Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет (НИУ)»

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ОТЧЕТ

о выполнении практического задания №2

по дисциплине

«Структуры и алгоритмы обработки данных»

Вариант 5

Проверил:

ст. преподаватель кафедры СП

Петрова Л.Н.

Выполнил:

Студент группы КЭз-391

Галиулин Р.Р.

Челябинск
2025

Содержание

1 Описание задачи **3**

2 Листинги программ **4**

3 Контрольные тесты **10**

3.1 Задание №2: Строка 10

4 Контрольные вопросы **10**

4.1 Что такое ссылка? 10

4.2 Линейный (односвязный) список — что это? 11

4.3 Какие операции можно выполнять с односвязным списком? 11

4.4 Понятие стека. Операции, выполняемые над стеком. 12

4.5 Представление стека с помощью массива. Выполнение основных операций. . 12

1. Описание задачи

Задание №1: Односвязанный список

Информационная запись о файле содержит следующие поля: каталог, имя файла, расширение, дата и время создания, атрибуты «только для чтения», «скрытый», «системный», количество выделенных секторов (размер сектора принять равным 512 байтам). Поиск и сортировка — по каталогу, дате создания файла. Выяснить, поместится ли файл на носитель с некоторым количеством секторов.

Входные данные:

- Размер носителя - целое положительное число больше нуля
- Параметр поиска - буква латинского алфавита: **c** или **d**
- Каталог для поиска - строка
- Дата создания для поиска - строка в формате DD.MM.YYYY, где DD - день, MM -месяц, YYYY - год

Все данные вводятся с помощью стандартного потока вывода.

Выходные данные:

- Список файлов, содержащий сведения о файлах с данными указанными в задании. Выводится вначале, после сортировок и поиска. Формат буквенно-числовой.

Все данные выводятся с помощью стандартного потока вывода.

Задание №2: Стек

Сформировать стек из N чисел. Извлечь элементы из стека, найти их сумму и произведение. Результат поместить в стек.

Входные данные:

- Размер стека (кол-во элементов) - целое положительное число больше нуля
- Элементы стека - целые числа

Все данные вводятся с помощью стандартного потока вывода.

Выходные данные:

- Начальный стек, список элементов стека - целых чисел
- Сумма числе элементов стека, целое число
- Произведение числе элементов стека, целое число
- Конечный стек, содержащие элементы начального стека, также сумму и произведение в качестве элементов - целые числа

Все данные выводятся с помощью стандартного потока вывода.

2. Листинги программ

Язык программирования: C++ 14. Среда разработки: Ubuntu 24.10, gcc 14.2.0, nvim

При реализации задачи намерено, в учебных целях, не использованы контейнеры стандартной библиотеки **forward_list** и его методы

Листинг 1: Задание 1: Односвязанный список

```
// Галиулин РР.. КЭз-391
// Структуры и алгоритмы обработки данных
// Практическое занятие №2

// Информационная запись о файле содержит следующие поля: каталог, имя файла,
// расширение, дата и время создания, атрибуты «только для чтения», «скрытый»,
// «системный», количество выделенных секторов размер( сектора принять равным
// 512 байтам). Поиск и сортировка - по каталогу, дате создания файла.
// Выяснить, поместится ли файл на носитель с некоторым количеством секторов.

#include <iostream>
#include <string>
#include <sstream>
#include <iomanip>
#include <limits>

// Структура для хранения информации о файле
struct FileInfo {
    std::string directory; //Каталог
    std::string file_name; //Имя файла
    std::string extension; //Расширение
    std::string creation_date; //Дата создания
    std::string creation_time; //Время создания
    bool read_only; //Параметр - Только для чтения
    bool hidden; //Параметр - Скрытый
    bool system; //Параметр - Системный
    int sectors_allocated; //Размер занимамых секторов
    FileInfo* next_file; //Указать на следующий элемент

    FileInfo(const std::string& dir, const std::string& f_name, const std::
    string& ext,
             const std::string& c_date, const std::string& c_time, bool
    r_only,
             bool hid, bool sys, int s_alloc)
        : directory(dir), file_name(f_name), extension(ext),
          creation_date(c_date), creation_time(c_time), read_only(r_only),
          hidden(hid), system(sys), sectors_allocated(s_alloc), next_file(
    nullptr) {}
};

// Функция очистки памяти от списка
void ClearFiles(FileInfo*& head //указать на первый элемент
                ) {
    while (head) {
        FileInfo* next = head->next_file;
        delete head;
        head = next;
    }
}

// Добавление нового файла в список
// FileInfo*& head - указать на первый элемент
```

```
50 void AddFile(FileInfo*& head, const std::string& directory, const std::
       string& file_name,
51                const std::string& extension, const std::string& creation_date,
       const std::string& creation_time,
52                bool read_only, bool hidden, bool system, int sectors_allocated)
        {
53     FileInfo* new_file = new FileInfo(directory, file_name, extension,
54                                       creation_date, creation_time,
       read_only, hidden, system, sectors_allocated);
55     if (!head) {
56         head = new_file;
57     } else {
58         FileInfo* temp = head;
59         while (temp->next_file) {
60             temp = temp->next_file;
61         }
62         temp->next_file = new_file;
63     }
64 }
65
66 // Функция для сравнения дат
67 bool CompareDates(const std::string& date1, const std::string& date2) {
68     std::istringstream ss1(date1);
69     std::istringstream ss2(date2);
70     std::tm tm1 = {}, tm2 = {};
71
72     if (!(ss1 >> std::get_time(&tm1, "%d.%m.%Y"))) {
73         std::cerr << "Ошибка: Неверный формат даты: " << date1 << std::endl;
74         return false;
75     }
76     if (!(ss2 >> std::get_time(&tm2, "%d.%m.%Y"))) {
77         std::cerr << "Ошибка: Неверный формат даты: " << date2 << std::endl;
78         return false;
79     }
80
81     return std::mktime(&tm1) < std::mktime(&tm2);
82 }
83
84 // Функция сортировки
85 void SortFiles(FileInfo*& head, bool byDate) {
86     if (!head || !head->next_file) return;
87
88     FileInfo* sorted = nullptr;
89     FileInfo* current = head;
90
91     while (current != nullptr) {
92         FileInfo* next = current->next_file;
93         FileInfo** prev = &sorted;
94
95         while (*prev != nullptr &&
96                (byDate ? CompareDates((*prev)->creation_date, current->
       creation_date) : (*prev)->directory < current->directory)) {
97             prev = &(*prev)->next_file;
98         }
99
100        current->next_file = *prev;
101        *prev = current;
102        current = next;
103    }
104
```

```
105     head = sorted;
106 }
107 
108 // Функция поиска
109 void SearchFiles(FileInfo* head, const std::string& search_term, bool byDate
    ) {
110     FileInfo* temp = head;
111     bool found = false;
112     while (temp) {
113         if ((byDate ? temp->creation_date == search_term : temp->directory
    == search_term)) {
114             std::cout << "DIR: " << temp->directory
115                       << ", NAME: " << temp->file_name
116                       << ", EXT: " << temp->extension
117                       << ", C_DATE: " << temp->creation_date
118                       << ", C_TIME: " << temp->creation_time
119                       << ", R: " << (temp->read_only ? "1" : "0")
120                       << ", H: " << (temp->hidden ? "1" : "0")
121                       << ", S: " << (temp->system ? "1" : "0")
122                       << ", SIZE6(512): " << temp->sectors_allocated
123                       << "\n";
124             found = true;
125         }
126         temp = temp->next_file;
127     }
128     if (!found) {
129         std::cout << (byDate ? "Файлы, созданные " : "Файлы в каталоге \"") <<
    search_term << (byDate ? ", не найдены.\n" : "\" не найдены.\n");
130     }
131 }
132 
133 
134 // Функция для вывода списка файлов
135 void PrintFiles(FileInfo* head) {
136     FileInfo* temp = head;
137     while (temp) {
138         std::cout << "DIR: " << temp->directory
139                   << ", NAME: " << temp->file_name
140                   << ", EXT: " << temp->extension
141                   << ", C_DATE: " << temp->creation_date
142                   << ", C_TIME: " << temp->creation_time
143                   << ", R: " << (temp->read_only ? "1" : "0")
144                   << ", H: " << (temp->hidden ? "1" : "0")
145                   << ", S: " << (temp->system ? "1" : "0")
146                   << ", SIZE6(512): " << temp->sectors_allocated
147                   << "\n";
148         temp = temp->next_file;
149     }
150 }
151 
152 // Функция для проверки, поместится ли файл на носитель
153 bool CanFitOnDisk(int sectors_allocated, int available_sectors) {
154     return sectors_allocated <= available_sectors;
155 }
156 // Функция для вывода списка файлов и проверки, поместится ли файл на носитель
157 void PrintFilesAndCheckFit(FileInfo* head, int available_sectors) {
158     FileInfo* temp = head;
159     while (temp) {
160         std::cout << "Каталог: " << temp->directory
161                   << ", Имя файла: " << temp->file_name
```

```
162                   << ", Размер в секторах б(512): " << temp->sectors_allocated
163                   << ((temp->sectors_allocated <= available_sectors) ? " -
    Поместится на носитель\n" : " - Не поместится на носитель\n");
164         temp = temp->next_file;
165     }
166 }
167
168 // Функция для проверки корректности даты (DD.MM.YYYY)
169 bool IsValidDate(const std::string& date) {
170     if (date.length() != 10) return false;
171     if (date[2] != '.' || date[5] != '.') return false;
172     for (int i = 0; i < 10; ++i) {
173         if (i == 2 || i == 5) continue;
174         if (!isdigit(date[i])) return false;
175     }
176     return true;
177 }
178
179
180
181 int main() {
182     //Добаявляем файлы
183     FileInfo* files = nullptr; //Указатель на первый элемент
184
185     AddFile(files, "/home/user/docs", "file1", ".txt", "10.01.2023", "12:00"
    , true, false, false, 50);
186     AddFile(files, "/home/user/images", "image1", ".jpg", "10.01.2022", "
    14:00", false, false, false, 100);
187     AddFile(files, "/home/user/docs", "file2", ".docx", "09.01.2023", "10:00
    ", false, true, false, 150);
188     AddFile(files, "/home/user/music", "song1", ".mp3", "05.01.2022", "18:30
    ", false, false, true, 200);
189     AddFile(files, "/home/user/videos", "video1", ".mp4", "01.01.2023", "
    09:00", true, true, false, 300);
190
191     std::cout << "Файлы до сортировки:\n";
192     PrintFiles(files);
193
194     SortFiles(files, false);
195     std::cout << "Файлы после сортировки по каталогу:\n";
196     PrintFiles(files);
197
198     SortFiles(files, true);
199     std::cout << "Файлы после сортировки по дате создания:\n";
200     PrintFiles(files);
201
202     int available_sectors;
203     std::cout << "Введите размер носителя в секторах (512 байт): ";
204     while (true) {
205         std::cin >> available_sectors;
206         if (std::cin.fail() || available_sectors <= 0) {
207             std::cin.clear();
208             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n
    ');
209             std::cout << "Введите корректное целое положительное число больше нуля: "
    ;
210         } else {
211             break;
212         }
213     }
```

```
    // Проверяем поместится ли файл на носистель с заданным размером
    PrintFilesAndCheckFit(files, available_sectors);

    //Поиск
    char search_choice;
    std::cout << "По какому параметру искать файлы? (d - по каталогу, c - по дате создания): ";
    while (true) {
        std::cin >> search_choice;
        if (search_choice != 'd' && search_choice != 'c') {
            std::cout << "Введите 'd' для поиска по каталогу или 'c' для поиска по дате создания: ";
        } else {
            break;
        }
    }

    if (search_choice == 'd') { //Поиск по каталогу
        std::string search_directory;
        std::cout << "Введите каталог для поиска: ";
        std::cin.ignore();
        std::getline(std::cin, search_directory);
        SearchFiles(files, search_directory, false);
    } else {                    //Поиск по дате создания
        std::string search_date;
        std::cout << "Введите дату создания для поиска формат( DD.MM.YYYY): ";
        std::cin.ignore();
        std::getline(std::cin, search_date);
        if (IsValidDate(search_date)) {
            SearchFiles(files, search_date, true);
        } else {
            std::cerr << "Ошибка: Неверный формат даты. Используйте формат DD.MM.YYYY" << std::endl;
        }

    }

    ClearFiles(files);
    return 0;
}
```

Листинг 2: Задание 2: Стек

```
#include <iostream>
#include <stack>
#include <limits>

// Функция печати стека
void PrintStack(const std::stack<int>& stack) {
    std::stack<int> temp_stack = stack;
    while (!temp_stack.empty()) {
        std::cout << temp_stack.top() << " ";
        temp_stack.pop();
    }
    std::cout << std::endl;
}

int main() {
    int count, sum = 0, product = 1;
    std::stack<int> oper_stack; // Основной стек
    std::stack<int> temp_stack; // Стек для операции
```

```
19
20      std::cout << "Введите количество элементов: ";
21      std::cin >> count;
22
23      // Проверка корректности введенного количества элементов
24      while (count <= 0) {
25          std::cerr << "Ошибка: количество элементов должно быть положительным числом.
    Попробуйте еще раз: ";
26          std::cin.clear();
27          std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
28          std::cin >> count;
29      }
30
31      std::cout << "Введите элементы: ";
32      for (int i = 0; i < count; ++i) {
33          int element;
34          std::cin >> element;
35
36          // Проверка корректности введенного элемента
37          while (std::cin.fail()) {
38              std::cerr << "Ошибка: введено некорректное значение. Попробуйте еще раз:
    ";
39              std::cin.clear();
40              std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n
    ');
41              std::cin >> element;
42          }
43
44          oper_stack.push(element);
45      }
46
47      std::cout << "Начальный стек: ";
48      PrintStack(oper_stack);
49
50      // Извлечение элементов из стека во временный стек,
51      // нахождение их суммы и произведения
52      while (!oper_stack.empty()) {
53          int top_element = oper_stack.top();
54          sum += top_element;
55          product *= top_element;
56          temp_stack.push(top_element);
57          oper_stack.pop();
58      }
59
60      // Возвращаем первоначальные элементы обратно в основной стек
61      while (!temp_stack.empty()) {
62          oper_stack.push(temp_stack.top());
63          temp_stack.pop();
64      }
65
66      // Помещаем результат в стек
67      oper_stack.push(sum);
68      oper_stack.push(product);
69
70      std::cout << "Сумма: " << sum << std::endl;
71      std::cout << "Произведение: " << product << std::endl;
72
73      std::cout << "Конечный стек: ";
74      PrintStack(oper_stack);
75
```

```
76      return 0;
77  }
```

3. Контрольные тесты

Задание №1: Односвязанный список

Пример работы

Файлы до сортировки: DIR: /home/user/docs, NAME: file1, EXT: .txt, C_DATE: 10.01.2023, C_TIME: 12:00, R: 1, H: 0, S: 0, SIZE(512б): 50
...
DIR: /home/user/videos, NAME: video1, EXT: .mp4, C_DATE: 01.01.2023, C_TIME: 09:00, R: 1, H: 1, S: 0, SIZE(512б): 300
Файлы после сортировки по каталогу:
DIR: /home/user/docs, NAME: file2, EXT: .docx, C_DATE: 09.01.2023, C_TIME: 10:00, R: 0, H: 1, S: 0, SIZE(512б): 150
...
DIR: /home/user/videos, NAME: video1, EXT: .mp4, C_DATE: 01.01.2023, C_TIME: 09:00, R: 1, H: 1, S: 0, SIZE(512б): 300
Файлы после сортировки по дате создания:
DIR: /home/user/music, NAME: song1, EXT: .mp3, C_DATE: 05.01.2022, C_TIME: 18:30, R: 0, H: 0, S: 1, SIZE(512б): 200
...
DIR: /home/user/docs, NAME: file1, EXT: .txt, C_DATE: 10.01.2023, C_TIME: 12:00, R: 1, H: 0, S: 0, SIZE(512б): 50
Введите размер носителя в секторах (512 байт): **150**
Каталог: /home/user/music, Имя файла: song1, Размер в секторах (512б): 200 - Не поместится на носитель
...
Каталог: /home/user/docs, Имя файла: file1, Размер в секторах (512б): 50 - Поместится на носитель
По какому параметру искать файлы? (d - по каталогу, c - по дате создания): **c**
Введите дату создания для поиска (формат DD.MM.YYYY): **09.01.2023**
DIR: /home/user/docs, NAME: file2, EXT: .docx, C_DATE: 09.01.2023, C_TIME: 10:00, R: 0, H: 1, S: 0, SIZE(512б): 150

3.1 Задание №2: Строка

4. Контрольные вопросы

4.1 Что такое ссылка?

Ссылка — это особый тип переменной, которая, аналогично указателю, указывает на область памяти. Однако, в отличие от указателя, ссылка должна быть инициализирована в месте объявления, не может быть переназначена на другой объект и не может ссылаться на неопределённую или несуществующую область памяти. Безопаснее чем указатель и позволяет также эффективно передавать переменные или структуры данных например в вызов функции.

Исходные данные	Результат
5 1 2 3 4 5	Начальный стек: 5 4 3 2 1 Сумма: 15 Произведение: 120 Конечный стек: 120 15 5 4 3 2 1
3 2 3 4	Начальный стек: 4 3 2 Сумма: 9 Произведение: 24 Конечный стек: 24 9 5 4 3 2 1

Таблица 1: Таблица с результатами контрольных тестов Задания №2

4.2 Линейный (односвязный) список — что это?

Односвязный список — это динамическая структура данных, представляющая собой последовательность, каждый элемент которой содержит данные и ссылку (или указатель) на следующий элемент (кроме последнего, который указывает на nullptr). Такая структура позволяет эффективно добавлять или удалять элементы, но имеет низкую эффективность при частом доступе к произвольным элементам, так как для этого требуется последовательный перебор от начала списка.

4.3 Какие операции можно выполнять с односвязным списком?

С односвязным списком возможно выполнять следующие операции:

- **Добавление элементов**
  - В начало или конец последовательности (без изменения указателей других элементов).
  - Вставка в середину списка требует изменения указателя предыдущего элемента.
- **Удаление элементов**
  - Удаление первого элемента выполняется путём обновления головы списка.
  - Удаление последнего элемента требует обхода списка до предпоследнего узла.
  - Удаление элемента из середины требует изменения указателя предыдущего узла.
- **Обращение всего списка**
  - Меняется направление указателей в каждом узле так, чтобы каждый элемент указывал на предыдущий, а первый элемент становился последним.
- **Обход списка**
  - Проход по всем узлам для выполнения операций, таких как поиск, изменение или обработка элементов.

4.4 Понятие стека. Операции, выполняемые над стеком.

Стек это структура данных реализующая прицип LIFO (последним пришел, первым вышел).

Основными операциями являются:

- **Push** - добавление элемента "сверху"
- **Pop** - извлечение элемента "сверху"

Также существуют, но не во всех реализациях, дополнительные операции:

- **Top** (или Peek) - просмотр элемента без извлечения
- Проверка на пустоту, наличия хотя-бы одного элемента.
- **Size** - размер стека

4.5 Представление стека с помощью массива. Выполнение основных операций.

Логичнее всего реализовать на основе динамического массива и работать с нём только через функции реализующие на ним операции указанные в пункте выше.

```
1    class Stack {
2      private:
3      int* array;
4      int top;
5      int capacity;
6
7      void resize() {
8        capacity *= 2;
9        int* new_array = new int[capacity];
10       for (int i = 0; i < top; ++i) {
11         new_array[i] = array[i];
12       }
13       delete[] arr;
14       arr = new_array;
15     }
16
17     public:
18     Stack() {
19       capacity = 10;
20       array = new int[capacity];
21       top = 0;
22     }
23
24     ~Stack() {
25       delete[] array;
26     }
27
28
29     void push(int value) {
30       if (top == capacity) {
31         resize();
```

```
32          }
33          array[top++] = value;
34      }
35
36      int pop() {
37          if (top == 0) {
38              std::cout << "Стек пуст!" << std::endl;
39              return -1;
40          }
41          return array[--top];
42      }
43
44      int peek() {
45          if (top == 0) {
46              std::cout << "Стек пуст!" << std::endl;
47              return -1;
48          }
49          return array[top - 1];
50      }
51
52
53      bool isEmpty() {
54          return top == 0;
55      }
56
57      int size() {
58          return top;
59      }
60  };
61
62
```