**Независимо от выбранной среды программирования, программный код должен быть скомпилирован в исполняемый \*.ехе файл с возможностью ввода исходных данных и вывода результата в экранную форму.**

**К каждой задачи прилагается пояснительная записка с листингом программного кода и скриншотами выполнения.**

**Тема 1. Поиск элемента в массиве**

1. Написать программу поиска перебором элемента массива равного заданному значению.

2. Написать программу поиска с барьером элемента массива равного заданному значению.

3. Написать программу поиска перебором и поиска с барьером элемента массива равного заданному значению. Оценить эффективность работы алгоритмов по количеству сравнений.

**4. Написать программу бинарного поиска элемента массива равного заданному значению.**

5. Написать программу бинарного поиска элемента массива равного заданному значению, в которой новое значение индекса m определялось бы не как среднее значение между L и R, а согласно правилу золотого сечения.

6. Провести анализ эффективности алгоритмов поиска: поиска перебором, поиска с барьером, бинарного поиска.

**Тема 2. Поиск образа в строке**

1. Написать программу поиска образа в строке по методу Кнута, Мор­риса и Пратта. Предусмотреть возможность существования в образе пробела. Ввести опцию чувствительности / нечувствительности к регистру.
2. Написать программу поиска образа в строке по методу Боуера и Мура. Предусмотреть возможность существования в образе пробела. Ввести опцию чувствительности / нечувствительности к регистру.
3. Ниже приведен листинг программы, формирующей таблицу d по КМП-алгоритму. При каком образе таблица d будет сформирована неверно? При какой строке и каком образе положительный результат не будет получен?

*m = strlen(img); n = strlen(str);*

j = 0; k = -1; d[0] = -1;

*while(j* < m)

{

while((k >= 0) && ( img[j] != img[k])) k = d[k]; j++; k++;

if (imgj] == img[k]) dj] = d[k]; else d j] = k;

}

1. Реализовать в программе алгоритм прямого поиска строки и КМП- алгоритм. Сравнить эффективность поиска образа в строке обоими алгорит­мами по количеству итераций.
2. **Реализовать в программе алгоритм прямого поиска строки и БМ- алгоритм. Сравнить эффективность поиска образа в строке обоими алгорит­мами по количеству итераций.**
3. Разработать и программно реализовать усовершенствованный алго­ритм, объединяющий БМ-алгоритм с КМП-алгоритмом, который при сдвиге образа использует две таблицы, полученные согласно данным алгоритмам.

**Тема 3. Сортировка массивов**

1. Реализовать в программе два алгоритма из указанных ниже:

а) сортировка с помощью прямого включения,

б) сортировка с помощью двоичного включения,

в) сортировка с помощью прямого выбора,

г) шейкерная сортировка,

д) сортировка Шелла.

Сравнить эффективность реализованных алгоритмов по числу перестановок.

1. Запрограммировать три метода прямой сортировки. Сравнить в про­грамме время выполнения каждого из методов.
2. Предположим, что необходимо отсортировать массив, состоящий из нескольких «случайных» элементов и следующих за ними упорядоченных элементов. Какой из рассмотренных в данной главе методов сортировки наи­более подходит для решения данной задачи?
3. Алгоритм называется *устойчивым,* если он сохраняет исходный по­рядок элементов с одинаковыми значениями. Какой из алгоритмов, пред­ставленных в данной главе, устойчив?
4. Рассмотрим алгоритм случайной сортировки, примененный к масси­ву *a[n]* целых чисел: выбирается случайное число *i* из интервала от 1 до *n* и меняются местами элементы a[0] и a[i], процесс повторяется до тех пор, пока не будут упорядочены все элементы массива. Каково ожидаемое время вы­полнения массива?
5. **Написать программу нахождения *к* наименьших элементов в массиве длиной *n*. Для каких значений *k* эффективнее сначала выполнить сортировку всего массива, а затем взять *к* наименьших элементов, вместо поиска наи­меньших элементов в неупорядоченном массиве?**