2022

**Постановка задачи**

Для интерполяционного поиска:   
Дан массив целых чисел (или символов, если это дано по заданию), выполнить задание из своего варианта, используя указанный метод поиска. Массив можно заполнить вручную или случайными числами, размерность массива любая (не меньше 5). Массив для интерполяционного поиска предварительно отсортировать.

**Исходные данные**

Вариант 10:

Найти и удалить все нечетные числа

#include<iostream>

using namespace std;

bool Destroy(int\* arr, int size, int indexN)

{

if (indexN < 0 || indexN > size)

return false;

for (int i = indexN; i < size; i++)

{

arr[i] = arr[i + 1];

}

size--;

return true;

}

void inprint(int\* arr, int n)

{

cout << "Заданное НЕотсортированное множество: "; // Вводите через пробел от 1 до 10 ( 1 4 2 3 7 5 8 6 9 10 ) - скопируйте для быстрого ввода

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

{

cin >> arr[i];

}

}

void outprint(int\* arr, int n)

{

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

{

cout << arr[i] << " ";

}

}

void sort(int\* arr, int n)

{

int flag;

int i;

int tmp;

do

{

flag = 0;

for (i = 0; i < n - 1; i++)

{

if (arr[i] > arr[i + 1])

{

flag = 1;

tmp = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = tmp;

}

}

} while (flag);

return;

}

int inter\_search(int\* arr, int size, int key)

{

int left = 0; //левая граница

int right = size; //правая граница

int mid = 0; //переменная для записи индекса искомого элемента по формуле

bool flag = false;

while ((left <= right) && flag != true)

{

mid = left + ((key - arr[left]) \* (right - left)) / (arr[right] - arr[left]);//Формула для вычисления mid

if (arr[mid] < key) //Если искомое значение элемента множества больше значения элемента множества с индексом mid, то алгоритм начинает новый поиск в правой половине множества

{

left = mid + 1;

}

else if (arr[mid] > key) //Иначе, если искомое значение элемента множества меньше значения элемента множества с индексом mid, то алгоритм начинает новый поиск в левой половине множества

{

right = mid + 1;

}

else flag = true; //Иначе, то поворот флага

}

if (arr[left] == key)

{

return left; //Вывод "левую границу"

}

else if (arr[right] == key)

{

return right; //Вывод "правую границу"

}

else

{

return -1;//Если не нашёл

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

const int n = 10;

const int k = 5;

int\* arr = new int[n];

inprint(arr, n);

sort(arr, n);

cout << "Отсортированное множество: ";

outprint(arr, n);

int nechet[k] = { 1,3,5,7,9 };

int indexN = 0;

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

{

for (int j = 0; j < k; j++)

{

if (arr[i] == nechet[j])

{

int key = arr[i];

indexN = inter\_search(arr, n, key);

Destroy(arr, n, indexN);

}

}

}

cout << endl;

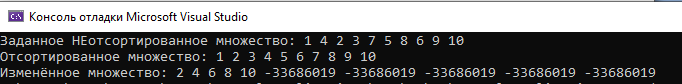
cout << "Изменённое множество: ";

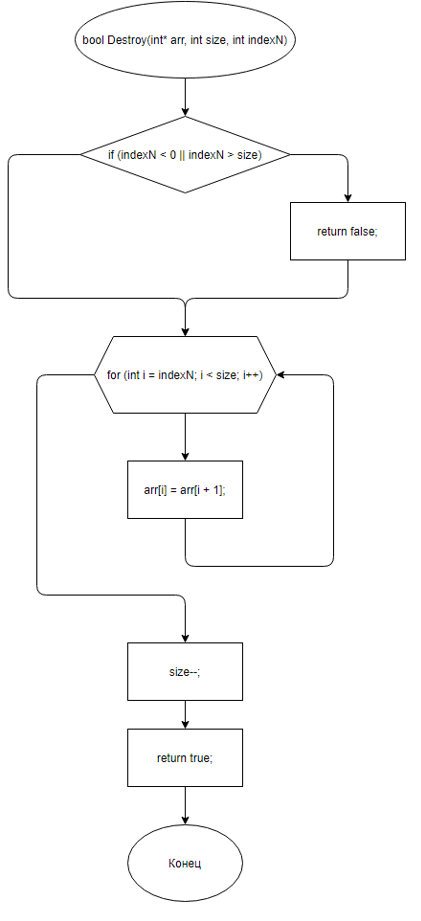
outprint(arr, n);

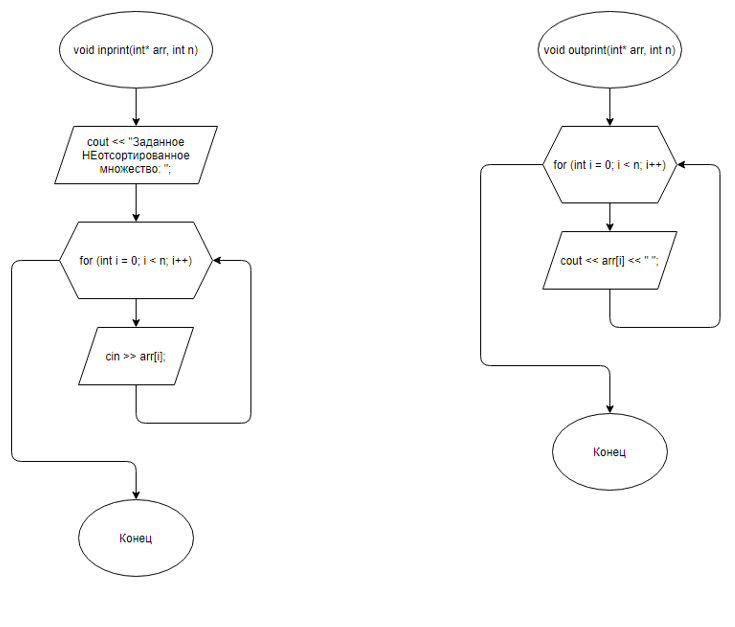
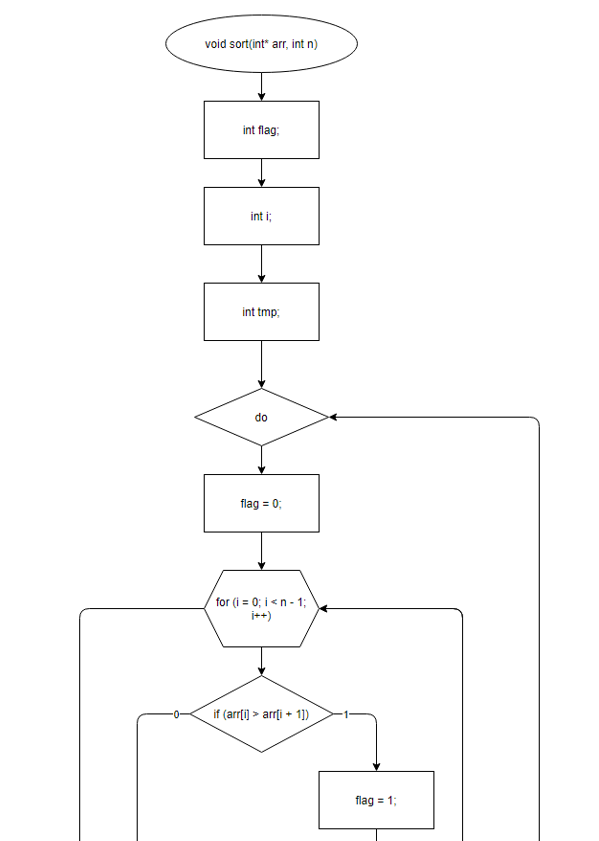
delete[] arr;

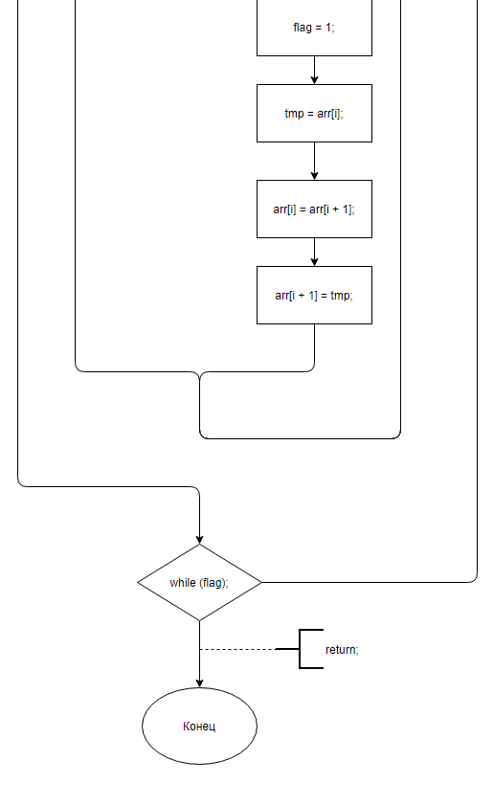
}

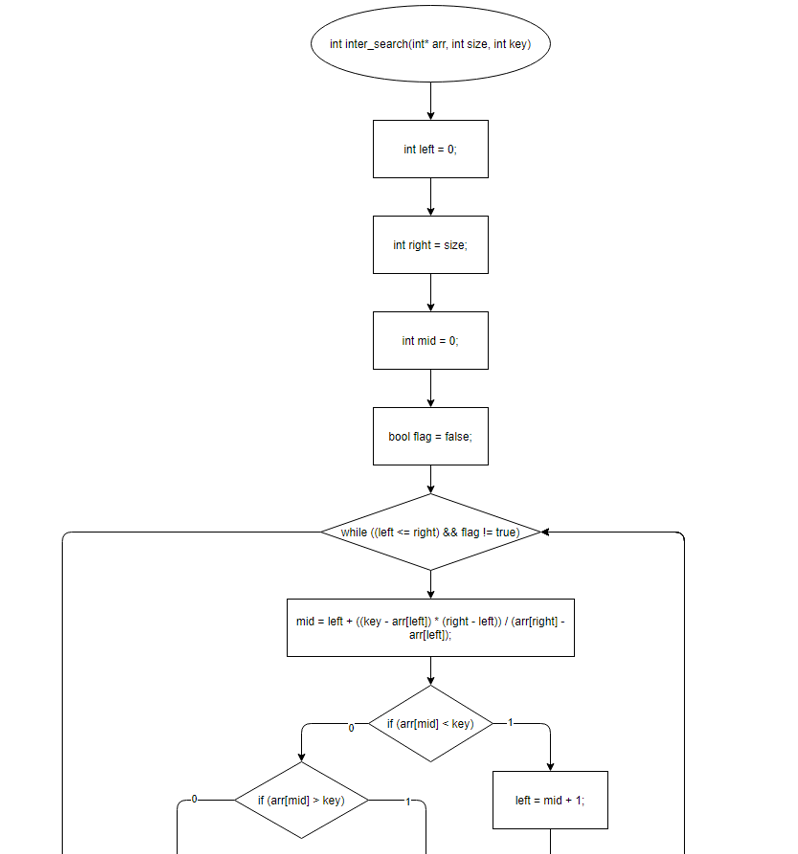
**Результат**

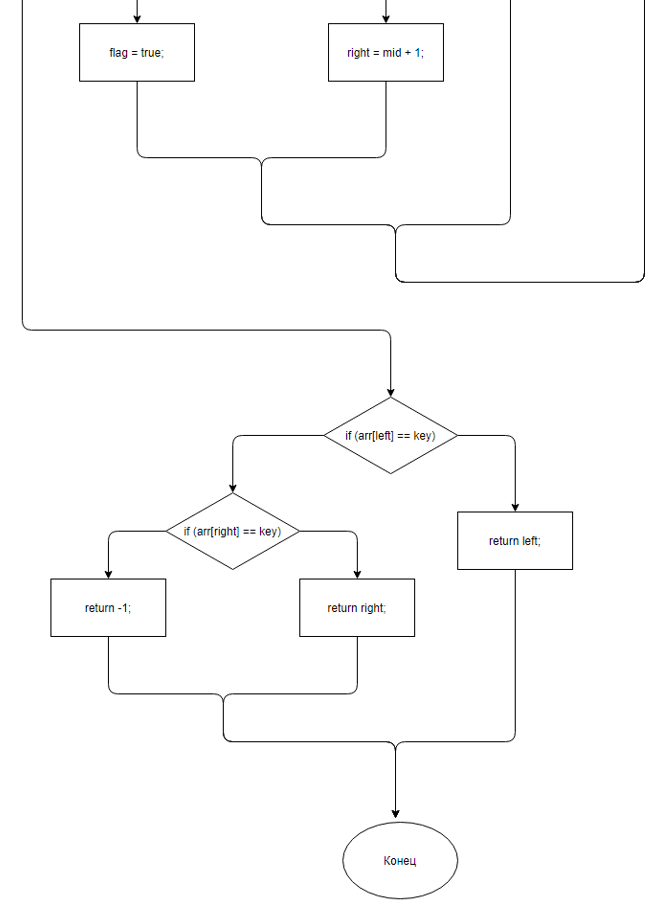


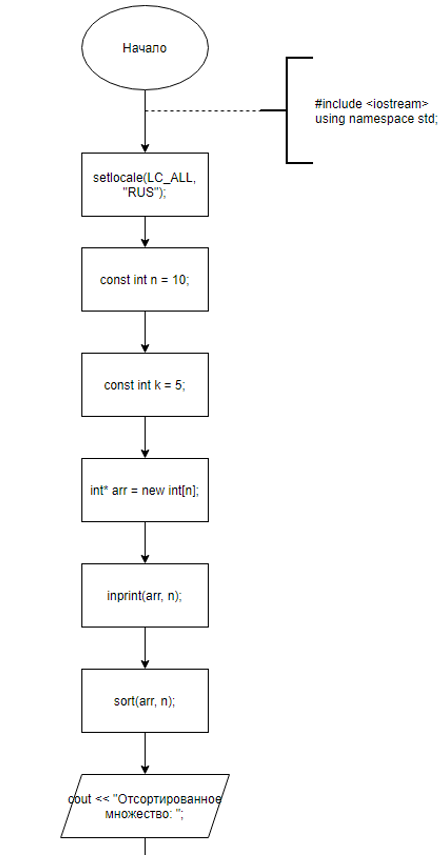
**Блок-схема  
  
**

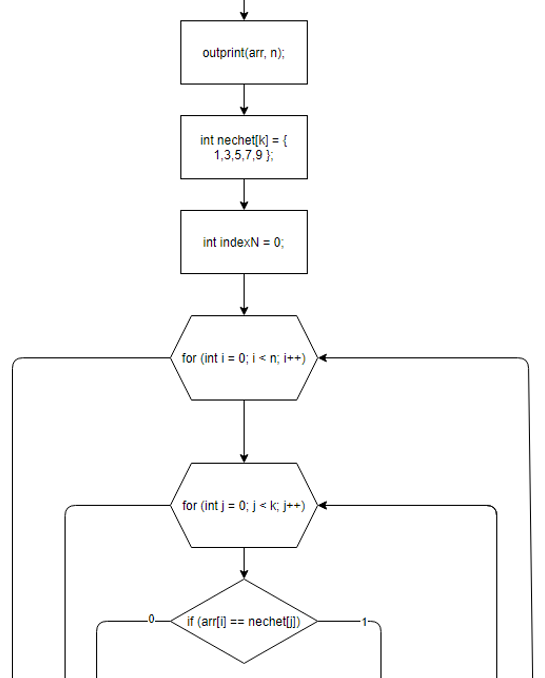
** **

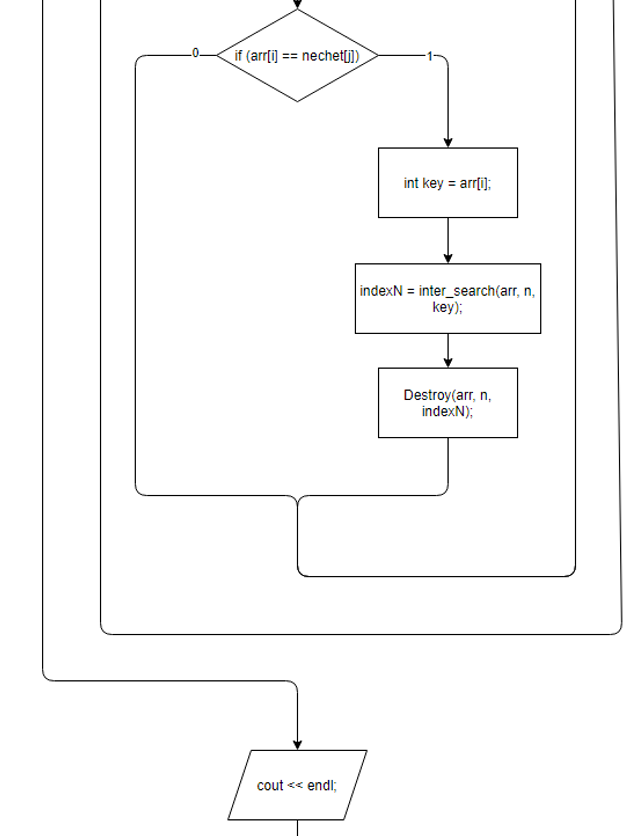
****

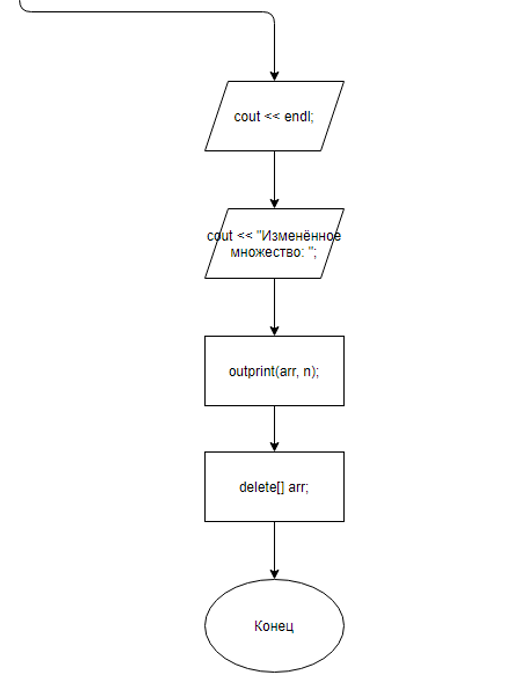
****

****

****

****

****

****