**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 ввода

cout << "input error" << endl;

exit(0);

}

}

if (count2 != 0) {

cout << count2<<endl;

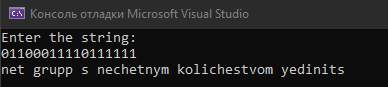
}

else cout << "net grupp s nechetnym kolichestvom edinits" << endl;

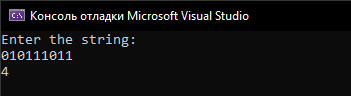
return 0;

}

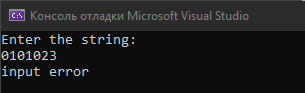
1. Когда во всех группах с единицами содержится четное количество единиц



1. Когда есть группы с нечетным количеством единиц



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



1. **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\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;

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) {

int t = 0;

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

for (i = 0; i < N; i++) {

if ((s[i].phys == 4 || s[i].phys == 5) && (s[i].math > 8) && (s[i].inf > 8)) {

if (t == 0) OutputTableHead();

t++;

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;

}

}

if (t == 0) cout << "Нет студентов, имеющих отметку 4 или 5 по физике и больше 8 по остальным предметам" << endl;

}

}

}

}

}

}

int main() {

setlocale(LC\_ALL, "RU");

system("chcp 1251");

int answer, N = 0, i = 0;

string file\_name;

while (1) {

cout << "\n-------- МЕНЮ -----------" << endl;

cout << "\n1. Создать файл\n2. Работа с существующим файлом\n3. Выход из программы" << endl;

cin >> answer;

if (answer == 3) exit(0); // выход из программы

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);

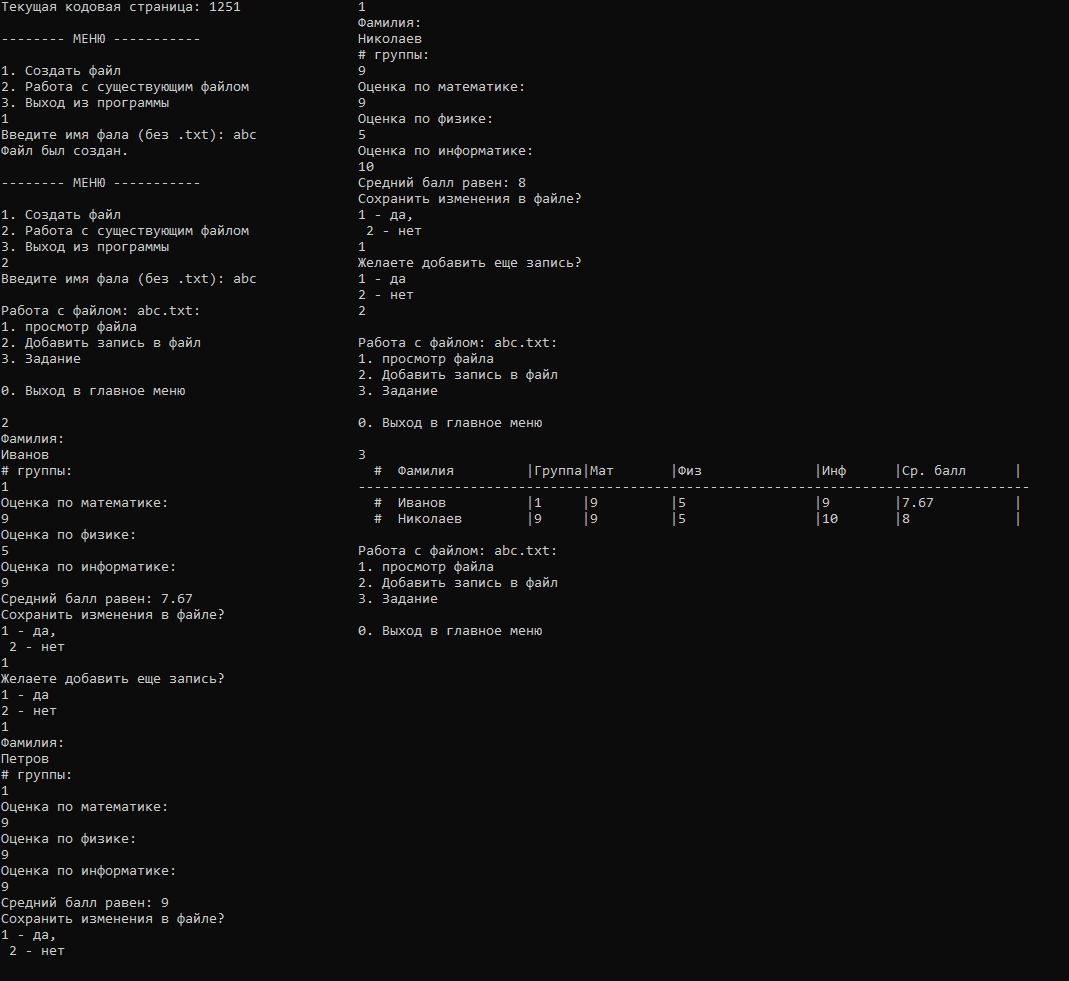
}

else cout << "Неверный ответ!" << endl;

}

return 0;

}



1. **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(O, n) = n + 1

A(m, O) = A(m-1, 1) при m > 0

A(m, n) = A(m-1, A(m, n-1)) при m > 0 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    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 | | 2 | 2 | 7 | | 2 | 3 | 9 | | 3 | 0 | 5 | | 3 | 1 | 13 | | 3 | 2 | 29 | | 3 | 3 | 61 | | 4 | 0 | 13 | |

1. **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 | 1. Добавление элементов в стек |
|  |  |
| 1. Просмотр стека | 1. Удаление стека |
|  |  |
| 1. Удаление всех элементов, заканчивающихся на 5 | 1. Сортировка |
|  |  |

1. **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 DelList(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;

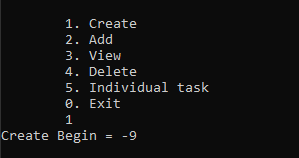
}

}

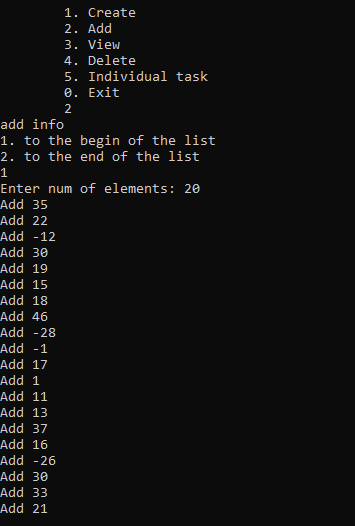
return 0;

}

1. Создание списка и первого элемента



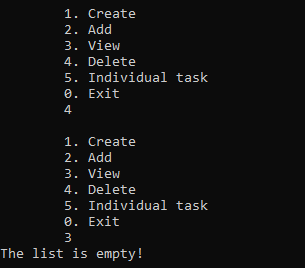
1. Добавить элементы в список (20 штук)



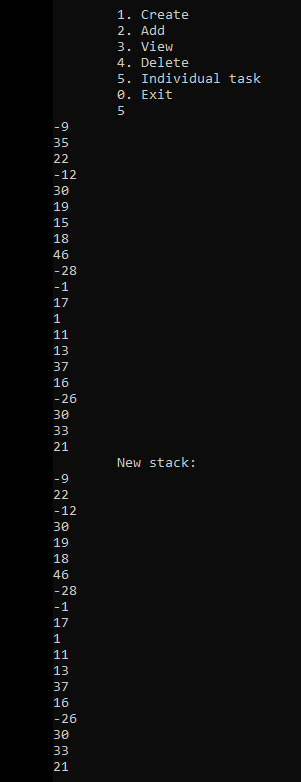
3) просмотр списка

|  |  |
| --- | --- |
| Просмотр с начала | Просмотр с конца |
|  |  |

1. Удалить список (удаление и последующий просмотр)



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



1. **Нелинейные списки (часть 2, лаб 6, стр 40)**

Определить число узлов на каждом уровне дерева.

1. **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;

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,\n 2 - 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()) { // считаем количество строк в файле

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 в массив структур

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) { // 1 - просмотр файла

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].code << " "

<< contact[index\_number].city << " "

<< contact[index\_number].time[0] << "."

<< contact[index\_number].time[1] << "."

<< contact[index\_number].time[2] << " "

<< contact[index\_number].tarif << " " << contact[index\_number].city\_num << " " << contact[index\_number].subscriber\_num << endl;

}

else if (answer3 == 4) {//time

cout

<< contact[index\_number].date[0] << "."

<< contact[index\_number].date[1] << "."

<< contact[index\_number].date[2] << " "

<< " " << contact[index\_number].code << " " << contact[index\_number].city

<< " " << temp\_time << " " << contact[index\_number].tarif << " "

<< contact[index\_number].city\_num << " " << contact[index\_number].subscriber\_num << endl;

}

else {

cout

<< contact[index\_number].date[0] << "."

<< contact[index\_number].date[1] << "."

<< contact[index\_number].date[2] << " "

<< 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].tarif << " " << contact[index\_number].city\_num

<< " " << contact[index\_number].subscriber\_num << endl;

}

cout << "Сохранить изменения в файле?\n1 - да,\n 2 - 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] << ".";

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

file

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

<< contact[i].code << " " << contact[i].city << " " << temp\_time << " "

<< contact[i].tarif << " " << contact[i].city\_num << " " << contact[i].subscriber\_num;

}

else {

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 << " ";

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;

}

if (i != N - 1) {

file << endl;

}

}

cout << "Данные в файле обновлены!" << endl << endl;

file.close();

}

else cout << "Данные в файле не были обновлены" << endl;

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,\n 2 - 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 << " ";

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;

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;

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;

LinearSearch(contact, N, task, count);

cout << endl << setw(6) << " code |" << setw(10) << " city |" << setw(8) << "tarif |" << setw(12) << "all time,s |" << setw(11) << "hh.mm.ss |" << setw(11) << "sum |" << endl << "----------------------------------------------------------- " << endl;

for (j = 0; j < count; j++) {

cout << setw(6) << task[j].code << "|"

<< setw(9) << task[j].city << "|"

<< setw(7) << task[j].tarif << "|"

<< setw(12) << task[j].time << "|";

if (task[j].time\_all[0] < 10)cout << setw(3) << "0" << task[j].time\_all[0] << ".";

else cout << setw(4) << task[j].time\_all[0] << ".";

if (task[j].time\_all[1] < 10)cout << "0" << task[j].time\_all[1] << ".";

else cout << task[j].time\_all[1] << ".";

if (task[j].time\_all[2] < 10)cout << "0" << task[j].time\_all[2] << "|";

else cout << task[j].time\_all[2] << "|";

cout << setw(10) << task[j].sum << "|" << endl;

}

cout << endl;

}

}

}

}

}

}

int main() {

setlocale(LC\_ALL, "RU");

system("chcp 1251");

int answer, N = 0, i = 0;

string file\_name;

while (1) {

cout << "\n-------- МЕНЮ -----------" << endl;

cout << "\n1. Создать файл\n2. Работа с существующим файлом\n3. Выход из программы" << endl;

cin >> answer;

if (answer == 3) exit(0); // выход из программы

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;

}