**онлайн-Платформа STEPIK**

**отчет**

**по практической работе №2**

**по курсу «Основы алгоритмизации и программирования на языке C++»**

**Тема: Одномерные статические массивы**

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. | Простаков А.В (User ID: 89907427) |  |
| Преподаватель | Глущенко А.Г. |  |

Москва

2023

**Цель работы.**

Отработка практических навыков работы с одномерными статическими массивами данных

**Постановка задачи**

Разработать алгоритм и написать программу, которая позволяет:

1) Создать целочисленный массив размерности N=100. Элементы массива должны принимать случайное значение в диапазоне от -99 до 99.

2)    Отсортировать заданный в пункте 1 массив […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.

3)    Найти максимальный и минимальный элемент массива. Подсчитайте время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.

4)    Выводит среднее значение (если необходимо, число нужно округлить) максимального и минимального значения в отсортированном и неотсортированном. Выводит индексы всех элементов, которые равны этому значению, и их количество. Подсчитайте время поиска.

5)    Выводит количество элементов в отсортированном массиве, которые меньше числа *a*, которое инициализируется пользователем.

6)    Выводит количество элементов в отсортированном массиве, которые больше числа *b*, которое инициализируется пользователем.

7)    Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором. (\*)

8)     Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.

**Выполнение работы.**

Код программы в редактируемом формате представлен в приложении А.

|  |
| --- |
| Блок описания кода и использованных алгоритмов. Скриншоты работы программы |
| Скриншот из VisualStudio |
|  |

|  |
| --- |
| Вид консоли после окончания работы программы |
|  |

**Выводы**

В данной практической работе рассмотрены приемы работы с элементами одномерного статического массива.

Так как я выполнял данную практическую работу уже после прохождения блока «Функции и структура программы», то я также активно использовал возможности работы с функциями для выполнения заданий.

**Приложение А**

**рабочий код**

|  |
| --- |
| *1)    Создает целочисленный массив размерности*N*= 100. Элементы массивы должны принимать случайное значение в диапазоне от -99 до 99.* |
| int main()  {  srand(time(NULL));  setlocale(LC\_ALL, "ru");  const int N = 100;  int mass[N];  int min;  int max;  int average;  int k = 0;  int a, b;  int value, result;  int n1, n2;    cout << "Неотсортированный массив: " << endl;  for (int i = 0; i < N; i++)  {    mass[i] = GetRand(-99, 99);  cout << mass[i] << endl;  } |
| *2)    Отсортировать заданный в пункте 1 массив […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.* |
| void Sort(int \*mass, int N)  {  system\_clock::time\_point start = system\_clock::now();  int temp;  temp = mass[0];  temp = 0;  for (int i = 0; i < N; i++)  {  for (int k = i + 1; k < N; k++)  {  if (mass[k] < mass[i])  {  temp = mass[k];  mass[k] = mass[i];  mass[i] = temp;  }  }  }  cout << "\n\n";    system\_clock::time\_point end = system\_clock::now();  duration<double> sec = end - start;  cout << "Время, затраченное на сортировку массива: " << sec.count() << " сек." << endl;  cout << "Отсортированный массив: " << endl;  for (int i = 0; i < N; i++)  {  cout << mass[i] << endl;  }  } |

|  |
| --- |
| *3) Найти максимальный и минимальный элемент массива. Подсчитайте время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.* |
| system\_clock::time\_point start = system\_clock::now();  min = mass[0];  max = mass[0];  for (int i = 1; i < N; i++)  {  if (mass[i] < min)  min = mass[i];  if (mass[i] > max)  max = mass[i];  }  cout << "\n\n";  system\_clock::time\_point end = system\_clock::now();  duration<double> sec = end - start;  cout << "Время, затраченное на поиск максимального/минимального элемента в неотсортированном массиве: " << sec.count() << " сек." << endl;  cout << "Минимальное значение =" << min << endl;  cout << "Максимальное значение = "<< max << endl;  average = 0.5 \* (max + min);  cout << "Среднее значение элементов в массиве: " << average << endl;    cout << "\n\n";  cout << "Номера элементов массива, равные среднему значению: " << endl;  system\_clock::time\_point start1 = system\_clock::now();  for (int i = 0; i < N; i++)  {  if (mass[i] == average)  {  cout << i << endl;  k = k + 1;  }  }  system\_clock::time\_point end1 = system\_clock::now();  duration<double> sec1 = end1 - start1;  cout << "Количество элементов в неотсортированном массиве, равное среднему: " << k << endl;  cout << "Время, затраченное на поиск элементов, равных среднему: " << sec1.count() << " сек." << endl; |
| int Minmax(int \* mass, int N)  {  int min;  int max;  int average;  int k = 0;  system\_clock::time\_point start = system\_clock::now();  min = mass[0];  max = mass[0];  for (int i = 1; i < N; i++)  {  if (mass[i] < min)  min = mass[i];  if (mass[i] > max)  max = mass[i];    }  cout << "\n\n";  system\_clock::time\_point end = system\_clock::now();  duration<double> sec = end - start;  cout << "Время, затраченное на поиск максимального/минимального элемента в отсортированном массиве: " << sec.count() << " сек." << endl;  cout << "Минимальное значение =" << min << endl;  cout << "Максимальное значение = " << max << endl;  average = 0.5 \* (max + min);  cout << "Среднее значение элементов в массиве: " << average << endl;  cout << "\n\n";  cout << "Номера элементов массива, равные среднему значению: " << endl;  system\_clock::time\_point start1 = system\_clock::now();  for (int i = 0; i < N; i++)  {  if (mass[i] == average)  {  cout << i << endl;  k = k + 1;  }  }  system\_clock::time\_point end1 = system\_clock::now();  duration<double> sec1 = end1 - start1;  cout << "Количество элементов в неотсортированном массиве, равное среднему: " << k << endl;  cout << "Время, затраченное на поиск элементов, равных среднему в отсортированном массиве: " << sec1.count() << " сек." << endl;  return 0;  } |
| *5) Вывести количество элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем.* |
| void Mina(int\* mass, int N, int a)  {  int k = 0;  for (int i = 0; i < N; i++)  {  if (mass[i] < a)  k = k + 1;  }  cout << "Количество элементов, значение которых меньше а: " << k << endl; |
| *6)    Выводит количество элементов в отсортированном массиве, которые больше числа b, которое инициализируется пользователем.* |
| void Maxb(int\* mass, int N, int b)  {  int k = 0;  for (int i = 0; i < N; i++)  {  if (mass[i] > b)  k = k + 1;  }  cout << "Количество элементов, значение которых больше b: " << k << endl; |
| *7) Вывести информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором. (\*)* |
| int binarySearch(int \*arr, int value, int start, int end)  {    if (end >= start) {  int mid = start + (end - start) / 2;  if (arr[mid] == value) {  return mid;  }  if (arr[mid] > value) {  return binarySearch(arr, value, start, mid - 1);  }  return binarySearch(arr, value, mid + 1, end);  }  return -1;    } |
| *8) Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.* |
| void Swap(int \* mass, int n1, int n2)  {  system\_clock::time\_point start = system\_clock::now();  int temp;  temp = mass[n2];  mass[n2] = mass[n1];  mass[n1] = temp;  system\_clock::time\_point end = system\_clock::now();  duration<double> sec = end - start;  cout << "\n\n";  cout << "Время, затраченное на изменения положения элементов массива: " << sec.count() << " сек." << endl;  } |