Документация бригады №4.

Тема №2: Поиск элемента в одномерном массиве.

1. Описание входных и выходных данных:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| класс | имя | описание | тип | структура |
| Входные данные | Massiv | Заданный массив | double | Одномерный массив |
| Var | Значение элемента для поиска | double | Простая переменная |
| Выходные данные | Massiv[i] | Значение элемента, которое нашли(равно заданному) | double | Простая переменная |
| i | Индекс элемента, который нашли, выводим (i+1) | int | Простая переменная |

1. Пример работы:

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Массив из 5 элементов:  2  3.45  1  0.23  4  Значение для поиска:  3.45 | Найдено значение 3.45 под номером 2 |

1. Краткое описание логики работы:

Дан входной одномерный массив из n элементов. n-константа. Предположим, что наш процессор способен выполнять как единые инструкции следующие операции:

Присваивать значение переменной.  
Находить значение конкретного элемента в массиве  
Сравнивать два значения

Запомнить индекс  
Инкрементировать значение  
Основные арифметические операции (например, сложение и умножение).

Проходимся по всему массиву с помощью цикла for, который выполняет присвоение, сложение, сравнение.

for(int i = 0; i<n; i++)

Теперь с помощью if сравниваем элементы массива и Var(заданное число).

(Massiv[i] == Var)

При выполнении равенства выводим индекс + 1, так как массив записывался с нуля. Выводим значение и индекс.

cout « "Найдено значение " « Massiv[i] « " под номером " « i + 1 «

1. Аналитический анализ сложности:

Сложность алгоритмов обычно оценивают по времени выполнения или по используемой памяти. В обоих случаях сложность зависит от размеров входных данных: массив из 100 элементов будет обработан быстрее, чем аналогичный из 1000. При этом точное время мало кого интересует: оно зависит от процессора, типа данных, языка программирования и множества других параметров. Важна лишь асимптотическая сложность, т. е. сложность при стремлении размера входных данных к бесконечности.

Алгоритм поиска элемента по заданному значению обладает линейной сложностью. Сложность нашей программы не зависит от времени выполнения, так как для поиска элемента по заданному значению не нужно ни запоминать элементы, ни проходить по ним сколько-то раз. Аналогично проводят оценку и по памяти, когда это важно. Однако алгоритмы могут использовать значительно больше памяти при увеличении размера входных данных, чем другие, но зато работать быстрее. И наоборот. Это помогает выбирать оптимальные пути решения задач исходя из текущих условий и требований.

1. Экспериментальное исследование вычислительной сложности решения:

