

Практическая работа №1

Алгоритмы поиска

Цель: Изучить алгоритмы поиска.

Оснащение:

Персональный компьютер под управлением ОС Windows 10.

Офисный пакет Microsoft Office 2016

Среда разработки Visual Studio 2019

Ход работы

1. Линейный поиск

Линейный, последовательный поиск — алгоритм нахождения заданного значения произвольной функции на некотором отрезке. Данный алгоритм является простейшим алгоритмом поиска.

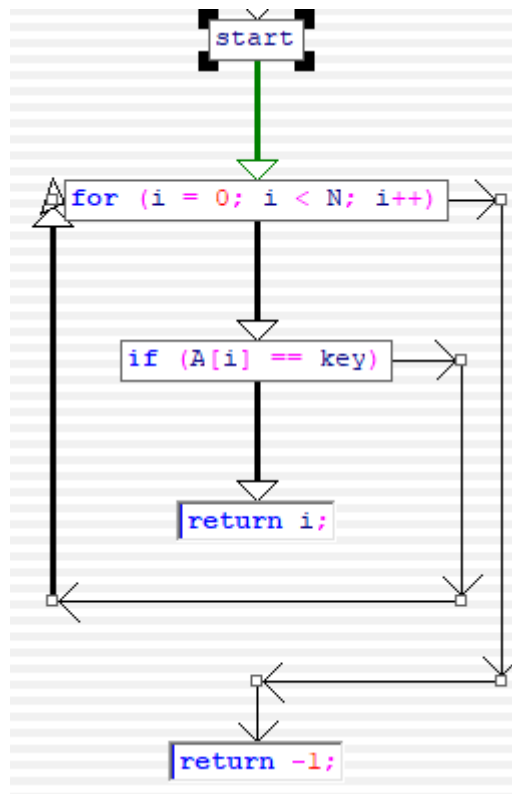


Рисунок 1 - Диаграмма деятельности алгоритм линейного поиска

Код программы

```
#include <iostream>
#include <ctime>
using namespace std;
```

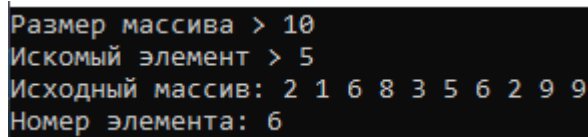
					<i>AuCD.09.03.02.050000 ПР</i>		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Ермошина В.А			Лабораторная работа 1 Алгоритмы поиска	Лит.	Лист
Провер.		Берёза А.Н.					Листов
Реценз							2
Н. Контр.							10
Утверд.						ИСОиП (филиал) ДГТУ в г.Шахты ИСТ-Тб21	

```

int i, N;
//линейный поиск
int LineSearch(int A[], int key)
{
    for (i = 0; i < N; i++)
        if (A[i] == key) return i;
    return -1;
}
//главная функция
void main()
{
    setlocale(LC_ALL, "Rus");
    int key, A[1000];
    srand(time(NULL));
    cout << "Размер массива > "; cin >> N;
    cout << "Искомый элемент > "; cin >> key;
    cout << "Исходный массив: ";
    for (i = 0; i < N; i++)
    {
        A[i] = rand() % 10;
        cout << A[i] << " ";
    }
    if (LineSearch(A, key) == -1) cout << "\nЭлемент не найден";
    else cout << "\nНомер элемента: " << LineSearch(A, key) + 1;
    system("pause>>void");
}

```

Результат



```

Размер массива > 10
Искомый элемент > 5
Исходный массив: 2 1 6 8 3 5 6 2 9 9
Номер элемента: 6

```

Рисунок 2 - Результат работы программы

2. Бинарный поиск

Двоичный (бинарный) поиск (также известен как метод деления пополам или дихотомия) — классический алгоритм поиска элемента в отсортированном массиве (векторе), использующий дробление массива на половины.

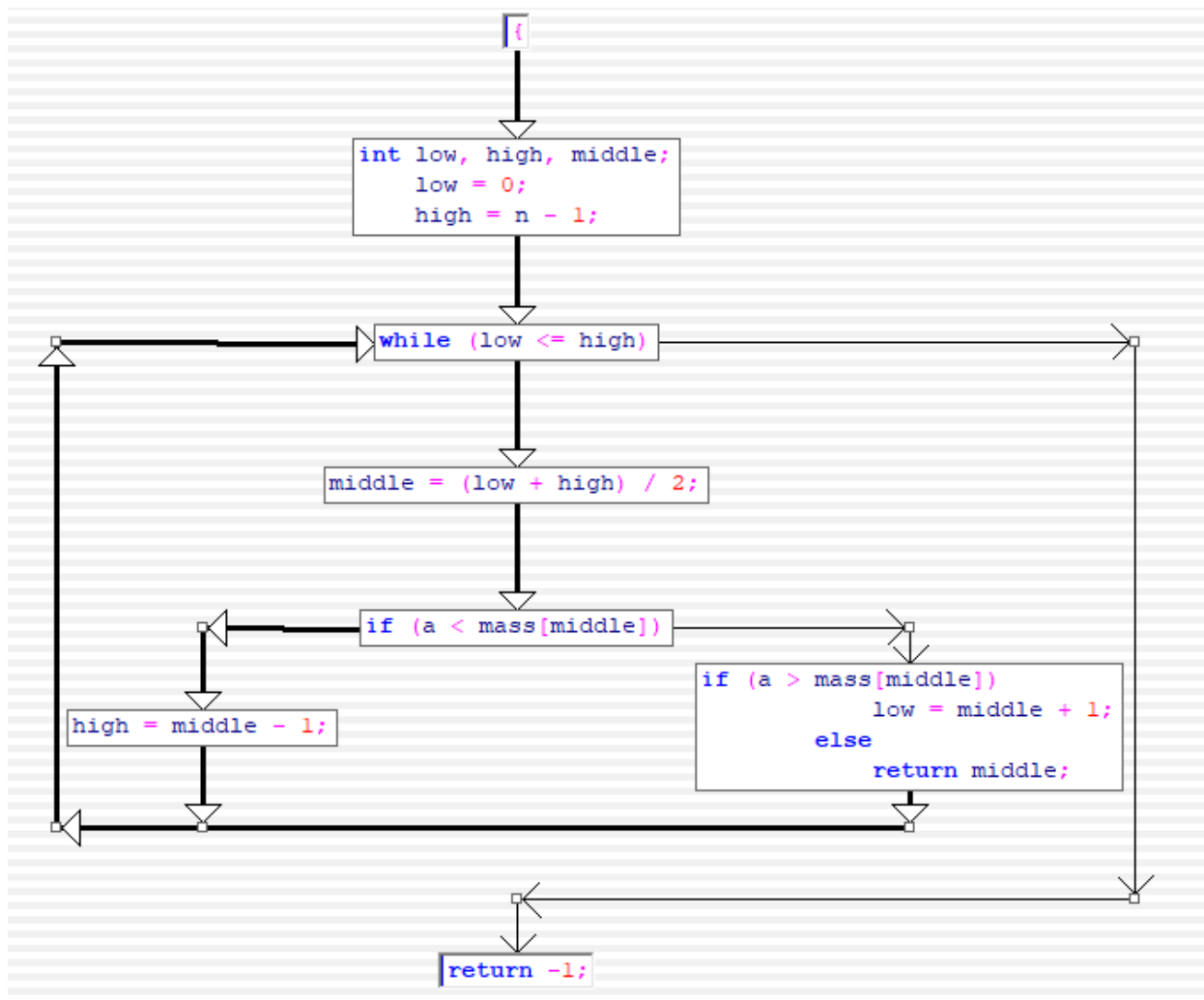


Рисунок 3 – Алгоритм бинарного поиска

Код программы

```

#include <stdio.h>
#include <malloc.h>
#include <conio.h>
#include <stdlib.h>
#include <ctime>
#include <iostream>
int binarysearch(int a, int mass[], int n);
void InsertionSort(int n, int mass[]);
int main()
{
    setlocale(LC_ALL, "Rus");
    //ВВОД N
    int N, a;
    printf("Введите число элементов массива: ");
    scanf_s("%d", &N);
    //выделение памяти под массив
    int* mass;
    mass = (int*)malloc(N * sizeof(int));
    //ВВОД элементов массива
    srand(time(NULL));
    for (int i = 0; i < N; i++)
        mass[i] = rand();

```

```

//сортировка вставками
InsertionSort(N, mass);
//Вывод отсортированного массива на экран
printf("Отсортированный массив:\n");
for (int i = 0; i < N; i++)
    printf("%d ", mass[i]);
printf("\n");
//ввод искомого элемента
printf("Введите искомый элемент: ");
scanf_s("%d", &a);
int k;
//двоичный поиск
k = binarysearch(a, mass, N);
if (k != -1)
{
    printf("Индекс элемента массива %d\n", k);
}
else
    printf("Искомый элемент не найден!\n");
//освобождение памяти
free(mass);
_getch();
return 0;
}

int binarysearch(int a, int mass[], int n)
{
    int low, high, middle;
    low = 0;
    high = n - 1;
    while (low <= high)
    {
        middle = (low + high) / 2;
        if (a < mass[middle])
            high = middle - 1;
        else if (a > mass[middle])
            low = middle + 1;
        else
            return middle;
    }
    return -1;
}

//сортировка вставками
void InsertionSort(int n, int mass[])
{
    int newElement, location;
    for (int i = 1; i < n; i++)
    {
        newElement = mass[i];
        location = i - 1;
        while (location >= 0 && mass[location] > newElement)
        {
            mass[location + 1] = mass[location];
            location = location - 1;
        }
    }
}

```

```

    }
    mass[location + 1] = newElement;
}
}

```

Результат

```

Введите число элементов массива: 10
Отсортированный массив:
3024 3142 6499 11374 12277 14975 18617 26291 28936 32168
Введите искомый элемент: 3024
Индекс элемента массива 0

```

Рисунок 4 – Результат работы программы

3. Числа Фибоначчи

Числами Фибоначчи называют элементы числовой последовательности. В ней каждое следующее число в ряду получается суммированием двух предыдущих чисел.

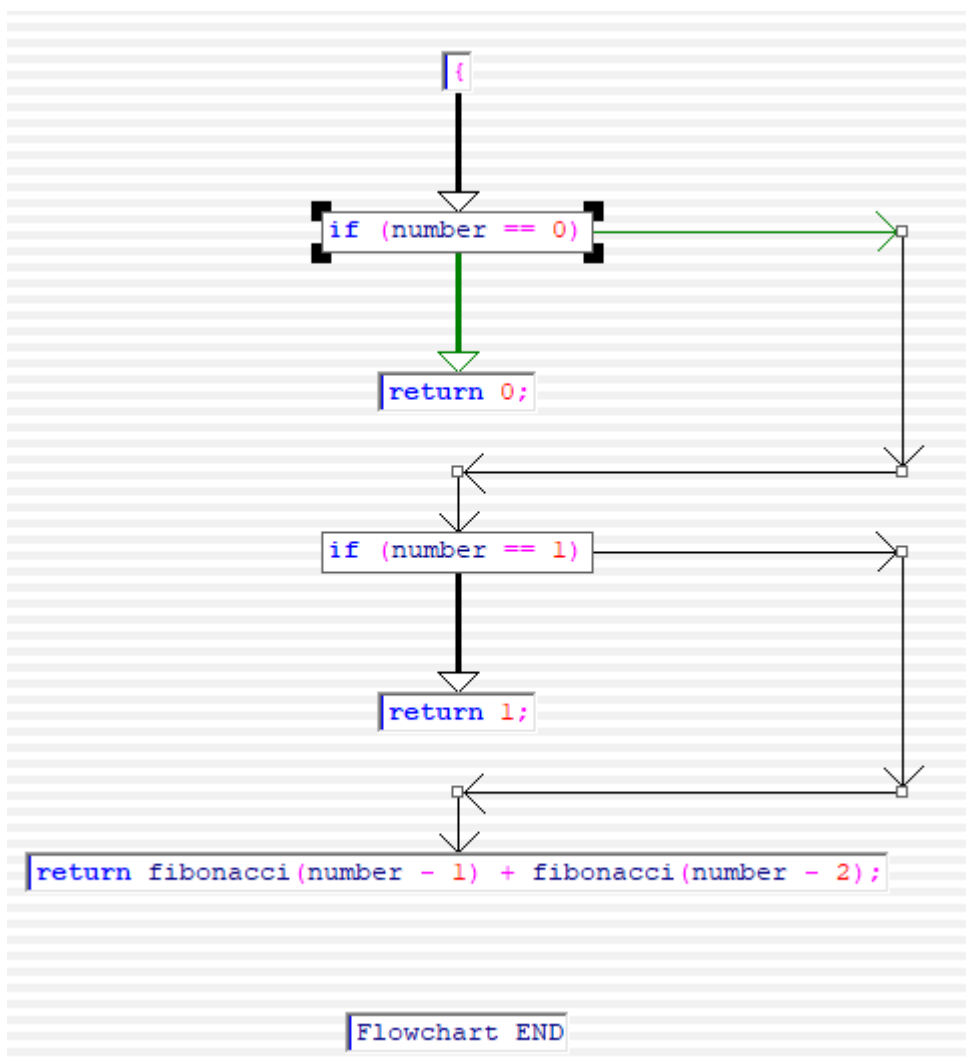


Рисунок 5 –Алгоритм вычисление чисел Фибоначчи

Код программы

```
#include <iostream>
```

```

int fibonacci(int number)
{
    if (number == 0)
        return 0; // базовый случай (условие завершения)
    if (number == 1)
        return 1; // базовый случай (условие завершения)
    return fibonacci(number - 1) + fibonacci(number - 2);
}
// Выводим первые 13 чисел Фибоначчи
int main()
{
    for (int count = 0; count < 13; ++count)
        std::cout << fibonacci(count) << " ";
    return 0;
}

```

Результат

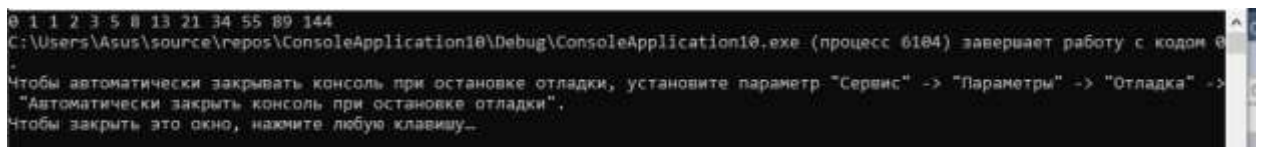


Рисунок 6 – Результат работы программы

4. Интерполяционный поиск

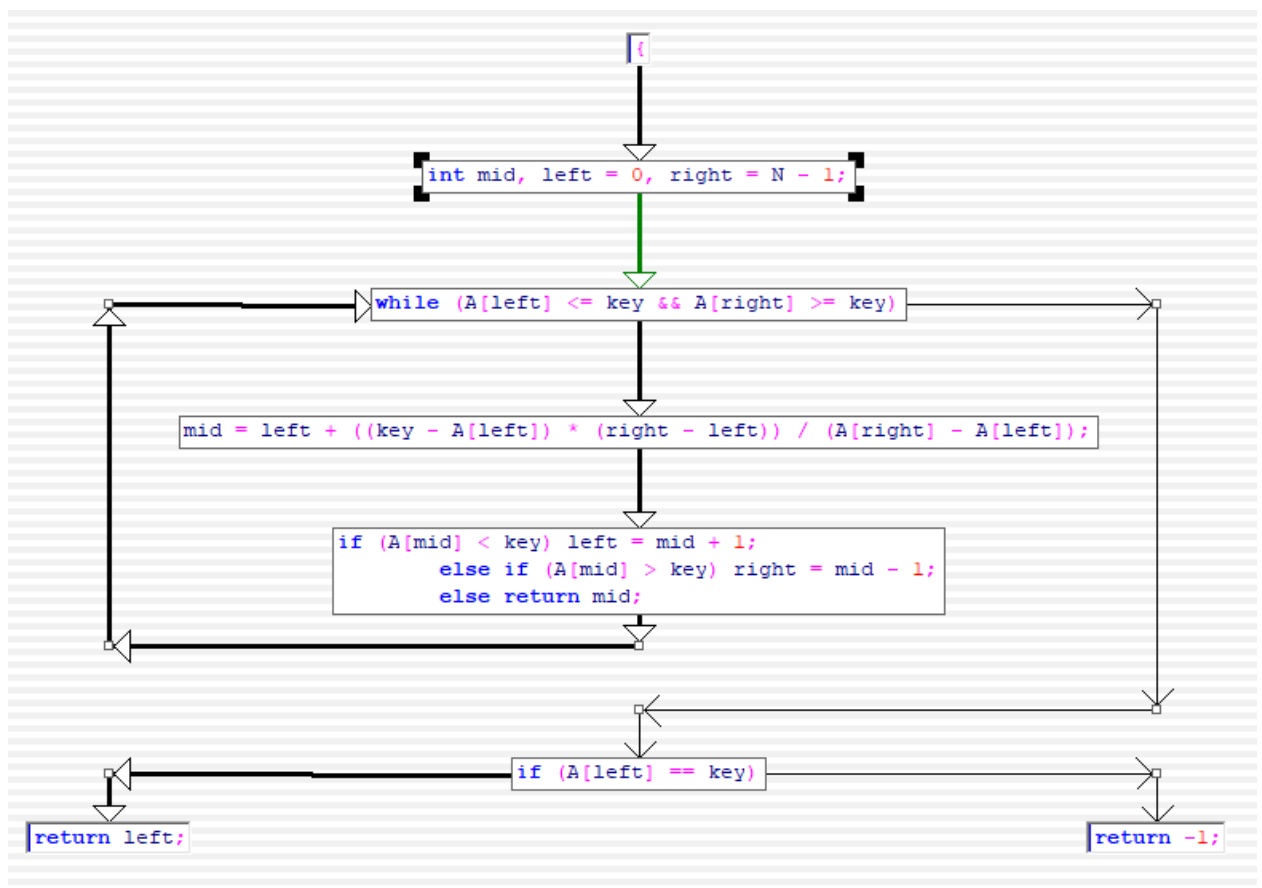


Рисунок 7 – Алгоритм интерполяционного поиска

Код программы

```
#include <iostream>
```

```

#include <cstdlib>
#include <ctime>

using namespace std;
const int N = 20;
int InterpolSearch(int A[], int key)
{
    int mid, left = 0, right = N - 1;
    while (A[left] <= key && A[right] >= key)
    {
        mid = left + ((key - A[left]) * (right - left)) / (A[right] - A[left]);
        if (A[mid] < key) left = mid + 1;
        else if (A[mid] > key) right = mid - 1;
        else return mid;
    }
    if (A[left] == key) return left;
    else return -1;
}
//главная функция
int main()
{
    setlocale(LC_ALL, "Rus");
    int i, key;
    srand(time(NULL));
    int A[N] = { };
    for (int i = 0; i < N; i++)
    {
        A[i] = rand();
    }
    cout << "Искомый элемент > "; cin >> key; //ввод ключа
    cout << "Исходный массив: ";
    for (i = 0; i < N; i++) cout << A[i] << " "; //вывод массива
    if (InterpolSearch(A, key) == -1) cout << "\nЭлемент не найден";
    else cout << "\nНомер элемента: " << InterpolSearch(A, key) + 1;
    system("pause>>void");
}

```

Результат



Искомый элемент > 10000
Исходный массив: 17391 23364 17125 16852 10118 802 28071 3004 22404 1350 32544 9442 25497 6579 23060 1166 10235 27156 27992 23247
Элемент не найден

Рисунок 8 - Результат работы программы

Вывод: В ходе выполнения практической работы изучила алгоритмы поиска и написала программы, реализующие линейный поиск, бинарный поиск, вычисление чисел Фибоначчи и интерполяционный поиск.