**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Лабораторная работа №2**

**по дисциплине «Языки программирования»**

Структурированный тип

Группа: **АВТ-615**

Студент: **Кузенков В.С.**

Преподаватель: **Балагуров М.В.**

НОВОСИБИРСК 2017

# Постановка задачи

Определить структурированный тип и набор функций для работы с таблицей записей, реализованной в массиве структур. В перечень функций входят:

·        ввод записи таблицы с клавиатуры;

·        загрузка и сохранение таблицы в текстовом файле;

·        просмотр таблицы;

·        сортировка таблицы в порядке возрастания заданного поля;

·        поиск в таблице элемента с заданным значением поля или с наиболее близким к нему по значению;

·        удаление записи;

·        изменение (редактирование) записи;

·        вычисление с проверкой и использованием всех записей по заданному условию и формуле (например, общая сумма на всех счетах).

7. Номер рейса, пункт назначения, время вылета, дата вылета, стоимость билета.

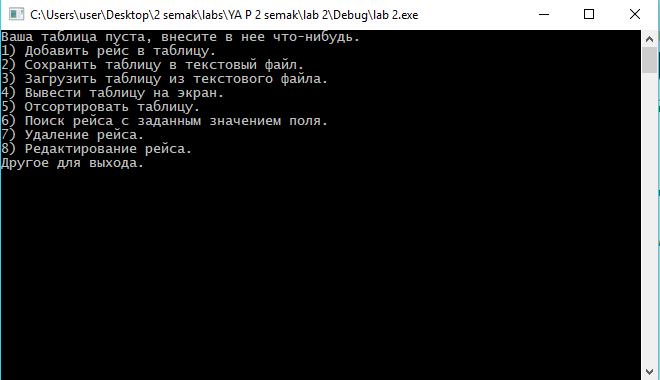
# Ограничения

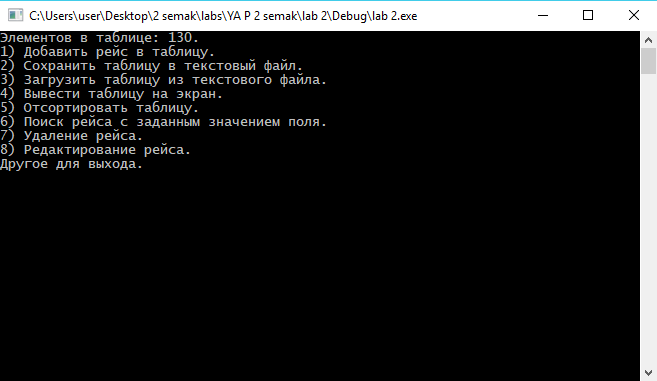
Отсутствуют

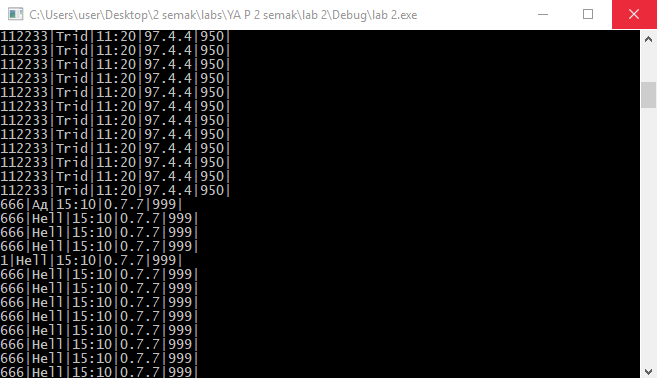
# Тестирование

**Входные данные**

**Выходные данные**







# Текст программы:

/\*Определить структурированный тип и набор функций для работы с таблицей записей, реализованной в массиве структур. В перечень функций входят:

· ввод записи таблицы с клавиатуры;

· загрузка и сохранение таблицы в текстовом файле;

· просмотр таблицы;

· сортировка таблицы в порядке возрастания заданного поля;

· поиск в таблице элемента с заданным значением поля или с наиболее близким к нему по значению;

· удаление записи;

· изменение (редактирование) записи;

· вычисление с проверкой и использованием всех записей по заданному условию и формуле(например, общая сумма на всех счетах).

Для: Номер рейса, пункт назначения, время вылета, дата вылета, стоимость билета.

\*/

#include <iostream>

#include <fstream>

#include <Windows.h>

#include <stdlib.h>

using namespace std;

struct Time //время

{

unsigned short hour;

unsigned short min;

void setTime(int h, int m)

{

this->hour = h%24;

this->min = m%60;

}

void showTime()

{

cout << this->hour << ':' << this->min << endl;

}

};

struct Date //дата

{

unsigned short year;

unsigned short mon;

unsigned short day;

void setDate(int y, int m, int d)

{

this->year = y;

this->mon = m%12;

this->day = d%31;

}

void showDate()

{

cout << this->year << '.' << this->mon << '.' << this->day << endl;

}

};

struct Flight //рейс

{

unsigned int flightNumber;

unsigned int ticketCost;

char destination[250];

Time departureTime;

Date departureDate;

bool empty = false;

void setFlight(int flightNumber, int ticketCost, char \*destination, Time departureTime, Date departureDate)

{

int i = 0;

this->flightNumber = flightNumber;

this->ticketCost = ticketCost;

this->departureTime = departureTime;

this->departureDate = departureDate;

for (; destination[i] != '\0';)

{

this->destination[i] = destination[i];

i++;

}

this->destination[i] = '\0';

}

void setDestination(char \*destination)

{

int i = 0;

for (; destination[i] != '\0';)

{

this->destination[i] = destination[i];

i++;

}

this->destination[i] = '\0';

}

void showFlight()

{

cout << this->flightNumber << "|" << this->destination << "|" << this->departureTime.hour << ":"

<< this->departureTime.min << "|"<< this->departureDate.year << "."

<< this->departureDate.mon << "." << this->departureDate.day << "|" << this->ticketCost << "|" << endl;

}

void fightToFile(ostream &fout)

{

fout << this->flightNumber << " " << this->destination << " " << this->departureTime.hour << ":"

<< this->departureTime.min << " " << this->departureDate.year << "."

<< this->departureDate.mon << "." << this->departureDate.day << " " << this->ticketCost << " " << endl;

}

};

struct Table //таблица указателей на рейс

{

unsigned int count = 0;

Flight \*flightTable = static\_cast<Flight\*>(malloc(sizeof(Flight)));

void saveTable() //сохранение таблицы в текстовом файле

{

if (count)

{

ofstream fout("table.txt");

for (unsigned int i = 0; i < count; ++i)

{

flightTable[i].fightToFile(fout);

}

}

else//если пусто

{

int wait;

cout << "Ваша таблица пуста, внесите в нее что-нибудь.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

menu(this);

};

void loadTable() //загрузка таблицы в текстовый файле

{

if (count)

{

//free(flightTable);

count = 0;

}

//Flight \*flightTable = static\_cast<Flight\*>(malloc(sizeof(Flight)));

ifstream fin("table.txt");

while (!fin.eof())

{

unsigned int flightNumber = 0;

unsigned int ticketCost = 0;

char destination[250] = "";

unsigned short year = 0;

unsigned short mon = 0;

unsigned short day = 0;

unsigned short hour = 0;

unsigned short min = 0;

Date date;

Time time;

char temp[255];

fin.getline(temp,255);

if (temp[0] == '\0') continue;

int spase = 0;

int destination\_i = 0;

int dot = 0;

int colon = 0;

for (int i = 0; temp[i] != 0;++i) //мучу костыль

{

if (temp[i] == ' ') {

spase++;

continue;

}

if (temp[i] == '.') {

dot++;

continue;

}

if (temp[i] == ':') {

colon++;

continue;

}

switch(spase)

{

case 0:

flightNumber \*= 10;

flightNumber += temp[i] - '0';

break;

case 1:

destination[destination\_i] = temp[i];

//cout << destination;

destination\_i++;

destination[destination\_i] += '\0';

break;

case 2:

if(colon)

{

min \*= 10;

min += temp[i] - '0';

}else

{

hour \*= 10;

hour += temp[i] - '0';

}

break;

case 3:

if (dot == 0)

{

year \*= 10;

year += temp[i] - '0';

}

if (dot == 1)

{

mon \*= 10;

mon += temp[i] - '0';

}

if (dot == 2)

{

day \*= 10;

day += temp[i] - '0';

}

break;

case 4:

ticketCost \*= 10;

ticketCost += temp[i] - '0';

break;

}

}

flightTable = static\_cast<Flight \*>(realloc(flightTable, sizeof(Flight)\*++count));

date.setDate(year, mon, day);

time.setTime(hour, min);

flightTable[count - 1].setFlight(flightNumber, ticketCost, destination, time, date);

//cout << destination;

//system("pause");

}

menu(this);

};

void showTable() //просмотр таблицы

{

if(count)

{

int wait;

system("cls");

cout << "Номер рейса|" << "Пункт назначения|" << "Время вылета|" << "Дата вылета|" << "Cтоимость билета $|" << endl;

for (unsigned int i = 0; i < count; ++i)

{

flightTable[i].showFlight();

}

cout << "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

else//если пусто

{

int wait;

cout << "Ваша таблица пуста, внесите в нее что-нибудь.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

menu(this);

};

void sortTable() //сортировка таблицы в порядке возрастания заданного поля

{

if (count)

{

unsigned short choose;

system("cls");

cout << "1) Сортировка по номеру рейса.\n"

<< "2) Сортировка по пункту назначения.\n"

<< "3) Сортировка по дате вылета.\n"

<< "4) Сортировка по цене.\n"

<< "0 для выхода в меню.\n";

cin >> choose;

if (choose > count) {

system("cls");

int wait;

cout << "Вы ввели не корректные данные.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

if (!choose)

menu(this);

switch (choose)

{

case 0: menu(this);

case 1:

{

Flight temp;

for (unsigned int i = 0; i < count - 1; ++i)

{

for (unsigned int j = 0; j < count - i - 1; j++) {

if (flightTable[j].flightNumber > flightTable[j + 1].flightNumber) {

// меняем элементы местами

temp = flightTable[j];

flightTable[j] = flightTable[j + 1];

flightTable[j + 1] = temp;

}

}

}

break;

}

case 2:

{

Flight temp;

for (unsigned int i = 0; i < count - 1; ++i)

{

for (unsigned int j = 0; j < count - i - 1; j++) {

if (strcmp(flightTable[j].destination, flightTable[j + 1].destination)>0) {

// меняем элементы местами

temp = flightTable[j];

flightTable[j] = flightTable[j + 1];

flightTable[j + 1] = temp;

}

}

}

break;

}

case 3:

{

Flight temp;

for (unsigned int i = 0; i < count - 1; ++i)

{

for (unsigned int j = 0; j < count - i - 1; j++) {

Date date1 = flightTable[j].departureDate;

Time time1 = flightTable[j].departureTime;

Date date2 = flightTable[j].departureDate;

Time time2 = flightTable[j].departureTime;

unsigned long full1 = date1.year\*525600+date1.mon\*44640+date1.day\*1440+time1.hour\*60+time1.min;

unsigned long full2 = date2.year \* 525600 + date2.mon \* 44640 + date2.day \* 1440 + time2.hour \* 60 + time2.min;

if (full1 > full2) {

// меняем элементы местами

temp = flightTable[j];

flightTable[j] = flightTable[j + 1];

flightTable[j + 1] = temp;

}

}

}

break;

}

case 4:

{

Flight temp;

for (unsigned int i = 0; i < count - 1; ++i)

{

for (unsigned int j = 0; j < count - i - 1; j++) {

if (flightTable[j].ticketCost > flightTable[j + 1].ticketCost) {

// меняем элементы местами

temp = flightTable[j];

flightTable[j] = flightTable[j + 1];

flightTable[j + 1] = temp;

}

}

}

break;

}

}

int wait;

cout << "Сортировка таблицы успешно завершена.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";;

cin >> wait;

}

else//если пусто

{

int wait;

cout << "Ваша таблица пуста, внесите в нее что-нибудь.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

menu(this);

};

void searchByEl() //поиск в таблице элемента с заданным значением поля или с наиболее близким к нему по значению

{

if (count)

{

unsigned short choose;

system("cls");

cout << "1) Поиск по номеру рейса.\n"

<< "2) Поиск по пункту назначения.\n"

<< "3) Поиск по дате вылета.\n"

<< "4) Поиск по цене.\n"

<< "0 для выхода в меню.\n";

cin >> choose;

if (choose > count) {

system("cls");

int wait;

cout << "Вы ввели не корректные данные.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

if (!choose)

menu(this);

switch(choose)

{

case 0: break;

case 1:

{

unsigned int num = 0;

cout << "Введите номер рейса\n";

cin >> num;

bool result = false;

//result.empty = true;

for (unsigned int i = 0; i < count; ++i)

{

if(flightTable[i].flightNumber == num)

{

int wait;

cout << "Ваш рейс:\n";

flightTable[i].showFlight();

cout << "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

result = true;

break;

}

}

if(!result)

{

int wait;

cout << "К сожалению мы ничего не нашли.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

break;

}

break;

}

case 2:

{

char destination[250];

cout << "Введите пункт назначения\n";

cin >> destination;

bool result = false;

//result.empty = true;

for (unsigned int i = 0; i < count; ++i)

{

if(!strcmp(destination, flightTable[i].destination))

{

int wait;

cout << "Ваш рейс:\n";

flightTable[i].showFlight();

cout << "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

result = true;

break;

}

}

if (!result)

{

int wait;

cout << "К сожалению мы ничего не нашли.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

break;

}

break;

}

case 3:

{

int year, mon, day;

int hour, min;

cout << "Введите через пробел: год, месяц, день, час, минуту\n";

cin >> year >> mon >> day >> hour >> min;

bool result = false;

//result.empty = true;

for (unsigned int i = 0; i < count; ++i)

{

if (flightTable[i].departureDate.year == year

&&flightTable[i].departureDate.mon == mon

&&flightTable[i].departureDate.day == day

&&flightTable[i].departureTime.hour == hour

&&flightTable[i].departureTime.min == min)

{

int wait;

cout << "Ваш рейс:\n";

flightTable[i].showFlight();

cout << "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

result = true;

break;

}

}

if (!result)

{

int wait;

cout << "К сожалению мы ничего не нашли.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

break;

}

break;

}

case 4:

{

unsigned int cost = 0;

cout << "Введите цену билеты\n";

cin >> cost;

bool result = false;

//result.empty = true;

for (unsigned int i = 0; i < count; ++i)

{

if (flightTable[i].ticketCost == cost)

{

int wait;

cout << "Ваш рейс:\n";

flightTable[i].showFlight();

cout << "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

result = true;

break;

}

}

if (!result)

{

int wait;

cout << "К сожалению мы ничего не нашли.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

break;

}

break;

}

}

}

else//если пусто

{

int wait;

cout << "Ваша таблица пуста, внесите в нее что-нибудь.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

menu(this);

};

void addAccount() //добавление записи

{

unsigned int flightNumber;

unsigned int ticketCost;

char destination[250] = "";

unsigned short year;

unsigned short mon;

unsigned short day;

unsigned short hour;

unsigned short min;

Date date;

Time time;

system("cls");

cout << "Введите номер рейса:\n";

cin >> flightNumber;

cout << "Введите стоимость билета:\n";

cin >> ticketCost;

cout << "Введите пункт назначения:\n";

cin >> destination;

cout << "Введите дату через пробел сначала день, затем месяц, затем год:\n";

cin >> day >> mon >> year;

cout << "Введите время рейса через пробел сначала часы, затем минуты:\n";

cin >> hour >> min;

flightTable = static\_cast<Flight \*>(realloc(flightTable, sizeof(Flight)\*++count));

date.setDate(year, mon, day);

time.setTime(hour, min);

flightTable[count - 1].setFlight(flightNumber,ticketCost,destination,time,date);

menu(this);

}

void deleteAccount() //удаление записи

{

if (count)

{

int wait;

unsigned int select = 0;

system("cls");

cout << "Номер рейса|" << "Пункт назначения|" << "Время вылета|" << "Дата вылета|" << "Cтоимость билета $|" << endl;

for (unsigned int i = 0; i < count; ++i)

{

flightTable[i].showFlight();

}

cout << "Введите порядковый номер строки записи, которую хотите удалить. 0 для выхода.\n";

cin >> select;

if (select == 0) menu(this);

if (select > count) {

system("cls");

//int wait;

cout << "Вы ввели не корректные данные.\n";

cout << "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}else{

count--;

for (unsigned int i = select-1; i < count;++i)

{

flightTable[i] = flightTable[i+1];

}

}

cout << "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

else//если пусто

{

int wait;

cout << "Ваша таблица пуста, внесите в нее что-нибудь.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

menu(this);

};

void editAccount() // редактирование записи

{

if (count)

{

int wait;

unsigned int num = 0;

unsigned int select = 0;

char destination[250];

int year, mon, day;

int hour, min;

unsigned int cost = 0;

system("cls");

cout << "Номер рейса|" << "Пункт назначения|" << "Время вылета|" << "Дата вылета|" << "Cтоимость билета $|" << endl;

for (unsigned int i = 0; i < count; ++i)

{

flightTable[i].showFlight();

}

cout << "Введите порядковый номер строки записи, которую хотите изменить. 0 для выхода.\n";

cin >> select;

if (select == 0) menu(this);

if (select > count) {

system("cls");

cout << "Вы ввели не корректные данные.\n";

}else{

system("cls");

cout << "Вы выбрали ";

flightTable[select-1].showFlight();

cout << "Какое поле желаете изменить?\n"

<< "1) Номер рейса\n"

<< "2) Пункт назначения\n"

<< "3) Время вылета\n"

<< "4) Дата вылета\n"

<< "5) Стоимость билета\n";

cin >> select;

switch(select)

{

case 1:

cout << "Введите новое значение\n";

cin >> num;

flightTable[select - 1].flightNumber = num;

break;

case 2:

cout << "Введите новое значение\n";

cin >> destination;

flightTable[select - 1].setDestination(destination);

break;

case 3:

cout << "Введите новое время рейса через пробел сначала часы, затем минуты:\n";

cin >> hour >> min;

flightTable[select - 1].departureTime.setTime(hour,min);

break;

case 4:

cout << "Введите новую дату через пробел сначала день, затем месяц, затем год:\n";

cin >> day >> mon >> year;

flightTable[select - 1].departureDate.setDate(year, mon, day);

break;

case 5:

cout << "Введите новое значение\n";

cin >> cost;

flightTable[select - 1].ticketCost = cost;

break;

}

}

cout << "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

else//если пусто

{

int wait;

cout << "Ваша таблица пуста, внесите в нее что-нибудь.\n"

<< "Для возврата в меню введите число и нажмите ввод.\n";

cin >> wait;

}

menu(this);

}

static int menu(Table \*table)

{

unsigned short choose = 0;

system("cls");

if (table->count)

{

cout << "Элементов в таблице: " << table->count << ".\n";

}

else

{

cout << "Ваша таблица пуста, внесите в нее что-нибудь.\n";

}

cout << "1) Добавить рейс в таблицу.\n"

<< "2) Сохранить таблицу в текстовый файл.\n"

<< "3) Загрузить таблицу из текстового файла.\n"

<< "4) Вывести таблицу на экран.\n"

<< "5) Отсортировать таблицу.\n"

<< "6) Поиск рейса с заданным значением поля.\n"

<< "7) Удаление рейса.\n"

<< "8) Редактирование рейса.\n"

<< "Другое для выхода.\n";

cin >> choose;

switch (choose)

{

case 1: table->addAccount(); break;

case 2: table->saveTable(); break;

case 3: table->loadTable(); break;

case 4: table->showTable(); break;

case 5: table->sortTable(); break;

case 6: table->searchByEl(); break;

case 7: table->deleteAccount(); break;

case 8: table->editAccount(); break;

default: return 0;

}

return 0;

}

};

int main()

{

//setlocale(0, "russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Table \*table = new Table;

table->menu(table);

system("pause");

return 0;

}

**Вывод:** Основываясь на практическом опыте было выяснено, что структурированная переменная в программе может выступать как «изображение» внешнего объекта, к которым работает программа. Это может быть физический объект внешнего мира, а может быть логический (программный) объект (например, файл). Структурированный тип в таком случае выступает как описание внешней сущности (или класса). Элементы (поля) структуры выступают как свойства объекта.