Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Тема: «Типы поисков в С++»

Выполнил

Студент РИС-24-1б

Конькова С. С.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2025

**Постановка задачи**

Изучить три вида поисков строки в C++.

**Анализ:**

Линейный:

1. Старт с первого элемента массива.
2. Текущий элемент сравнивается с искомым значением.
   1. Если совпадение найдено → возвращается индекс.
   2. Если нет → переход к следующему элементу.
3. При нахождении элемента → успешное завершение.
4. Если массив пройден до конца без совпадений → возврат -1.

Бинарный:

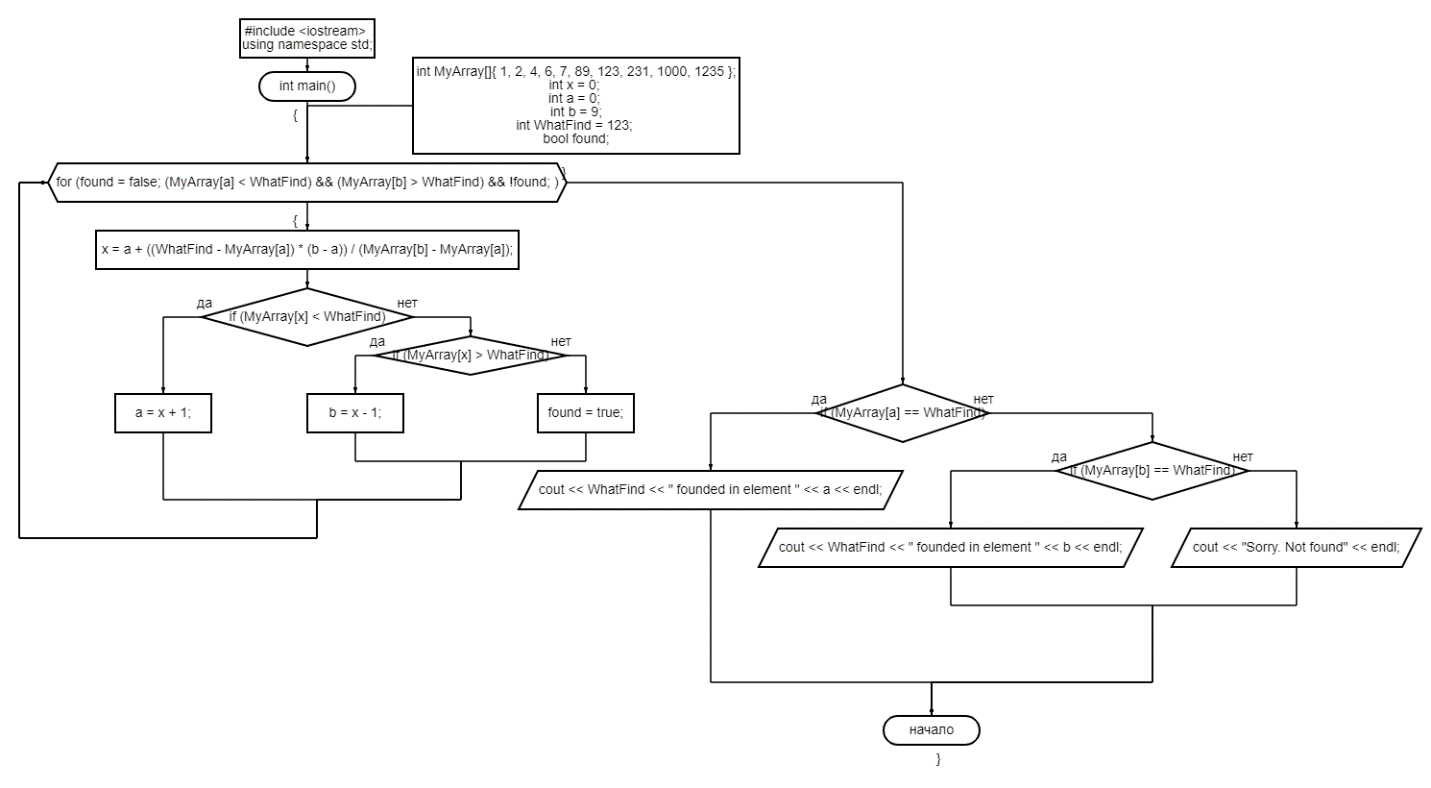
1. Массив должен быть отсортирован.
2. Вычисляется средний элемент (mid = left + (right - left) / 2).
3. Если arr[mid] == target → возврат mid.
4. Если arr[mid] < target → поиск в правой половине (left = mid + 1).
5. Если arr[mid] > target → поиск в левой половине (right = mid - 1).
6. Повтор: Шаги 2–3 выполняются, пока left <= right.
7. Завершение: Если элемент не найден → возврат -1.

Интерполяционный:

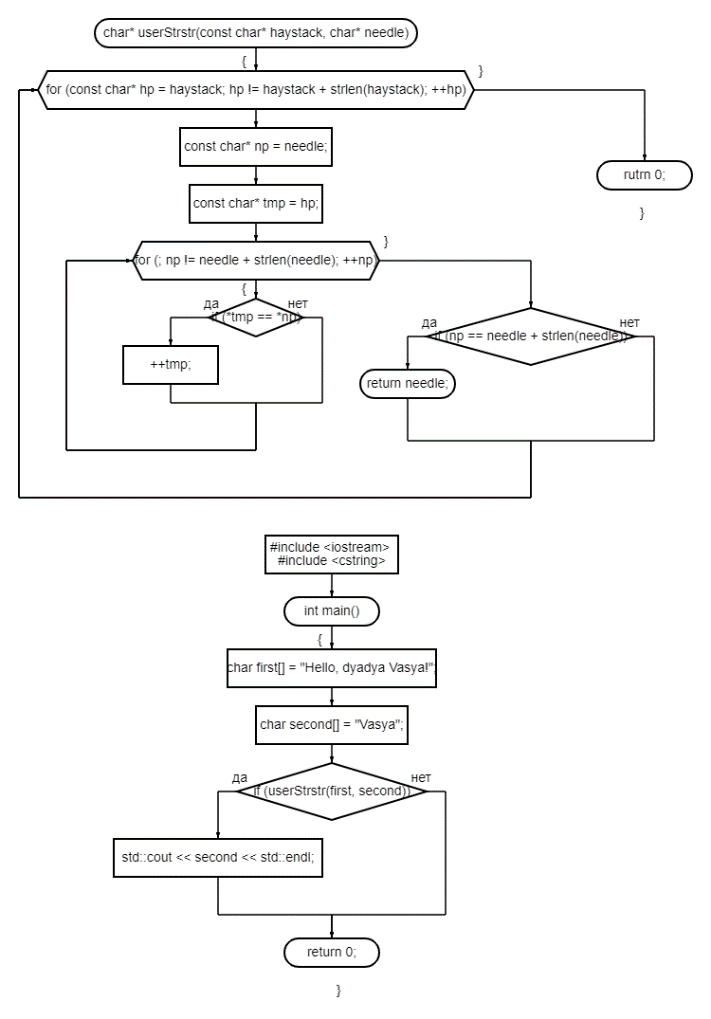
1. Массив отсортирован и значения распределены равномерно (например, [10, 20, 30, 40]).
2. Позиция делителя: Вычисляется по формуле: pos = low + ((target - arr[low]) \* (high - low)) / (arr[high] - arr[low])
3. Если arr[pos] == target → возврат pos.
4. Если arr[pos] < target → поиск в правой части (low = pos + 1).
5. Если arr[pos] > target → поиск в левой части (high = pos - 1).
6. Повтор: Шаги 2–3, пока low <= high и target в границах [arr[low], arr[high]].
7. Завершение: Если элемент не найден → возврат

**Блок – схема**

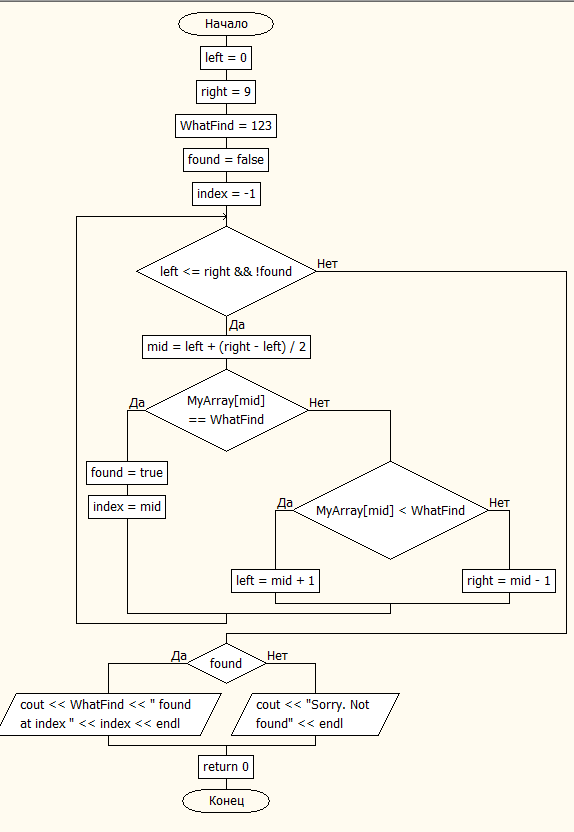
1. Интерполяционный



1. Линейный



1. Бинарный



**Коды программ**

1. Интерполяционный

#include <iostream>

using namespace std;

int main()

{

int MyArray[]{ 1, 2, 4, 6, 7, 89, 123, 231, 1000, 1235 };

int x = 0;

int a = 0;

int b = 9;

int WhatFind = 123;

bool found;

for (found = false; (MyArray[a] < WhatFind) && (MyArray[b] > WhatFind) && !found; )

{

x = a + ((WhatFind - MyArray[a]) \* (b - a)) / (MyArray[b] - MyArray[a]);

if (MyArray[x] < WhatFind)

a = x + 1;

else if (MyArray[x] > WhatFind)

b = x - 1;

else

found = true;

}

if (MyArray[a] == WhatFind)

cout << WhatFind << " founded in element " << a << endl;

else if (MyArray[b] == WhatFind)

cout << WhatFind << " founded in element " << b << endl;

else

cout << "Sorry. Not found" << endl;

return 0;

}

1. Линейный

#include <iostream>

#include <cstring>

char\* userStrstr(const char\* haystack, char\* needle)

{

for (const char\* hp = haystack; hp != haystack + strlen(haystack); ++hp)

{

const char\* np = needle;

const char\* tmp = hp;

for (; np != needle + strlen(needle); ++np)

{

if (\*tmp != \*np)

{

break;

}

else

{

++tmp;

}

}

if (np == needle + strlen(needle))

{

return needle;

}

}

return 0;

}

int main()

{

char first[] = "Hello, dyadya Vasya!";

char second[] = "Vasya";

if (userStrstr(first, second))

{

std::cout << second << std::endl;

}

}

1. Бинарный

#include <iostream>

using namespace std;

int main() {

int MyArray[]{ 1, 2, 4, 6, 7, 89, 123, 231, 1000, 1235 };

int left = 0; // Левая граница поиска

int right = 9; // Правая граница поиска (размер массива - 1)

int WhatFind = 123; // Искомый элемент

bool found = false; // Флаг нахождения элемента

int index = -1; // Индекс найденного элемента

while (left <= right && !found) {

int mid = left + (right - left) / 2; // Средний элемент

if (MyArray[mid] == WhatFind) {

found = true;

index = mid;

} else if (MyArray[mid] < WhatFind) {

left = mid + 1; // Ищем в правой половине

} else {

right = mid - 1; // Ищем в левой половине

}

}

if (found) {

cout << WhatFind << " found at index " << index << endl;

} else {

cout << "Sorry. Not found" << endl;

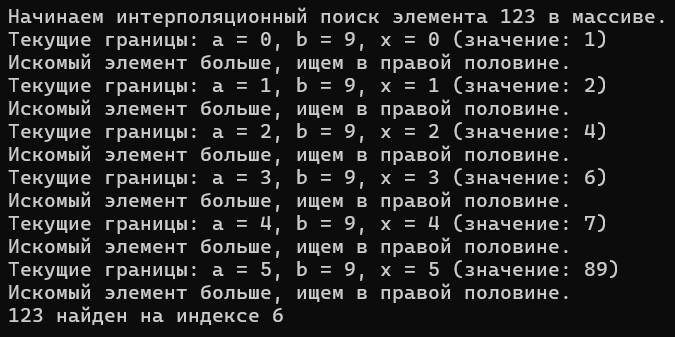
}

return 0;

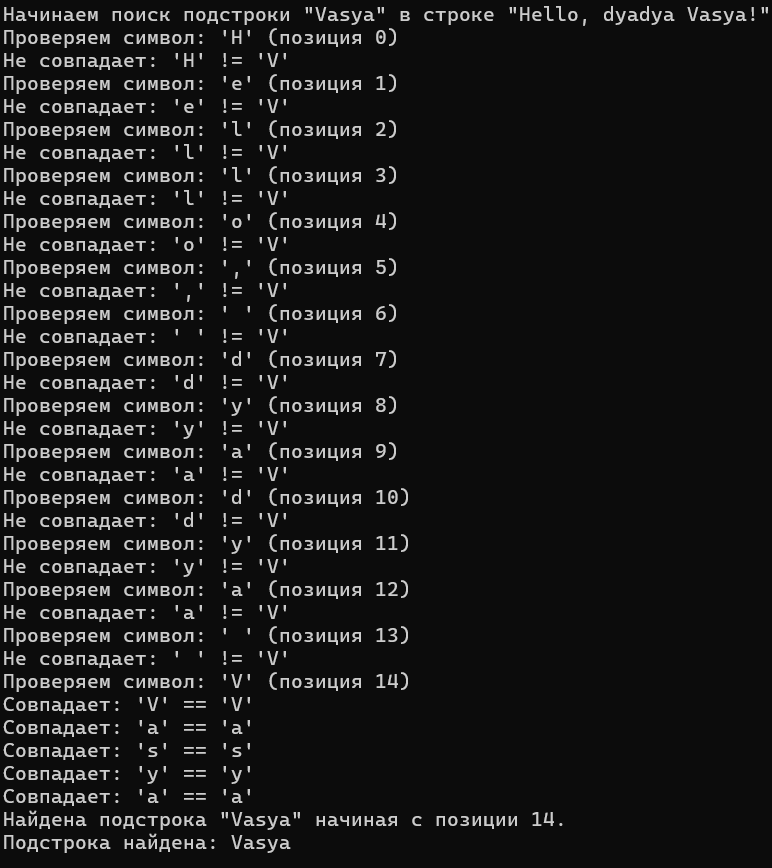
}

**Результаты программы**

1. Интерполяционный



1. Прямой



1. Бинарный

