

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Алтайский государственный технический университет  
им. И. И. Ползунова»

Факультет информационных  
технологий Кафедра прикладной  
математики

Отчет защищен с оценкой \_

Преподаватель \_  
(подпись)  
«\_\_\_» 2024 г.

Отчет  
По лабораторной работе №1  
« Внутренняя сортировка. Бинарный поиск.  
Библиотека стандартных шаблонов (STL) »  
по дисциплине «Типы и структуры данных»

Студент группы ПИ-21 Лихтинфельд А.А.

Преподаватель Потупчик А.И.

Барнаул 2024

## Вариант №13

Тип данных – char. Сортировка по возрастанию.

### Задание №1

Разработать и отладить функцию на языке C++, реализующую один из известных простых методов сортировки (сортировка выбором, сортировка вставками, обменная сортировка). Выполнить оценку временной и емкостной сложности программы.


#### Сортировка выбором:

```
#include <iostream>
```

```
void sortArray(char* a, int size)
{
    int k; char x;
    for (int i = 0; i < size; i++) // i - номер текущего шага
    {
        k = i; x = a[i];
        for (int j = i + 1; j < size; j++) { // цикл выбора наименьшего элемента
            if (a[j] < x) {
                k = j; x = a[j]; // k - индекс наименьшего элемента
            }
        }
        a[k] = a[i]; a[i] = x; // меняем местами наименьший с a[i]
    }
}
```

```
int main()
{
    char arr[] = { 'd', 'b', 'c', 'a', 'e' };
    int size = sizeof(arr) / sizeof(char);
    sortArray(arr, size);

    for (int i = 0; i < size; i++)
    {
        std::cout << arr[i] << " ";
    }
    return 0;
}
```

 Выбрать Консоль отладки Microsoft Visual Studio

```
a b c d e
C:\Users\Sweety\source\repos\TASD1\x64\Debug\TASD1.exe (процесс 11648) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:
```

### Оценка временной сложности:

Для нахождения наименьшего элемента из  $n+1$  рассматриваемых алгоритм выполняет  $n$  сравнений. С учетом того, что количество рассматриваемых на очередном шаге элементов уменьшается на единицу, общее количество операций:  $n + (n-1) + (n-2) + (n-3) + \dots + 1 = 1/2 * (n^2 + n) = O(n^2)$ . Так как число обменов всегда будет меньше числа сравнений, время сортировки растет квадратично относительно количества элементов.

### Оценка ёмкостной сложности:

Алгоритм не использует дополнительной памяти: все операции происходят "на месте".

### **Задание №2**

Разработать и отладить функцию на языке C++, реализующую один из известных улучшенных методов сортировки. Выполнить оценку временной и емкостной сложности программы.

### Быстрая сортировка Хоара.

```
#include <iostream>

void Quick_Sort(int left, int right, char* x) {
    int i, j, m, h;
    i = left;
    j = right;
    m = x[(i + j + 1) / 2];

    do {
        while (x[i] < m) i++; while (x[j] > m) j--; if (i <= j) {
            std::swap(x[i], x[j]);
            i++; j--;
        }
    } while (i <= j);
    if (left < j)
        Quick_Sort(left, j, x); if (i < right)
        Quick_Sort(i, right, x);
}

int main()
{
    char arr[] = { 'd', 'b', 'c', 'a', 'e', 'h', 'y', 'p', 'f', 'w', 'q', 'n' };
    int size = sizeof(arr) / sizeof(char);
    Quick_Sort(0, size-1, arr);
}
```

```
for (int i = 0; i < size; i++)
{
    std::cout << arr[i] << " ";
}

return 0;
```

```
}
```

Выбрать Консоль отладки Microsoft Visual Studio

```
a b c d e f h n p q w y
C:\Users\Sweety\source\repos\TASD1\x64\Debug\TASD1.exe (процесс 11276) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:
```

### Оценка временной сложности:

Временная сложность алгоритма в данном случае (при делении массива на две равные части) выражается через нее же по формуле:  $T(n) = n + T(1/2 * n) + T((1-1/2) * n)$ . Таким образом, когда мы вызываем сортировку массива из  $n$  элементов, тратится порядка  $n$  операций на создание отсортированных частей и на выполнения себя же 2 раза с параметрами  $1/2 * n$  и  $(1 - 1/2) * n$ .  $T(n) = n + 2 * T(n / 2) = n + 2 * (n / 2 + 2 * T(n / 4)) = n + n + 4 * T(n / 4) = n + n + 4 * (n / 4 + 2 * T(n / 8)) = n + n + n + 8 * T(n / 8) = \dots$ . Итого будет  $\log(n)$  слагаемых, потому как слагаемые появляются до тех пор, пока аргумент не уменьшится до 1. В результате  $T(n) = O(n * \log(n))$

### Оценка ёмкостной сложности:

Алгоритм не использует дополнительной памяти: все операции происходят "на месте".

### **Задание №3**

Разработать и отладить функцию на языке C++, реализующую бинарный поиск. Выполнить оценку временной и емкостной сложности программы.

### Бинарный поиск:

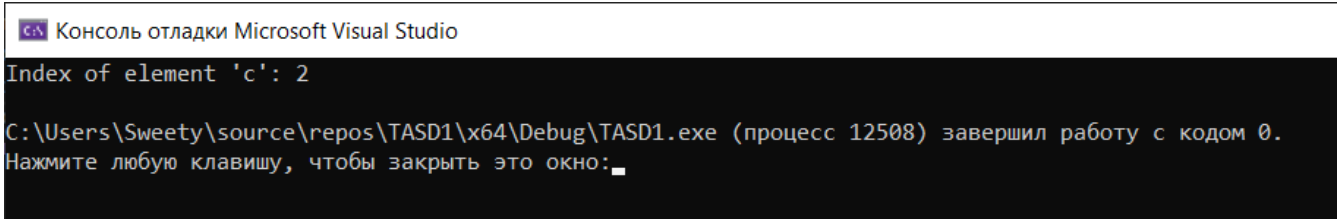
```
#include <iostream>
using namespace std;
int binarySearch(char arr[], int left, int right, char x) {
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x) {
            return mid;
        }
        if (arr[mid] < x) {
            left = mid + 1;
        }
        else {
            right = mid - 1;
        }
    }
}
```

```

    }
    return -1;
}

int main() {
    char arr[] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g' };
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << "И Index of element 'c': " << binarySearch(arr, 0, n - 1, 'c') << endl;
    return 0;
}

```



Консоль отладки Microsoft Visual Studio

```

Index of element 'c': 2

C:\Users\Sweet\source\repos\TASD1\x64\Debug\TASD1.exe (процесс 12508) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:

```

#### Оценка временной сложности:

Временная сложность бинарного поиска составляет  $O(\log n)$ , так как на каждой итерации размер пространства поиска уменьшается вдвое.

#### Оценка ёмкостной сложности:

Алгоритм не использует дополнительной памяти: все операции происходят "на месте".

#### **Задание №4**

Разработать и отладить программу на языке C++, реализующую сортировку и бинарный поиск, используя библиотеку стандартных шаблонов (STL).

#### Код программы и пример работы:

```

#include <iostream>
#include <algorithm>
#include <locale>

int main() {
    setlocale(LC_ALL, "Russian");
    char arr[] = { 'a', 'g', 'c', 'd', 'e', 'f', 'b' };
    int n = sizeof(arr) / sizeof(arr[0]);

    // Сортировка массива
    std::sort(arr, arr + n);

    // Вывод отсортированного массива
    std::cout << "Отсортированный массив: ";

```


```

for (int i = 0; i < n; i++) {
    std::cout << arr[i] << " ";
}
std::cout << std::endl;

// Бинарный поиск элемента
char key = 'c';
if (std::binary_search(arr, arr + n, key)) {
    std::cout << "Элемент " << key << " найден в массиве." << std::endl;
}
else {
    std::cout << "Элемент " << key << " не найден в массиве." << std::endl;
}

return 0;
}

```


 Выбрать Консоль отладки Microsoft Visual Studio

Отсортированный массив: a b c d e f g  
 Элемент c найден в массиве.  
  
 C:\Users\Sweetey\source\repos\TASD1\x64\Debug\TASD1.exe (процесс 7672) завершил работу с кодом 0.  
 Нажмите любую клавишу, чтобы закрыть это окно:

#### Оценка временной сложности:

Временная сложность сортировки массива -  $O(n \cdot \log(n))$ , где  $n$  - количество элементов в массиве. Временная сложность бинарного поиска -  $O(\log(n))$ , где  $n$  - количество элементов в массиве.

Бинарный поиск выполняется после сортировки массива. Поэтому, общая временная сложность алгоритма для сортировки и последующего бинарного поиска будет  $O(n \cdot \log(n))$ , поскольку сначала мы выполняем сортировку с такой временной сложностью, а затем выполняем бинарный поиск.

#### Оценка ёмкостной сложности:

Алгоритм не использует дополнительной памяти: все операции происходят "на месте".