МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

Направление подготовки: «Фундаментальная информатика и информационные технологии»

Программа бакалавриата: «Инженерия программного обеспечения»

Отчёт

на тему

«Реализация и сравнительный анализ алгоритмов сортировки строк: пузырьковая сортировка и сортировка вставками»

Выполнил:

студент группы 3825Б1ФИсп1

Плакунов Ф. П.

Оглавление

1. Введение

Сортировка данных является одной из фундаментальных задач в информатике и программировании. Эффективная организация данных позволяет ускорить последующий поиск, анализ и представление информации. В данной работе рассматриваются два классических алгоритма сортировки строк: **пузырьковая сортировка** и **сортировка вставками**. Оба метода реализованы на языке C с учётом возможности выбора порядка сортировки — по возрастанию (A–Z) или убыванию (Z–A). Работа направлена на сравнение производительности указанных алгоритмов при обработке массива строк, введённых пользователем.

2. Постановка задачи

Входные данные:

- Целое число k — количество строк (массивов) для сортировки (k > 0).
- Целое число a — максимальная длина одной строки (a > 0).
- k строк, введённых пользователем, каждая длиной не более a - 1 символов.
- Выбор метода сортировки: 1 — пузырьковая, 2 — вставками.
- Направление сортировки: 1 — по возрастанию (A–Z), 2 — по убыванию (Z–A).

Выходные данные:

- Отсортированный массив строк в соответствии с выбранным методом и направлением.
- Время выполнения сортировки в секундах с точностью до 6 знаков после запятой.

Требования к программе:

- Поддержка сортировки строк на русском языке (обеспечено через setlocale(LC_ALL, "Rus")).
- Динамическое выделение памяти под массив строк и их содержимое.
- Измерение времени выполнения выбранного алгоритма.
- Возможность многократного выбора метода сортировки без перезапуска программы.
- Корректная очистка выделенной памяти.

3. Программная реализация

Программа написана на языке C и включает следующие компоненты:

Основные структуры данных:

- char** data — массив указателей на строки; размер — k.
- Каждая строка выделяется динамически через calloc(a, sizeof(char)).

Константы и переменные:

- k — количество строк.
- a — максимальная длина строки (включая завершающий нуль).
- mode — выбор метода сортировки.
- alph — направление сортировки.
- j — коэффициент направления: 1 для A–Z, -1 для Z–A.
- start, end — метки времени до и после сортировки.
- replay — флаг повторного запуска выбора метода.

Функции:

1. **bubble(char **data, int k, int j)**
   Реализует пузырьковую сортировку. Сравнение строк осуществляется через strcmp. Направление определяется умножением результата strcmp на j. После сортировки выводит результат.
2. **insertion(char **data, int k, int j, int a)**
   Реализует сортировку вставками. Использует временный буфер key (выделенный через malloc) для хранения текущей строки. Для копирования строк применяется безопасная функция strncpy_s. Также выводит результат после сортировки.

Особенности реализации:

- Использование _s-версий функций (scanf_s, gets_s, strncpy_s) для повышения безопасности.
- Очистка входного буфера после ввода числа a с помощью цикла while ((c = getchar()) != '\n' && c != EOF).
- Измерение времени через clock() и расчёт в секундах: ((double)(end - start)) / CLOCKS_PER_SEC.

4. Результаты экспериментов

```
Введите количество массивов для сортировки.
4
Введите максимально возможный размер массива.
20
Введите массивы
banana
apple
cherry
date
Ваши массивы:
banana apple cherry date

Выберите метод сортировки: 1-пузырьковая 2-вставками
1
Выберите сортировку: 1)A-Z 2)Z-A
1
apple banana cherry date
Время выполнения: 0,001000 секунд
Хотите выбрать другой метод сортировки? 0-НЕТ 1-ДА
```

```
Введите количество массивов для сортировки.
4
Введите максимально возможный размер массива.
20
Введите массивы
banana
apple
cherry
date
Ваши массивы:
banana apple cherry date

Выберите метод сортировки: 1-пузырьковая 2-вставками
2
Выберите сортировку: 1)A-Z 2)Z-A
2
date cherry banana apple
Время выполнения: 0,001000 секунд
Хотите выбрать другой метод сортировки? 0-НЕТ 1-ДА
```

5. Заключение

Была реализована программа, выполняющая сортировку массива строк с возможностью выбора метода (пузырьковая или вставками) и направления (A–Z / Z–A). Входные данные задаются пользователем вручную. Программа корректно управляет памятью, измеряет время выполнения и обеспечивает повторный запуск сортировки без перезапуска. Задача успешно решена в соответствии с поставленными требованиями.

6. Приложение

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

void bubble(char **data, int k, int j) { //пузырькувая сортировка

for (int i = 0; i < k - 1; i++) {
for (int p = 0; p < k - i - 1; p++) {
if (strcmp(data[p], data[p + 1]) * j > 0) {
char* a = data[p];
data[p] = data[p + 1];
data[p + 1] = a;
}
}
}

for (int i = 0; i < k; i++) {
printf("%s ", data[i]);
}
printf("\n");
}


void insertion(char** data, int k, int j, int a) { //сортировка
вставками
char* key = malloc(a * sizeof(char));
for (int i = 1; i < k; i++) {
strncpy_s(key, a, data[i], a);
int c = i - 1;

while (c >= 0 && strcmp(data[c], key) * j > 0) {
strncpy_s(data[c + 1], a, data[c], a);
c--;
}
```

```
strncpy_s(data[c + 1], a, key, a);
}
for (int i = 0; i < k; i++)
printf("%s ", data[i]);
printf("\n");
free(key);
}



int main() {
setlocale(LC_ALL, "Rus");
int k, replay=1;
clock_t start, end;
do {
  //количество массивов с проверкой
printf("Введите количество массивов для сортировки.\n");
scanf_s("%d", &k);
if(k <= 0)
printf("Введите значение больше 0.\n");
} while (k <= 0);

char** data = malloc(k * sizeof(char*));
int a, mode, alph;

do {
  //максимальная возможная длинна массива с проверкой
printf("Введите максимально возможный размер массива.\n");
scanf_s("%d", &a);
if (a <= 0)
printf("Введите значение больше 0.\n");
} while (a <= 0);
a++;

int c;                                          // убирание \n
while ((c = getchar()) != '\n' && c != EOF);

printf("Введите массивы\n");
for (int i = 0; i < k; i++) {
data[i] = calloc(a, sizeof(char));
gets_s(data[i], a);
}

printf("Ваши массивы:\n");

for (int i = 0; i < k; i++) {
printf("%s ", data[i]);
}
printf("\n\n");
```

```
74. do {
75. printf("Выберите метод сортировки: 1-пузырьковая 2-вставками\n");
76. scanf_s("%d", &mode);
77. printf("Выберите сортировку: 1)A-Z 2)Z-A\n");
78. scanf_s("%d", &alph);
79. int j = (alph == 1) ? 1 : -1;

80. start = clock();
81. switch (mode) {
82. case 1: bubble(data, k, j); break;
83. case 2: insertion(data, k, j, a); break;
84. default: printf("Введите предложенный метод");
85. }
86. end = clock();
87. double s_t = ((double)(end - start)) / CLOCKS_PER_SEC;
88. printf("Время выполнения: %.6f секунд\nХотите выбрать другой
    метод сортировки? 0-НЕТ 1-ДА\n", s_t);
89. scanf_s("%d", &replay);
90. } while (replay != 0);


91. for (int i = 0; i < k; i++)
92. free(data[i]);
93. free(data);
94. return 0;
95. }
```