Лабораторная работа №10
Тема: «Алгоритмы сортировки и поиска»

Цель работы: Сформировать знания и умения по изучению методов внутренних сортировок. Приобретение навыков реализации алгоритмов сортировок.

Время выполнения: 6 часа.

Теоретические сведения

Алгоритмы сортировок подробно рассмотрены в лекции по данной теме.

Рассмотрим алгоритмы поиска.

В ниже приведенном примере объявим массив на 50 элементов и заполним его используя генератор случайных чисел rand(). Предложим пользователю ввести искомое значение с клавиатуры и реализуем проверку на наличие этого значения в нашем массиве. Если значение будет найдено в каком-либо элементе массива – выведем на экран индекс этого элемента. Это алгоритм линейного поиска.

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

//прототипы функций
int linSearch(int arr[], int requiredKey, int size); // линейный поиск
void showArr(int arr[], int size); // показ массива

int main()
{
	setlocale(LC_ALL, "rus");
	const int arrSize = 50;
	int arr[arrSize];
	int requiredKey = 0; // искомое значение (ключ)
	int nElement = 0; // номер элемента массива
	srand(time(NULL));

	//запись случ. чисел в массив от 1 до 50
	for (int i = 0; i < arrSize; i++)
	{
		arr[i] = 1 + rand() % 50;
```

```
        }

        showArr(arr, arrSize);

        cout << "Какое число необходимо искать? ";
        cin >> requiredKey; // ввод искомого числа

        //поиск искомого числа и запись номера элемента
        nElement = linSearch(arr, requiredKey, arrSize);

        if (nElement != -1)
        {
            //если в массиве найдено искомое число - выводим индекс элемента на экран
            cout << "Значение " << requiredKey << " находится в ячейке с индексом: " << nElement << endl;
        }
        else
        {
            //если в массиве не найдено искомое число
            cout << "В массиве нет такого значения" << endl;
        }
        return 0;
    }

    //вывод массива на экран
    void showArr(int arr[], int arrSize)
    {
        for (int i = 0; i < arrSize; i++)
        {
            cout << setw(4) << arr[i];
            if ((i + 1) % 10 == 0)
            {
                cout << endl;
            }

        }
        cout << endl << endl;
    }
```

```
//линейный поиск
int linSearch(int arr[], int requiredKey, int arrSize)
{
	for (int i = 0; i < arrSize; i++)
	{
		if (arr[i] == requiredKey)
			return i;
	}
	return -1;
}

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

//прототипы функций
int linSearch(int arr[], int requiredKey, int size); // линейный поиск
void showArr(int arr[], int size); // показ массива

int main()
{
 setlocale(LC_ALL, "rus");
 const int arrSize = 50;
 int arr[arrSize];
 int requiredKey = 0; // искомое значение (ключ)
 int nElement = 0; // номер элемента массива
 srand(time(NULL));

 //запись случ. чисел в массив от 1 до 50
 for (int i = 0; i < arrSize; i++)
 {
 arr[i] = 1 + rand() % 50;
 }

 showArr(arr, arrSize);

 cout << "Какое число необходимо искать? ";
 cin >> requiredKey; // ввод искомого числа
```

```
	//поиск искомого числа и запись номера элемента
	nElement = linSearch(arr, requiredKey, arrSize);

	if (nElement != -1)
	{
	//если в массиве найдено искомое число - выводим индекс элемента на экран
	cout << "Значение " << requiredKey << " находится в ячейке с индексом: " << nElement << endl;
	}
	else
	{
	//если в массиве не найдено искомое число
	cout << "В массиве нет такого значения" << endl;
	}
	return 0;
	}

	//вывод массива на экран
	void showArr(int arr[], int arrSize)
	{
	for (int i = 0; i < arrSize; i++)
	{
	cout << setw(4) << arr[i];
	if ((i + 1) % 10 == 0)
	{
	cout << endl;
	}

	}
	cout << endl << endl;
	}

	//линейный поиск
	int linSearch(int arr[], int requiredKey, int arrSize)
	{
	for (int i = 0; i < arrSize; i++)
	{
	if (arr[i] == requiredKey)
	return i;
```

```
 }
 return -1;
}
```

Функция выполняющая линейный поиск определена в строках 62-70. Она возвращает в программу -1 в том случае, если значение, которое ищет пользователь, не будет найдено в массиве. Если же значение будет найдено – функция вернет индекс элемента массива, в котором это значение хранится.

Двоичный (бинарный) поиск является более эффективным (проверяется асимптотическим анализом алгоритмов) решением в случае, если массив заранее отсортирован.

Предположим, что массив из 12-ти элементов отсортирован по возрастанию.

Пользователь задает искомое значение (ключ поиска). Допустим 4. На первой итерации массив делится на две части (ищем средний элемент – midd): (0 + 11) / 2 = 5 (0.5 отбрасываются). Сначала, проверяется значение среднего элемента массива. Если оно совпадает с ключом – алгоритм прекратит работу и программа выведет сообщение, что значение найдено. В нашем случае, ключ не совпадает со значением среднего элемента.

Если ключ меньше значения среднего элемента, алгоритм не будет проводить поиск в той половине массива, которая содержит значения больше ключа (т.е. от среднего элемента до конца массива). Правая граница поиска сместится (midd – 1). Далее снова деление массива на 2.

Ключ снова не равен среднему элементу. Он больше него. Теперь левая граница поиска сместится (midd + 1).

На третьей итерации средний элемент – это ячейка с индексом 3: (3 + 4) / 2 = 3. Он равен ключу. Алгоритм завершает работу.

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

// функция с алгоритмом двоичного поиска
int Search_Binary (int arr[], int left, int right, int key)
{
 int midd = 0;
 while (1)
 {
 midd = (left + right) / 2;

 if (key < arr[midd])       // если искомое меньше значения в ячейке
 right = midd - 1;      // смещаем правую границу поиска
```

```
        else if (key > arr[midd])  // если искомое больше значения в ячейке
        left = midd + 1;    // смещаем левую границу поиска
        else                // иначе (значения равны)
        return midd;          // функция возвращает индекс ячейки

        if (left > right)         // если границы сомкнулись
        return -1;
        }
       }

       int main()
       {
        setlocale (LC_ALL, "rus");

        const int SIZE = 12;
        int array[SIZE] = {};
        int key = 0;
        int index = 0; // индекс ячейки с искомым значением

        for (int i = 0; i < SIZE; i++) // заполняем и показываем массив
        {
        array[i] = i + 1;
        cout << array[i] << " | ";
        }

        cout << "\n\nВведите любое число: ";
        cin >> key;

        index = Search_Binary (array, 0, SIZE, key);

        if (index >= 0)
        cout << "Указанное число находится в ячейке с индексом: " << index <<
"\n\n";
        else
        cout << "В массиве нет такого числа!\n\n";

        return 0;
       }
```

Двоичный поиск является эффективным алгоритмом — его оценка сложности O(log2(n)), в то время как у обычного последовательного поиска

O(n). Это значит, что например, для массива из 1024 элементов линейный поиск в худшем случае (когда искомого элемента нет в массиве) обработает все 1024 элемента, но бинарным поиском достаточно обработать log2(1024) = 10 элементов. Такой результат достигается за счет того, что после первого шага цикла область поиска сужается до 512 элементов, после второго – до 256 и т.д.

Недостатками такого алгоритма является требование упорядоченности данных, а также возможности доступа к любому элементу данных за постоянное (не зависящее от количества данных) время. Таким образом алгоритм не может работать на неупорядоченных массивах и любых структурах данных, построенных на базе связных списков.

Индивидуальные задания к лабораторной работе №10

!!! Не использовать встроенные методы сортировок.

1. Отсортировать строки массива целых чисел по убыванию. Сортировка включением.
2. Отсортировать столбцы массива целых чисел по возрастанию. Шейкерная сортировка.
3. Отсортировать нечетные столбцы массива по возрастанию. Сортировка прямой выбор.
4. Отсортировать элементы нечетных строк массива целых чисел по убыванию. Сортировка разделением.
5. Отсортировать элементы квадратной вещественной матрицы размерности n, применив сортировку бинарным включением.
6. Отсортировать элементы квадратной вещественной матрицы размерности n, применив пузырьковую сортировку.
7. Отсортировать положительные элементы одномерного массива, отрицательные оставить на местах. Пузырьковая сортировка.
8. Отсортировать строки массива целых чисел по убыванию. Шейкерная сортировка.
9. Отсортировать столбцы массива целых чисел по возрастанию. Сортировка включением.
10. Отсортировать нечетные столбцы массива по возрастанию. Сортировка разделением.
11. Отсортировать элементы нечетных строк массива целых чисел по убыванию. Сортировка прямой выбор.
12. Отсортировать элементы квадратной вещественной матрицы размерности n, применив пузырьковую сортировку слева направо.
13. Заданный одномерный массив отсортировать по возрастанию цифры десятков каждого элемента. Сортировка прямой выбор.

14. Отсортировать элементы квадратной вещественной матрицы размерности n, стоящие на побочной диагонали, применив сортировку бинарным включением.

15. Заданный одномерный массив отсортировать по возрастанию цифры десятков каждого элемента. Сортировка включением.

16. В одномерном массиве упорядочить отрицательные элементы, оставив положительные на местах. Сортировка включением.

17. В одномерном массиве упорядочить нечетные элементы, оставив четные на местах. Сортировка прямой выбор.

18. Упорядочить одномерный массив так, чтобы в начале располагались четные элементы в порядке возрастания их значений, а затем нечетные – в порядке убывания их значений.

19. В одномерном массиве упорядочить нечетные элементы, оставив четные на местах. Сортировка шейкерная.

20. Отсортировать положительные элементы одномерного массива, отрицательные оставить на местах. Сортировка прямой выбор.

21. Отсортировать строки массива целых чисел по убыванию. Шейкерная сортировка.

22. Отсортировать столбцы массива целых чисел по возрастанию. Сортировка включением.

23. Отсортировать нечетные столбцы массива по возрастанию. Сортировка разделением.

24. Отсортировать элементы нечетных строк массива целых чисел по убыванию. Сортировка прямой выбор.

25. Отсортировать элементы квадратной вещественной матрицы размерности n, стоящие на главной диагонали, применив пузырьковую сортировку слева направо.

26. Заданный одномерный массив отсортировать по возрастанию цифры десятков каждого элемента. Сортировка прямой выбор.

27. Отсортировать элементы квадратной вещественной матрицы размерности n, стоящие на побочной диагонали, применив сортировку бинарным включением.

28. Заданный одномерный массив отсортировать по возрастанию цифры десятков каждого элемента. Сортировка включением.

29. В одномерном массиве упорядочить отрицательные элементы, оставив положительные на местах. Сортировка включением.

30. В одномерном массиве упорядочить нечетные элементы, оставив четные на местах. Сортировка прямой выбор.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретическую часть лабораторной работы.

2. Реализовать индивидуальное задание по вариантам, представленные в теоретических сведениях, сделать скриншоты работающих программ. Написать комментарии.

3. Написать отчет, содержащий:

1. Титульный лист, на котором указывается:

а) полное наименование министерства образование и название учебного заведения;

б) название дисциплины;

в) номер практического занятия;

г) фамилия преподавателя, ведущего занятие;

д) фамилия, имя и номер группы студента;

е) год выполнения лабораторной работы.

2. Индивидуальное задание из раздела «Теоретические сведения» с кодом, комментариями и скриншотами работающих программ.

3. Построение блок-схем.