

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Стандартные типы данных для представления многоэлементных не однородных структур данных

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-06-22 |  | Кликушин В.И. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

ОГЛАВЛЕНИЕ

[1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА 3](#_Toc132712641)

[2 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc132712642)

[3 МОДЕЛЬ РЕШЕНИЯ 5](#_Toc132712643)

[3.1 СТРУКТУРА ЭЛЕМЕНТА ТАБЛИЦЫ 5](#_Toc132712644)

[3.2 СТРУКТУРА ТАБЛИЦЫ 5](#_Toc132712645)

[3.3 ОПРЕДЕЛЕНИЕ АТД 5](#_Toc132712646)

[3.4 КОД СТРУКТУР 8](#_Toc132712647)

[4 ТЕСТЫ 8](#_Toc132712648)

[5 РЕАЛИЗАЦИЯ ЗАДАНИЯ №1 11](#_Toc132712649)

[5.1 КОД ПРОГРАММЫ 11](#_Toc132712650)

[5.2 СКРИНЫ РЕЗУЛЬТАТА ТЕСТИРОВАНИЯ 19](#_Toc132712651)

[6 РЕАЛИЗАЦИЯ ЗАДАНИЯ №2 26](#_Toc132712652)

[6.2 КОД ПРОГРАММЫ 26](#_Toc132712653)

[6.2 СКРИНЫ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ 35](#_Toc132712654)

[7 ВЫВОДЫ 40](#_Toc132712655)

[8 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ 41](#_Toc132712656)

# 1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА

Дано множество из n элементов, тип элемента определен вариантом. Дан набор задач (операций), которые требуется выполнить над исходным множеством (таблицей). Набор задач определен в варианте задания. В определении структуры элемента определено поле, которое является ключом. Несколько элементов могут иметь одинаковые ключи. Разработать программу, которая управляет таблицей, выполняя операции варианта. Управление выбором операции осуществляется посредством текстового меню.

Задание варианта: Учет проданных на поезд билетов. Структура элемента: номер поезда, номер вагона, номер места, дата отправления, стоимость билета. Таблица хранит данные по всем вагонам поезда. Операции: 1) Добавить запись о продаже билета в таблицу; 2) Определить количество билетов, проданных на поезд заданного номера и дате отправления; 3) Удалить запись из таблицы по проданному билету; 4) Вывести сведения о количестве проданных билетов в каждый вагон поезда.

# 2 ПОСТАНОВКА ЗАДАЧИ

Необходимо изучить, как составлять таблицы из элементов неоднородных структур данных, а затем применить полученные знания для выполнения операций над таблицей.

Операции, необходимые для реализации задания: заполнение таблицы с клавиатуры; заполнение таблицы из файла; удаление элемента из таблицы; нахождение элемента в таблице; добавление нового элемента в таблицу; подсчёт количества элементов таблицы, удовлетворяющих определённому условию.

# 3 МОДЕЛЬ РЕШЕНИЯ

## 3.1 СТРУКТУРА ЭЛЕМЕНТА ТАБЛИЦЫ

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 1 – Структура элемента таблицы

## 3.2 СТРУКТУРА ТАБЛИЦЫ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Структура таблицы

## 3.3 ОПРЕДЕЛЕНИЕ АТД

АТД Tickets

{

Данные

train\_number – номер поезда;

car\_number – номер вагона;

seat\_number – номер места;

departure\_date – дата отправления;

ticket\_price – стоимость билета.

Операции

1. Заполнение данных элемента типа Tickets с клавиатуры

Предусловие. t – элемент типа Tickets.

Постусловие. Возвращает заполненный элемент t.

**input\_ticket(Tickets t);**

1. Отображение элемента множества

Предусловие. t – элемента типа Tickets.

Постусловие. Выводит данные об элементе. Нет возвращаемого значения.

**output\_ticket(Tickets t);**

АТД Table

{

Данные

n – количество записей в таблице

data – таблица проданных билетов;

Tickets – тип элементов таблицы.

Операции

1. Заполнение таблицы элементами с клавиатуры

Предусловие. arr – адрес таблицы типа Table, n – количество элементов в таблице.

Постусловие. Заполненная таблицы arr из n элементов. Возвращает 0, если число элементов в таблице корректное и удалось заполнить таблицу, и 1 в противном случае.

**input\_table(Table& arr, int n);**

1. Заполнение таблицы элементами из файла

Предусловие. file\_name – имя файла, arr – адрес на таблицу типа Table.

Постусловие. Заполненная таблицы arr из n элементов. Возвращает 0, если считывание из файла и заполнение таблицы произошли успешно, и 1 в противном случае.

**input\_from\_file(Table& arr, string file\_name);**

1. Вывод таблицы в консоль

Предусловие. arr – таблица типа Table.

Постусловие. Вывод значений таблицы. Возвращает 0, если таблица заполнена, и 1 в противном случае.

**output\_table(Table arr);**

1. Нахождение позиции билета в таблице по номеру поезда

Предусловие. arr – адрес таблицы типа Table.

Постусловие. Возвращает позицию элемента, чей номер поезда равен key, и -1, если такой элемент не найден.

**find\_train\_number(Table arr, int key);**

1. Добавление записи о продаже билета

Предусловие. arr – ссылка на таблицу типа Table, key – номер поезда.

Постусловие. Таблица arr с (n + 1) элементом, в конце стоит элемент, равный t. Возвращает 0, если в таблицу можно добавить новый элемент, и 1 в противном случае.

**append\_ticket(Table& arr, Tickets t);**

1. Определить количество билетов, проданных на поезд заданного номера и даты отправления.

Предусловие. arr – таблица типа Table, key – номер поезда, date – дата отправления.

Постусловие. Возвращает количество билетов в соответствие с условием.

**number\_sold\_tickets\_by\_train\_number(Table arr, int key, string date);**

1. Удалить запись по проданному билету

Предусловие. arr – адрес таблицы типа Table, key – номер поезда.

Постусловие. Возвращает 0, если удалось удалить запись, иначе 1.

**delete\_ticket(Table& arr, int key);**

1. Вывести сведения о количестве проданных билетов в каждый вагон поезда

Предусловие. arr – таблица типа Table, key – номер поезда

Постусловие. Возвращает 1, если не найдено записей, иначе выводит информацию о проданных билетах и выводит 0.

**number\_sold\_tickets\_by\_car\_number(Table arr, int key);**

1. Вставить элемент в заданную позицию

Предусловие. arr – адрес таблицы типа Table, pos – позиция вставляемого элемента, t – вставляемый элемент

Постусловие. Таблица arr с (n+1) элементами, если удалось вставить элемент. Возвращает 0, если в таблицу можно вставить новый элемент, и 1 в противном случае.

**insert\_ticket(Table& arr, int pos, Tickets t);**

## 3.4 КОД СТРУКТУР

Код Tickets:

struct Tickets

{

int train\_number; // номер поезда

int car\_number; // номер вагона

int seat\_number; // номер места

string departure\_date; // дата отправления

int ticket\_price; // стоимость билета

Tickets();

};

Код Table:

struct Table

{

int n;

Tickets data[N];

Table();

};

# 4 ТЕСТЫ

Таблица 1. Таблица тестирования функции append\_ticket

|  |  |  |
| --- | --- | --- |
| Добавить запись о продаже билета в таблицу  append\_ticket(Table& arr, Tickets t); | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | arr.n = 4  arr.data = {elem1, elem2, elem3, elem4}  t | arr.n = 5  arr.data = {elem1, elem2, elem3, elem4, t}  Код возврата: 0 |
| 2 (для массива) | arr.n = N  arr.data = {elem1, …, elemN} | Код возврата: 1 |
| 3 | arr.n = 0  t | arr.data = {t}  Код возврата: 0 |

Таблица 2. Таблица тестирования функции number\_sold\_tickets\_by\_train\_number

|  |  |  |
| --- | --- | --- |
| Определить количество билетов, проданных на поезд заданного номера и даты отправления  number\_sold\_tickets\_by\_train\_number(Table arr, int key, string date); | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | arr.n = 5  t.table = {elem1(14), elem2 (7), elem3(91), elem4(2), elem5(10)}  key = 10 | count = 1 |
| 2 | arr.n = 3  arr.data = {elem1(1), elem2(2), elem3(1)} (разные даты при одинаковых ключах)  key = 1 | count = 1 |

Таблица 3. Таблица тестирования функции delete\_ticket

|  |  |  |
| --- | --- | --- |
| Удалить запись из таблицы по проданному билету  delete\_ticket(Table& arr, int key); | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | arr.n = 3  arr.data = {elem1 (7), elem2 (6), elem3(18)}  key = 18 | arr.n = 2  arr.data = {elem1, elem2}  Код возврата: 0 |
| 2 | arr.n = 1  arr.data = {elem1 (5)}  key = 5 | arr.n = 0  arr.data = {}  Код возврата: 0 |
| 3 | arr.n = 0 | Код возврата: 1 |

Таблица 4. Таблица тестирования функции number\_sold\_tickets\_by\_car\_number

|  |  |  |
| --- | --- | --- |
| Вывести сведения о количестве проданных билетов в каждый вагон поезда number\_sold\_tickets\_by\_car\_number(Table arr, int key); | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | arr.n = 5  arr.data = {elem1 (14), elem2 (7), elem3 (91), elem4 (2), elem5(10)}  key = 10 | Код работы: 0 |
| 2 | arr.n = 2  arr.data = {elem1 (3), elem2 (3)}  key = 3 | Код работы: 0 |

# 5 РЕАЛИЗАЦИЯ ЗАДАНИЯ №1

## 5.1 КОД ПРОГРАММЫ

Код файла Tickets.h:

#pragma once

#include <string>

using namespace std;

struct Tickets

{

int train\_number; // номер поезда

int car\_number; // номер вагона

int seat\_number; // номер места

string departure\_date; // дата отправления

int ticket\_price; // стоимость билета

Tickets();

};

Tickets input\_ticket(Tickets t);

void output\_ticket(Tickets t);

Код файла Tickets.cpp:

#include "Tickets.h"

#include <iostream>

using namespace std;

Tickets::Tickets()

{

train\_number = 0;

car\_number = 0;

seat\_number = 0;

departure\_date = "";

ticket\_price = 0;

}

Tickets input\_ticket(Tickets t)

{

cout << " Введите номер поезда: ";

cin >> t.train\_number;

cout << " Введите номер вагона: ";

cin >> t.car\_number;

cout << " Введите номер места: ";

cin >> t.seat\_number;

cout << " Введите дату отправления: ";

cin >> t.departure\_date;

cout << " Введите стоимость билета: ";

cin >> t.ticket\_price;

return t;

}

void output\_ticket(Tickets t)

{

cout << " Номер поезда: " << t.train\_number << endl << " Номер вагона: " << t.car\_number << endl << " Номер места: " << t.seat\_number << endl << " Дата отправления: " << t.departure\_date << endl << " Стоимость билета: " << t.ticket\_price << endl;

}

Код файла Table.h:

#pragma once

#include "Tickets.h"

const int N = 100;

struct Table

{

int n;

Tickets data[N];

Table();

};

int input\_from\_file(Table& arr, string file\_name);

int input\_table(Table& arr, int n);

int output\_table(Table arr);

int find\_train\_number(Table arr, int key);

int append\_ticket(Table& arr, Tickets t);

int number\_sold\_tickets\_by\_train\_number(Table arr, int key, string date);

int delete\_ticket(Table& arr, int key);

int insert\_ticket(Table& arr, int pos, Tickets t);

int number\_sold\_tickets\_by\_car\_number(Table arr, int key);

Код файла Table.cpp:

#include "Table.h"

#include <iostream>

#include <fstream>

using namespace std;

Table::Table()

{

n = 0;

}

int input\_table(Table& arr, int n)

{

if (n >= 0 and n <= N)

{

arr.n = 0;

for (int i = 0; i < n; i++)

{

cout << " Введите информацию о(об) " << (i + 1) << " - ом(ем) проданном билете:\n";

arr.data[i] = input\_ticket(arr.data[i]);

arr.n++;

}

return 0;

}

return 1;

}

int input\_from\_file(Table& arr, string file\_name)

{

ifstream itf(file\_name);

if (itf.is\_open())

{

int i = 0;

while (!itf.eof())

{

itf >> arr.data[i].train\_number >> arr.data[i].car\_number >> arr.data[i].seat\_number >> arr.data[i].departure\_date >> arr.data[i].ticket\_price;

i++;

}

arr.n = i;

itf.close();

return 0;

}

return 1;

}

int output\_table(Table arr)

{

if (arr.n > 0 and arr.n <= N)

{

cout << "---------------------------------------------------------------------------------------------------------------|\n";

cout << "Номер поезда" << " | " << "Номер вагона" << " | " << "Номер места" << " | " << "Дата отправления" << " | " << "Стоимость билета" << " |" << "\n";

cout << "-----------------|-------------------|---------------------|--------------------------|------------------------" << "|" << endl;

for (int i = 0; i < arr.n; i++)

{

cout.width(10);

cout << arr.data[i].train\_number;

cout.width(8);

cout << "|";

cout.width(10);

cout << arr.data[i].car\_number;

cout.width(10);

cout << "|";

cout.width(12);

cout << arr.data[i].seat\_number;

cout.width(10);

cout << "|";

cout.width(18);

cout << arr.data[i].departure\_date;

cout.width(9);

cout << "|";

cout.width(14);

cout << arr.data[i].ticket\_price;

cout.width(11);

cout << "|" << endl;

cout << "---------------------------------------------------------------------------------------------------------------|\n";

}

return 0;

}

return 1;

}

int find\_train\_number(Table arr, int key)

{

for (int i = 0; i < arr.n; i++)

{

if (arr.data[i].train\_number == key)

{

return i;

}

}

return -1;

}

int append\_ticket(Table& arr, Tickets t)

{

if (arr.n >= 0 and arr.n < N)

{

arr.data[arr.n++] = t;

return 0;

}

return 1;

}

int number\_sold\_tickets\_by\_train\_number(Table arr, int key,string date)

{

int count = 0;

for (int i = 0; i < arr.n; i++)

{

if (arr.data[i].train\_number == key and arr.data[i].departure\_date == date)

{

count++;

}

}

return count;

}

int delete\_ticket(Table& arr, int key)

{

if (arr.n > 0 and arr.n <= N)

{

int pos = find\_train\_number(arr, key);

if (pos != -1)

{

for (int i = pos + 1; i < arr.n; i++)

{

arr.data[i-1] = arr.data[i];

}

arr.n--;

return 0;

}

return 1;

}

return 1;

}

int insert\_ticket(Table& arr, int pos, Tickets t)

{

if (pos >= 0 and pos <= arr.n)

{

for (int i = arr.n; i > pos; i--)

{

arr.data[i] = arr.data[i - 1];

}

arr.data[pos] = t;

arr.n++;

return 0;

}

return 1;

}

int number\_sold\_tickets\_by\_car\_number(Table arr, int key)

{

int temp[512]{0};

if (arr.n > 0 and arr.n <= N)

{

for (int i = 0; i < arr.n; i++)

{

if (arr.data[i].train\_number == key)

{

temp[arr.data[i].car\_number] ++;

}

}

for (int i = 0; i < 500; i++)

{

if (temp[i] > 0)

{

cout << " В вагон номер " << i << " было продано " << temp[i] << " билетов\n";

}

}

return 0;

}

return 1;

}

Код файла siaod\_5:

#include <iostream>

#include "Table.h"

#include "Tickets.h"

using namespace std;

int main()

{

system("chcp 1251");

Table arr;

int num;

string date;

while (true)

{

cout << "-----------------------------------------------------------------------------------------------------------" << endl;

cout << " Меню\n";

cout << " 1. Заполнить информацию о проданных билетах с клавиатуры\n";

cout << " 2. Заполнить информацию о проданных билетах из файла\n";

cout << " 3. Вывести таблицу проданных билетов\n";

cout << " 4. Добавить запись о продаже билета\n";

cout << " 5. Определить количество билетов, проданных на поезд заданного номера и дате отправления\n";

cout << " 6. Удалить запись о продаже билета\n";

cout << " 7. Вывести сведения о количестве проданных билетов в каждый вагон поезда\n";

cout << " 8. Вставить запись о продаже билета\n";

cout << " 9. Завершение работы программы\n";

cin >> num;

switch (num)

{

case 1:

cout << " Введите количество проданных билетов\n";

cin >> num;

if (input\_table(arr, num) == 1)

{

cout << " Неверное количество проданных билетов\n";

}

break;

case 2:

if (input\_from\_file(arr, "Tickets.txt") == 1)

{

cout << " Произошла ошибка при чтении из файла\n";

}

else

{

output\_table(arr);

}

break;

case 3:

if (output\_table(arr) == 1)

{

cout << " Пустая таблица\n";

}

break;

case 4:

if (append\_ticket(arr, input\_ticket(Tickets())) == 1)

{

cout << " Невозможно добавить запись о продаже билета\n";

}

else

{

output\_table(arr);

}

break;

case 5:

cout << " Введите номер поезда: ";

cin >> num;

cout << " Введите дату отправления: ";

cin >> date;

if (number\_sold\_tickets\_by\_train\_number(arr,num,date) == 0)

{

cout << " Не найдено билетов, проданных на поезд с заданным номером и датой отправления\n";

}

else

{

cout << " Было продано " << number\_sold\_tickets\_by\_train\_number(arr, num, date) << " билетов на поезд с заданным номером и датой отправления" << endl;

}

break;

case 6:

cout << " Введите номер поезда, для которого будет удалена запись о проданном билете\n";

cin >> num;

if (delete\_ticket(arr, num) == 1)

{

cout << " Не удалось удалить запись о билете по заданному номеру поезда\n";

}

else

{

output\_table(arr);

}

break;

case 7:

cout << " Введите номер поезда, для которого нужно найти количество проданных билетов в каждый вагон\n";

cin >> num;

if (number\_sold\_tickets\_by\_car\_number(arr, num) == 1)

{

cout << " Не найдено записей по продаже билетов в вагоны поезда с заданным номером\n";

}

break;

case 8:

cout << " Введите позицию вставляемой записи о продаже билета\n";

cin >> num;

if (insert\_ticket(arr, num, input\_ticket(Tickets())) == 1)

{

cout << " Невозможно вставить элемент\n";

}

else

{

output\_table(arr);

}

break;

case 9:

return 0;

default:

cout << " Нет такого пункта\n";

}

}

return 0;

}

## 5.2 СКРИНЫ РЕЗУЛЬТАТА ТЕСТИРОВАНИЯ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Результат теста 1.1.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Результат теста 1.1.2

Изображение выглядит как текст

Автоматически созданное описание Рисунок 5 – Результат теста 1.1.3

Изображение выглядит как текст

Автоматически созданное описание Рисунок 6 – Результат теста 1.2.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Результат теста 1.2.2

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 8 – Результат теста 1.3.1

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 9 – Результат теста 1.3.2

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 10 – Результат теста 1.3.3

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 11 – Результат теста 1.4.1

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 12 – Результат теста 1.4.2

# 6 РЕАЛИЗАЦИЯ ЗАДАНИЯ №2

## 6.2 КОД ПРОГРАММЫ

Код Tickets.h:

#pragma once

#include <string>

using namespace std;

struct Tickets

{

int train\_number; // номер поезда

int car\_number; // номер вагона

int seat\_number; // номер места

string departure\_date; // дата отправления

int ticket\_price; // стоимость билета

Tickets();

};

Tickets input\_ticket(Tickets t);

void output\_ticket(Tickets t);

Код Tickets.cpp:

#include "Tickets.h"

#include <iostream>

using namespace std;

Tickets::Tickets()

{

train\_number = 0;

car\_number = 0;

seat\_number = 0;

departure\_date = "";

ticket\_price = 0;

}

Tickets input\_ticket(Tickets t)

{

cout << " Введите номер поезда: ";

cin >> t.train\_number;

cout << " Введите номер вагона: ";

cin >> t.car\_number;

cout << " Введите номер места: ";

cin >> t.seat\_number;

cout << " Введите дату отправления: ";

cin >> t.departure\_date;

cout << " Введите стоимость билета: ";

cin >> t.ticket\_price;

return t;

}

void output\_ticket(Tickets t)

{

cout << " Номер поезда: " << t.train\_number << endl << " Номер вагона: " << t.car\_number << endl << " Номер места: " << t.seat\_number << endl << " Дата отправления: " << t.departure\_date << endl << " Стоимость билета: " << t.ticket\_price << endl;

}

Код Table.h:

#pragma once

#include "Tickets.h"

struct Table

{

int n;

vector <Tickets> data;

Table();

};

int input\_from\_file(Table& arr, string file\_name);

int input\_table(Table& arr, int n);

int output\_table(Table arr);

int find\_train\_number(Table arr, int key);

int append\_ticket(Table& arr, Tickets t);

int number\_sold\_tickets\_by\_train\_number(Table arr, int key, string date);

int delete\_ticket(Table& arr, int key);

int insert\_ticket(Table& arr, int pos, Tickets t);

int number\_sold\_tickets\_by\_car\_number(Table arr, int key);

Код Table.cpp:

#include "Table.h"

#include <iostream>

#include <fstream>

using namespace std;

Table::Table()

{

data = vector <Tickets> ();

}

int input\_table(Table& arr, int n)

{

arr.data.clear();

for (int i = 0; i < n; i++)

{

cout << " Введите информацию о(об) " << (i + 1) << " - ом(ем) проданном билете:\n";

arr.data.push\_back(input\_ticket(Tickets()));

}

return 0;

}

int input\_from\_file(Table& arr, string file\_name)

{

ifstream itf(file\_name);

if (itf.is\_open())

{

arr.data.clear();

int i = 0;

while (!itf.eof())

{

Tickets t;

itf >> t.train\_number >> t.car\_number >> t.seat\_number >> t.departure\_date >> t.ticket\_price;

arr.data.push\_back(t);

}

itf.close();

return 0;

}

return 1;

}

int output\_table(Table arr)

{

if (arr.data.size() > 0)

{

cout << "---------------------------------------------------------------------------------------------------------------|\n";

cout << "Номер поезда" << " | " << "Номер вагона" << " | " << "Номер места" << " | " << "Дата отправления" << " | " << "Стоимость билета" << " |" << "\n";

cout << "-----------------|-------------------|---------------------|--------------------------|------------------------" << "|" << endl;

for (int i = 0; i < arr.data.size(); i++)

{

cout.width(10);

cout << arr.data[i].train\_number;

cout.width(8);

cout << "|";

cout.width(10);

cout << arr.data[i].car\_number;

cout.width(10);

cout << "|";

cout.width(12);

cout << arr.data[i].seat\_number;

cout.width(10);

cout << "|";

cout.width(18);

cout << arr.data[i].departure\_date;

cout.width(9);

cout << "|";

cout.width(14);

cout << arr.data[i].ticket\_price;

cout.width(11);

cout << "|" << endl;

cout << "---------------------------------------------------------------------------------------------------------------|\n";

}

return 0;

}

return 1;

}

int find\_train\_number(Table arr, int key)

{

for (int i = 0; i < arr.data.size(); i++)

{

if (arr.data[i].train\_number == key)

{

return i;

}

}

return -1;

}

int append\_ticket(Table& arr, Tickets t)

{

arr.data.push\_back(t);

return 0;

}

int number\_sold\_tickets\_by\_train\_number(Table arr, int key, string date)

{

int count = 0;

for (int i = 0; i < arr.data.size(); i++)

{

if (arr.data[i].train\_number == key and arr.data[i].departure\_date == date)

{

count++;

}

}

return count;

}

int delete\_ticket(Table& arr, int key)

{

if (arr.data.size()!=0)

{

int pos = find\_train\_number(arr, key);

if (pos != -1)

{

arr.data.erase(arr.data.begin() + pos);

return 0;

}

return 1;

}

return 1;

}

int insert\_ticket(Table& arr, int pos, Tickets t)

{

if (pos >= 0 and pos <= arr.data.size())

{

arr.data.insert(arr.data.begin() + pos, t);

return 0;

}

return 1;

}

int number\_sold\_tickets\_by\_car\_number(Table arr, int key)

{

const int N = 500;

int temp[N]{ 0 };

if (arr.data.size()!=0)

{

for (int i = 0; i < arr.data.size(); i++)

{

if (arr.data[i].train\_number == key)

{

temp[arr.data[i].car\_number] ++;

}

}

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

{

if (temp[i] > 0)

{

cout << " В вагон номер " << i << " было продано " << temp[i] << " билетов\n";

}

}

return 0;

}

return 1;

}

Код siaod\_5\_vector.cpp:

#include <iostream>

#include "Table.h"

#include "Tickets.h"

using namespace std;

int main()

{

system("chcp 1251");

Table arr;

int num;

string date;

while (true)

{

cout << "-----------------------------------------------------------------------------------------------------------" << endl;

cout << " Меню\n";

cout << " 1. Заполнить информацию о проданных билетах с клавиатуры\n";

cout << " 2. Заполнить информацию о проданных билетах из файла\n";

cout << " 3. Вывести таблицу проданных билетов\n";

cout << " 4. Добавить запись о продаже билета\n";

cout << " 5. Определить количество билетов, проданных на поезд заданного номера и дате отправления\n";

cout << " 6. Удалить запись о продаже билета\n";

cout << " 7. Вывести сведения о количестве проданных билетов в каждый вагон поезда\n";

cout << " 8. Вставить запись о продаже билета\n";

cout << " 9. Завершение работы программы\n";

cin >> num;

switch (num)

{

case 1:

cout << " Введите количество проданных билетов\n";

cin >> num;

if (input\_table(arr, num) == 1)

{

cout << " Неверное количество проданных билетов\n";

}

break;

case 2:

if (input\_from\_file(arr, "Tickets.txt") == 1)

{

cout << " Произошла ошибка при чтении из файла\n";

}

else

{

output\_table(arr);

}

break;

case 3:

if (output\_table(arr) == 1)

{

cout << " Пустая таблица\n";

}

break;

case 4:

if (append\_ticket(arr, input\_ticket(Tickets())) == 1)

{

cout << " Невозможно добавить запись о продаже билета\n";

}

else

{

output\_table(arr);

}

break;

case 5:

cout << " Введите номер поезда: ";

cin >> num;

cout << " Введите дату отправления: ";

cin >> date;

if (number\_sold\_tickets\_by\_train\_number(arr, num, date) == 0)

{

cout << " Не найдено билетов, проданных на поезд с заданным номером и датой отправления\n";

}

else

{

cout << " Было продано " << number\_sold\_tickets\_by\_train\_number(arr, num, date) << " билетов на поезд с заданным номером и датой отправления" << endl;

}

break;

case 6:

cout << " Введите номер поезда, для которого будет удалена запись о проданном билете\n";

cin >> num;

if (delete\_ticket(arr, num) == 1)

{

cout << " Не удалось удалить запись о билете по заданному номеру поезда\n";

}

else

{

output\_table(arr);

}

break;

case 7:

cout << " Введите номер поезда, для которого нужно найти количество проданных билетов в каждый вагон\n";

cin >> num;

if (number\_sold\_tickets\_by\_car\_number(arr, num) == 1)

{

cout << " Не найдено записей по продаже билетов в вагоны поезда с заданным номером\n";

}

break;

case 8:

cout << " Введите позицию вставляемой записи о продаже билета\n";

cin >> num;

if (insert\_ticket(arr, num, input\_ticket(Tickets())) == 1)

{

cout << " Невозможно вставить элемент\n";

}

else

{

output\_table(arr);

}

break;

case 9:

return 0;

default:

cout << " Нет такого пункта\n";

}

}

return 0;

}

## 6.2 СКРИНЫ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 – Результат теста 2.1.1

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 14 – Результат теста 2.1.3

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 15 – Результат теста 2.2.1

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 16 – Результат теста 2.2.2

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 17 – Результат теста 2.3.1

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 18 – Результат теста 2.3.2

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 19 – Результат теста 2.3.3

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 20 – Результат теста 2.4.1

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 21 – Результат теста 2.4.2

# 7 ВЫВОДЫ

В ходе выполнения практической работы были получены навыки работы с неоднородными структурами данных – таблицами, построенными на статическом массиве и векторе. Были освоены навыки построения таблиц и выполнения различных операция поиска, вставки, удаления элемента.

# 8 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Учебно-методическое пособие СиАОД (часть 1)
2. Приложение к практическим работам – СДО (online-edu.mirea.ru)