МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

ОТЧЁТ
ЛАБОРАТОРНАЯ РАБОТА №11

Работа с файлами

Руководитель,
старший преподаватель _____ Помогалова А. В.
подпись, дата

Исполнитель,
группа ИКПИ-33 _____ Коньков М. Д.
подпись, дата

Санкт-Петербург 2024

УСЛОВИЯ РАБОТЫ

Вариант № 12:

Необходимо выполнить заданную обработку файла:

12	Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить сумму содержащихся в ней чисел, а затем выполнить сортировку строк файла в порядке убывания суммы. Результаты сортировки записать в новый файл.

Общая формулировка:

Необходимо написать программу для высчитывания суммы чисел в строках файла ‘input.txt’ (правила создания которого указаны в задании). После подсчёта суммы нужно отсортировать итоговый массив сумм в порядке убывания суммы в файл ‘output.txt’.

Переменные:

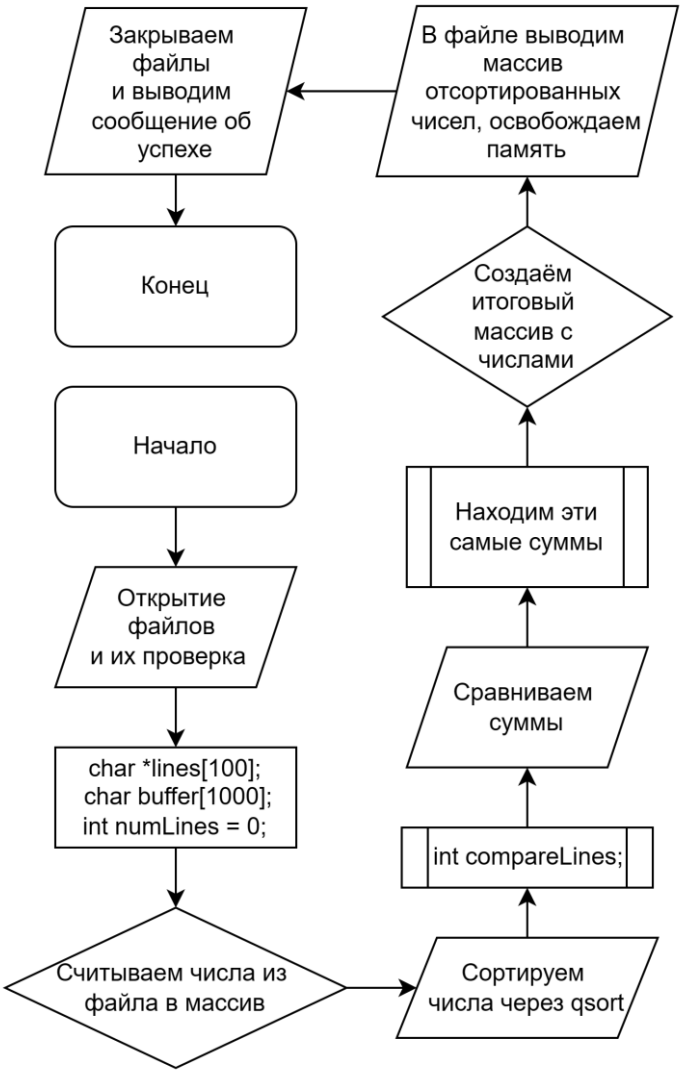
N	Обозначение в задаче	Идентификатор	Назначение
1	*lines*	*char *lines*	Массив указателей на строки
2	*buffer*	*char buffer*	Буфе для чтения строки из файла
3	*numLines*	*int numLines*	Число строк
4	*newLine*	*char *newline*	Строка без \0
5	*inputFile*	*FILE *inputFile*	Исходный файл
6	*outputFile*	*FILE *outputFile*	Итоговый файл
7	*i*	*int i*	Итератор цикла

1 Общий алгоритм решения

Процесс выполнения лабораторной работы можно разбить на **подзадачи**:

1. С помощью программы ‘**generator.c**’ генерируем файл ‘input.txt’, содержащий исходные строки с числами (необходимо учесть начальное условие!);
2. После того, как файл был создан, основная программа открывает оба файла (‘**input.txt**’ и ‘**output.txt**’) и делает проверку их открытия, где в случае провала программа останавливает выполнение;
3. Считываем строки из файла и сохраняем в массиве и закрываем файл ‘input.txt’ с исходными числами;
4. Сортируем строки в порядке убывания суммы, для чего используем встроенную функцию ‘**qsort’**, где, в качестве метода сортировки, пишем свою функцию ‘**compareLines**’;
5. В последней будет выполняться ещё одна вложенная функция ‘sumNumbersInLine’, которая будет разделять строки на отступы и считать сумму чисел в строке;
6. Записываем итоговые суммы в массив ‘output.txt’ и освобождаем память, выделенную для каждой строки;
7. Закрываем итоговый файл и выводим сообщение об успешности выполнения программы, числа в файле ‘output.txt’ отсортированы.

1.1 Общий алгоритм решения



1.2 Тестирование:

Для тестирования в программе ‘generator.c’ создадим файл ‘output.txt’ с параметрами: кол-во строк – 20, максимум чисел в строке – 10, диапазон чисел в строке – от -100 до 100.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ ФАЙЛ INPUT.TXT (TXT)

```
30.11 88.93 54.37 56.85 79.42 42.34 -0.15 73.94 55.47
-50.79 68.96 -39.76
25.88 63.15 55.50 56.90 -55.42 24.14 2.87 -21.26 -78.10 -40.72
-12.90 -67.39 -28.26 81.99 4.63
-89.11 5.47 -14.78 51.51 -52.01
-54.59 71.25 -91.45
5.48 -20.03 92.54 20.09 21.64
13.77
90.42 -43.19 -10.29
-93.38 86.27 12.86 -63.59 56.02 -74.55 35.04 44.72 24.08
95.89 -72.20 1.91 3.47
75.24 70.81 72.92 -39.45
46.48 77.09 -18.53
-35.67 37.58 -58.90 -61.25
-79.22 -22.96 55.94
29.75
-23.23 -15.02 41.72 17.53 -18.47 -73.61 -52.47 -81.94 1.46 -24.03
-74.65 47.77 12.53 72.89
70.98 -94.96 -50.00 57.49
50.94 -71.26
```

ПРИЛОЖЕНИЕ Б
ЛИСТИНГ ПРОГРАММЫ LAB11.C (C)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Функция для вычисления суммы чисел в строке
double sumNumbersInLine(const char *line) {
    double sum = 0;
    char *token;

    // Используем функцию strtod для преобразования строкового представления числа в
double
    token = strtok((char *)line, " ");
    while (token != NULL) {
        sum += strtod(token, NULL);
        token = strtok(NULL, " ");
    }

    return sum;
}

// Функция сравнения для сортировки строк по убыванию суммы чисел
int compareLines(const void *a, const void *b) {
    const char **line1 = (const char **)a;
    const char **line2 = (const char **)b;

    double sum1 = sumNumbersInLine(*line1);
    double sum2 = sumNumbersInLine(*line2);

    if (sum1 < sum2) return 1;
    if (sum1 > sum2) return -1;
    return 0;
}

int main() {
    FILE *inputFile = fopen("input.txt", "r");
    FILE *outputFile = fopen("output.txt", "w");

    if (inputFile == NULL || outputFile == NULL) {
```

```
        printf("Error opening files.\n");
        return 1;
    }

    char *lines[100];    // Массив указателей на строки
    char buffer[1000];   // Буфер для чтения строки из файла
    int numLines = 0;

    // Читаем строки из файла и сохраняем их в массиве
    while (fgets(buffer, sizeof(buffer), inputFile) != NULL && numLines < 100) {
        // Удаляем символ новой строки, если он присутствует
        char *newline = strchr(buffer, '\n');
        if (newline != NULL) *newline = '\0';

        lines[numLines] = strdup(buffer);    // strdup копирует строку и возвращает
указатель на новую копию
        numLines++;
    }

    fclose(inputFile);

    // Сортируем строки в порядке убывания суммы чисел (рекурсивно)
    qsort(lines, numLines, sizeof(char *), compareLines);

    // Записываем отсортированные строки в новый файл
    for (int i = 0; i < numLines; i++) {
        fprintf(outputFile, "%s\n", lines[i]);
        free(lines[i]);    // Освобождаем память, выделенную для каждой строки
    }

    fclose(outputFile);

    printf("Sorting completed.\n");

    return 0;
}
```

ПРИЛОЖЕНИЕ В

ЛИСТИНГ ФАЙЛА GENERATOR.C (C)

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Генерирует случайное действительное число в заданном диапазоне
double random_double(double min, double max) {
    return min + ((double)rand() / RAND_MAX) * (max - min);
}

// Генерирует файл с исходными данными
void generate_input_file(const char* filename, int num_lines, int
max_numbers_per_line, double min_value, double max_value) {
    FILE* file = fopen(filename, "w");
    if (file == NULL) {
        printf("Error while openning file.\n");
        return;
    }

    // Задаем seed для генерации случайных чисел
    srand(time(NULL));

    // Генерируем каждую строку файла
    for (int i = 0; i < num_lines; i++) {
        // Генерируем случайное количество чисел для каждой строки
        int num_numbers = rand() % max_numbers_per_line + 1;
        // Генерируем числа и записываем их в строку через пробел
        for (int j = 0; j < num_numbers; j++) {
            double number = random_double(min_value, max_value);
            fprintf(file, "%.2f ", number); // "%.2f" используется для вывода
чисел с двумя десятичными знаками
        }
        fprintf(file, "\n");
    }

    fclose(file);
}

int main() {
```

```
    const char* filename = "input.txt";
    int num_lines = 20;
    int max_numbers_per_line = 10;
    double min_value = -100.0;
    double max_value = 100.0;

    generate_input_file(filename, num_lines, max_numbers_per_line, min_value,
max_value);
    printf("Generated file '%s' with %d string of random numbers.\n", filename,
num_lines);

    return 0;
}
```

ПРИЛОЖЕНИЕ Г

РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ LAB11.C (C)

```
                                Вывод командной строки:
D:\University\Programming\NotGit\LabsC1.2\Lab11>Lab11.exe
Sorting completed.
D:\University\Programming\NotGit\LabsC1.2\Lab11>

                              Итоговый файл ‘output.txt’:
                                        75.24
                                        46.48
                                        30.11
                                        95.89
                                         5.48
                                        90.42
                                        70.98
                                        50.94
                                        29.75
                                        25.88
                                        13.77
                                       -12.90
                                       -23.23
                                       -35.67
                                       -50.79
                                       -54.59
                                       -74.65
                                       -79.22
                                       -89.11
                                       -93.38
```