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: «Системное программирование»

Отчёт

на тему

«Разработка прототипа файлового менеджера с функцией сортировки файлов»

Выполнил: студент группы 3825Б1ФИсп1

Марков А. А.

Нижний Новгород

2025 год

Оглавление

1. Введение

1.1. Структуры данных

Структура — составной тип данных, объединяющий набор полей различных типов под одним именем. В языке C структуры используются для представления объектов предметной области, например, файлов в файловой системе. Доступ к полям структуры осуществляется через оператор → для указателей и для обычных переменных.

1.2. Алгоритмы сортировки

Сортировка — процесс упорядочивания элементов в соответствии с заданным критерием. В работе реализованы два классических алгоритма:

- Сортировка пузырьком — простой алгоритм, многократно проходящий по массиву и меняющий местами соседние элементы, если они расположены в неправильном порядке.
- Сортировка слиянием — эффективный алгоритм «разделяй и властвуй», который рекурсивно делит массив на две части, сортирует их, а затем объединяет в один отсортированный массив.

2. Постановка задачи

Задача: Разработать программу-прототип файлового менеджера, позволяющую пользователю работать с тремя директориями (наборами файлов), сортировать файлы внутри выбранной директории по размеру в порядке возрастания или убывания двумя методами: пузырьком и слиянием.

Для решения было выполнено следующее:

- Определена структура file для хранения имени и размера файла.
- Предопределены три массива структур, имитирующих директории с файлами.
- Реализована функция-компаратор compare_size для сравнения файлов по размеру.
- Реализованы алгоритмы сортировки пузырьком (bubble_sort) и слиянием (Mergesort).
- Реализовано консольное меню для выбора директории, порядка и метода сортировки.

- Реализованы функции вывода содержимого директории и копирования массива файлов.

3. Программная реализация

3.1 Структура данных программы

Структура file содержит два поля:

```
typedef struct{
      char name[20];
      int size;
} file;
```

name – строка для хранения имени файла (максимум 20 символов)

size – число типа int представляющее размер файла

Также в программе определены три статических массива структур, имитирущих директории:

```
file folder_1[20] = {
    {"Bread", 15}, {"Cheese", 21},
    {"Okak", 3}, {"Vot", 1},
    {"Document", 45}, {"Presentation", 67},
    {"Spreadsheet", 89}, {"Image", 12},
    {"Video", 156}, {"Audio", 34},
    {"Archive", 78}, {"Executable", 92},
    {"SourceCode", 54}, {"Database", 123},
    {"Backup", 200}, {"LogFile", 23},
        {"Config", 8}, {"Readme", 5},
        {"License", 19}, {"Script", 31}
    };

    file folder_2[20] = {
        {"Coffee", 25}, {"Sugar", 18},
        {"Milk", 32}, {"Eggs", 24},
        {"Report", 56}, {"Budget", 73},
        {"Invoice", 91}, {"Photo", 14},
        {"Movie", 210}, {"Podcast", 42},
        {"ZipFile", 65}, {"Installer", 88},
        {"Python", 47}, {"BackupDB", 145},
        {"Snapshot", 180}, {"ErrorLog", 29},
        {"Settings", 11}, {"Changelog", 7},
        {"Agreement", 22}, {"BatchFile", 38}
    };

    file folder_3[20] = {
        {"Butter", 12}, {"Yogurt", 26},
        {"Rice", 8}, {"Salt", 2},
        {"Manual", 51}, {"Schedule", 69},
        {"Database", 94}, {"Diagram", 16},
        {"Tutorial", 187}, {"Music", 53},
        {"RarArchive", 71}, {"AppImage", 96},
        {"JavaScript", 49}, {"LogArchive", 134},
```

```
        {"SystemBackup", 220}, {"AccessLog", 31},
        {"Profile", 9}, {"Instructions", 6},
        {"Warranty", 17}, {"AutoScript", 44}
    };
```

3.2 Функция сравнения

Функция compare_size сравнивает два файла по размеру в зависимости от выбранного порядка:

```
int compare_size(file *a, file *b, int* c){
    if (*c == 1){
        return (a->size < b->size ? 1 : 0);
    }
    else if (*c == 2){
        return (a->size > b->size ? 1 : 0);
    }
}
```

Глобальная переменная global_for_comparator хранит выбранный порядок (1 — возрастание, 2 — убывание).

3.3 Сортировка пузырьком

Алгоритм сортировки пузырьком реализован в функции bubble_sort:

```
void bubble_sort(file arr[], int sz, int (*comparator)(file*,
file*, int*)){
    for (int i = 0; i < sz; i++){
        for (int j = 0; j < sz; j++){
            if (comparator(&arr[i], &arr[j],
&global_for_comparator)){
                file tmp = arr[j];
                arr[j] = arr[i];
                arr[i] = tmp;
            }
        }
     }
    }
```

3.4 Сортировка слиянием

Алгоритм сортировки слиянием реализован рекурсивно в функциях Mergesort и Merge, функция Merge объединяет два отсортированных подмассива.

```
void Merge(file arr[], int st, int fin, file res[], int
(*comparator)(file*, file*, int*)){
    int i, j, k, mid = st + (fin - st)/2;
    i = st; j = mid+1;
    k = st;
    while (i<=mid && j <= fin){
        if (comparator(&arr[i], &arr[j], &global_for_comparator)){
            res[k++] = arr[i++];
        }
```

```
        else{
                res[k++] = arr[j++];
            }
        }
        while (i <= mid){
            res[k++] = arr[i++];
        }
        while (j <= fin){
            res[k++] = arr[j++];
        }
    }

    void Mergesort (file arr[], int st, int fin, file res[], int
(*comparator)(file*, file*, int*)){
        if (st == fin) return;
        int mid = st + (fin-st)/2;
        Mergesort(arr, st, mid, res, comparator);
        Mergesort(arr, mid+1, fin, res, comparator);
        Merge(arr, st, fin, res, comparator);
        for (int i = st; i <= fin; i++){
            arr[i] = res[i];
        }
    }
```

3.5 Функция main

Основная функция main организует взаимодействие с пользователем через консольное меню:

1. Инициализация локали и генератора случайных чисел.
2. Выбор текущей директории.
3. Цикл меню с вариантами:
    - Вывод исходного списка файлов.
    - Выбор порядка и метода сортировки.
    - Вывод отсортированного списка.
    - Смена директории.
    - Выход.

Для сохранения исходных данных используется временный массив temp_folder, в который копируется выбранная директория перед сортировкой.

```
int main() {
    setlocale(LC_ALL, "Russian");
    srand(time(NULL));

    int choice = -1, is_sorted = 0;
    int n = 20;

    file *folders[3] = {folder_1, folder_2, folder_3};

        printf("          \"Сортировщик файлов\"        \n\n");
```

```
11.         printf("Выберите директорию для работы(1, 2 или 3): ");
12.          scanf(" %d", &current_folder); current_folder--;
13.
14.         printf("\nДоступные функции\n1. Вывести исходный список
    файлов\n2. Выбрать метод и тип сортировки\n3. Вывести
    отсортированный список\n4. Сменить директорию\n0. Выход");
15.
16.         file temp_folder[20], merge_temp[20];
17.
18.         do {
19.             printf("\nВыберите операцию: ");
20.             scanf(" %d", &choice);
21.
22.             switch(choice) {
23.                 case 1:
24.                    printf("\nТекущая директория: %d",
    current_folder+1);
25.                     printf("\nИсходный список файлов:\n");
26.                     print_folder(folders[current_folder], n);
27.                     break;
28.                 case 2:
29.                    int sort_choice, order_choice;
30.                     copy_folder(folders[current_folder],
    temp_folder, &n);
31.
32.                     printf("\nВыберите порядок сортировки:\n");
33.                     printf("1. По возрастанию размера\n2. По
    убыванию размера\n");
34.                     do{
35.                         printf("Ваш выбор: ");
36.                         scanf(" %d", &order_choice);
37.                         if(order_choice > 2 || order_choice < 1){
38.                             printf("Неверный выбор!\n\n");
39.                             }
40.                             else printf("\n");
41.                         } while(order_choice > 2 || order_choice
    < 1);
42.
43.                     global_for_comparator = order_choice;
44.
45.                     printf("Выберите метод сортировки:\n");
46.                     printf("1. Сортировка пузырьком\n2. Сортировка
    слиянием\n");
47.                     do{
48.                         printf("Ваш выбор: ");
49.                         scanf(" %d", &sort_choice);
50.                         if(sort_choice > 2 || sort_choice < 1){
51.                             printf("Неверный выбор!\n");
52.                             }
53.                             else printf("\n");
54.                         } while(sort_choice > 2 || sort_choice <
    1);
55.
56.                     if (sort_choice == 1) {
57.                         bubble_sort(temp_folder, n, compare_size);
58.                     }
59.                     else if (sort_choice == 2) {
60.                         Mergesort(temp_folder, 0, n-1, merge_temp,
    compare_size);
61.                     }
62.                     printf("Сортировка завершена!\n");
63.                     is_sorted = 1;
```

```
64.                     break;
65.                 case 3:
66.                     if (is_sorted == 0){
67.                         printf("\nСписок не отсортирован, сначала
   выполните сортировку!\n");
68.                         break;
69.                         }
70.                     printf("\nТекущая директория: %d",
   current_folder+1);
71.                      printf("\nОтсортированный список файлов:\n");
72.                      print_folder(temp_folder, n);
73.                      break;
74.                 case 4:
75.                     printf("\nВыберите директорию для работы(1, 2
   или 3): ");
76.                         scanf(" %d", &current_folder);
   current_folder--;
77.                         is_sorted = 0;
78.                     break;
79.                 case 0:
80.                     printf("\nВыход из программы.");
81.                     break;
82.                 default:
83.                      printf("Неверный выбор!\n");
84.                      break;
85.             }
86.         } while (choice);
87.
88.         return 0;
89.     }
```

4. Результаты тестирования

Программа была протестирована вручную с использованием всех вариантов меню:

- Вывод исходных данных для каждой из трёх директорий корректно отображает имена и размеры файлов.

```
Выберите операцию: 1

Текущая директория: 1
Исходный список файлов:

Имя файла            Размер
--------------------------------
Bread                15
Cheese               21
Okak                 3
Vot                  1
Document             45
Presentation         67
Spreadsheet          89
Image                12
Video                156
Audio                34
Archive              78
Executable           92
SourceCode           54
Database             123
Backup               200
LogFile              23
Config               8
Readme               5
License              19
Script               31
--------------------------------
```

- Сортировка по возрастанию и убыванию размера обоими методами выполняется корректно. Файлы упорядочиваются в соответствии с выбранным порядком.

```
Выберите операцию: 3

Текущая директория: 1
Отсортированный список файлов:

Имя файла            Размер
--------------------------------
Vot                  1
Okak                 3
Readme               5
Config               8
Image                12
Bread                15
License              19
Cheese               21
LogFile              23
Script               31
Audio                34
Document             45
SourceCode           54
Presentation         67
Archive              78
Spreadsheet          89
Executable           92
Database             123
Video                156
Backup               200
--------------------------------
```

```
Выберите операцию: 2

Выберите порядок сортировки:
1. По возрастанию размера
2. По убыванию размера
Ваш выбор: 2

Выберите метод сортировки:
1. Сортировка пузырьком
2. Сортировка слиянием
Ваш выбор: 2

Сортировка завершена!

Выберите операцию: 3

Текущая директория: 1
Отсортированный список файлов:

Имя файла            Размер
--------------------------------
Backup               200
Video                156
Database             123
Executable           92
Spreadsheet          89
Archive              78
Presentation         67
SourceCode           54
Document             45
Audio                34
Script               31
LogFile              23
Cheese               21
License              19
Bread                15
Image                12
Config               8
Readme               5
Okak                 3
Vot                  1
--------------------------------
```

- Попытка вывода отсортированного списка до выполнения сортировки обрабатывается корректно: выводится предупреждение.

```
          "Сортировщик файлов"

Выберите директорию для работы(1, 2 или 3): 2

Доступные функции
1. Вывести исходный список файлов
2. Выбрать метод и тип сортировки
3. Вывести отсортированный список
4. Сменить директорию
0. Выход
Выберите операцию: 3

Список не отсортирован, сначала выполните сортировку!

Выберите операцию:
```

- Смена директории сбрасывает флаг сортировки (переменная is_sorted), что предотвращает вывод неактуальных данных.
- Все пункты меню обрабатываются, некорректный ввод отклоняется с сообщением.

```
Выберите операцию: 30
Неверный выбор!
```

5. Заключение

В работе был реализован прототип файлового менеджера с функциями сортировки файлов по размеру. Программа демонстрирует применение структур, указателей на функции, классических алгоритмов сортировки и организацию консольного интерфейса. Все поставленные задачи выполнены. Возможные направления улучшения: добавление сортировки по имени, поддержка динамического изменения количества файлов, реализация графического интерфейса.

6. Приложение А: листинг кода программы

```
1. #include<stdio.h>
2. #include<locale.h>
3. #include<string.h>
```

```c
#include<stdlib.h>
#include<math.h>
#include<time.h>

 typedef struct{
     char name[20];
          int size;
     } file;

     file folder_1[20] = {
         {"Bread", 15}, {"Cheese", 21},
         {"Okak", 3}, {"Vot", 1},
         {"Document", 45}, {"Presentation", 67},
         {"Spreadsheet", 89}, {"Image", 12},
         {"Video", 156}, {"Audio", 34},
         {"Archive", 78}, {"Executable", 92},
         {"SourceCode", 54}, {"Database", 123},
         {"Backup", 200}, {"LogFile", 23},
         {"Config", 8}, {"Readme", 5},
         {"License", 19}, {"Script", 31}
     };

     file folder_2[20] = {
         {"Coffee", 25}, {"Sugar", 18},
         {"Milk", 32}, {"Eggs", 24},
         {"Report", 56}, {"Budget", 73},
         {"Invoice", 91}, {"Photo", 14},
         {"Movie", 210}, {"Podcast", 42},
         {"ZipFile", 65}, {"Installer", 88},
         {"Python", 47}, {"BackupDB", 145},
         {"Snapshot", 180}, {"ErrorLog", 29},
         {"Settings", 11}, {"Changelog", 7},
         {"Agreement", 22}, {"BatchFile", 38}
     };

     file folder_3[20] = {
         {"Butter", 12}, {"Yogurt", 26},
         {"Rice", 8}, {"Salt", 2},
         {"Manual", 51}, {"Schedule", 69},
         {"Database", 94}, {"Diagram", 16},
         {"Tutorial", 187}, {"Music", 53},
         {"RarArchive", 71}, {"AppImage", 96},
         {"JavaScript", 49}, {"LogArchive", 134},
         {"SystemBackup", 220}, {"AccessLog", 31},
         {"Profile", 9}, {"Instructions", 6},
         {"Warranty", 17}, {"AutoScript", 44}
     };

```

```
52.     int global_for_comparator = 0;
53.     int current_folder;
54.
55.     int compare_size(file *a, file *b, int* c){
56.          if (*c == 1){
57.               return (a->size < b->size ? 1 : 0);
58.          }
59.          else if (*c == 2){
60.               return (a->size > b->size ? 1 : 0);
61.          }
62.     }
63.
64.     void bubble_sort(file arr[], int sz, int (*comparator)(file*,
   file*, int*)){
65.          for (int i = 0; i < sz; i++){
66.               for (int j = 0; j < sz; j++){
67.                    if        (comparator(&arr[i],        &arr[j],
   &global_for_comparator)){
68.                         file tmp = arr[j];
69.                         arr[j] = arr[i];
70.                         arr[i] = tmp;
71.                    }
72.               }
73.          }
74.     }
75.
76.     void  Merge(file  arr[],  int  st,  int  fin,  file  res[],  int
   (*comparator)(file*, file*, int*)){
77.          int i, j, k, mid = st + (fin - st)/2;
78.          i = st; j = mid+1;
79.          k = st;
80.          while (i<=mid && j <= fin){
81.              if            (comparator(&arr[i],            &arr[j],
   &global_for_comparator)){
82.                  res[k++] = arr[i++];
83.              }
84.              else{
85.                  res[k++] = arr[j++];
86.              }
87.          }
88.          while (i <= mid){
89.              res[k++] = arr[i++];
90.          }
91.          while (j <= fin){
92.              res[k++] = arr[j++];
93.          }
94.     }
95.
```

```
96.     void Mergesort (file arr[], int st, int fin, file res[], int
  (*comparator)(file*, file*, int*)){
97.         if (st == fin) return;
98.         int mid = st + (fin-st)/2;
99.         Mergesort(arr, st, mid, res, comparator);
100.        Mergesort(arr, mid+1, fin, res, comparator);
101.        Merge(arr, st, fin, res, comparator);
102.        for (int i = st; i <= fin; i++){
103.            arr[i] = res[i];
104.        }
105.    }
106.
107.    void print_folder(file arr[], int size) {
108.        printf("\n%-20s %s\n", "Имя файла", "Размер");
109.        printf("--------------------------------\n");
110.        for (int i = 0; i < size; i++) {
111.            printf("%-20s %d\n", arr[i].name, arr[i].size);
112.        }
113.        printf("--------------------------------\n");
114.    }
115.
116.    void copy_folder(file a[], file b[], int* size){
117.         for (int i = 0; i < *size; i++){
118.              b[i] = a[i];
119.         }
120.    }
121.
122.    int main() {
123.        setlocale(LC_ALL, "Russian");
124.        srand(time(NULL));
125.
126.        int choice = -1, is_sorted = 0;
127.        int n = 20;
128.
129.        file *folders[3] = {folder_1, folder_2, folder_3};
130.
131.        printf("          \"Сортировщик файлов\"        \n\n");
132.        printf("Выберите директорию для работы(1, 2 или 3): ");
133.         scanf(" %d", &current_folder); current_folder--;
134.
135.        printf("\nДоступные функции\n1. Вывести исходный список
  файлов\n2. Выбрать метод и тип сортировки\n3. Вывести отсортированный
  список\n4. Сменить директорию\n0. Выход");
136.
137.        file temp_folder[20], merge_temp[20];
138.
139.        do {
140.            printf("\nВыберите операцию: ");
```

```
141.            scanf(" %d", &choice);
142.
143.            switch(choice) {
144.                case 1:
145.                   printf("\nТекущая       директория:       %d",
  current_folder+1);
146.                    printf("\nИсходный список файлов:\n");
147.                    print_folder(folders[current_folder], n);
148.                    break;
149.                case 2:
150.                   int sort_choice, order_choice;
151.                    copy_folder(folders[current_folder],
  temp_folder, &n);
152.
153.                    printf("\nВыберите порядок сортировки:\n");
154.                    printf("1.  По  возрастанию  размера\n2.  По
  убыванию размера\n");
155.                    do{
156.                        printf("Ваш выбор: ");
157.                        scanf(" %d", &order_choice);
158.                        if(order_choice > 2 || order_choice < 1){
159.                            printf("Неверный выбор!\n\n");
160.                            }
161.                            else printf("\n");
162.                        } while(order_choice > 2 || order_choice <
  1);
163.
164.                    global_for_comparator = order_choice;
165.
166.                    printf("Выберите метод сортировки:\n");
167.                    printf("1. Сортировка пузырьком\n2. Сортировка
  слиянием\n");
168.                    do{
169.                        printf("Ваш выбор: ");
170.                        scanf(" %d", &sort_choice);
171.                        if(sort_choice > 2 || sort_choice < 1){
172.                            printf("Неверный выбор!\n");
173.                            }
174.                            else printf("\n");
175.                        } while(sort_choice > 2 || sort_choice <
  1);
176.
177.                    if (sort_choice == 1) {
178.                        bubble_sort(temp_folder, n, compare_size);
179.                    }
180.                    else if (sort_choice == 2) {
181.                        Mergesort(temp_folder, 0, n-1, merge_temp,
  compare_size);
```

```
                    }
                    printf("Сортировка завершена!\n");
                    is_sorted = 1;
                    break;
                case 3:
                    if (is_sorted == 0){
                        printf("\nСписок не отсортирован, сначала выполните сортировку!\n");
                        break;
                        }
                    printf("\nТекущая директория: %d", current_folder+1);
                    printf("\nОтсортированный список файлов:\n");
                    print_folder(temp_folder, n);
                    break;
                case 4:
                    printf("\nВыберите директорию для работы(1, 2 или 3): ");
                        scanf(" %d", &current_folder); current_folder--;
                        is_sorted = 0;
                    break;
                case 0:
                    printf("\nВыход из программы.");
                    break;
                default:
                    printf("Неверный выбор!\n");
                    break;
            }
        } while (choice);

        return 0;
    }
```