### Задание на выполнение лабораторной работы №1

1. Написать компьютерную программу, предоставляющую следующие возможности:
   1. Создание массива целых чисел, имеющего заданный пользователем размер.
   2. Заполнение созданного массива случайными числами или заполнение числами, введенными с клавиатуры (по выбору пользователя).
   3. Сортировка этого массива по возрастанию или по убыванию.
   4. Многократный линейный поиск различных чисел в несортированном массиве и бинарный поиск различных чисел в отсортированном массиве. Количество операций поиска задается пользователем.
   5. Подсчет количества сравнений чисел, выполняемых при операциях сортировки, многократного линейного и многократного бинарного поиска, и расчет среднего количества сравнений на одну операцию линейного и бинарного поиска.
2. При написании программы использовать собственные реализации алгоритмов сортировки, линейного и бинарного поиска.
3. Оценить временную сложность алгоритмов сортировки, линейного и бинарного поиска (в среднем и в худшем случаях).
4. Произвести практическую оценку временных затрат для операций сортировки, линейного и бинарного поиска путем многократного запуска программы и ввода различных размеров массива (10, 1000, 100000) и количеств операций поиска (10, 1000, 100000) (во всех комбинациях). Результаты свести в таблицу и включить в раздел 5 отчета.

## Теория

**Линейный, последовательный поиск** — алгоритм нахождения заданного значения произвольной функции на некотором отрезке. Данный алгоритм является простейшим алгоритмом поиска и, в отличие, например, от двоичного поиска, не накладывает никаких ограничений на функцию и имеет простейшую реализацию. Поиск значения функции осуществляется простым сравнением очередного рассматриваемого значения (как правило, поиск происходит слева направо, то есть от меньших значений аргумента к большим) и, если значения совпадают (с той или иной точностью), то поиск считается завершённым.

Если отрезок имеет длину N, то найти решение с точностью до {\displaystyle \epsilon }E можно за время {\displaystyle N \over \epsilon }N/E. Т.о. асимптотическая сложность алгоритма — {\displaystyle O(n)}O(n) В связи с малой эффективностью по сравнению с другими алгоритмами линейный поиск обычно используют, только если отрезок поиска содержит очень мало элементов, тем не менее, линейный поиск не требует дополнительной памяти или обработки/анализа функции, так что может работать в потоковом режиме при непосредственном получении данных из любого источника. Также линейный поиск часто используется в виде линейных алгоритмов поиска максимума/минимума.

**Двоичный (бинарный) поиск** (также известен как метод деления пополам и дихотомия) — классический алгоритм поиска элемента в отсортированном массиве (векторе), использующий дробление массива на половины. Используется в информатике, вычислительной математике и математическом программировании.

Частным случаем двоичного поиска является метод бисекции, который применяется для поиска корней заданной непрерывной функции на заданном отрезке.

Двоичный поиск заключается в том, что на каждом шаге множество объектов делится на две части и в работе остаётся та часть множества, где находится искомый объект. Или же, в зависимости от постановки задачи, мы можем остановить процесс, когда мы получим первый или же последний индекс вхождения элемента. Последнее условие — это левосторонний/правосторонний двоичный поиск.

## Метод решения

Для начала необходимо реализовать 3 функции: функцию линейного поиска, проверяющую на совпадение искомого элемента с каждым элементом массива; функцию бинарного поиска, реализующую алгоритм половинного деления отсортированного массива на 2 равных подмассива, один из которых в свою очередь опять делится на две части и так, пока не будет найден искомый элемент; функцию сортировки, отсортировывающую массив по возрастанию значений элементов. Затем необходимо реализовать логическую структуру программы:

### Структура программы:

1. Ввод размерности массива
2. Выбор способа заполнения массива (пользовательский ввод или заполнение случайными числами)
3. Создание массива чисел с размерностью, заданной пользователем
4. Заполнение массива в соответствии с выбранным пользователем способом заполнения
5. Сортировка массива по возрастанию
6. Ввод числа повторений операции поиска числа в массиве
7. Выбор способа ввода чисел для поиска (пользовательский ввод или поиск случайных чисел)
8. Повторение операции поиска в соответствии с выбором способа поиска пользователем и вывод результатов линейного и бинарного поиска, а также число итераций, произведенных функциями линейного и бинарного поиска
9. Выбор действия: выход или повторное использование программы

## Руководство программиста

int linearSearch(int \*arr, int N, int val); //функция линейного поиска числа val в массиве arr с длинной N

int binarySearch(int \*arr, int N, int val); //функция бинарного поиска числа val в массиве arr с длинной N

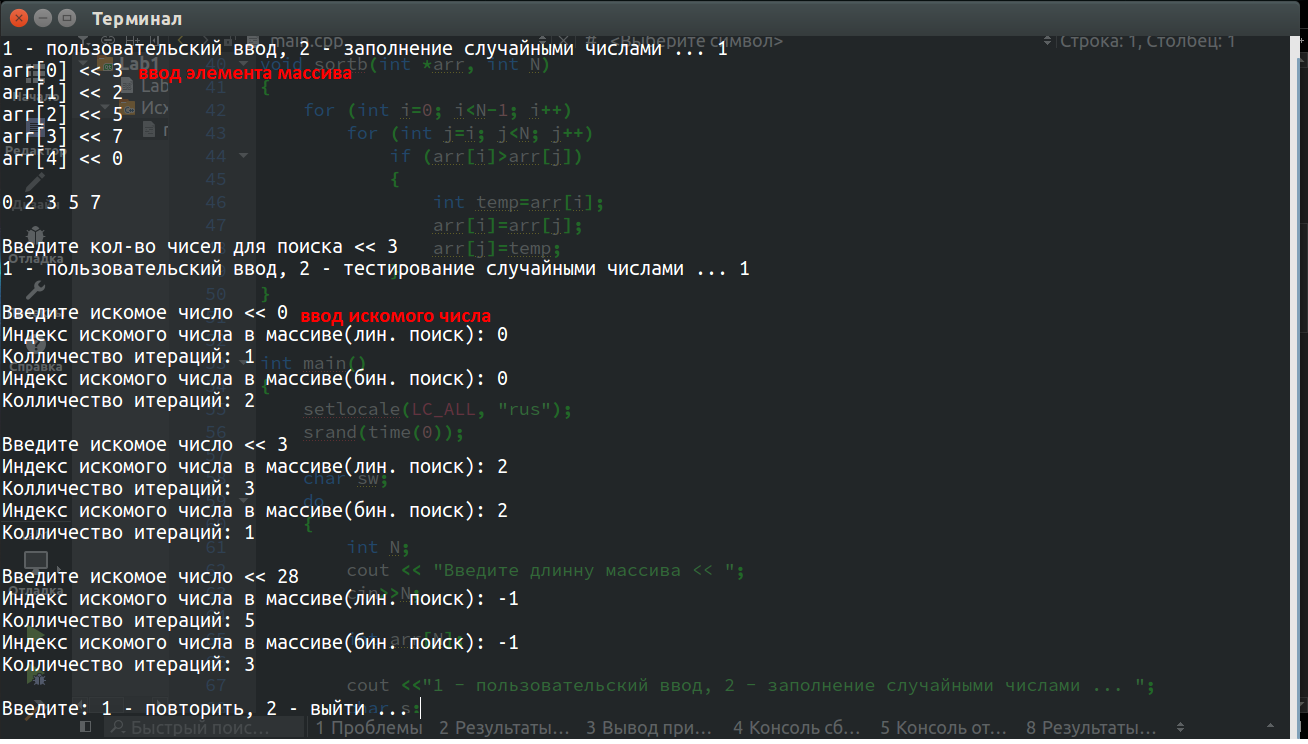
void sortb(int \*arr, int N); //сортировка «пузырьком» массива arr с длинной N

int LC; //счётчик итераций для функции линейного поиска, обнуляется в начале каждого вызова функции линейного поиска

int BC; //счётчик итераций для функции бинарного поиска, обнуляется в начале каждого вызова функции бинарного поиска

## Руководство пользователя со скриншотами





## Анализ результатов работы программы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Размерность | Кол-во повторений поиска | Среднее Число итераций для линейного поиска | Среднее Число итераций для бинарного поиска | Среднее Число итераций для пузырьковой сортировки |
| 1 | 10 | 10 | 6.8 | 3 | 54 |
| 2 | 10 | 100 | 6.7 | 2.96 | 54 |
| 3 | 10 | 1000 | 7.233 | 2.95 | 54 |
| 4 | 10 | 100000 | 6.1991 | 2.99562 | 54 |
| 5 | 100 | 10 | 76.2 | 5.3 | 5049 |
| 6 | 100 | 100 | 67.61 | 5.68 | 5049 |
| 7 | 100 | 1000 | 67.044 | 5.916 | 5049 |
| 8 | 100 | 100000 | 69.6585 | 5.82395 | 5049 |
| 9 | 1000 | 10 | 614.4 | 8.8 | 500499 |
| 10 | 1000 | 100 | 632.81 | 9.1 | 500499 |
| 11 | 1000 | 1000 | 666.35 | 8.986 | 500499 |
| 12 | 1000 | 100000 | 680.671 | 9.03972 | 500499 |
| 13 | 100000 | 10 | 55588.3 | 15.9 | 705082703 |
| 14 | 100000 | 100 | 68617.8 | 15.77 | 705082703 |
| 15 | 100000 | 1000 | 68980.9 | 15.637 | 705082703 |
| 16 | 100000 | 100000 | 68545 | 15.7322 | 705082703 |

В теории сложность линейного поиска в среднем и худшем случаях будет O(n), для бинарного поиска сложность для среднего и худшего случаев будет O(ln2(n)), что подтверждает таблица выше. Для сортировки пузырьком сложность для любого случая будет O(n\*n).

## Листинг программы

**#include** <iostream>

**#include** <cstdlib>

**#include** <ctime>

**using** **namespace** std;

**int** LC;

**int** BC;

**int** **linearSearch**(**int** \*arr, **int** N, **int** val)

{

LC = 0;

**for** (**int** i=0; i<N; i++)

{

LC++;

**if** (arr[i]==val) **return** i;

}

**return** -1;

}

**int** **binarySearch**(**int** \*arr,**int** N, **int** val)

{

BC = 0;

**int** left = 0, right = N-1;

**while** (1)

{

BC++;

**int** midl = (left + right) / 2;

**if** (val < arr[midl]) right = midl-1;

**else** **if** (val > arr[midl]) left = midl+1;

**else** **return** midl;

**if** (left > right) **return** -1;

}

}

**void** **sortb**(**int** \*arr, **int** N)

{

**for** (**int** i=0; i<N-1; i++)

**for** (**int** j=i; j<N; j++)

**if** (arr[i]>arr[j])

{

**int** temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

**int** **main**()

{

setlocale(**LC\_ALL**, "rus");

srand(time(0));

**char** sw;

**do**

{

**int** N;

cout << "Введите длинну массива << ";

cin>>N;

**int** arr[N];

cout <<"1 - пользовательский ввод, 2 - заполнение случайными числами ... ";

**char** s;

cin >> s;

**switch** (s)

{

**case** '1':

{

**for** (**int** i=0; i<N; i++)

{

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

cin >> arr[i];

}

} **break**;

**case** '2':

{

**for** (**int** i=0; i<N; i++) arr[i] = rand() % N;

} **break**;

**default**:

{

cout << "Ошибка!" << endl;

**return** 0;

}

}

sortb(arr,N);

cout << endl;

**for** (**int** i = 0; i < N; i++) cout << arr[i] << " ";

cout << endl << endl;

cout << "Введите кол-во чисел для поиска << ";

**int** val\_n;

cin >> val\_n;

cout << "1 - пользовательский ввод, 2 - тестирование случайными числами ... ";

cin >> s;

**switch**(s)

{

**case** '1':

{

**for** (**int** i = 0; i < val\_n; i++)

{

cout << endl << "Введите искомое число << ";

**int** val;

cin >> val;

cout << "Индекс искомого числа в массиве(лин. поиск): "

<< linearSearch(arr, N, val) << endl << "Колличество итераций: ";

cout << LC << endl;

cout << "Индекс искомого числа в массиве(бин. поиск): "

<< binarySearch(arr, N, val) << endl

<< "Колличество итераций: "; cout << BC << endl;

}

} **break**;

**case** '2':

{

**for** (**int** i = 0; i < val\_n; i++)

{

**int** val = rand() % N;

cout << endl << "Искомое число: " << val << endl;

cout << "Индекс искомого числа в массиве(лин. поиск): "

<< linearSearch(arr, N, val) << endl << "Колличество итераций: ";

cout << LC << endl;

cout << "Индекс искомого числа в массиве(бин. поиск): "

<< binarySearch(arr, N, val) << endl

<< "Колличество итераций: "; cout << BC << endl;

}

} **break**;

**default**:

{

cout << "Ошибка!" << endl;

**return** 0;

}

}

cout << endl << "Введите: 1 - повторить, 2 - выйти ... ";

cin >> sw;

cout << endl;

} **while**(sw == '1');

**return** 0;

}