Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Тульский государственный университет»
Институт прикладной информатики и компьютерных наук
Кафедра «Информационная безопасность»

ОТЧЕТ

Вид практики	Ознакомительная практика
Курс	1 курс
Направление подготовки / Специальность	Прикладная информатика в промышленности
ФИО обучающегося	Артемов Александр Евгеньевич
Место прохождения практики	Самостоятельно
Период прохождения практики	23.10.2023 — 26.10.2023 г.г.

Тула 2023

Содержание

Введение

Номер зачетки — 220085, номер индивидуального задания 13 — (36 + 36 + 13 = 85), номер темы реферата 36 — (49 + 36 = 85).

Задания первой части практики выполнены на языке программирования C++17 (стандарт C++ 2017 года) при помощи среды разработки Visual Studio Code, компилятор GCC 11.4.0. Исходные коды программ расположены соответственно в папках «задание_1», «задание_2» и «задание_3».

1. Задание 1

1.1 Описание задачи

Поступает последовательность целых положительных чисел, 0 – конец последовательности. Найти среднее арифметическое простых чисел в этой последовательности (с использованием процедуры, с использованием функции). Определение простого числа оформить в виде функций.

Функцию определения простого числа будем использовать одну и ту же в обоих случаях.

1.2 Схема алгоритма решения задачи (с использованием процедуры)

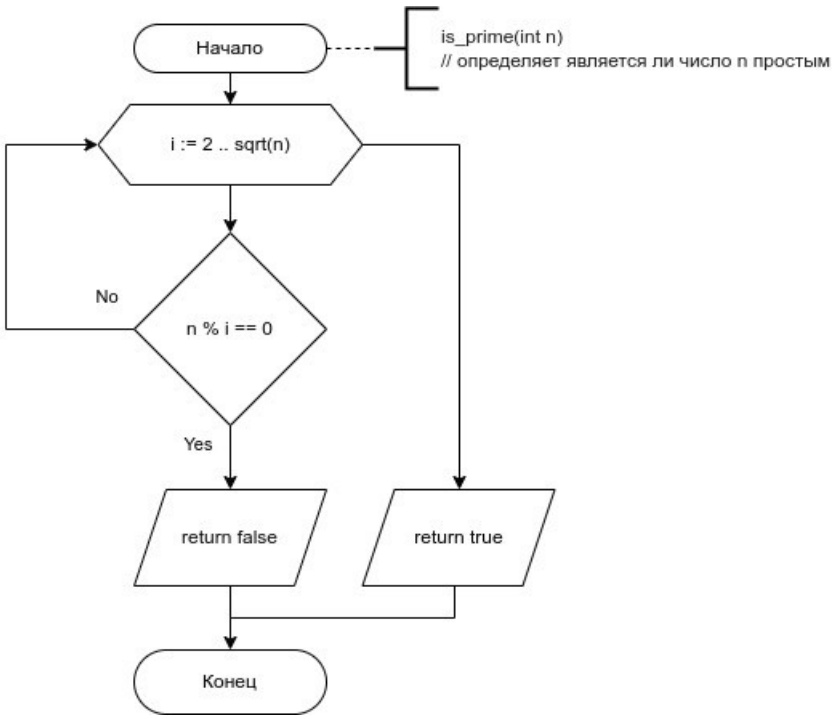

Рис 1. Блок-схема алгоритма функции определения простого числа

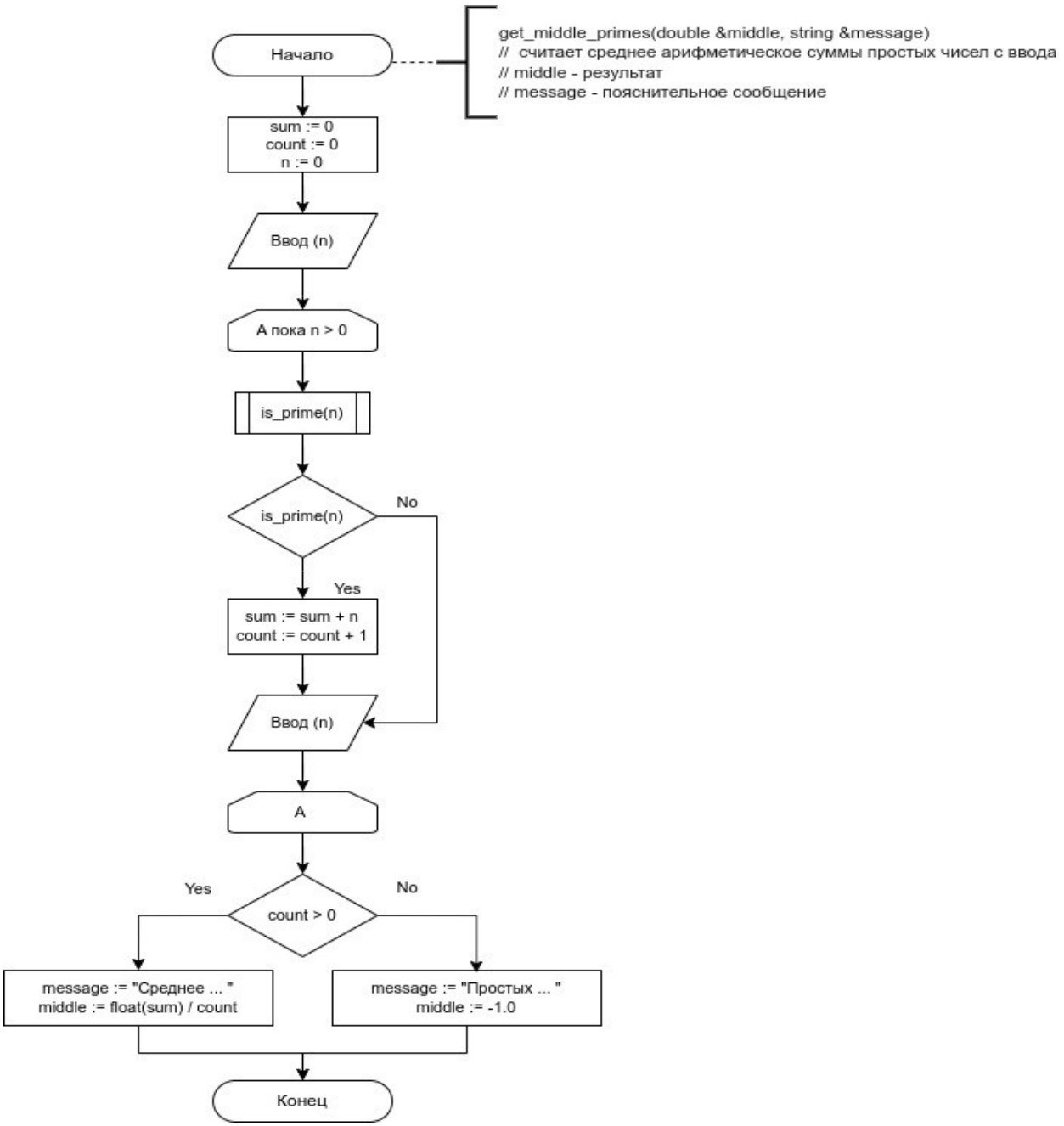

Рис 2. Блок-схема алгоритма процедуры нахождения среднего арифметического простых чисел в последовательности

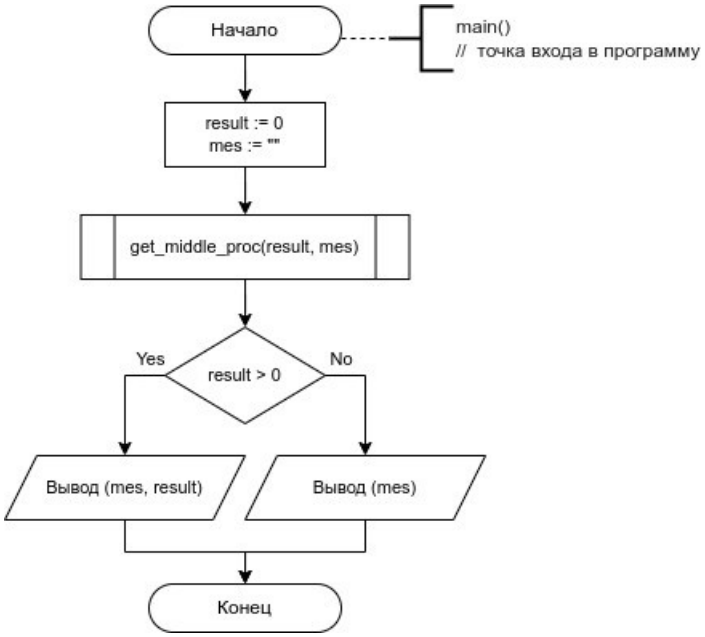

Рис 3. Блок-схема алгоритма программы с использованием процедуры

1.3 Текст программы (с использованием процедуры)

```cpp
#include <iostream>
#include <cmath>

using namespace std;

bool is_prime(int n) {
    for (int i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) return false;
    }
    return true;
}

void get_middle_primes(double &middle, string &message) {
    int sum{0};
    int count{0};

    int n{0};
    cin >> n;

    while (n > 0) {
        if (is_prime(n)) {
            sum += n;
            count++;
        }
        cin >> n;
    }

    if (count) {
        message = "Среднее арифметическое простых чисел: ";
        middle = float(sum) / count;
    } else {
        message = "Простых чисел в последовательности нет.";
        middle = -1.0;
    }
}

int main() {
    double result{0.0};
    string mes{""};

    get_middle_primes(result, mes);

    if (result > 0) {
        cout << mes << result << endl;
    } else {
        cout << mes << endl;
    }

    return 0;
}
```

Рис 4. Текст программы с использованием процедуры

1.4 Схема алгоритма решения задачи (с использованием функции)

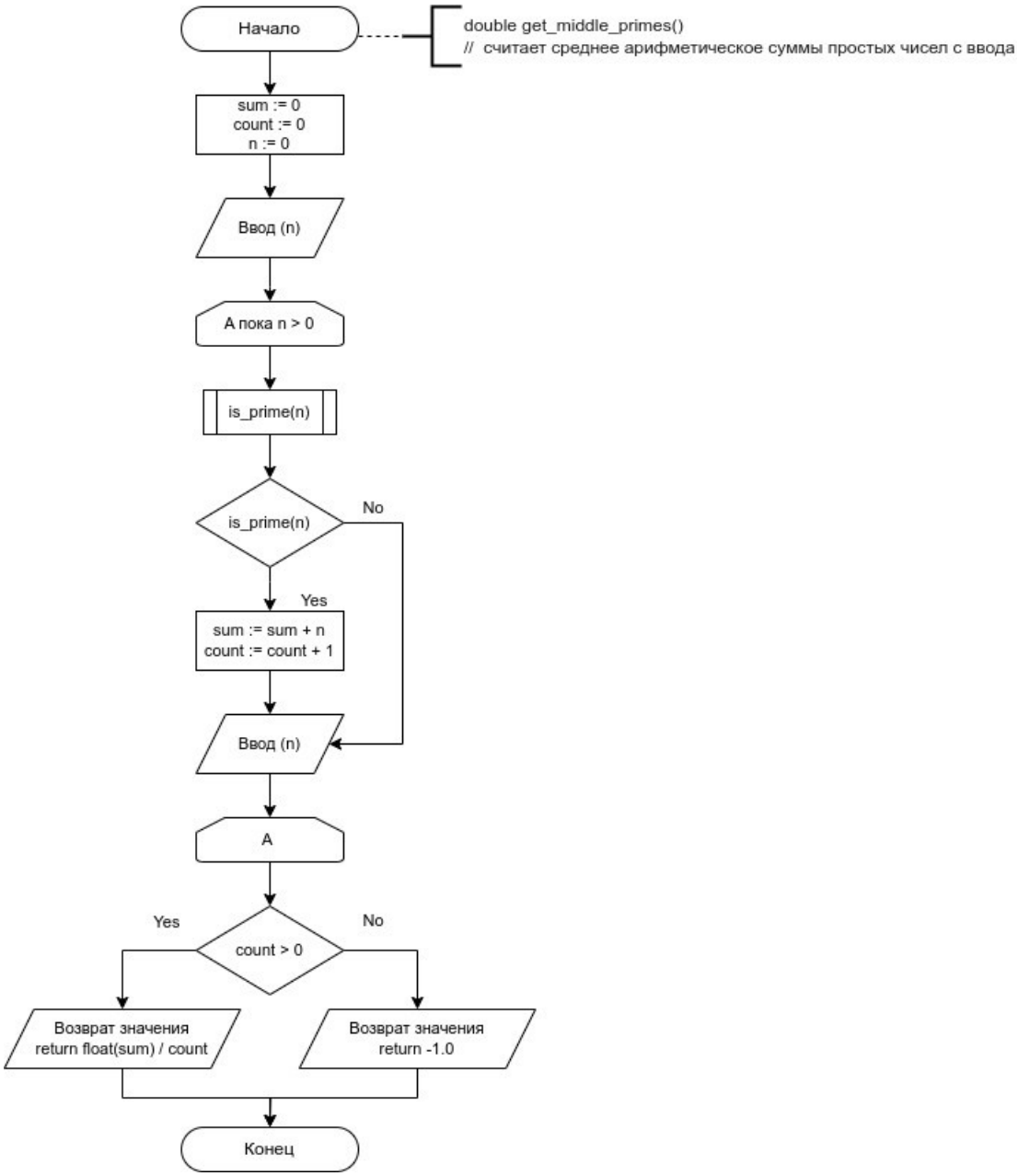

Рис 5. Блок-схема алгоритма функции нахождения среднего арифметического простых чисел в последовательности

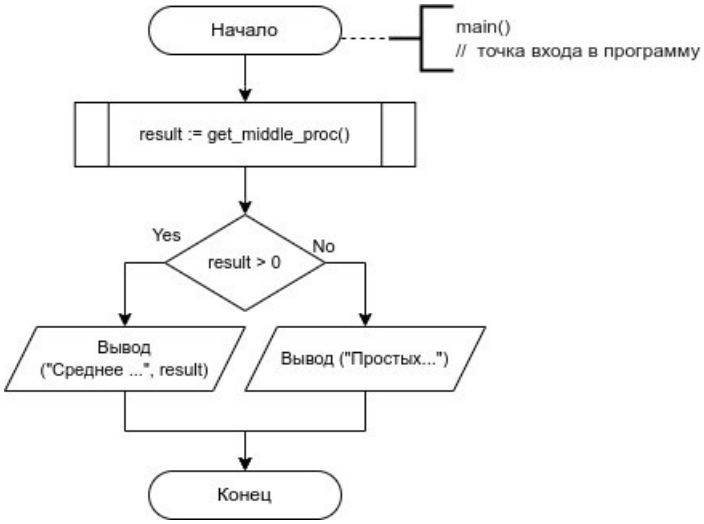

Рис 6. Блок-схема алгоритма программы с использованием функции

1.5 Текст программы (с использованием функции)

```cpp
#include <iostream>
#include <cmath>

using namespace std;

bool is_prime(int n) {
    for (int i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) return false;
    }
    return true;
}

double get_middle_primes() {
    int sum{0};
    int count{0};

    int n{0};
    cin >> n;

    while (n > 0) {
        if (is_prime(n)) {
            sum += n;
            count++;
        }
        cin >> n;
    }

    if (count) {
        return float(sum) / count;
    } else {
        return -1.0;
    }
}

int main() {
    double result = get_middle_primes();

    if (result > 0) {
        cout << "Среднее арифметическое простых чисел: " << result << endl;
    } else {
        cout << "Простых чисел в последовательности нет." << endl;
    }

    return 0;
}
```

Рис 7. Текст программы с использованием функции

1.6 Результаты выполнения программы

Для вывода работы программы используем последовательность чисел:
1 2 3 4 5 6 7 8 9 10

Из них простыми являются: 1 2 3 5 7, соответственно, среднее арифметическое равно (1+2+3+5+7) / 5 = 3,6.

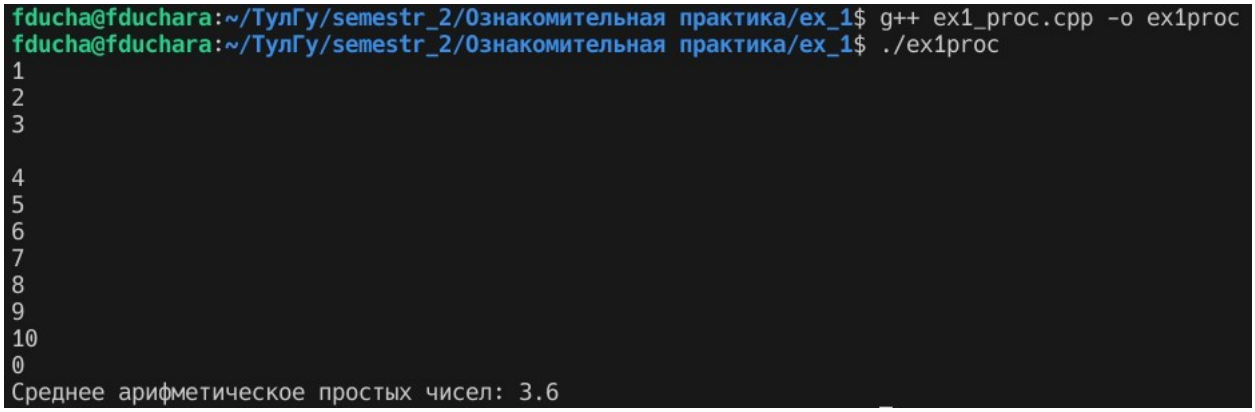

Рис 8. Компиляция и выполнение программы с использованием процедуры

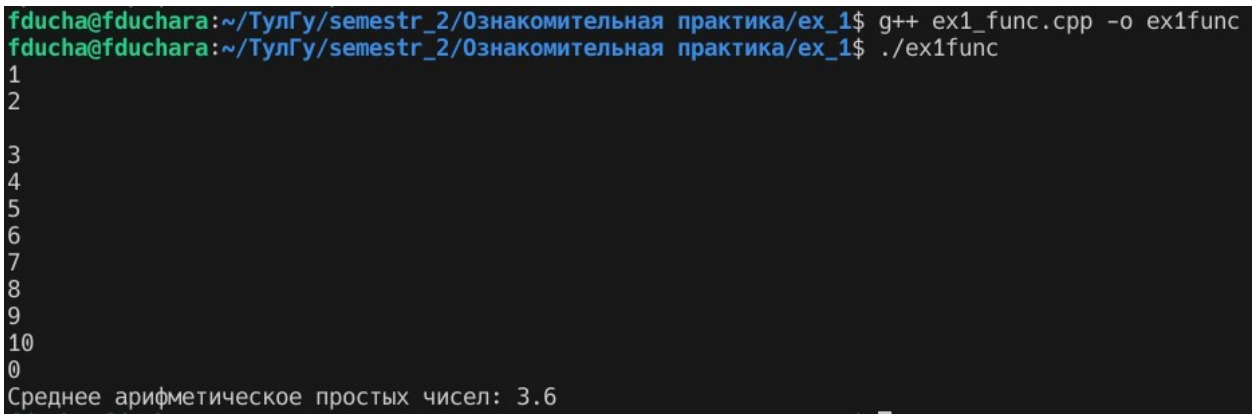

Рис 9. Компиляция и выполнение программы с использованием функции

2. Задание 2

2.1 Описание задачи

Ввести и сохранить в файле данные следующей структуры: ф.,и.,о. пассажира, пункт назначения (не менее 5 пунктов); стоимость билета; номер поезда. Организовать просмотр исходных данных и вывести список всех пассажиров, отсортированный по стоимости билета (возрастание метод вставки), с указанием пункта назначения и номера поезда. Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

2.2 Схема алгоритма решения задачи

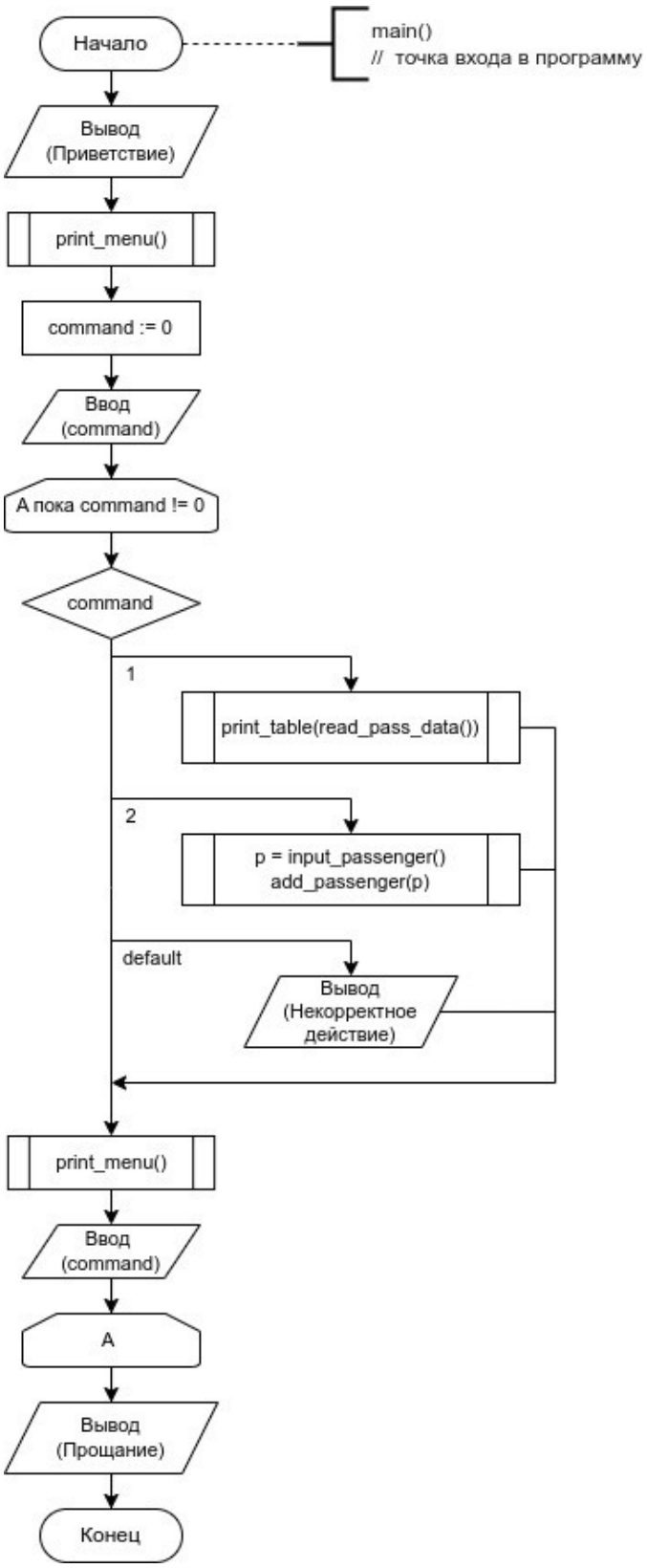

Рис 10: Блок-схема алгоритма программы

2.3 Текст программы

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <iomanip>
#include <bits/stdc++.h>

using namespace std;

const string FILE_NAME = "data.dat";

struct Passenger {
    string name;
    string destination;
    double price;
    string train_number;
};

Passenger input_passenger() {
    cout << "Введите ФИО: ";
    Passenger p;
    cin.ignore();
    getline(cin, p.name);
    cout << "Введите пункт назначения: ";
    cin.clear();
    cin >> p.destination;
    cout << "Введите стоимость билета: ";
    string price_str;
    cin >> price_str;
    p.price = stod(price_str);
    cout << "Введите номер поезда: ";
    cin >> p.train_number;

    return p;
}

void add_passenger(string name, string destination, double price, string train_number) {
    ofstream file;
    file.open(FILE_NAME, ios::app);
    file << endl << name + "&" + destination + "&" +
                std::to_string(price) + "&" + train_number;
    file.close();
}

Passenger parse_from_string(const string &data) {
    size_t pos = 0;
    Passenger p;

    stringstream ss{data};
    string str;

    while (getline(ss, str, '&'))
    {
        if (pos == 0) {
            p.name = str;
            pos++;
            continue;
        }
        if (pos == 1) {
            p.destination = str;
            pos++;
            continue;
        }
        if (pos == 2) {
            p.price = stod(str);
            pos++;
            continue;
        }
        if (pos == 3) {
            p.train_number = str;
            pos++;
            continue;
        }
    }

    return p;
}
```

```cpp
void swap_data(Passenger &p1, Passenger &p2) {
    Passenger tmp;
    tmp = p1;
    p1 = p2;
    p2 = tmp;
}

void sort_passengers_by_price(vector<Passenger> &data) {
    for (size_t i = 1; i < data.size(); i++)    {
        for (size_t j = i; j > 0 && data[j-1].price > data[j].price; j--) {
            swap_data(data[j-1], data[j]);
        }
    }
}

vector<Passenger> read_pass_data() {
    ifstream file;
    file.open(FILE_NAME, ios::in);
    vector<Passenger> result;
    while (!file.eof()) {
        string data;
        getline(file, data);
        Passenger p = parse_from_string(data);
        result.push_back(p);
    }
    file.close();
    sort_passengers_by_price(result);
    return result;
}

void print_menu() {
    cout << endl;
    cout << "Выберете действие и нажмите Ввод:" << endl;
    cout << "1 - Вывести список всех пассажиров;" << endl;
    cout << "2 - Добавить пассажира;" << endl;
    cout << "0 - Выйти из приложения." << endl;
}

void print_line() {
    for (size_t i = 0; i < 96; i++) {
        cout << "_";
    }
    cout << endl;
}

void print_table(const vector<Passenger> &data) {
    print_line();
    cout << "| " <<  "ФИО пассажира" << "\t\t\t\t| " << "Пункт назначения" << "\t| "
         << "Стоимость" << "\t| " << "Номер" << "\t\t|" <<  endl;
    cout << "|\t\t\t\t\t| \t\t\t| "  << "билета, руб." << "\t| " << "поезда" << "\t|" <<  endl;
    print_line();
    for (auto p: data) {
        string holder1 = p.name.length() <= 40 ? "\t\t\t| " : "\t\t| ";
        string holder2 = p.destination.length() < 10 ? "\t\t\t| " : p.destination.length() > 25 ? "\t| " : "\t\t| ";
        cout << "| " <<  p.name << holder1 << p.destination << holder2
            << fixed << setprecision(2) << p.price << "\t| " << p.train_number << "\t\t|" <<  endl;
    }
    print_line();
}
```

```
int main() {
    cout << "Вас приветствует приложение РЖД Пассажирам" << endl;
    print_menu();
    int command{0};
    cin >> command;
    while (command != 0) {
        switch (command) {
        case 1:
            print_table(read_pass_data());
            break;
        case 2: {
            Passenger p = input_passenger();
            add_passenger(p.name, p.destination, p.price, p.train_number);
        }
            break;
        default:
            cout << "Выбран некорректное действие" << endl;
            break;
        }
        print_menu();
        cin >> command;
    }
    cout << "До скорой встречи!!" << endl;
    return 0;
}
```

Пояснение.

Организация хранения данных пассажиров. Данные пассажиров сохраняются в текстовый файл data.dat, находящийся в одном каталоге с исполняемым файлом программы. Данные одного пассажира записываются как строка, содержащая символы «&» между полями в порядке согласно задания, например, «Иванов Иван Иванович&Краснодар&7599.00&030С».

Исходный код программы:

Passenger – структура хранения данных пассажира;

Passenger input_passenger() – функция организует ввод данных пассажира и возвращает их как объект Passenger;

void add_passenger(const Passenger &p) – функция записывает данные пассажира **p** в конец файла data.dat;

Passenger parse_from_string(const string &data) – функция производит парсинг строки (например, из файла data.dat), из данных которой создает объект структуры **Passenger** и возвращает его;

void swap_data(Passenger &p1, Passenger &p2) – функция меняет поля данных объектов местами, то есть значение полей **p1**, переносится в соответствующие поля **p2** и наоборот;

void sort_passengers_by_price(vector<Passenger> &data) – функция производит сортировку вставкой вектора, содержащего данные пассажиров, по полю стоимости билета;

vector<Passenger> read_pass_data() - функция возвращает сортированный вектор данных пассажиров, прочитанных из файла data.dat;

void print_menu() - выводит на экран меню выбора действий с программой (1 - Вывести список всех пассажиров, 2 - Добавить пассажира, 0 - Выйти из приложения);

void print_table(const vector<Passenger> &data) – выводит на экран данные пассажиров в виде таблицы.

2.4 Результаты выполнения программы

```
fducha@fduchara:~/ТулГу/semestr_2/Ознакомительная практика/ex2$ g++ main.cpp -o main && ./main
Вас приветствует приложение РЖД Пассажирам

Выберете действие и нажмите Ввод:
1 - Вывести список всех пассажиров;
2 - Добавить пассажира;
0 - Выйти из приложения.
1

| ФИО пассажира                    | Пункт назначения      | Стоимость билета, руб. | Номер поезда |
|----------------------------------|-----------------------|------------------------|--------------|
| Сидоров Николай Иванович         | Тула                  | 1180.00                | 113А         |
| Петров Петр Петрович             | Санкт-Петербург       | 1619.00                | 728А         |
| Ющенко Юлия Александровна        | Ярославль             | 1807.00                | 002Э         |
| Попов Владимир Юрьевич           | Воронеж               | 1872.00                | 126Э         |
| Сидоров Николай Иванович         | Орел                  | 2060.00                | 081А         |
| Коросень Иван Алиевич            | Вологда               | 2636.00                | 022Я         |
| Полетаев Вячеслав Игоревич       | Белгород              | 3083.00                | 081А         |
| Кутько Олег Антонович            | Котлас                | 4307.00                | 022Я         |
| Коновалов Алексей Сергеевич      | Архангельск           | 4511.00                | 016М         |
| Теркин Василий Алибабаевич       | Ташкент               | 5236.00                | 635Е         |
| Алабаев Исаак Рамзанович         | Махачкала             | 5932.00                | 133М         |
| Потапов Михаил Владимирович      | Воркута               | 6047.00                | 376Я         |
| Александрова Анна Ивановна       | Екатеринбург          | 6305.00                | 145А         |
| Марина Лариса Ивановна           | Санкт-Петербург       | 6415.00                | 020У         |
| Иванов Иван Иванович             | Краснодар             | 7599.00                | 030С         |
| Иванов Иван Иванович             | Сочи                  | 9233.00                | 102М         |
| Пупкин Василий Бедросович        | Уренгой               | 23334.00               | 232А         |

Выберете действие и нажмите Ввод:
1 - Вывести список всех пассажиров;
2 - Добавить пассажира;
0 - Выйти из приложения.
2
Введите ФИО: Киркоров Филипп Бедросович
Введите пункт назначения: Тель-Авив
Введите стоимость билета: 17350.50
Введите номер поезда: 353К

Выберете действие и нажмите Ввод:
1 - Вывести список всех пассажиров;
2 - Добавить пассажира;
0 - Выйти из приложения.
1

| ФИО пассажира                    | Пункт назначения      | Стоимость билета, руб. | Номер поезда |
|----------------------------------|-----------------------|------------------------|--------------|
| Сидоров Николай Иванович         | Тула                  | 1180.00                | 113А         |
| Петров Петр Петрович             | Санкт-Петербург       | 1619.00                | 728А         |
| Ющенко Юлия Александровна        | Ярославль             | 1807.00                | 002Э         |
| Попов Владимир Юрьевич           | Воронеж               | 1872.00                | 126Э         |
| Сидоров Николай Иванович         | Орел                  | 2060.00                | 081А         |
| Коросень Иван Алиевич            | Вологда               | 2636.00                | 022Я         |
| Полетаев Вячеслав Игоревич       | Белгород              | 3083.00                | 081А         |
| Кутько Олег Антонович            | Котлас                | 4307.00                | 022Я         |
| Коновалов Алексей Сергеевич      | Архангельск           | 4511.00                | 016М         |
| Теркин Василий Алибабаевич       | Ташкент               | 5236.00                | 635Е         |
| Алабаев Исаак Рамзанович         | Махачкала             | 5932.00                | 133М         |
| Потапов Михаил Владимирович      | Воркута               | 6047.00                | 376Я         |
| Александрова Анна Ивановна       | Екатеринбург          | 6305.00                | 145А         |
| Марина Лариса Ивановна           | Санкт-Петербург       | 6415.00                | 020У         |
| Иванов Иван Иванович             | Краснодар             | 7599.00                | 030С         |
| Иванов Иван Иванович             | Сочи                  | 9233.00                | 102М         |
| Киркоров Филипп Бедросович       | Тель-Авив             | 17350.50               | 353К         |
| Пупкин Василий Бедросович        | Уренгой               | 23334.00               | 232А         |

Выберете действие и нажмите Ввод:
1 - Вывести список всех пассажиров;
2 - Добавить пассажира;
0 - Выйти из приложения.
0
До скорой встречи!!
fducha@fduchara:~/ТулГу/semestr_2/Ознакомительная практика/ex2$
```

3. Задание 3

3.1 Описание задачи

Разработать процедуру или функцию, которая в списке L меняет местами первый и последний элементы.

Пояснение.

Для представления односвязного списка будем использовать собственную структуру LinkedList описанную в заголовочном файле /задание_3/linked_list.h. Узлом списка является структура Node, имеющая 2 поля: строка value и указатель next на следующий элемент списка. Структура LinkedList имеет 2 поля: указатели на первый и последний элементы списка, first и last соответственно, и 3 члена- функции: is_empty() - проверка является ли список пустым, push_back(string val) — добавляет новый элемент со значением строки val в конец списка и print(), которая выводит на экран значение value элемента и значение указателя на следующий элемент. Блок-схема структуры LinkedList не приводится.

Для решения задачи реализуется способ перемещения непосредственно элементов списка, а не их значений .

3.2 Схема алгоритма решения задачи

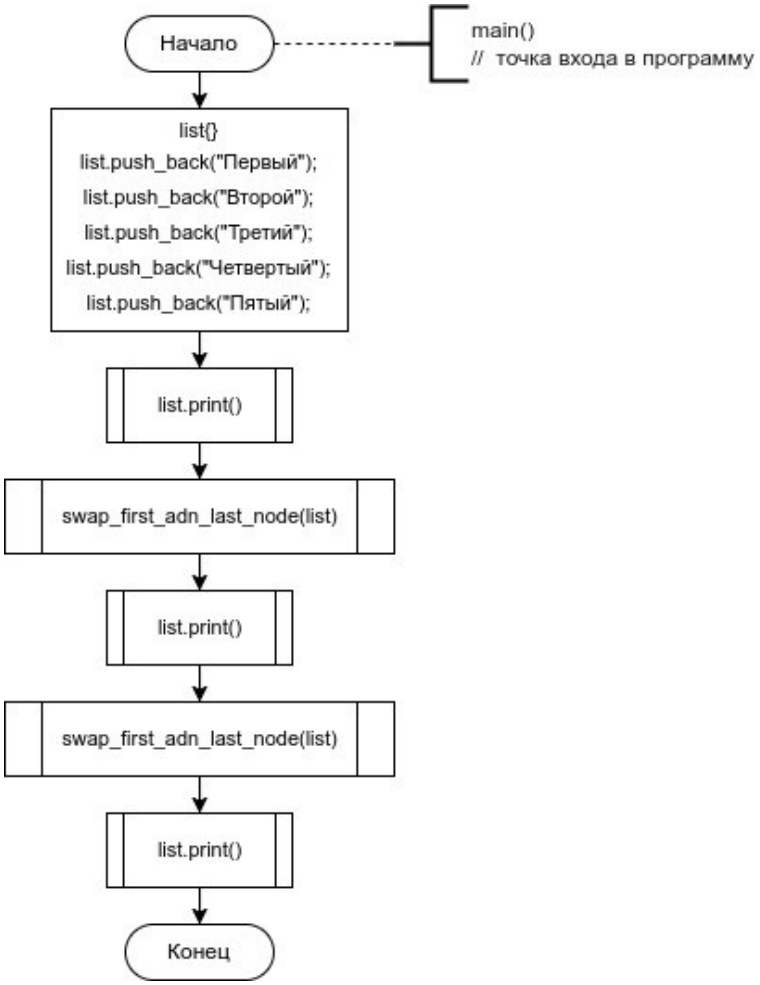

Рис 11: Блок-схема выполнения основного тела программы

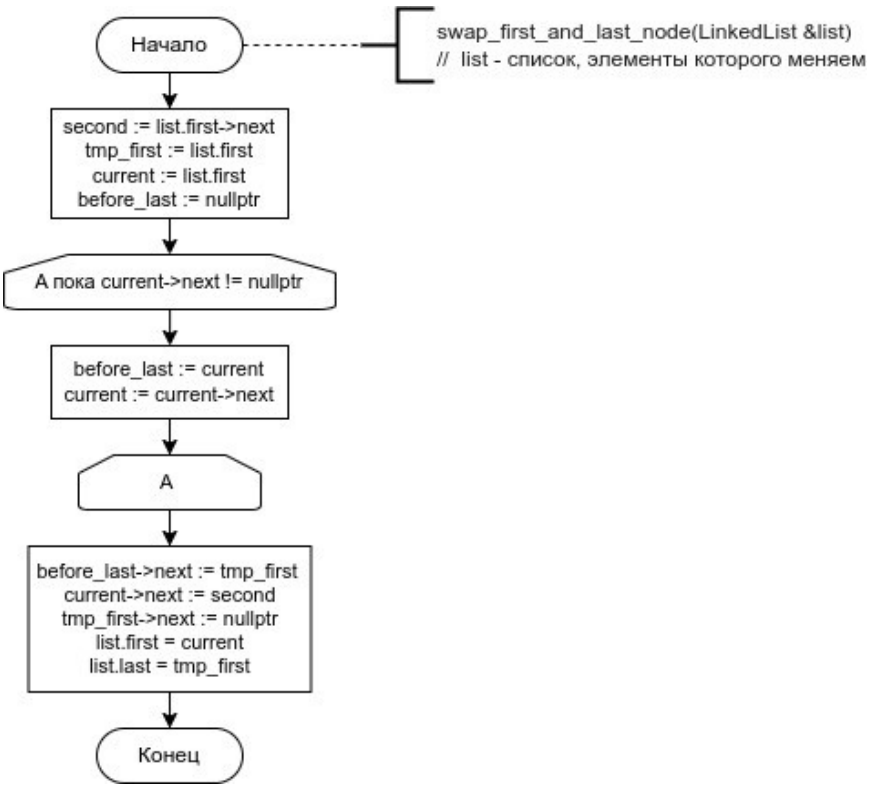

Рис 12: Блок-схема функции, которая меняет местами первый и последний элементы односвязного списка

3.3 Текст программы

Текст исходного кода программы

```
#include "linked_list.h"


void swap_first_and_last_node(LinkedList &list) {
    auto second = list.first->next;
    auto tmp_first = list.first;
    auto current = list.first;
    Node* before_last = nullptr;
    while (current->next) {
        before_last = current;
        current = current->next;
    }
    before_last->next = tmp_first;
    current->next = second;
    tmp_first->next = nullptr;
    list.first = current;
    list.last = tmp_first;
}

int main() {
    LinkedList list{};
    list.push_back("Первый");
    list.push_back("Второй");
    list.push_back("Третий");
    list.push_back("Четвертый");
    list.push_back("Пятый");

    list.print();

    swap_first_and_last_node(list);

    list.print();

    swap_first_and_last_node(list);

    list.print();

    return 0;
}

```

Текст исходного кода структуры LinkedList

```cpp
#include <iostream>
#include <string>

using namespace std;

struct Node {
    string value;
    Node* next;

    Node(string val) : value(val), next(nullptr) {}
};

struct LinkedList {
    Node* first;
    Node* last;

    LinkedList() : first(nullptr), last(nullptr) {}

    bool is_empty() const {
        return first == nullptr;
    }

    void push_back(string val) {
        auto p = new Node(val);
        if (is_empty()) {
            first = p;
            last = p;
        }

        last->next = p;
        last = p;
    }

    void print() {
        if (is_empty()) return;
        auto p = first;
        while (p) {
            cout << p->value << " (" << p << ")" << endl;
            p = p->next;
        }
        cout << endl;
    }
};
```

3.4 Результаты выполнения программы

```
fducha@fduchara:~/ТулГу/semestr_2/Ознакомительная практика/ex_3$ g++ main.cpp -o main && ./main
Первый (0x56018e018eb0)
Второй (0x56018e018ee0)
Третий (0x56018e018f10)
Четвертый (0x56018e018f80)
Пятый (0x56018e018fd0)

Пятый (0x56018e018fd0)
Второй (0x56018e018ee0)
Третий (0x56018e018f10)
Четвертый (0x56018e018f80)
Первый (0x56018e018eb0)

Первый (0x56018e018eb0)
Второй (0x56018e018ee0)
Третий (0x56018e018f10)
Четвертый (0x56018e018f80)
Пятый (0x56018e018fd0)

fducha@fduchara:~/ТулГу/semestr_2/Ознакомительная практика/ex_3$
```

В теле программы создается список из пяти элементов и выводится на экран. Затем первый и последний элементы списка меняются местами посредством функции, и список опять выводится. Обратим внимание, что обменялись местами не только значения элементов, но и сами указатели элементов. На следующем шаге происходит обратный обмен элементов и снова вывод списка. Как видно, элементы встали на свои изначальные места.

4. Реферат на тему «Уровни представления баз данных»

Концепции многоуровневой архитектуры систем управления базами данных (далее СУБД) служат основой современной технологии баз данных. Эти идеи впервые были сформулированы в отчёте рабочей группы по базам данных Комитета по планированию стандартов Американского национального института стандартов (ANSI/X3/SPARC), опубликованному в 1975 г. В нем была предложена обобщенная трехуровневая модель архитектуры СУБД, включающая концептуальный, внешний и внутренний уровни (рис. 13).

Рис. 13: Уровни представления данных

Первая попытка создания стандартной терминологии и общей архитектуры СУБД была предпринята в 1971 году группой, называемой DBTG. Она была создана после конференции CODASYL (Conference on Data Systems and Languages — Конференция по языкам и системам данных), прошедшей в этом же году. Группа DBTG признала необходимость использования двухуровневого подхода, построенного на основе использования системного представления, то есть схемы (schema), и пользовательских представлений, то есть подсхем (subschema). Сходные терминология и архитектура были предложены в 1975 году Комитетом планирования стандартов и норм SPARC (Standards Planning and Requirements Committee) Национального Института Стандартизации США (American National Standard Institute— ANSI), ANSI/X3/SPARC (ANSI, 1975). Комитет ANSI/SPARC признал необходимость использования трехуровневого подхода. В этих материалах отражены предложения, которые были сделаны организациями Guide/Share, состоящими из пользователей продуктов корпорации IBM, и опубликованы за несколько лет до этого. Основное

внимание в них было сконцентрировано на необходимости воплощения независимого уровня для изоляции программ от особенностей представления данных на более низком уровне (Guide/Share, 1970). Хотя модель ANSI/SPARC не стала стандартом, тем не менее она все еще представляет собой основу для понимания некоторых функциональных особенностей СУБД.

В данном случае наиболее фундаментальным моментом в этих и последующих отчетах исследовательских групп является идентификация трех уровней абстракции, то есть трех различных уровней описания элементов данных. Эти уровни формируют трехуровневую архитектуру, которая охватывает внешний, концептуальный и внутренний уровни. Цель трехуровневой архитектуры заключается в отделении пользовательского представления базы данных от ее физического представления. Ниже перечислено несколько причин, по которым желательно выполнять такое разделение:

1. Каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя свое собственное представление о них. Каждый пользователь должен иметь возможность изменять свое представление о данных, причем это изменение не должно оказывать влияния на других пользователей.

2. Пользователи не должны непосредственно иметь дело с такими подробностями физического хранения данных в базе, как индексирование и хеширование. Иначе говоря, взаимодействие пользователя с базой не должно зависеть от особенностей хранения в ней данных.

3. Администратор базы данных должен иметь возможность изменять структуру хранения данных в базе, не оказывая влияния на пользовательские представления.

4. Внутренняя структура базы данных не должна зависеть от таких изменений физических аспектов хранения информации, как переключение на новое устройство хранения.

5. Администратор базы данных должен иметь возможность изменять концептуальную или глобальную структуру базы данных без какого-либо влияния на всех пользователей.

Уровень, на котором воспринимают данные пользователи, называется внешним уровнем, тогда как СУБД и операционная система воспринимают данные на внутреннем уровне. Именно на внутреннем уровне данные реально сохраняются с использованием структур и файловой организации. Концептуальный уровень представления данных предназначен для

отображения внешнего уровня на внутренний и обеспечения необходимой независимости друг от друга.

Рис 14: Схематичное представление архитектуры ANSI/SPARC

Внешний уровень.

Индивидуальный уровень пользователя называется внешним уровнем. Пользователи могут относиться к различным группам: прикладные программисты, конечные пользователи, администраторы (они работают также с внутренним и концептуальным уровнем). У каждого пользователя есть свой язык общения с СУБД. У конечного пользователя это может быть язык запросов или специальный язык, основанный на формах и меню. Для прикладных программистов им может быть один из языков высокого уровня. Все эти языки включают подъязык данных, то есть подмножество операторов базового языка, связанное только с объектами и операциями баз данных. Наиболее распространенный подобный язык – SQL (англ. Structured Query Language – язык структурированных запросов), он поддерживается большинством систем реляционного типа.

Любой язык данных является комбинацией, по крайней мере, двух подчиненных языков – языка определения данных (англ. Data Definition Language, DDL), который поддерживает определение и объявление объектов базы данных, и языка обработки данных (англ. Data Manipulation Language, DML), который поддерживает операции с объектами баз данных, их обработку.

Отдельного пользователя интересует только некоторая часть всей баз данных. Кроме того, пользовательское представление этих данных может существенно отличаться от того, как они хранятся. В соответствии с терминологией ANSI/SPARC представление отдельного пользователя называется внешним представлением. Внешнее представление – это содержимое базы данных, каким его видят определенный конечный

пользователь или группа пользователей. Можно сказать, что для пользователя его внешнее представление и есть база данных.

Внешних представлений обычно бывает несколько. Например, пользователь из отдела кадров может рассматривать базу данных как набор записей об отделах и служащих. Он может ничего не знать про записи о деталях и поставщиках, с которыми работают пользователи в отделе обеспечения.

В общем случае, внешнее представление состоит из множества экземпляров внешних записей, которые могут не совпадать с хранимыми записями. В системах, отличных от баз данных, логическая (внешняя) запись обычно совпадает с хранимой.

Концептуальный уровень.

Концептуальный уровень состоит из одного представления. Концептуальное представление – это представление всей информации базы данных в несколько более абстрактной форме (как и в случае внешнего представления) по сравнению с физическим способом хранения данных.

Концептуальное представление состоит из множества экземпляров каждого типа концептуальной записи. Концептуальная запись не обязательно должна совпадать с внешней записью, с одной стороны, и с хранимой записью – с другой. Хранимая запись – это набор связанных хранимых полей. Хранимое поле – это наименьшая единица хранимых данных. База данных содержит экземпляры каждого из нескольких типов хранимых полей.

Концептуальное представление – это представление всего содержимого базы данных, а концептуальная схема – это определение такого представления. Определения в концептуальной схеме могут включать определения многих дополнительных средств, таких как средства безопасности или правила для обеспечения целостности. Администратор данных, определяя, какие характеристики предметной области требуется сохранять в системе, фактически определяет основу концептуальной схемы.

Этот уровень осведомлён о различиях между базами данных и способен построить путь выполнения операций во всех случаях. Однако концептуальный уровень отступает к физическому уровню для фактической реализации каждой отдельной операции.

Концептуальный уровень архитектуры ANSI/SPARC служит для поддержки единого взгляда на базу данных, общего для всех её приложений и независимого от них. Концептуальный уровень представляет собой формализованную информационно-логическую модель программного обеспечения.

Внутренний уровень (физический уровень).

Так же как и концептуальный, внутренний уровень состоит только из одного представления. Внутреннее представление описывает все подробности, связанные с хранением данных в базе. Оно состоит из экземпляров каждого типа внутренней записи. Термин "внутренняя запись" принадлежит терминологии ANSI/SPARC и фактически соответствует хранимой записи. Внутреннее представление, так же как внешнее и концептуальное, не связано с аппаратным уровнем и не включает подробностей, связанных с размещением данных на дисках, таких как номера секторов и тому подобное.

Внутреннее представление описывается с помощью внутренней схемы, которая определяет типы хранимых записей, индексы (служебные структуры, упрощающие поиск данных), способы представления хранимых полей, физическую последовательность хранимых записей и так далее.

Внутренний уровень архитектуры поддерживает представление базы данных в среде хранения – хранимую базу данных. На этом архитектурном уровне базы данных представлена в полностью «материализованном» виде, тогда как на других уровнях идёт работа на уровне отдельных экземпляров или множества экземпляров записей. Описание базы данных на внутреннем уровне называется внутренней схемой или схемой хранения.

Отображения.

В представленной архитектуре присутствует отображение двух уровней. Отображение концептуального уровня на внутренний определяет соответствие между концептуальным представлением и хранимой базой данных. При изменении структур хранения изменяется и отображение «концептуальный – внутренний» таким образом, чтобы концептуальная схема осталась неизменной. Это обеспечивает так называемую физическую независимость данных.

Отображение внешнего уровня на концептуальный определяет соответствие между внешними представлениями и концептуальным. Например, несколько концептуальных полей «индекс», «города», «улица», «дом» для пользователя могут быть объединены в одно внешнее поле «адрес». Появляется возможность менять отдельные внешние представления, дополнительно изменяя только отображения и не затрагивая остальные уровни системы. Отделение внешнего уровня от концептуального обеспечивает логическую независимость данных.

Определения представлений каждого из уровней и отображений СУБД должна хранить вместе с прочей метаинформацией и использовать их при обработке запросов.

Совокупность схем всех уровней называется схемой базы данных.

Каждый из этих уровней может считаться управляемым, если он обладает внешним интерфейсом, который поддерживает возможности определения данных. В этом случае становится возможными формирование и системная поддержка независимого взгляда на базу данных для какой-либо группы персонала или пользователей, взаимодействующих с базу данных через интерфейс данного уровня.

Архитектура ANSI/SPARC имеет большое теоретическое значение, определяя пути обеспечения логической и физической независимости данных. Однако два уровня отображения приводят к дополнительным накладным расходам при обработке запросов пользователя, поэтому разработчики СУБД, стараясь увеличить быстродействие систем, обычно отходят от строгой реализации этой архитектуры.

Используемые материалы:

Нестеров, С.А. Базы данных: учебник и практикум для вузов/ С.А.Нестеров. - 2-е изд., перераб. и доп. - Москва: Издательство Юрайт, 2023. - 258 с. - (Высшее образование);

SQL: https://ru.wikipedia.org/wiki/SQL;

Уровень абстракции базы данных: https://ru.wikipedia.org/wiki/Уровень_абстракции_базы_данных;

Трехуровневая архитектура ANSI-SPARC: http://andrey-2119163.narod.ru/management_data/lection_02.htm;