Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

**Лабораторная работа №4**

**По дисциплине АиСД за 2 семестр**  
**Тема: «ИССЛЕДОВАНИЕ И ОЦЕНКА АЛГОРИТМОВ ПОИСКА»**

**Вариант 4**

**Выполнил:**

Студент группы ПО-6(1)  
 1-го курса

Мартынович Даниил

**Проверила:**

Гирель Т. Н.

**Цель работы**. Разработка программ, реализующих различные алгоритмы поиска, и оценка их временной и пространственной сложности.

**Ход работы**:

Алгоритм дихотомического поиска

Этот алгоритм является более быстрым, чем линейный, но применяется только к упорядоченным массивам. Среднее время дихотомического поиска пропорционально величине log2n, где n — количество элементов таблицы эталонов. Таким образом, ускорение достигается за счет дополнительной информации о расположении элементов, а асимптотическая оценка алгоритма О(n) = log2n.

Метод основан на последовательном делении на 2 диапазона поиска. При этом на каждом шаге либо находится элемент, либо происходит переход в одну из половин диапазона. В процессе поиска выполняется не только сравнение на равенство, но и на больше - меньше. Последняя операция позволяет выбрать очередную половину диапазона таблицы. Если массив эталонов не упорядочен, то выбор будет сделан неверно, и результат можно не получить никогда (говорят, что алгоритм расходится).

Для корректности работы алгоритма необходимо упорядочить ключи (массив эталонов) по возрастанию. Для простоты их значения можно задать с помощью датчика случайных чисел. После этого необходимо выполнить операцию сортировки полученных величин

**Задание:** разработать алгоритм и программу дихотомического поиска. В качестве исходных данных использовать массив целых чисел, который формируется с помощью датчика случайных чисел с диапазоном от 0 до 100. Аргумент поиска – число.

#include <iostream>

#include <vector>

#include <ctime>

using namespace std;

int main()

{

srand(time(0));

setlocale(LC\_ALL, "RU");

int a;

cout << "Введите размер массива: ";

cin >> a;

vector<int> array(a);

for (size\_t i = 0; i < a; i++)

{

array[i] = (0 + rand() % 100);

}

for (size\_t i = 0; i < a; i++)

{

cout << array[i] << "\t";

}

cout << endl;

cout << "Сортировка" << endl;

for (size\_t i = 0; i < a; i++)

{

for (size\_t j = 0; j < a; j++)

{

if (array[i] < array[j])

{

swap(array[i], array[j]);

}

}

}

for (size\_t i = 0; i < a; i++)

{

cout << array[i] << "\t";

}

cout << endl;

unsigned int start\_time = clock();

int b,k;

bool otvet;

do

{

otvet = false;

cout << "Ввести аргумент поиска (целое число): ";

cin >> b;

int l = 0;

int r = a - 1;

if (b == array[a - 1])

{

otvet = true;

k = a - 1;

}

else

{

otvet = false;

do

{

k = ((l + r) / 2);

if (b == array[k])

{

otvet = true;

break;

}

else

{

if (b > array[k])

{

l = k;

}

else

{

r = k;

}

}

} while (otvet==false && l!=r-1);

}

if (otvet==true)

{

cout << "Найдено под номером: " << k<<endl;

}

else

{

cout << "Такого ключа в массиве нет, введите другой" << endl;

}

} while (otvet!=true);

cout << endl;

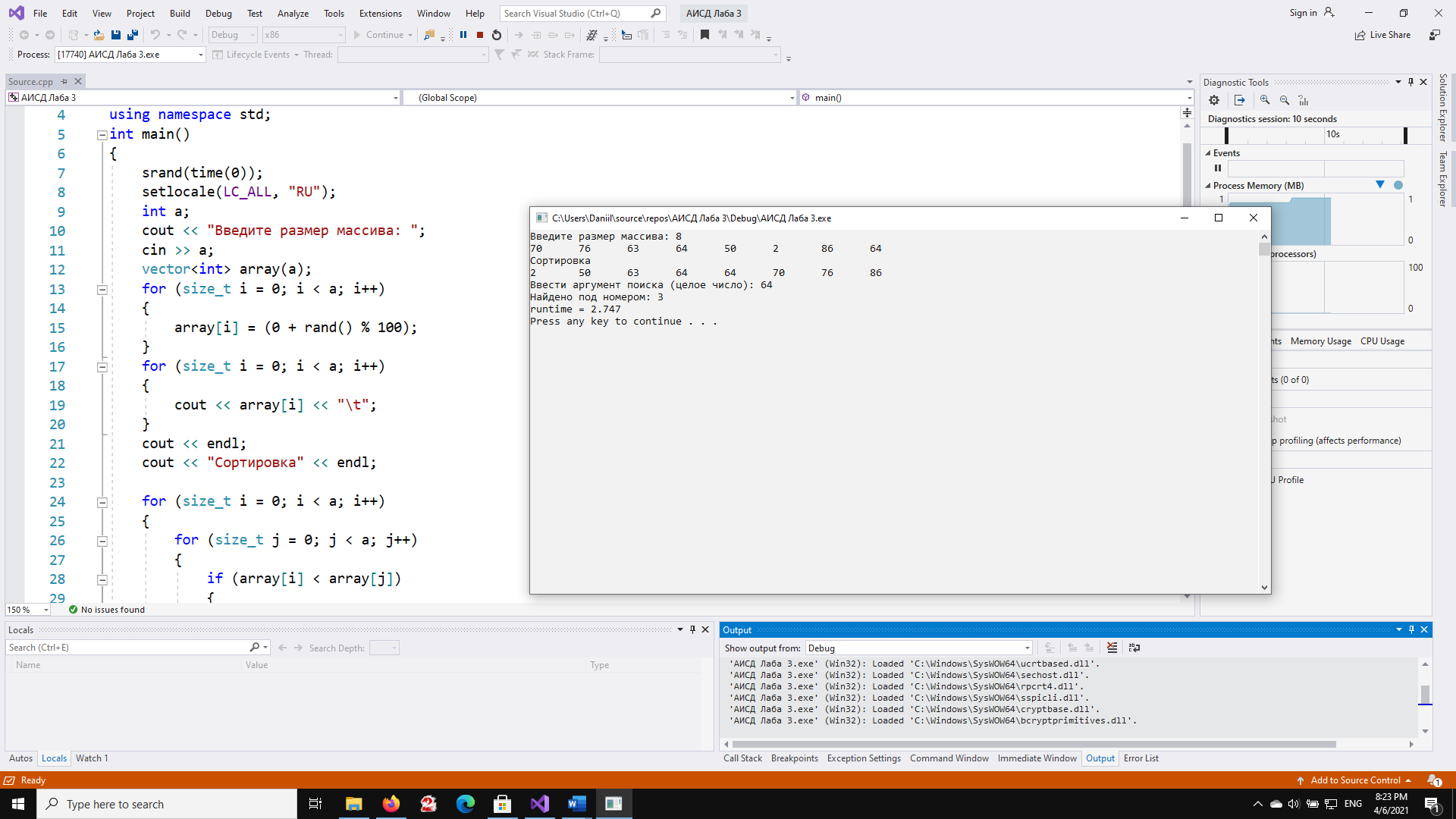
unsigned int end\_time = clock();

unsigned int search\_time = end\_time - start\_time;

cout << "runtime = " << search\_time / 1000.0 << endl;

system("pause");

}



Вывод: разработал программу, реализующую различные алгоритмы поиска, и оценил их временную и пространственную сложность.