

Вариант 5

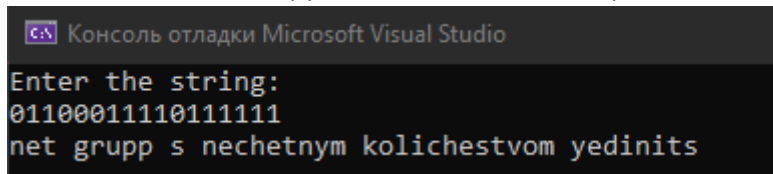
1) Строки (часть 1, лаб 6, 6.4.2)

В строке, состоящей из групп нулей и единиц, подсчитать количество единиц в группах с нечетным количеством символов.

```
#include <iostream>
#include <string>
#include <stdlib.h>

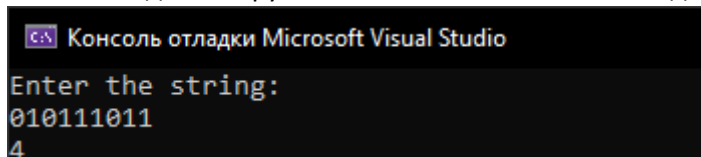
int main() {
    using namespace std;
    string s;
    int count1 = 0, count2 = 0; // 1 - счетчик единиц в группе, 2 - счетчик единиц по
    всем группам
    cout << "Enter the string: " << endl;
    getline(cin, s);
    for (int i = 0; i < s.length(); i++) {
        if (s[i] == '1') {
            count1++;
        }
        else if (s[i] == '0') {
            if (count1 % 2) {
                count2 += count1;
            }
            count1 = 0;
        }
        else { //если элемент строки не является 0 или 1, выводим сообщение об ошибке
            cout << "input error" << endl;
            exit(0);
        }
    }
    if (count2 != 0) {
        cout << count2 << endl;
    }
    else cout << "net grupp s nechetyym kolichestvom edynits" << endl;
    return 0;
}
```

1. Когда во всех группах с единицами содержится четное количество единиц



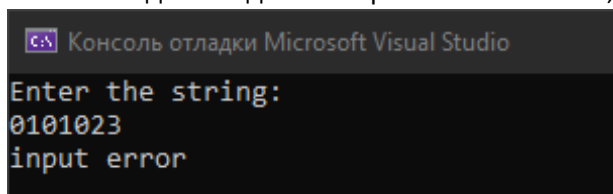
```
Консоль отладки Microsoft Visual Studio
Enter the string:
0110001111011111
net grupp s nechetyym kolichestvom yedynits
```

2. Когда есть группы с нечетным количеством единиц



```
Консоль отладки Microsoft Visual Studio
Enter the string:
010111011
4
```

3. Когда в введенной строке есть символы, не равные «0» или «1».



```
Консоль отладки Microsoft Visual Studio
Enter the string:
0101023
input error
```

2) Файлы (часть 1, лаб 7.4.1)

Написать программу обработки файла записей, содержащую следующие подпункты меню: «Создание», «Просмотр», «Добавление», «Решение индивидуального задания». Каждая запись должна содержать следующую информацию о студентах:

- фамилия
- номер группы
- оценки за семестр: по физике, математике и информатике
- средний балл

Организовать ввод исходных данных, средний балл рассчитать по введенным оценкам. Содержимое всего файла и результаты решения индивидуального задания записать в текстовый файл.

Задание:

Найти информацию о студентах, имеющих отметку 4 или 5 по физике и больше 8 по остальным предметам

```
/*
Написать программу обработки файла записей, содержащую следующие подпункты меню :
«Создание», «Просмотр», «Добавление», «Решение индивидуального задания».
Каждая запись должна содержать следующую информацию о студентах :
- фамилия
- номер группы
- оценки за семестр : по физике, математике и информатике
- средний балл
Организовать ввод исходных данных, средний балл рассчитать по введенным
оценкам. Содержимое всего файла и результаты решения индивидуального задания записать
в текстовый файл.

Найти информацию о студентах, имеющих отметку 4 или 5 по физике и больше 8 по
остальным предметам.
*/
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cmath>
#include <string>

using namespace std;

struct student {
    string surname;
    int group = 0;
    int math = 0;
    int phys = 0;

    int inf = 0;
    double sr_ball;
};

void OutputTableHead() {
    cout << setw(5) << left << " #"
         << setw(16) << left << "Фамилия" << "|"
         << setw(6) << "Группа" << "|"
         << setw(10) << "Мат" << "|"
         << setw(17) << "Физ" << "|"
         << setw(9) << "Инф" << "|"
         << setw(14) << "Ср. балл" << "|"
         << endl;
    cout << "-----"
         << endl;
```

```

}

void OutputTableBody(student s[], int N) {
    for (int i = 0; i < N; i++) {
        cout << setw(5) << left << " #"
            << setw(16) << left << s[i].surname << "|"
            << setw(6) << s[i].group << "|"
            << setw(10) << s[i].math << "|"
            << setw(17) << s[i].phys << "|"
            << setw(9) << s[i].inf << "|"
            << setw(14) << s[i].sr_ball << "|"
            << endl;
    }
}

void FileEdition(string file_name) { // работа с файлом
    int loop = 1;

    while (loop == 1) {
        int temp_answ = 0;
        int answer = 0;
        string temp_string;
        cout << "\nРабота с файлом: " << file_name
            << ": \n1. просмотр файла \n2. Добавить запись в файл \n3.
Задание \n \n 0. Выход в главное меню \n \n";
        cin >> answer;

        if (answer == 0) { // 0 - выход в главное меню
            loop = 0;
        }
        if (answer == 2) { // 2 - добавить запись в файл
            int answer1 = 1;

            ofstream file;
            // поток записи в файл
            file.open(file_name, ios::app);
            // открыть файл в режиме app, чтобы указатель переместился в конец файла и
            // предыдущие данные не стерлись
            if (!file.is_open()) {
                cout << "Ошибка. Файл не был открыт!" << endl;
            }
            else {
                string t_surname;
                int t_group = 0, t_phys = 0, t_math = 0, t_inf = 0;
                double t_sr_ball;
                answer1 = 1;
                while (answer1 == 1) {
                    cout << "Фамилия: " << endl;
                    cin >> t_surname;
                    cout << "# группы: " << endl;
                    cin >> t_group;
                    cout << "Оценка по математике: " << endl;
                    cin >> t_math;
                    cout << "Оценка по физике: " << endl;
                    cin >> t_phys;
                    cout << "Оценка по информатике: " << endl;
                    cin >> t_inf;

                    t_sr_ball = round(100*(t_phys + t_math + t_inf)/3.)
/ 100;

                    cout << "Средний балл равен: " << t_sr_ball << endl;
                    cout << "Сохранить изменения в файле? \n1 - да, \n 2 -
нет" << endl;

                    cin >> temp_answ;

```

```

        if (temp_answ == 1) {
            //if (!file.eof()) file << "\n";
            file << t_surname << " " << t_group << " " <<
t_math << " " << t_phys << " " << t_inf << " " << t_sr_ball;
        }
        else cout << "Данные не были добавлены в файл!" <<
endl;
        cout << "Желаете добавить еще запись? \n1 - да\n2 -
нет" << endl;

        cin >> answer1;
        if ((temp_answ == 1) && (answer1 == 1)) file <<
"\n";
    }
    file.close();
}

// 1 - просмотр файла
// 3 - выполнение задания

if (answer == 1 || answer == 3 ) {

    int N = 0;
    ifstream file;
    file.open(file_name);
    if (!file.is_open()) cout << "Ошибка. Файл не был открыт!" <<
endl;

    else {
        file.seekg(0);
        int oN = 0;
        string f_str;
        getline(file, f_str);
        if (file.eof() || f_str == "") cout << "Файл пуст!" <<
endl;

        else {
            file.seekg(0);
            while (!file.eof()) {
                // считаем количество строк в файле
                getline(file, temp_string, '\n');
                oN++;
            }
            N = oN;
            oN = 0;
            file.seekg(0);
            student* s = new student[N]; // массив структур
            int i = 0;
            while (!file.eof() && i <= N - 1) { // вытягиваем
данные с файла в массив структур
                file >> s[i].surname >> s[i].group >>
s[i].math >> s[i].phys >> s[i].inf >> s[i].sr_ball;
                i++;
            }
            file.close();
            i = 0;
            if (answer == 1) { // 1 -
просмотр файла
                OutputTableHead();
                OutputTableBody(s, N);
            }

            if (answer == 3) {

```



```

    }
}

else if (answer == 2) { // Работа с
    существующим файлом
        FileEdition(file_name);
    }
    else cout << "Неверный ответ!" << endl;
}
return 0;
}

```

Текущая кодовая страница: 1251

----- МЕНЮ -----

1. Создать файл
2. Работа с существующим файлом
3. Выход из программы

1
Введите имя файла (без .txt): abc
Файл был создан.

----- МЕНЮ -----

1. Создать файл
2. Работа с существующим файлом
3. Выход из программы

2
Введите имя файла (без .txt): abc

Работа с файлом: abc.txt:

1. просмотр файла
2. Добавить запись в файл
3. Задание

0. Выход в главное меню

2

Фамилия:

Иванов

группы:

1

Оценка по математике:

9

Оценка по физике:

5

Оценка по информатике:

9

Средний балл равен: 7.67

Сохранить изменения в файле?

1 - да,

2 - нет

1

Желаете добавить еще запись?

1 - да

2 - нет

1

Фамилия:

Петров

группы:

1

Оценка по математике:

9

Оценка по физике:

9

Оценка по информатике:

9

Средний балл равен: 9

Сохранить изменения в файле?

1 - да,

2 - нет

1

Фамилия:

Николаев

группы:

9

Оценка по математике:

9

Оценка по физике:

5

Оценка по информатике:

10

Средний балл равен: 8

Сохранить изменения в файле?

1 - да,

2 - нет

1

Желаете добавить еще запись?

1 - да

2 - нет

2

Работа с файлом: abc.txt:

1. просмотр файла
2. Добавить запись в файл
3. Задание

0. Выход в главное меню

3

#	Фамилия	Группа	Мат	Физ	Инф	Ср. балл	
#	Иванов	1	9	5	9	7.67	
#	Николаев	9	9	5	10	8	

Работа с файлом: abc.txt:

1. просмотр файла
2. Добавить запись в файл
3. Задание

0. Выход в главное меню

3) Рекурсия (часть 2, лаб 1, 1.3, стр 7)

Найти значение функции Аккермана $A(m, n)$, которое определяется для всех неотрицательных целых аргументов m и n следующим образом:

$$A(0, n) = n + 1$$

$$A(m, 0) = A(m-1, 1) \text{ при } m > 0$$

$$A(m, n) = A(m-1, A(m, n-1)) \text{ при } m > 0 \text{ и } n > 0$$

```
#include <iostream>

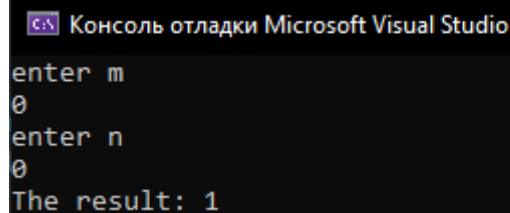
using namespace std;

double number(string text) { // проверка на то, чтобы введенные данные были числом
    int t = 0;
    double x;
    while (t == 0) {
        cout << "enter " << text << endl;
        cin >> x;
        if (cin.fail()) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Input error. Enter NUMBER" << endl;
        }
        else t = 1;
    }
    return x;
}

int A(int m, int n) {
    if (m == 0) return n + 1;
    else {
        if ((m > 0) && (n == 0)) return A(m - 1, 1);
        else if ((m > 0) && (n > 0)) return A(m - 1, A(m, n - 1));
    }
}

int main() {
    int m = 0, n = 0;
    m = number("m");
    n = number("n");
    if ((m >= 0) && (n >= 0)) cout << "The result: " << A(m, n) << endl;
    else cout << "One of the numbers is negative" << endl; //проверка, чтобы m и
n были положительными
    return 0;
}
```

Работа программы



```
Консоль отладки Microsoft Visual Studio
enter m
0
enter n
0
The result: 1
```

Проверка введенных данных на тип и отрицательность

Результат выполнения программы

m	n	result
0	0	1
0	1	2
0	2	3
0	3	4
1	0	2
1	1	3
1	2	4
1	3	5
2	0	3
2	1	5

<div>Консоль отладки Microsoft Visual Studio</div> <div>enter m п Input error. Enter NUMBER enter m в Input error. Enter NUMBER enter m к Input error. Enter NUMBER enter m 6 enter n -32 One of the numbers is negative</div>			2	2	7
			2	3	9
			3	0	5
			3	1	13
			3	2	29
			3	3	61
			4	0	13

4) Стеки (часть 2, лаб 3, 3.3, стр 20)

Написать программу по созданию, добавление, просмотру и решению приведенных дальше задач для однонаправленного линейного списка типа Stack. Реализовать сортировку стека методами, рассмотренными в подразделе 3.1.

Из созданного списка удалить элементы, заканчивающиеся на цифру 5

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <limits>

using namespace std;

struct Stack {
    int data;
    Stack* next;
};

// функция формирования элемента стека
Stack* s_push(Stack* p, int in){ // указатель на вершину и введенное значение
    Stack* t = new Stack;        // захватываем память для элемента
    t->data = in;                 // формирование информационной части
    t->next = p;                  // формирование адресной части
    return t;                    // возвращаем измененное значение вершины
}

// просмотр стека
void s_view(Stack* p) {
    Stack* t = p;
    while (t != NULL) {
        cout << " " << t->data << endl;
        t = t->next;
    }
}

// функция освобождения памяти, занятой стеком
void s_del(Stack** p) {
    Stack* t;
    while (*p != NULL) {
        t = *p;
        *p = (*p)->next;
        delete t;
    }
}

void StackSort(Stack* p) {
    Stack* t = NULL;
    Stack* tl;
    int r, i = 0;
    do {
        for (tl = p; tl->next != t; tl = tl->next) {
            if (tl->data > tl->next->data) {
                r = tl->data;
                tl->data = tl->next->data;
                tl->next->data = r;
            }
        }
        t = tl;
    } while (p->next != t);
}

int main() {
    Stack* begin = NULL;
    Stack* temp_begin = NULL;
    int i, in, n, kod, el;
    while (true) {
```

```

        cout << "\n\t1. Create\n\t2. Add\n\t3. View\n\t4. Delete\n\t5. Individual
task\n\t6. Sorting\n\t0. Exit\n\t";
    cin >> kod;
    switch (kod) {
    case 1: case 2: //create , add
        cout << "Input the num of elements: ";
        cin >> n;
        if ((kod == 1) && (begin != NULL)) {
            s_del(&begin);
        }
        for (i = 1; i <= n; i++) {
            in = rand() % 101 - 50;
            begin = s_push(begin, in);
        }
        if (kod == 1) cout << "Create " << n << " elements." << endl;
        else cout << "Add " << n << " elements." << endl;
        cout << "The stack: " << endl;
        s_view(begin);
        break;

    case 3: // view
        if (!begin) {
            cout << "The Stack is empty!" << endl;
            break;
        }
        s_view(begin);
        break;

    case 4: //delete
        s_del(&begin);
        cout << "The memory is free!" << endl;
        break;

    case 5:
        if (!begin) {
            cout << "The Stack is empty!" << endl;
            break;
        }
        while (begin != NULL) {
            if (begin->data % 5 == 0) {
                if (begin->data % 10 != 0) cout << "Delete " << begin-
>data << endl;

                else temp_begin = s_push(temp_begin, begin->data);
            }
            else temp_begin = s_push(temp_begin, begin->data);
            begin = begin->next;
        }

        while (temp_begin != NULL) {
            begin = s_push(begin, temp_begin->data);
            temp_begin = temp_begin->next;
        }
        delete temp_begin;
        cout << "\tNew stack: " << endl;
        s_view(begin);
        break;

    case 6: // сортировка
        if (!begin) {
            cout << "The Stack is empty!" << endl;
            break;
        }
        StackSort(begin);
        cout << "\tNew stack: " << endl;
        s_view(begin);
        break;
    }
}

```

```

case 0:
    if (begin != NULL) s_del(&begin);
    exit(0);

default:
    cout << "The answer doesn't exist. Try again!";
    break;
}}}

```

1. Создание стека	2. Добавление элементов в стек
<pre> 1. Create 2. Add 3. View 4. Delete 5. Individual task 6. Sorting 0. Exit 1 Input the num of elements: 5 Create 5 elements. The stack: 30 -12 22 35 -9 </pre>	<pre> 1. Create 2. Add 3. View 4. Delete 5. Individual task 6. Sorting 0. Exit 2 Input the num of elements: 4 Add 4 elements. The stack: 46 18 15 19 30 -12 22 35 -9 </pre>
3. Просмотр стека	4. Удаление стека
<pre> 1. Create 2. Add 3. View 4. Delete 5. Individual task 6. Sorting 0. Exit 3 46 18 15 19 30 -12 22 35 -9 </pre>	<pre> 1. Create 2. Add 3. View 4. Delete 5. Individual task 6. Sorting 0. Exit 4 The memory is free! </pre>
5. Удаление всех элементов, заканчивающихся на 5	6. Сортировка

	<pre>1. Create 2. Add 3. View 4. Delete 5. Individual task 6. Sorting 0. Exit 3 46 18 15 19 30 -12 22 35 -9</pre>			<pre>1. Create 2. Add 3. View 4. Delete 5. Individual task 6. Sorting 0. Exit 6 New stack: -12 -9 18 19 22 30 46</pre>	
--	---	--	--	--	--

5) Очереди (часть 2, лаб 4, стр 26)

Написать программу по созданию, добавлению (в начало, в конец), просмотру (с начала, с конца) и решению приведенной в подразделе 3.3.задачи для двунаправленных линейных списков

Из созданного списка удалить элементы, заканчивающиеся на цифру 5

```
/*
Написать программу по созданию, добавлению (в начало, в конец),
просмотру (с начала, с конца) и решению приведенной в подразделе 3.3.задачи для
двунаправленных линейных списков
Из созданного списка удалить элементы, заканчивающиеся на цифру 5
*/
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <limits>

using namespace std;

struct Spisok {
    int info;
    Spisok* next;
    Spisok* prev;
};

void CreateList(Spisok** begin_spisok, Spisok** end_spisok, int in) { // создание
первого элемента
    Spisok* t = new Spisok;
    t->info = in;
    t->next = t->prev = NULL;
    *begin_spisok = *end_spisok = t;
}

//добавление элемента в список
void AddElement(int kod, Spisok** begin_spisok, Spisok** end_spisok, int in) {
    Spisok* t = new Spisok;
    t->info = in;
    if (kod == 0) {
        t->prev = NULL;
        t->next = *begin_spisok;
        (*begin_spisok)->prev = t;
        (*begin_spisok) = t;
    }
    else {
        t->next = NULL;
        t->prev = *end_spisok;
        (*end_spisok)->next = t;
        *end_spisok = t;
    }
}

//просмотр списка
void ViewList(int kod, Spisok** begin_spisok, Spisok** end_spisok) {
    //просмотр списка с начала
    if (kod == 1) {
        Spisok* t = *begin_spisok ;
        while (t != NULL) {
            cout << t->info << endl;
            t = t->next;
        }
    }
}
```

```

        //просмотр списка с конца
        if (kod == 2) {
            Spisok* t = *end_spisok;
            while (t != NULL) {
                cout << t->info << endl;
                t = t->prev;
            }
        }
    }
    void DellList(Spisok** p) { //удаление списка
        Spisok* t;
        while (*p != NULL) {
            t = *p;
            *p = (*p)->next;
            delete t;
        }
    }

    int main() {
        Spisok* begin = NULL, * end = NULL, * key = NULL;
        Spisok* t = new Spisok;
        int i, in, n, kod, kod1, num;
        while (true) {
            cout << "\n\t1. Create\n\t2. Add\n\t3. View\n\t4. Delete\n\t5.
Individual task\n\t0. Exit\n\t";
            cin >> kod;

            switch (kod) {

                case 1://Create
                    if (begin != 0) {
                        cout << "the list isn't empty!";
                        break;
                    }
                    in = rand() % 101 - 50;
                    Createlist(&begin, &end, in);
                    cout << "Create Begin = " << begin->info << endl;
                    break;

                case 2://add
                    cout << "add info\n1. to the begin of the list\n2. to the end of
the list" << endl;
                    cin >> kod1;
                    if (kod1 != 1 && kod1 != 2) {
                        cout << "Error answer!";
                        break;
                    }
                    cout << "Enter num of elements: "; cin >> num;
                    for (i = 1; i <= num; i++) {
                        in = rand() % 101 - 50;
                        AddElement(kod1, &begin, &end, in);
                        cout << "Add " << in << endl;
                    }
                    break;

                case 3://view
                    if (!begin) {
                        cout << "The list is empty!" << endl;
                        break;
                    }
                    cout << "1 - View from begin\n2 - View from the end " << endl;
                    cin >> kod1;
                    if (kod1 == 1) {
                        cout << "\tFrom begin: " << endl;

```

```

        ViewList(kod1, &begin, &end);
    }
    else if (kod1 == 2) {
        cout << "\tFrom end: " << endl;
        ViewList(kod1, &begin, &end);
    }
    else {
        cout << "error answer!" << endl;
        break;
    }

    break;

case 4: //delete
    DelList(&begin);
    break;

case 5: //удалить из списка элементы заканчивающиеся на 5
    t = begin; //текущий указатель выставляем на начало списка
    ViewList(1, &begin, &end);
    while (t != NULL) {
        if (t->info % 5 == 0) {
            if (t->info % 10 != 0) {
                key = t;
                if (key == begin) {
                    t = t->next;
                    t->prev = NULL;
                }
                else if (key == end) {
                    end = end->prev;
                    end->next = NULL;
                }
                else {
                    key->prev->next = key->next;
                    key->next->prev = key->prev;
                }
            }
            t = t->next;
        }

        if (key == NULL) {
            cout << "There are no items in this list that end in 5" <<
endl;

            break;
        }

        cout << "\tNew stack: " << endl;
        ViewList(1, &begin, &end);
        break;

case 0: //exit
    if (begin != NULL) DelList(&begin);
    exit(0);

default:
    cout << "The answer doesn't exist. Try again!";
    break;
    }
}
return 0;
}

```

1. Создание списка и первого элемента


```
1. Create
2. Add
3. View
4. Delete
5. Individual task
0. Exit
1
Create Begin = -9
```

2. Добавить элементы в список (20 штук)

```
1. Create
2. Add
3. View
4. Delete
5. Individual task
0. Exit
2
add info
1. to the begin of the list
2. to the end of the list
1
Enter num of elements: 20
Add 35
Add 22
Add -12
Add 30
Add 19
Add 15
Add 18
Add 46
Add -28
Add -1
Add 17
Add 1
Add 11
Add 13
Add 37
Add 16
Add -26
Add 30
Add 33
Add 21
```

3) просмотр списка

Просмотр с начала	Просмотр с конца
<pre> 1. Create 2. Add 3. View 4. Delete 5. Individual task 0. Exit 3 1 - View from begin 2 - View from the end 1 From begin: -9 35 22 -12 30 19 15 18 46 -28 -1 17 1 11 13 37 16 -26 30 33 21 </pre>	<pre> 1. Create 2. Add 3. View 4. Delete 5. Individual task 0. Exit 3 1 - View from begin 2 - View from the end 2 From end: 21 33 30 -26 16 37 13 11 1 17 -1 -28 46 18 15 19 30 -12 22 35 -9 </pre>

3. Удалить список (удаление и последующий просмотр)

```

1. Create
2. Add
3. View
4. Delete
5. Individual task
0. Exit
4

1. Create
2. Add
3. View
4. Delete
5. Individual task
0. Exit
3
The list is empty!

```

5.) выполнение индивидуального задания: отображает старый и новый стек (с начала)

- 1. Create
- 2. Add
- 3. View
- 4. Delete
- 5. Individual task
- 0. Exit
- 5

-9
35
22
-12
30
19
15
18
46
-28
-1
17
1
11
13
37
16
-26
30
33
21

New stack:

-9
22
-12
30
19
18
46
-28
-1
17
1
11
13
37
16
-26
30
33
21

6) Нелинейные списки (часть 2, лаб 6, стр 40)

Определить число узлов на каждом уровне дерева.

7) Курсач

Написать программу обработки файла данных, состоящих из структур, в которой реализованы следующие функции:

Список разговоров на международной АТС

Вид списка: дата разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента

Здание: вывести по каждому городу общее время разговоров с ним и сумму.

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>

using namespace std;

struct atc {
    int date[3] = { 0,0,0 }; // дата разговора {число.месяц.год}
    int code = 0; // код города

    string city; // название города

    int time[3] = { 0,0,0 }; // время разговора {часы.минуты.секунды}
    double tarif = 0; // тариф

    string city_num; // номер телефона в городе

    string subscriber_num; // номер телефона абонента
};

struct memory {
    int code = 0;
    string city;
    int time=0;
    double tarif=0;
    double sum=0;
    int time_all[3] = { 0,0,0 };
};

void QuickSort(atc contact[], int left, int right) { //сортируем по коду города
    if (left > right) return;
    atc c = contact[(left + right) / 2];
    int i = left;
    int j = right;
    atc temp;
    while (i <= j) {
        while (contact[i].code < c.code) i++;
        while (contact[j].code > c.code) j--;
        if (i <= j) {
            temp = contact[i];
            contact[i] = contact[j];
            contact[j] = temp;
        }
    }
}
```

```

        contact[i] = contact[j];
        contact[j] = temp;
        i++;
        j--;
    }
}
QuickSort(contact, left, j);
QuickSort(contact, i, right);
}

void LinearSearch(atc contact[], int N, memory task[], int count) {
    int j = 0;
    for (int i = 1; i < N; i++) {
        if (contact[i].code == contact[i - 1].code) {
            task[j].time += contact[i].time[0] * 3600 + contact[i].time[1] * 60 +
contact[i].time[2];
            task[j].code = contact[i].code;
            task[j].city = contact[i].city;
            task[j].tarif = contact[i].tarif;
            task[j].sum += task[j].tarif * task[j].time;

            task[j].time_all[0] = task[j].time / 3600;
            task[j].time_all[1] = task[j].time % 3600;
            task[j].time_all[2] = task[j].time_all[1] % 60;
            task[j].time_all[1] = task[j].time_all[1] / 60;
        }
        else {
            j++;
            task[j].time += contact[i].time[0] * 3600 + contact[i].time[1] * 60 +
contact[i].time[2];
            task[j].code = contact[i].code;
            task[j].city = contact[i].city;
            task[j].tarif = contact[i].tarif;
            task[j].sum += task[j].tarif * task[j].time;

            task[j].time_all[0] = task[j].time / 3600;
            task[j].time_all[1] = task[j].time % 3600;
            task[j].time_all[2] = task[j].time_all[1] % 60;
            task[j].time_all[1] = task[j].time_all[1] / 60;
        }
    }
}

void OutputTableHead() {
    cout << setw(5) << left << " #"
        << setw(16) << left << "Дата разговора" << "|"
        << setw(6) << " Код" << "|"
        << setw(10) << " Город" << "|"
        << setw(17) << " Время разговора" << "|"
        << setw(9) << " Тариф" << "|";
}

```

```

        << setw(14) << " Номер города" << "|"
        << setw(16) << " Номер абонента" << "|"
        << endl;
    cout << "-----" << endl;

}

void OutputTableBody(atc contact[], int N) {
    for (int i = 0; i < N; i++) {
        cout << setw(3) << right << i + 1 << ".";

        if (contact[i].date[0] < 10) cout << right << setw(8) << "0" << contact[i].date[0] << ".";
        else cout << right << setw(9) << contact[i].date[0] << ".";
        if (contact[i].date[1] < 10) cout << "0" << contact[i].date[1] << ".";
        else cout << contact[i].date[1] << ".";
        cout
            << contact[i].date[2] << "|"
            << setw(6) << contact[i].code << "|"
            << setw(10) << contact[i].city << "|";
        if (contact[i].time[0] < 10) cout << setw(10) << "0" << contact[i].time[0] << ".";
        else cout << setw(11) << contact[i].time[0] << ".";
        if (contact[i].time[1] < 10) cout << "0" << contact[i].time[1] << ".";
        else cout << contact[i].time[1] << ".";
        if (contact[i].time[2] < 10) cout << "0" << contact[i].time[2] << " ";
        else cout << contact[i].time[2] << "|";
        cout
            << setw(9) << contact[i].tarif << "|"
            << setw(14) << contact[i].city_num << "|"
            << setw(16) << contact[i].subscriber_num << "|"
            << endl;
    }
}

void FileEdition(string file_name) { // работа с файлом
    int loop = 1;

    while (loop == 1) {
        int temp_answ = 0;
        int answer = 0;
        string temp_string;
        cout << "\nРабота с файлом: " << file_name
            << ": \n1. просмотр файла\n2. Добавить запись в файл\n3. Изменить запись в
        файле\n4. Удалить запись из файла\n5. Задание\n\n0. Выход в главное меню\n\n";
        cin >> answer;

        if (answer == 0) { // 0 - выход в главное меню
            loop = 0;
        }
        if (answer == 2) { // 2 - добавить запись в файл
            int answer1 = 1;

```

```

        fstream file;                                // поток записи в файл
        file.open(file_name, ios::app);                // открыть файл в режиме app,
чтобы указатель переместился в конец файла и предыдущие данные не стерлись
        if (!file.is_open()) {
            cout << "Ошибка. Файл не был открыт!" << endl;
        }
        else {
            string t_date, t_code, t_city, t_time, t_city_num, t_subscriber_num;
            double t_tarif;
            answer1 = 1;
            while (answer1 == 1) {
                cout << "Дата разговора (в формате ДД.ММ.ГГ 13.04.2025): "
<< endl;

                cin >> t_date;
                cout << "Код города: " << endl;
                cin >> t_code;
                cout << "Город: " << endl;
                cin >> t_city;
                cout << "Продолжительность разговора (в формате
ЧАСЫ.МИНУТЫ.СЕКУНДЫ): " << endl;

                cin >> t_time;
                cout << "Тариф: " << endl;
                cin >> t_tarif;
                cout << "Номер города: " << endl;
                cin >> t_city_num;
                cout << "Номер абонента: " << endl;
                cin >> t_subscriber_num;

                cout << "Сохранить изменения в файле?\n1 - да,\n2 - нет" <<
endl;

                cin >> temp_answ;

                if (temp_answ == 1) {
                    cout << "Размер файла " << file.tellg() << endl;
                    file << "\n";
                    file << t_date << " " << t_code << " " << t_city << " " <<
t_time << " " << t_tarif << " " << t_city_num << " " << t_subscriber_num;
                }
                else cout << "Данные не были добавлены в файл!" << endl;
                cout << "Желаете добавить еще запись? \n1 - да\n2 - нет" <<
endl;

                cin >> answer1;
                //if ((temp_answ == 1) && (answer1 == 1)) file << "\n";
            }
            file.close();
        }
    }
}

```



```

//      1 - просмотр файла+      3 - изменить запись в
файле
//      4 - удалить запись из файла      5 - выполнение задания

if (answer == 1 || answer == 3 || answer == 4 || answer == 5) {

    int N = 0;
    ifstream file;
    file.open(file_name);
    if (!file.is_open()) cout << "Ошибка. Файл не был открыт!" << endl;

    else {
        file.seekg(0);
        int oN = 0;
        string f_str;
        getline(file, f_str);
        if (file.eof() || f_str == "") cout << "Файл пуст!" << endl;
        else {
            file.seekg(0);
            while (!file.eof()) {
                getline(file, temp_string, '\n');
                oN++;
            }
            N = oN;
            oN = 0;
            file.seekg(0);
            atc* contact = new atc[N]; // массив структур
            int i = 0;
            string temp_d, temp_t;
            while (!file.eof() && i <= N - 1) { // вытягиваем данные с файла

                file >> temp_d >> contact[i].code >> contact[i].city >>
temp_t >> contact[i].tarif >> contact[i].city_num >> contact[i].subscriber_num;
                contact[i].date[0] = stoi(temp_d.substr(0, 2));
                contact[i].date[1] = stoi(temp_d.substr(3, 2));
                contact[i].date[2] = stoi(temp_d.substr(6, 4));

                contact[i].time[0] = stoi(temp_t.substr(0, 2));
                contact[i].time[1] = stoi(temp_t.substr(3, 2));
                contact[i].time[2] = stoi(temp_t.substr(6, 2));
                i++;
            }
            file.close();
            i = 0;
            if (answer == 1) {
                OutputTableHead();
                OutputTableBody(contact, N);
            }
        }
    }
}

```

```

if (answer == 3) {
    cout << "Исходный список: " << endl;
    OutputTableHead();
    OutputTableBody(contact, N);
    int index_number = 0, answer3 = 0;
    string temp_date, temp_time;
    cout << "\nВведите порядковый номер записи,
которую желаете изменить: ";

    cin >> index_number;
    index_number--;
    if (index_number <= N) {
        cout << "\nЧто вы желаете изменить? \n1 -
дата разговора, \n2 - код города, \n3 - название города, \n4 - время разговора, \n5 - тариф, \n6 -
номер телефона города, \n7 - номер телефона абонента" << endl;
        cin >> answer3;
        switch (answer3) {
            case 1:
                cout << "Введите дату разговора в
формате ДД.ММ.ГГ:" << endl;

                cin >> temp_date;
                break;
            case 2:
                cout << "Введите индекс города:" <<
endl;

                cin >> contact[index_number].code;
                break;
            case 3:
                cout << "Введите название города:" <<
endl;

                cin >> contact[index_number].city;
                break;
            case 4:
                cout << "Время разговора
(ЧЧ.ММ.СС):" << endl;

                cin >> temp_time;
                break;
            case 5:
                cout << "Введите тариф:" << endl;
                cin >> contact[index_number].tarif;
                break;
            case 6:
                cout << "Введите номер телефона
города: " << endl;

                cin >> contact[index_number].city_num;
                break;
            case 7:
                cout << "Введите номер телефона
абонента:" << endl;

```

```

        cin >>

contact[index_number].subscriber_num;

        break;
default:
        cout << "Неверный ответ. Попробуйте

еще раз.\n";

    }

    cout << "Измененная запись: " << endl;
    if (answer3 == 1) { //date
        cout
            << temp_date << " " <<

            << contact[index_number].city

            <<

            <<

            <<

            << contact[index_number].tarif
            << " " << contact[index_number].city_num << " " << contact[index_number].subscriber_num << endl;
        }
        else if (answer3 == 4) { //time
            cout
                <<

                <<

                <<

                << " " <<

                << " " << temp_time << " " <<

                <<

                << contact[index_number].city_num << " " << contact[index_number].subscriber_num << endl;
            }
        else {
            cout
                <<

                <<

                <<

                << contact[index_number].code

            << " " << contact[index_number].city << " "

```

```

contact[index_number].time[0] << "."
contact[index_number].time[1] << "."
contact[index_number].time[2] << " "
<< " " << contact[index_number].city_num
contact[index_number].subscriber_num << endl;

    }
    cout << "Сохранить изменения в файле?\n1 -
да,\n2 - нет" << endl;

    cin >> temp_answ;

    if (temp_answ == 1) {
        ofstream file;
        file.open(file_name);
        for (int i = 0; i < N; i++) {
            if (i == index_number &&

answer3 == 1) {

                file << temp_date << " "
                    <<

contact[i].code << " " << contact[i].city << " ";

                if (contact[i].time[0] <
10) file << "0" << contact[i].time[0] << ".";

                else file <<

contact[i].time[0] << ".";

                if (contact[i].time[1] <
10) file << "0" << contact[i].time[1] << ".";

                else file <<

contact[i].time[1] << ".";

                if (contact[i].time[2] <
10) file << "0" << contact[i].time[2] << " ";

                else file <<

contact[i].time[2] << " ";

                file << contact[i].tarif <<

" " << contact[i].city_num << " " << contact[i].subscriber_num;

            }

        }

    else if (i == index_number &&

answer3 == 4) {

                if (contact[i].date[0] <
10) file << "0" << contact[i].date[0] << ".";

                else file <<

contact[i].date[0] << ".";

                if (contact[i].date[1] <
10) file << "0" << contact[i].date[1] << ".";

```

```

contact[i].date[1] << ".";

contact[i].date[2] << " "

contact[i].code << " " << contact[i].city << " " << temp_time << " "

<< " " << contact[i].city_num << " " << contact[i].subscriber_num;

10) file << "0" << contact[i].date[0] << ".";

contact[i].date[0] << ".";

10) file << "0" << contact[i].date[1] << ".";

contact[i].date[1] << ".";

<< " " << contact[i].code << " " << contact[i].city << " ";

10) file << "0" << contact[i].time[0] << ".";

contact[i].time[0] << ".";

10) file << "0" << contact[i].time[1] << ".";

contact[i].time[1] << ".";

10) file << "0" << contact[i].time[2] << " ";

contact[i].time[2] << " ";

" " << contact[i].city_num << " " << contact[i].subscriber_num;

else file <<

file

<<

<<

<< contact[i].tarif

}
else {
    if (contact[i].date[0] <

    else file <<

    if (contact[i].date[1] <

    else file <<

    file << contact[i].date[2]

    if (contact[i].time[0] <

    else file <<

    if (contact[i].time[1] <

    else file <<

    if (contact[i].time[2] <

    else file <<

    file << contact[i].tarif <<

}

if (i != N - 1) {
    file << endl;
}

}

cout << "Данные в файле обновлены!"

<< endl << endl;

file.close();

}

else cout << "Данные в файле не были

temp_answ = 0;

}

else {

```

```

        cout << "Нет записи с таким номером" << endl;
        file.close();
    }
}

if (answer == 4) { //удалить запись из файла
    cout << "Исходный список: " << endl;
    OutputTableHead();
    OutputTableBody(contact, N);
    int index_number = 0;
    cout << "\nВведите порядковый номер записи,
которую желаете удалить: ";

    cin >> index_number;
    cout << "Сохранить изменения в файле?\n1 - да,\n2 -
нет" << endl;

    cin >> temp_answ;
    if (temp_answ == 1) {
        for (int i = index_number - 1; i < N - 1; i++) {
            contact[i].date[0] = contact[i +
1].date[0];
            contact[i].date[1] = contact[i +
1].date[1];
            contact[i].date[2] = contact[i +
1].date[2];

            contact[i].code = contact[i + 1].code;
            contact[i].city = contact[i + 1].city;
            contact[i].time[0] = contact[i +
1].time[0];
            contact[i].time[1] = contact[i +
1].time[1];
            contact[i].time[2] = contact[i +
1].time[2];

            contact[i].tarif = contact[i + 1].tarif;
            contact[i].city_num = contact[i +
1].city_num;
            contact[i].subscriber_num = contact[i +
1].subscriber_num;
        }
        N--;
        cout << endl << "Новый список:" << endl;
        OutputTableBody(contact, N);

        ofstream file;
        file.open(file_name);
        for (int i = 0; i < N; i++) {
            if (contact[i].date[0] < 10) file << "0" <<
contact[i].date[0] << ".";
            else file << contact[i].date[0] << ".";
            if (contact[i].date[1] < 10) file << "0" <<
contact[i].date[1] << ".";

```

```

        else file << contact[i].date[1] << ".";
        file
            << contact[i].date[2] << " "
            << contact[i].code << " " <<

contact[i].city << " ";

contact[i].time[0] << ".";

contact[i].time[1] << ".";

contact[i].time[2] << " ";

contact[i].city_num << " " << contact[i].subscriber_num;

        file << contact[i].tarif << " " <<

        if (i != N - 1) file << "\n";
    }
    cout << "Данные в файле обновлены!" << endl;
    file.close();
}
else cout << "Данные в файле не были обновлены" <<

endl;

temp_answ = 0;
}

if (answer == 5) {
    OutputTableBody(contact, N);
    QuickSort(contact, 0, N - 1);

    cout << endl << endl << "сортировка по коду города: "

<< endl;

    OutputTableHead();
    OutputTableBody(contact, N);
    cout << endl;
    int count = 1;
    for (i = 1; i < N; i++) {
        if (contact[i].code != contact[i - 1].code) count++;
    }

    memory* task = new memory[count];
    int j = 0;
    task[0].code = contact[0].code;
    task[0].city = contact[0].city;
    task[0].tarif = contact[0].tarif;
    task[0].time = contact[0].time[0] * 3600 +
contact[0].time[1] * 60 + contact[0].time[2];

    task[0].sum += task[0].tarif * task[0].time;
    task[j].time_all[0] = task[j].time / 3600;

```



```

    if (answer == 1 || answer == 2) {
        cout << "Введите имя фала (без .txt): ";
        cin >> file_name;
        file_name = file_name + ".txt";
    }

    if (answer == 1) {                                     //        создать файл
        ofstream file;
        file.open(file_name);
        if (!file.is_open()) {
            cout << "Ошибка. Файл не был открыт!" << endl;
        }
        else {
            cout << "Файл был создан." << endl;
            file.close();
        }
    }

    else if (answer == 2) {                                //        Работа с
    существующим файлом
        FileEdition(file_name);
    }
    else cout << "Неверный ответ!" << endl;
}
return 0;
}

```