|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 4** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Структуризация многоэлементных структур средствами struct»** | |
|  | |
| Выполнил студент группы ИКБО-08-22 | Сенькевич Г.Д. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2023

# **Цели работы**

* получение навыков по реализации многоэлементных структур данных задачи С++;
* получение навыков по реализации многоэлементных данных средствами структуры данных – таблица.

# **Постановка задачи**

1. Разработать набор операций для управления таблицей, созданной на основе динамического массива. Структура записи элемента таблицы определена вариантом индивидуального задания.
   1. Создать проект, добавить заголовочный файл. Включить в заголовочный файл:
      1. разработанный тип данных, определяющий структуру элемента таблицы (записи).
      2. определенная на базе типа struct структура хранения данных по таблице, содержащая: размер таблицы – n, массив из n элементов типа записи варианта.
      3. реализованная функция вывода таблицы на экран.
      4. операции варианта задания, оформленные как функции с параметрами.
   2. Для заполнения отдельной записи с клавиатуры разработать функцию, которая принимает в качестве параметра пустую запись, а возвращает заполненную.
   3. Наполнение таблицы данными должна выполнять функция вставки или добавления записи, включенная в список операций варианта. Эта функция должна приять через параметр заполненную запись, выполнить действие по вставке или добавления, согласно алгоритму варианта этой операции.
2. Разработать программу, демонстрирующую работу всех функций с массивом или записей.
3. Разработать набор тестовых данных по наполнению таблицы. Включить в меню программы возможность автоматического ввода разработанных тестовых данных.
4. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Вариант №23. Условие задания:

|  |  |
| --- | --- |
| Упражнение 1 | Касса кинотеатра. Структура записи о проданном месте: название фильма, дата, время, номер ряда, номер места, номер зала (1-малый, 2-большой). Справочная информация о количестве мест в зале кинотеатра: массив с указанием количества мест всего в каждом зале.  Операции   1. Заполнение записи по одному проданному билету с клавиатуры. 2. Вставить запись в таблицу, упорядочивая по номеру зала (сортировку не использовать). Новая запись вставляется в начало подсписка по залу. 3. Удалить записи по заданной дате. 4. Определить количество свободных мест на заданный фильм в указанную дату и время. |

# **Решение**

Структура – тип, определяемый пользователем. Этот тип данных используется для представления в программе объектов, обладающих несколькими свойствами возможно различного типа. Структура позволяет представить объект, обладающий несколькими свойствами, одной переменной.

Структуры могут быть использованы для описания объектов реального мира, таких как студенты, автомобили, компьютеры и т.д. Каждый объект может быть представлен как структура, состоящая из нескольких элементов, таких как имя, возраст, марка и т.д.

Структуры в C++ могут быть вложенными, то есть структура может содержать в себе другую структуру. Это позволяет создавать более сложные типы данных, такие как связанные списки или деревья.

С помощью структур также можно создавать массивы структур, что позволяет обрабатывать группы данных как единое целое. В том числе таким образом можно представить таблицу.

Таблица – это массив, элемент которого имеет сложную структуру, реализуемую в языке С++ посредством типа struct. Элемент таблицы определяет объект предметной области, который характеризуется несколькими свойствами различного типа.

Кроме того, структуры в C++ могут иметь функции-члены, которые являются функциями, определенными внутри структуры, и работают с ее элементами. Такие функции могут быть полезны для обработки данных, хранящихся в структуре.

Для реализации первой операции была написана функция read\_sit\_info, считывающая информацию о купленном билете с клавиатуры. Функция принимает на вход ссылку на экземпляр структуры Sit и ничего не возвращает. В качестве вспомогательных были также написаны функции read\_movie\_name, read\_date и read\_time, которые считывают с клавиатуры название фильма, дату и время соответственно.

|  |
| --- |
| void read\_movie\_name(char movie\_name[MAX\_MOVIE\_NAME\_LENGTH])  {  std::cout << "Введите название фильма: ";  std::cin >> movie\_name;  }  void read\_date(Date& date)  {  std::cout << "Введите дату (день, месяц и год через пробел): ";  std::cin >> date.day >> date.month >> date.year;  }  void read\_time(Time& time)  {  std::cout << "Введите время (часы, минуты и секунды через пробел): ";  std::cin >> time.hours >> time.minutes >> time.seconds;  }  void read\_sit\_info(Sit& sit)  {  read\_movie\_name(sit.movie\_name);  read\_date(sit.date);  read\_time(sit.time);  std::cout << "Введите место в зале (номер ряда и номер места через пробел): ";  std::cin >> sit.row >> sit.place;  std::cout << "Введите номер зала (маленький - 1, большой - 2): ";  std::cin >> sit.room;  } |

Для реализации второй операции была написана функция Sits::insert, вставляющая информацию о купленном билете в таблицу. Функция принимает на вход экземпляр структуры Sit и ничего не возвращает.

|  |
| --- |
| void Sits::insert(Sit sit)  {  if (size == 0)  {  sits = (Sit\*) malloc(sizeof(\*sits) \* (size + 1));  }  else  {  sits = (Sit\*) realloc(sits, sizeof(\*sits) \* (size + 1));  }  size++;  if (sit.room == Room::big)  {  for (int i = size - 1; i > first\_in\_big\_room; i--)  {  sits[i] = sits[i - 1];  }  sits[first\_in\_big\_room] = sit;  }  else if (sit.room == Room::small)  {  for (int i = size - 1; i > 0; i--)  {  sits[i] = sits[i - 1];  }  first\_in\_big\_room++;  sits[0] = sit;  }  } |

Для реализации третьей операции была написана функция Sits::delete\_by\_date, удаляющая записи в таблице по заданной дате. Функция принимает на вход экземпляр структуры Date и ничего не возвращает.

|  |
| --- |
| void Sits::delete\_by\_date(Date date)  {  for (int i = 0; i < size; i++)  {  if (sits[i].date == date)  {  for (int j = i; j < size; j++)  {  sits[j] = sits[j + 1];  }  sits = (Sit\*) realloc(sits, sizeof(\*sits) \* (size - 1));  size--;  if (i < first\_in\_big\_room)  {  first\_in\_big\_room--;  }  i--;  }  }  } |

Для реализации четвёртой операции была написана функция Sits::free\_places, вычисляющая количество свободных мест на заданный фильм и заданные дату и время. Функция принимает на вход экземпляр строку (char[]), структуры Date, экземпляр структуры Time и возвращает целое число – количество свободных мест.

|  |
| --- |
| int Sits::free\_places(char movie\_name[MAX\_MOVIE\_NAME\_LENGTH], Date date, Time time)  {  int not\_free = 0;  for (int i = 0; i < size; i++)  {  if (strcmp(sits[i].movie\_name, movie\_name) == 0&&  sits[i].date == date &&  sits[i].time == time)  {  not\_free++;  }  }  return RoomsCapacities[Room::small] +  RoomsCapacities[Room::big] - not\_free;  } |

При запуске программы пользователь видит пользовательское меню, позволяющее выбрать между вставкой записи о купленном билете, удалением записи по дате и проверке количества свободных мест на определённый сеанс.

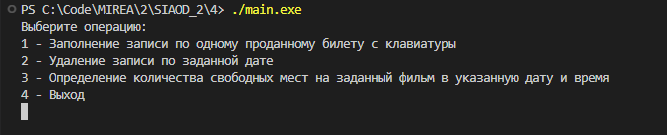


Рисунок 1. Интерфейс программы

# **Тестирование**

Протестируем выполнение программой первой и второй операций. Для этого выберем в меню заполнение информации о проданной билете с клавиатуры (1) и введём информацию о фильме. На рисунке 2 видим, как программа вывела верный ответ.

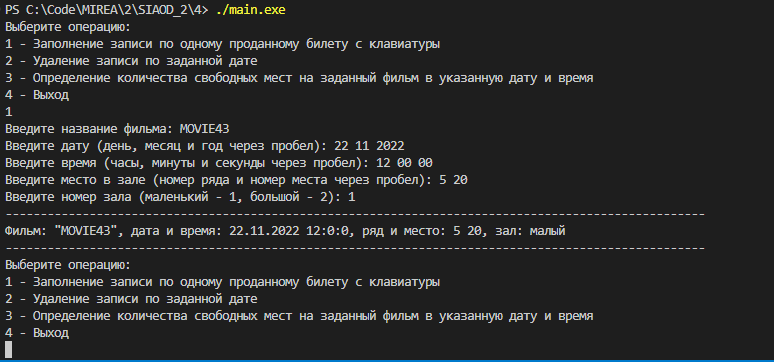


Рисунок 2. Тестирование программы

Протестируем выполнение программой третьей операции. Для этого выберем в меню удаление записи по заданной дате (2) и введём дату только что добавленного сеанса. На рисунке 3 видим, как программа вывела верный ответ.

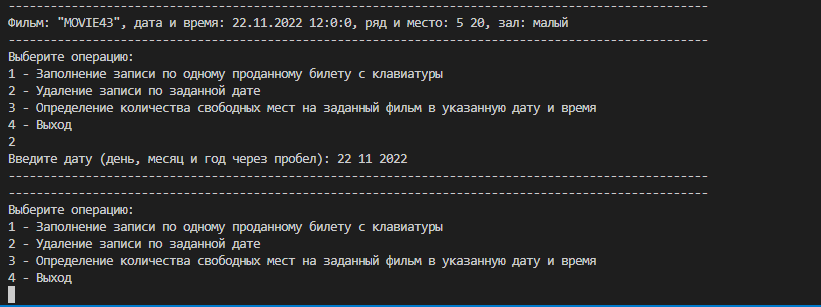


Рисунок 3. Тестирование программы

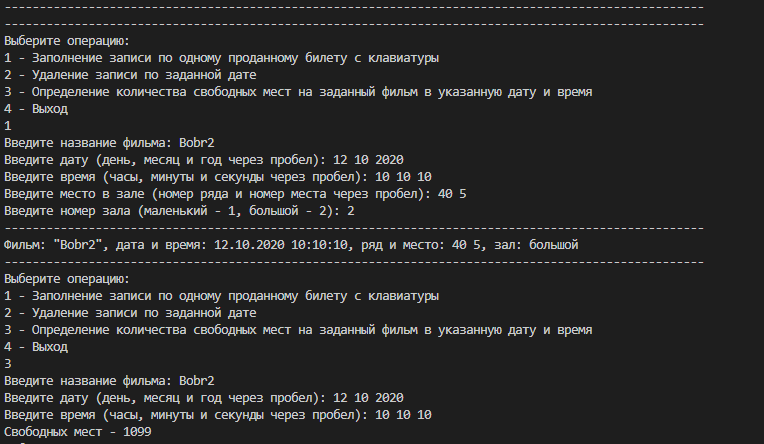
Протестируем выполнение программой четвёртой операции. Для этого сначала снова добавим один сеанс, а потом выберем в меню определение количества свободных мест на заданный фильм (3) и введём дату только что добавленного сеанса. На рисунке 4 видим, как программа вывела верный ответ. 

Рисунок 4. Тестирование программы

Из результатов выполнения программы видно, что программа работает корректно, решая все поставленные задачи

# **Вывод**

В результате выполнения работы я:

1. Освоил алгоритмы работы со структурами (struct) в языке C++;
2. Освоил реализацию многоэлементных структур данных в С++ с помощью struct.

# **Исходный код программы**

|  |
| --- |
| // ФАЙЛ “main.cpp”  #include <iostream>  #include <string>  #include "Sits.h"  #define DELIMITER\_LENGTH 100  void print\_sit\_info(const Sit& sit)  {  std::cout << "Фильм: \"" << sit.movie\_name << "\", "  << "дата и время: " << sit.date.day << '.'  << sit.date.month << '.' << sit.date.year << ' '  << sit.time.hours << ':' << sit.time.minutes  << ':' << sit.time.seconds << ", "  << "ряд и место: " << sit.row << ' ' << sit.place << ", "  << "зал: " << (sit.room == Room::big ? "большой" : "малый")  << '\n';  }  void print\_sits\_info(const Sits& sits)  {  for (int i = 0; i < DELIMITER\_LENGTH; i++)  {  std::cout << '-';  }  std::cout << '\n';  for (int i = 0; i < sits.size; i++)  {  print\_sit\_info(sits.sits[i]);  }  for (int i = 0; i < DELIMITER\_LENGTH; i++)  {  std::cout << '-';  }  std::cout << '\n';  }  void read\_movie\_name(char movie\_name[MAX\_MOVIE\_NAME\_LENGTH])  {  std::cout << "Введите название фильма: ";  std::cin >> movie\_name;  }  void read\_date(Date& date)  {  std::cout << "Введите дату (день, месяц и год через пробел): ";  std::cin >> date.day >> date.month >> date.year;  }  void read\_time(Time& time)  {  std::cout << "Введите время (часы, минуты и секунды через пробел): ";  std::cin >> time.hours >> time.minutes >> time.seconds;  }  void read\_sit\_info(Sit& sit)  {  read\_movie\_name(sit.movie\_name);  read\_date(sit.date);  read\_time(sit.time);  std::cout << "Введите место в зале (номер ряда и номер места через пробел): ";  std::cin >> sit.row >> sit.place;  std::cout << "Введите номер зала (маленький - 1, большой - 2): ";  std::cin >> sit.room;  }  int main()  {  Sits sits;    while (true)  {  std::cout << "Выберите операцию:\n"  << "1 - Заполнение записи по одному проданному билету с клавиатуры\n"  << "2 - Удаление записи по заданной дате\n"  << "3 - Определение количества свободных мест на заданный фильм в указанную дату и время\n"  << "4 - Выход\n";  int choice = 0;  std::cin >> choice;  if (choice == 1)  {  Sit sit;  read\_sit\_info(sit);  sits.insert(sit);  print\_sits\_info(sits);  }  else if (choice == 2)  {  Date date;  read\_date(date);  sits.delete\_by\_date(date);  print\_sits\_info(sits);  }  else if (choice == 3)  {  char movie\_name[MAX\_MOVIE\_NAME\_LENGTH];  Date date;  Time time;  read\_movie\_name(movie\_name);  read\_date(date);  read\_time(time);  std::cout << "Свободных мест - "  << sits.free\_places(movie\_name, date, time)  << '\n';  }  else  {  break;  }  }  return 0;  }  // ФАЙЛ “Sits.h”  #ifndef SITS\_H  #define SITS\_H  #define SMALL\_ROOM\_CAPACITY 100  #define BIG\_ROOM\_CAPACITY 1000  #define MAX\_MOVIE\_NAME\_LENGTH 128  #include <stdlib.h>  #include <string.h>  enum Room  {  none = 0,  small = 1,  big = 2  };  extern int RoomsCapacities[3];  struct Date  {  int day;  int month;  int year;  bool operator==(Date other);  };  struct Time  {  int seconds;  int minutes;  int hours;  bool operator==(Time other);  };  struct Sit  {  char movie\_name[MAX\_MOVIE\_NAME\_LENGTH];  Date date;  Time time;  int row;  int place;  int room;  };  struct Sits  {  int size;  int first\_in\_big\_room;  Sit\* sits;  Sits();  ~Sits();  void insert(Sit sit);  void delete\_by\_date(Date date);  int free\_places(char movie\_name[MAX\_MOVIE\_NAME\_LENGTH], Date date, Time time);  };  #endif  // ФАЙЛ “Sits.cpp”  #include "Sits.h"  int RoomsCapacities[3] = {  0,  SMALL\_ROOM\_CAPACITY,  BIG\_ROOM\_CAPACITY  };  bool Date::operator==(Date other)  {  return (this->day == other.day &&  this->month == other.month &&  this->year == other.year);  }  bool Time::operator==(Time other)  {  return (this->hours == other.hours &&  this->minutes == other.minutes &&  this->seconds == other.seconds);  }  Sits::Sits()  {  size = 0;  first\_in\_big\_room = 0;  sits = nullptr;  }  Sits::~Sits()  {  free(sits);  }  void Sits::insert(Sit sit)  {  if (size == 0)  {  sits = (Sit\*) malloc(sizeof(\*sits) \* (size + 1));  }  else  {  sits = (Sit\*) realloc(sits, sizeof(\*sits) \* (size + 1));  }  size++;  if (sit.room == Room::big)  {  for (int i = size - 1; i > first\_in\_big\_room; i--)  {  sits[i] = sits[i - 1];  }  sits[first\_in\_big\_room] = sit;  }  else if (sit.room == Room::small)  {  for (int i = size - 1; i > 0; i--)  {  sits[i] = sits[i - 1];  }  first\_in\_big\_room++;  sits[0] = sit;  }  }  void Sits::delete\_by\_date(Date date)  {  for (int i = 0; i < size; i++)  {  if (sits[i].date == date)  {  for (int j = i; j < size; j++)  {  sits[j] = sits[j + 1];  }  sits = (Sit\*) realloc(sits, sizeof(\*sits) \* (size - 1));  size--;  if (i < first\_in\_big\_room)  {  first\_in\_big\_room--;  }  i--;  }  }  }  int Sits::free\_places(char movie\_name[MAX\_MOVIE\_NAME\_LENGTH], Date date, Time time)  {  int not\_free = 0;  for (int i = 0; i < size; i++)  {  if (strcmp(sits[i].movie\_name, movie\_name) == 0&&  sits[i].date == date &&  sits[i].time == time)  {  not\_free++;  }  }  return RoomsCapacities[Room::small] +  RoomsCapacities[Room::big] - not\_free;  } |