МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ | **Факультет прикладной математики и информатики**

Кафедра прикладной математики

Практическое задание № 5

по дисциплине «Структуры данных и алгоритмы»

АЛГОРИТМЫ СОРТИРОВКИ

Бригада 2	ТАДЖИБАЕВ ЗАВКИДДИН
Группа ПМ-25	ЯГОДКИН АЛЕКСНАДР
Вариант 7	СУХАРЕВА СОФЬЯ

Преподаватель ТРАКИМУС ЮРИЙ ВИКТОРОВИЧ

Новосибирск, 2023

1 Задание

Упорядочить таблицу, построенную в практическом задании «ТАБЛИЦЫ» (варианты 3б, 3в), по новому ключу – по возрастанию номеров команд, не вошедших в претенденты ни на первое, ни на последнее место.

Для упорядочения использовать метод:

а) быстрой сортировки;

б) сортировки с использованием структуры дерева;

в) метод Шелла.

2 Анализ программы

Входные данные: В файле "Table.txt" содержится хеш-таблица. Каждый элемент этой таблицы состоит из номера команды и количества её вхождений в первую, последнюю и топ-3 местах рейтинга.

Выходные данные: Упорядоченная таблица по возрастанию номеров команд, не вошедших в претенденты ни на первое, ни на последнее место, "Не удалось открыть файл Table.txt.", "Файл Table.txt пуст.".

Анализ работы программы: Данная программа предназначена для сортировки данных, которые считываются из файла "Table.txt". Пользователь может выбрать один из трех методов сортировки: Quick sort, Tree sort или Shell sort. Алгоритм Quick sort основан на разделении и сортировке массива с использованием опорного элемента. Рекурсивно происходит разбиение массива на подмассивы и их сортировка, пока весь массив не будет отсортирован. Алгоритм Tree sort использует структуру данных "бинарное дерево поиска". Считанные данные из файла вставляются в это дерево, а затем извлекаются в отсортированном порядке. Этот метод позволяет получить отсортированный результат без использования циклов. Алгоритм Shell sort основан на методе сортировки Шелла, который использует последовательность шагов с постепенным уменьшением. Происходит сравнение и обмен элементов в таблице. Длину шага вычисляем при помощи последовательности Седжвика так-как он выполняется быстрее. После выбора метода сортировки и выполнения соответствующего алгоритма, отсортированные данные выводятся на экран. Таким образом, программа предоставляет возможность пользователю сортировать данные из файла с помощью различных методов сортировки и получать отсортированный результат для дальнейшего использования.

3 Программа

```
#include <stdio.h>
#include <locale.h>
#include <windows.h>
#include <math.h>
#include <algorithm>

const UINT N = 1000;
UINT K = 0, L = 0, H = 0;

struct elem
{
   UINT Num, first, last, top;
   elem(UINT _Num = 0, UINT _first = 0, UINT _last = 0, UINT _top = 0) :
      Num(_Num), first(_first), last(_last), top(_top) { };
};
```

```
struct tree
{
   elem *elm;
   tree *l, *r;
   tree(elem *_elm = NULL, tree *_l = NULL, tree *_r = NULL) :
      elm(_elm), l(_l), r(_r) { };
} *t1 = NULL, *t2 = NULL;

tree *add(tree *t, elem *e)
{
   if (t)
      e->Num < t->elm->Num ? t->l = add(t->l, e) : t->r = add(t->r, e);
   else
      t = new tree(e);
   return t;
}

struct table
{
   elem *e[N]{ };
   UINT size = 0;

   void Q_sort(int l, int r) // Быстрая сортировка
   {
      if (l >= r) return;

      int i = l, j = r, m = (l + r) / 2;
      for ( ; i <= j;)
      {
         for ( ; e[i]->Num < e[m]->Num; i++); // Находим элемент меньше
опорного
         for ( ; e[j]->Num > e[m]->Num; j--); // Находим элемент больше
опорного

         if (i <= j) // Инвертируем элементы в правильном порядке
         {
            swap(e[i], e[j]);
            i++;
            j--;
         }
      }

      if (l < j) Q_sort(l, j); // Для левой половины таблицы
      if (i < r) Q_sort(i, r); // Для правой половины таблицы
   }


   void Tree_sort(tree *t) // Сортировка бинарным деревом
   {
      if (t)
      {
```

```
        Tree_sort(t->l); // Для левого поддерева
        e[L] = t->elm; L++;
        Tree_sort(t->r); // Для правого поддерева
    }
}

void Shell_sort(UINT l, UINT r)
{
    UINT d = 0, i = 1, j = 0;
    UINT p1 = 1, p2 = 1, p3 = 1;
    for ( ; d < (r - l + 1) / 3; i++,
        d = i % 2 ? 8 * p1 - 6 * p2 + 1 : 9 * p1 - 9 * p3 + 1, p1 *= 2,
        p2 = i % 2 ? 2 * p2 : p2,
        p3 = i % 2 ? 2 * p3 : p3) // Вычисляем шаг сортировки

        for (j = l + d; j <= r; j++) // Сортировка элементов
        {
            elem *tmp = e[j];
            for (UINT k = j; k >= l + d && e[k - d]->Num > tmp->Num;
                k -= d) // Сдвиг элементов на расстояние d
            {
                e[k] = e[k - d];
                e[k - d] = tmp;
            }
        }
}

void insert(elem *el)
{
    if (el->first || el->last)
    {
        e[size] = el;
        t2 = add(t2, el);
    }
    else
    {
        for (UINT i = size; i > H; i--)
            e[i] = e[i - 1];
        e[H] = el;
        t1 = add(t1, el);
        H++;
    }
    size++;
}

inline bool empty() { return size > 0; }

void print_table()
{
    printf_s("   №   First, %%    Last, %%    Top3, %%\n");
    for (UINT i = 0; i < size; i++)
```

```
            printf_s("%14.d %10.1f %10.1f %10.1f\n", e[i]->Num,
               e[i]->first * 100. / K, e[i]->last * 100. / K,
               e[i]->top * 100. / K);
      }

} T{ };

bool input()
{
   FILE *f = NULL;

   fopen_s(&f, "Table.txt", "r");
   if (f)
   {
      UINT Num = 0, first = 0, last = 0, top = 0;
      for (elem *e = NULL;
         fscanf_s(f, "%d %d %d %d ", &Num, &first, &last, &top) == 4;
         e = new elem(Num, first, last, top), T.insert(e), K++);

      fclose(f);
      return true;
   }
   else
      return false;
}

int main()
{
   setlocale(0, "");
   UINT cp = GetConsoleCP(), outcp = GetConsoleOutputCP();
   SetConsoleCP(1251);
   SetConsoleOutputCP(1251);

   if (input())
   {
      if (T.empty())
      {
         bool exitFlag = false;
         for ( ; !exitFlag; )
         {
            printf_s("Выберите способ сортировки:\n1 - Quick sort\n2 -
Tree_sort\n3 - Shell_sort\n");
            USHORT a = 0;
            L = 0;
            scanf_s("%hu", &a);

            switch (a)
            {
               case 1: T.Q_sort(0, H - 1);
                       T.Q_sort(H, T.size - 1); break;
               case 2: T.Tree_sort(t1);
```

```
                        T.Tree_sort(t2); break;
                    case 3: T.Shell_sort(0, H - 1);
                        T.Shell_sort(H, T.size - 1); break;
                    default:
                    printf_s("Некорректный вариант способа сортировки.\n");
                }
                if (a <= 3);

                printf_s("Хотите продолжить работу:\n1 – ДА\n0 - НЕТ\n");
                UCHAR c = 0;
                scanf_s("%hhu", &c);

                exitFlag = c == 0;
            }
        }
        else
            printf_s("Файл Table.txt пуст.");
    }
    else
        printf_s("Не удалось открыть файл Table.txt.");

    SetConsoleCP(cp);
    SetConsoleOutputCP(outcp);
    return 0;
}
```

4 Набор тестов

№	Входные данные	Назначение
1		Файл Table.txt отсутствует.
2		Файл Table.txt пуст.
3	11 2 0 1 3 0 0 1 7 1 1 1 5 2 3 0 8 0 0 1	Проверка работоспособности программы.
4	4 0 2 2 3 0 2 3 2 1 1 4 1 0 0 1 5 0 0 2	Проверка работоспособности программы.

5 Результаты работы программы

№	Ввод/Вывод программы
1	Не удалось открыть файл Table.txt.
2	Файл Table.txt. пуст.
3	Выберите способ сортировки: 1 - Quick sort 2 - Tree_sort 3 - Shell_sort 1 № First, % Last, % Top3, % 3 0.0 0.0 20.0 8 0.0 0.0 20.0 5 40.0 60.0 0.0 7 20.0 20.0 20.0 11 40.0 0.0 20.0 Хотите продолжить работу: 1 – ДА 0 – НЕТ 1 Выберите способ сортировки: 1 - Quick sort 2 - Tree_sort 3 - Shell_sort 3 № First, % Last, % Top3, % 3 0.0 0.0 20.0 8 0.0 0.0 20.0 5 40.0 60.0 0.0 7 20.0 20.0 20.0 11 40.0 0.0 20.0 Хотите продолжить работу: 1 – ДА 0 – НЕТ 1

	Выберите способ сортировки: 1 - Quick sort 2 - Tree_sort 3 - Shell_sort 2 № First, % Last, % Top3, % 3 0.0 0.0 20.0 8 0.0 0.0 20.0 5 40.0 60.0 0.0 7 20.0 20.0 20.0 11 40.0 0.0 20.0 Хотите продолжить работу: 1 – ДА 0 – НЕТ 0
4	Выберите способ сортировки: 1 - Quick sort 2 - Tree_sort 3 - Shell_sort 2 № First, % Last, % Top3, % 1 0.0 0.0 20.0 5 0.0 0.0 40.0 2 20.0 20.0 80.0 3 0.0 40.0 60.0 4 0.0 40.0 40.0 Хотите продолжить работу: 1 – ДА 0 – НЕТ 1 Выберите способ сортировки: 1 - Quick sort 2 - Tree_sort 3 - Shell_sort 3 № First, % Last, % Top3, % 1 0.0 0.0 20.0 5 0.0 0.0 40.0 2 20.0 20.0 80.0 3 0.0 40.0 60.0 4 0.0 40.0 40.0 Хотите продолжить работу: 1 – ДА 0 – НЕТ 1 Выберите способ сортировки: 1 - Quick sort 2 - Tree_sort 3 - Shell_sort 1 № First, % Last, % Top3, %

```
   1        0.0        0.0       20.0
   5        0.0        0.0       40.0
   2       20.0       20.0       80.0
   3        0.0       40.0       60.0
   4        0.0       40.0       40.0
Хотите продолжить работу:
1 – ДА
0 – НЕТ
0
```