

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Конструирование программ и языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

ПРОГРАММА ОПЕРАТОРА КИНОТЕАТРА

БГУИР КП 1-40 02 01 107 ПЗ

Студент: гр. 850501 Кочерга М. В.

Руководитель: Ковальчук А. М.

Минск 2019

Учреждение образования
«Белорусский государственный университет информатики и
радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Б.В.Никульшин
(подпись)
«__» _____ 2019 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту Кочерга Маргарите Владимировне

1. Тема проекта Программа оператора кинотеатра
2. Срок сдачи студентом законченного проекта 23.12.2019 г.
3. Исходные данные к проекту Films.csv (хранится вся информация о фильмах)
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение. 1. Постановка задачи. 2. Структура входных и выходных данных.

3. Диаграмма классов. 4. Описание классов. 5. Структурная схема 6. Разработка алгоритмов. 6.1. Схема алгоритмов. 6.2. Алгоритмы по шагам. 7. Код программы. 8. Результаты работы программы. Заключение. Литература.

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. Схема структурная

2. Диаграмма классов

3. Схемы алгоритмов

4. Скриншоты работающей программы

6. Консультант по проекту (с обозначением разделов проекта) Ковальчук А.М

7. Дата выдачи задания 10.09.2019 г.

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

разделы 1, 2, 3 к. 10.10.2019 – 25 %;

раздел 4 к. 05.11.2019 – 20 %;

разделы 5, 6 к. 25.11.2019 – 20 %;

раздел 7 к. 10.12.2019 – 20 %;

оформление пояснительной записки и графического материала к 15.12.2019 – 15 %

Защита курсового проекта с 20 по 31.12.2019

РУКОВОДИТЕЛЬ _____ А. М. Ковальчук
(подпись)

Задание принял к исполнению _____ М. В. Кочерга
(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 ПОСТАНОВКА ЗАДАЧИ.....	6
2 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ	7
3 ДИАГРАММА КЛАССОВ	9
4 ОПИСАНИЕ КЛАССОВ	10
5 РАЗРАБОТКА АЛГОРИТМА	19
6 КОД ПРОГРАММЫ.....	21
7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	22
ЗАКЛЮЧЕНИЕ	30
ЛИТЕРАТУРА	31
ПРИЛОЖЕНИЕ А	32
ПРИЛОЖЕНИЕ Б.....	33
ПРИЛОЖЕНИЕ В	34
ПРИЛОЖЕНИЕ Г.....	35
ПРИЛОЖЕНИЕ Д	36

ВВЕДЕНИЕ

Язык программирования C++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. Он поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Основными принципами объектно-ориентированного программирования являются инкапсуляция, полиморфизм и наследование. Данный язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Язык C++ гибок и универсален. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Благодаря этому он подходит для решения многих задач, например, написание операционных систем, компьютерных игр, баз данных, создание высокопроизводительных серверов и прочее. С помощью языка C++ написаны такие популярные продукты как веб-браузер Google Chrome, операционные системы Windows и Unix, Microsoft Office, а также игра World of Warcraft.

Исходя из этого, можно считать, что C++ подходит для выполнения данной курсовой работы.

1 ПОСТАНОВКА ЗАДАЧИ

Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. В программе должна быть предусмотрена возможность создания информации в виде файлов, связанных определенным образом. Программа должна содержать:

- данные о времени проведения сеансов, названия фильмов, жанр фильма, количество мест в зале, цена билета в зависимости от места;
- выдавать отчеты о расписании сеансов со стоимостью билетов;
- билет на определенный сеанс;
- ведомости проданных билетов на определенный день.

Реализовать иерархию классов с использованием наследования. Разработать и использовать в программе классы контейнеров и итераторов (свои и STL). Производить обработку исключительных ситуаций. При реализации операций редактирования, добавления, удаления информации необходимо предусмотреть операцию отмены последних действий.

Аналогами программы являются сервисы по покупке билетов:

1. Сайт для покупки билетов в кино <http://www.bycard.by> ;
2. Сайт сети кинотеатров Silver Screen cinemas <https://silverscreen.by> .

2 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ

В программе используется 1 текстовый файл формата CSV. Формат CSV используют, чтобы хранить таблицы в текстовых файлах. CSV расшифровывается как comma-separated values — «значения, разделенные запятыми». Но разделителями столбцов в CSV-файле могут также служить точки с запятой и знаки табуляции.

Файл Films.csv. Хранит данные о сеансах фильмов.

Таблица 1 – структура файла Films.txt

Название фильма	Жанр	Дата	Время	Номера купленных мест
Maleficent: Mistress of Evil	Fantasy	22.01.2020	16:00	2 3 10 17 18 21 44 63
Frozen 2	Fantasy	21.01.2020	09:00	14 15 20 21 50 120

Также в ходе работы программы для хранения промежуточных данных используются контейнеры библиотеки Qt:

- QStringList (список строк, используется для хранения заголовков таблиц);
- QStack (используется для отмены последних действий. Хранит объекты класса Seance. Если была выполнена операция редактирования, то записываются объекты до редактирования).

Также был использован контейнер Queue, написанный вручную, в котором хранится информация о сеансах фильмов. Шаблонный класс Queue содержит поля:

- int size (количество элементов очереди);
- Node<T>* first (указатель на первый элемент очереди);
- Node<T>* last (указатель на последний элемент очереди).

При этом шаблонная структура Node<T> имеет следующие поля:

-T data (информация об объекте);

-Node<T>* left (указатель на следующий элемент очереди).

3 ДИАГРАММА КЛАССОВ

Разработанная диаграмма классов представлена в приложении А.

4 ОПИСАНИЕ КЛАССОВ

1 Класс `Algorithm` – класс алгоритмов, осуществляющий работу с итераторами

```
class Algorithm {
public:
    //Функция, обменивающая значения объектов,
    //на которые указывают два итератора
    template <class T>
    static void swapIterators(Iterator<T>& itr1, Iterator<T>& itr2);

    //Функция сортировки контейнера пузырьком
    template <class T>
    static void sort(Iterator<T> first, Iterator<T> last);
};
```

2 Класс формы `Qt CalendarWindow` – класс визуальной формы, отвечающий за экран выбора даты

```
class CalendarWindow : public QDialog
{
    Q_OBJECT

public:
    //Конструктор класса CalendarWindow
    explicit CalendarWindow(QWidget *parent = nullptr);
    //Деструктор класса CalendarWindow
    ~CalendarWindow();

private slots:
    //Обработчик изменения выбранной даты, выводит
    //изменения на экран с помощью QLabel
    void on_calendarWidget_selectionChanged();
    //Обработчик нажатия на кнопку chooseFilmButton,
    //осуществляет переход на экран выбора фильмов
    void on_chooseFilmButton_clicked();
    //Обработчик нажатия на кнопку listButton,
    //осуществляет переход к окну с ведомостью проданных
    //билетов на определенный день
    void on_listButton_clicked();
    //Обработчик нажатия на кнопку reportButton,
    //осуществляет переход к окну с отчетом о расписании
    //сеансов со стоимостью билетов
    void on_reportButton_clicked();

private:
    //Указатель на графическую составляющую класса
    Ui::CalendarWindow *ui;
};
```

3 Класс ConstIterator – шаблонный класс константного итератора

```
template <class T>
class ConstIterator {
public:
    //Конструктор класса ConstIterator
    ConstIterator();
    //Оператор * возвращает ссылку на элемент
    const T& operator*() const;
    //Перегрузка операторов сравнения
    bool operator==(const ConstIterator& itr) const;
    bool operator!=(const ConstIterator& itr) const;
    //Перегрузка префиксного инкремента
    ConstIterator& operator++();
    //Перегрузка постфиксного инкремента
    ConstIterator operator++(int);
protected:
    //Указатель на текущий узел
    Node<T>* current;
    //Защищенный конструктор
    ConstIterator(Node<T>* ptr);
    //Дружественный класс Queue<T>
    friend class Queue<T>;
};
```

4 Класс формы Qt EditWindow – класс визуальной формы, отвечающий за экран редактирования и добавления информации о сеансе

```
class EditWindow : public QDialog
{
    Q_OBJECT
    //Очередь сеансов на определенный день
    Queue<Seance> arrayOfSeances;
    //Выбранная дата
    QString selectedDate;
    //Флаг редактирования/добавления объекта
    QString buttonName;
    //Индекс редактируемого объекта
    int index;

public:
    //Конструктор класса EditWindow
    explicit EditWindow(QString _buttonName, QString _selectedDate,
                        Queue<Seance> _arrayOfSeances = {},
                        int _index = 0, *parent = nullptr);
    //Деструктор класса EditWindow
    ~EditWindow();

private slots:
    //Обработчик нажатия на кнопку addButton, добавляет
    //новый сеанс или изменяет старый в зависимости от флага
    void on_addButton_clicked();

private:
    //Указатель на графическую составляющую класса
    Ui::EditWindow *ui;
};
```

5 Класс Exception – класс обработки исключений

```
class Exception: public QException
{
    //Сообщение об ошибке
    QString message;
public:
    //Конструктор класса Exception
    Exception(QString _message = "");
    //Деструктор класса Exception
    ~Exception() noexcept {}
    //Функция получения сообщения об ошибке
    virtual QString Message();
    //Функция проверки, открыт ли файл для чтения
    static bool isFileOpenForReading(QWidget *parent, ifstream& ifile,
                                     const string& fileName);
    //Функция проверки, открыт ли файл для записи
    static bool isFileOpenForWriting(QWidget *parent, ofstream& ofile,
                                     const string& fileName);
    //Функция проверки на ошибки таблиц
    static bool tableProblems(QWidget *parent, QTableWidgetItem& tableWidget);
    //Функция проверки ввода строки
    static bool isString(QWidget *parent, const QString& str,
                        string language = "ENG");
    //Функция проверки ввода времени
    static bool isValidTime(QWidget *parent, const string& str);
};
```

6 Класс Film – класс для работы с фильмами

```
class Film
{
    //Название фильма
    QString name;
    //Жанр фильма
    QString genre;
    //Статическое поле с именем файла с фильмами
    static QString fileName;
public:
    //Конструктор класса Film
    Film(QString _name = "", QString _genre = "");
    //Деструктор класса Film
    virtual ~Film();
    //Перегрузка считывания из файла
    friend ifstream & operator >> (ifstream &ifile, Film &film);
    //Перегрузка записи в файл
    friend ofstream & operator << (ofstream &ofile, Film &film);
    //Функции получения и установки значений полей класса
    QString getName() const;
    void setName(const QString &value);
    QString getGenre() const;
    void setGenre(const QString &value);
    static QString getFileName();
    static void setFileName(const QString &value);
    //Перегрузка сравнения объектов
```

```

        friend bool operator>(Film obj1, Film obj2);
};

```

7 Класс формы Qt FilmsWindow – класс визуальной формы, отвечающий за экран с таблицей сеансов на определенный день

```

class FilmsWindow : public QDialog
{
    Q_OBJECT
    //Очередь сеансов на определенный день
    Queue<Seance> arrayOfSeances;
    //Выбранная дата
    QString selectedDate;
public:
    //Конструктор класса FilmsWindow
    explicit FilmsWindow(QString _selectedDate = "",
                        QWidget *parent = nullptr);
    //Деструктор класса FilmsWindow
    ~FilmsWindow();
    //Функция создания таблицы с сеансами
    //путем считывания информации из файла
    void configure();
    //Функция перезаписи информации в файл
    static void rewriteFile(QWidget *parent,
                            Queue<Seance> arrayOfSeances,
                            QString selectedDate);
    //Функции получения и установки значений полей
    Queue<Seance> getArrayOfSeances() const;
    void setArrayOfSeances(const Queue<Seance> &value);
private slots:
    //Обработчик изменения выбранного сеанса, выводит
    //изменения на экран с помощью QLabel
    void on_filmsTableWidget_itemSelectionChanged();
    //Обработчик нажатия на кнопку choosePlacesButton,
    //осуществляет переход на экран выбора мест
    void on_choosePlacesButton_clicked();
    //Обработчик нажатия на кнопку backButton,
    //осуществляет возврат на предыдущий экран
    void on_backButton_clicked();
    //Обработчик нажатия на кнопку addButton, осуществляет
    //переход на экран добавления нового сеанса
    void on_addButton_clicked();
    //Обработчик нажатия на кнопку deleteButton, осуществляет
    //удаляет выбранный сеанс
    void on_deleteButton_clicked();
    //Обработчик нажатия на кнопку editButton, осуществляет
    //переход на экран редактирования выбранного фильма
    void on_editButton_clicked();
    //Обработчик нажатия на кнопку addButton,
    //осуществляет отмену последнего действия
    void on_undoButton_clicked();
private:
    //Указатель на графическую составляющую класса
    Ui::FilmsWindow *ui;
};

```

8 Класс формы Qt InformationWindow – класс визуальной формы, отвечающий за экран с ведомостью проданных билетов или отчетом о расписании сеансов

```
class InformationWindow : public QDialog
{
    Q_OBJECT
    //Выбранная дата
    QString selectedDate;
    //Очередь сеансов на определенный день
    Queue<Seance> arrayOfSeances;
public:
    //Конструктор класса InformationWindow
    explicit InformationWindow(QString _selectedDate,
                                QString config = "LIST",
                                QWidget *parent = nullptr);
    //Деструктор класса InformationWindow
    ~InformationWindow();
    //Функция создания таблицы с сеансами
    //путем считывания информации из файла
    void configure(QString config);
private slots:
    //Обработчик нажатия на кнопку backButton,
    //осуществляет возврат на предыдущий экран
    void on_backButton_clicked();
private:
    //Указатель на графическую составляющую класса
    Ui::InformationWindow *ui;
};
```

9 Класс Iterator – шаблонный класс итератора

```
template <class T>
class Iterator: public ConstIterator<T> {
public:
    //Конструктор класса Iterator
    Iterator();
    //Оператор * возвращает изменяемую ссылку на
    //данные в текущем узле
    T& operator*();
    //Оператор * возвращает ссылку на элемент
    const T& operator*() const;
    //Перегрузка префиксного инкремента
    Iterator<T>& operator++();
    //Перегрузка постфиксного инкремента
    Iterator<T> operator++(int);
    //Перегрузки операторов сравнения итераторов
    bool operator>(const Iterator<T> itr);
    bool operator<(const Iterator<T> itr);
protected:
    //Защищенный конструктор класса Iterator
    Iterator(Node<T>* ptr);
    //Дружественный класс Queue<T>
    friend class Queue<T>;
};
```

10 Класс формы Qt MainWindow – класс визуальной формы, отвечающий за главный экран программы

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    //Конструктор класса MainWindow
    MainWindow(QWidget *parent = nullptr);
    //Деструктор класса MainWindow
    ~MainWindow();

private slots:
    //Обработчик нажатия на кнопку buyTicketsButton,
    //осуществляет переход на следующий экран
    void on_buyTicketsButton_clicked();

private:
    //Указатель на графическую составляющую класса
    Ui::MainWindow *ui;
};
```

11 Структура Node – шаблонная структура узла очереди

```
template <class T>
struct Node {
    //Информация об объекте
    T data;
    //Указатель на следующий элемент очереди
    Node* next;
    //Конструктор структуры Node
    Node(const T& _data = T{}, Node* _next = nullptr) :
        data{ _data }, next{ _next } {}
};
```

12 Класс формы Qt PlacesWindow – класс визуальной формы, отвечающий за экран выбора мест в зрительном зале

```
class PlacesWindow : public QDialog
{
    Q_OBJECT

public:
    //Очередь сеансов на определенный день
    Queue<Seance> arrayOfSeances;
    //Индекс выбранного сеанса
    int index;
    //Конструктор класса PlacesWindow
    explicit PlacesWindow(Queue<Seance> _arrayOfSeances, int _index,
        QWidget *parent = nullptr);
    //Деструктор класса PlacesWindow
};
```

```

~PlacesWindow();
//Функция создания группы кнопок (зрительный зал)
void createPlacesGroup();
private slots:
    //Обработчик нажатия на место в зале,
    //изменяет цвет выбранного места
    void placeClicked(int i);
    //Обработчик нажатия на кнопку backButton,
    //осуществляет возврат на предыдущий экран
    void on_backButton_clicked();
    //Обработчик нажатия на кнопку choosePlacesButton,
    //осуществляет переход на экран
    //с информацией о купленном билете
    void on_choosePlacesButton_clicked();
private:
    //Указатель на графическую составляющую класса
    Ui::PlacesWindow *ui;
    //Указатель на группу кнопок (зрительный зал)
    QPushButton *placesGroup;
};

```

13 Класс Queue – шаблонный класс контейнера Очередь

```

template <class T>
class Queue {
    int size;           //Количество элементов очереди
    Node<T>* first;     //Указатель на первый элемент очереди
    Node<T>* last;     //Указатель на последний элемент очереди
public:
    //Конструктор по умолчанию класса Queue
    Queue();
    //Конструктор с параметром класса Queue
    Queue(const Queue& obj);
    //Деструктор класса Queue
    ~Queue();
    //Функции получения значений полей класса
    int getSize();
    T getFirst();
    T getLast();
    //Добавление элемента в конец очереди
    void push(const T& value);
    //Удаление элемента с начала очереди
    void pop();
    //Проверка, пуста ли очередь
    bool isEmpty();
    //Очистка очереди
    void eraseAll();
    //Удаление элемента по индексу
    void eraseByIndex(int index);
    //Функция, обменивающая значения объектов
    void swap(Queue<T>& obj);
    //Итератор на начало очереди
    Iterator<T> begin();
    ConstIterator<T> begin() const;
    //Итератор на конец очереди
    Iterator<T> end();
    ConstIterator<T> end() const;
    //Перегрузка оператора присваивания
    const Queue<T>& operator=(const Queue<T>& obj);
    //Получение элемента по индексу

```



```

T& operator[] (int index);
//Перегрузка операторов сравнения
friend bool operator==(const Queue<T>& obj1,
                        const Queue<T>& obj2);
inline friend bool operator!=(const Queue<T>& obj1,
                              const Queue<T>& obj2);
};

```

14 Класс Seance – класс для работы с сеансами

```

class Seance: public Film
{
    QString date;           //Дата сеанса
    QString time;           //Время сеанса
    Queue<int> soldPlaces;   //Проданные места
public:
    //Конструктор класса Seance
    Seance(QString _name = "", QString _genre = "",
            QString _date = "", QString _time = "",
            Queue<int> _soldPlaces = {});
    //Деструктор класса Seance
    virtual ~Seance();
    //Перегрузка считывания из файла
    friend ifstream & operator >> (ifstream &ifile, Seance &seance);
    //Перегрузка записи в файл
    friend ofstream & operator << (ofstream &ofile, Seance &seance);
    //Функции получения и установки значений полей класса
    QString getTime() const;
    void setTime(const QString &value);
    Queue<int> getSoldPlaces() const;
    void setSoldPlaces(const Queue<int> &value);
    QString getDate() const;
    void setDate(const QString &value);
    //Статическое поле со стеком для отмены последнего действия
    static QStack< Queue<Seance> > undoStack;
    //Перегрузка сравнения объектов
    friend bool operator>(Seance obj1, Seance obj2);
};

```

15 Класс Ticket – класс для работы с билетами

```

class Ticket: public Seance
{
    //Купленные места
    Queue<int> places;
    //Цена билета
    int price;
public:
    //Конструктор класса Ticket
    Ticket(QString _name = "", QString _genre = "",
            QString _date = "", QString _time = "",
            Queue<int> _soldPlaces = {},
            Queue<int> _places = {}, int _price = 0);
    //Деструктор класса Ticket
    ~Ticket();
};

```

```

//Функции получения и установки значений полей класса
Queue<int> getPlaces() const;
void setPlaces(const Queue<int> &value);
int getPrice() const;
void setPrice(int value);
};

```

16 Класс формы Qt TicketWindow – класс визуальной формы, отвечающий за экран с данными о купленных местах

```

class TicketWindow : public QDialog
{
    Q_OBJECT
    //Билет
    Ticket ticket;
public:
    //Конструктор класса TicketWindow
    explicit TicketWindow(Seance selectedSeance, Queue<int> selectedPlaces,
                           QWidget *parent = nullptr);
    //Деструктор класса TicketWindow
    ~TicketWindow();
private slots:
    //Обработчик нажатия на кнопку exitButton,
    //осуществляет выход из программы
    void on_exitButton_clicked();
    //Обработчик нажатия на кнопку buyMoreButton,
    //осуществляет переход на экран выбора даты
    void on_buyMoreButton_clicked();
private:
    //Указатель на графическую составляющую класса
    Ui::TicketWindow *ui;
};

```

5 РАЗРАБОТКА АЛГОРИТМА

В данном разделе рассмотрены описания алгоритмов и схемы алгоритмов, используемые в программе.

5.1 Схема алгоритмов

Схема алгоритма метода `Queue<T>::push()` добавления элемента в конец очереди представлена в приложении Б.

Схема алгоритма метода `Queue<T>::eraseByIndex(int index)` удаления элемента очереди по индексу представлена в приложении В.

5.2 Алгоритмы по шагам

5.2.1 Алгоритм поиска объектов по определённым параметрам

Рассмотрим функцию `sort` в классе `Algorithm` сортировки пузырьком очереди с использованием итераторов.

1. Начало.
2. Входные данные: итераторы `Iterator<T> first` и `Iterator<T> last`, указывающие на начало и конец очереди.
Промежуточные данные: итераторы `Iterator<T> i, j, k`, с помощью которых происходит сортировка.
3. Цикл с параметром `Iterator<T> i==first`.
4. Цикл с параметрами `Iterator<T> j==first` и `Iterator<T> k==j`.
5. Инкрементация итератора `k`.
6. Сравнение значений, на которые указывают `j` и `k`. Если значение, на которое указывает итератор `j`, больше, чем значение, на которое указывает итератор `k`, то вызывается метод `Algorithm::swapIterators(j, k)`, который обменивает значения объектов, на которые указывают итераторы;

7. Конец цикла по j и k.
8. Конец цикла по i.
9. Конец.

5.2.2 Алгоритм префиксного инкремента объекта шаблонного класса Iterator

Для алгоритма по шагам рассмотрим перегрузку операции ++ в шаблонном классе Iterator.

1. Начало.
2. Входные данные: итератор *this в очереди.
Выходные данные: итератор *this в очереди.
3. Присваивание текущему указателю this->current следующий в очереди this->current->next.
4. Если текущий указатель this->current равен nullptr, то генерируется исключение throw Exception().
5. Возврат *this;
6. Конец.

6 КОД ПРОГРАММЫ

Текст программы представлен в приложении Г.

7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для запуска программы, необходимо открыть исполняемый файл MovieTickets.app.

После запуска программы открывается окно, продемонстрированное на рисунке 7.1. Перемещение указателя для выбора режима выполняется курсором мыши. Для перехода к следующему окну необходимо нажать левой кнопкой мыши на кнопку «Buy tickets».

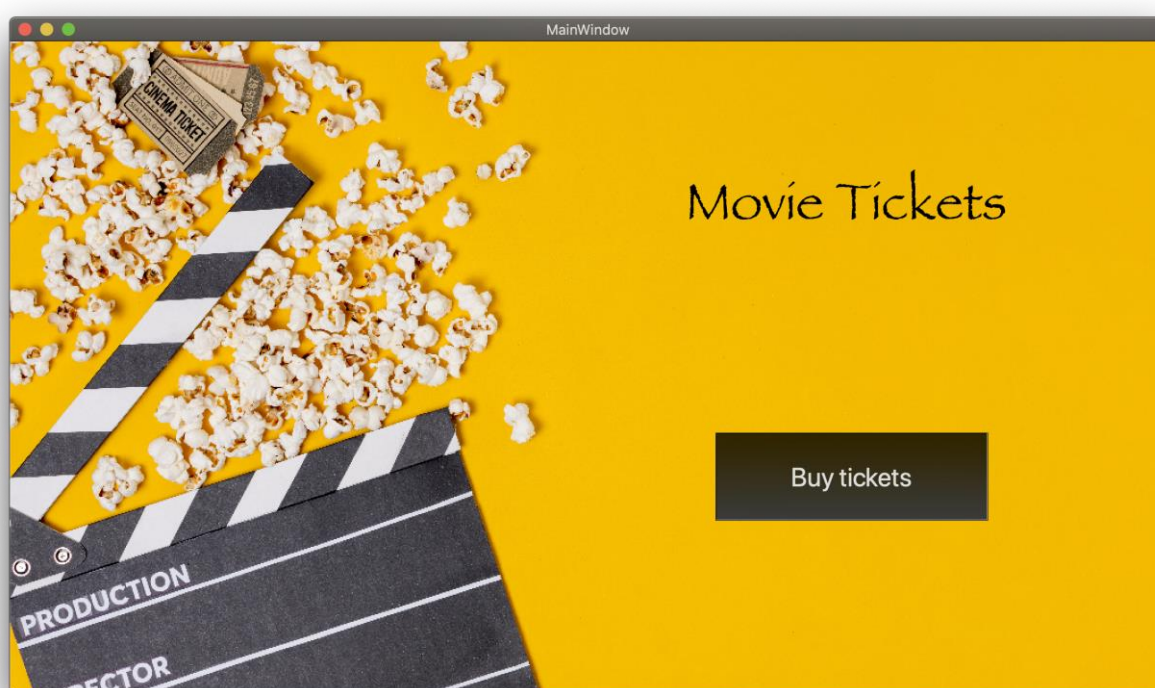


Рисунок 7.1 – Главное окно приложения

После нажатия на кнопку «Buy tickets», открывается окно (рисунок 7.2) выбора файла, в котором должны храниться данные о сеансах. Данное окно позволяет пользователю выбирать файлы с расширением .csv вне зависимости от того, какое имя у файла.

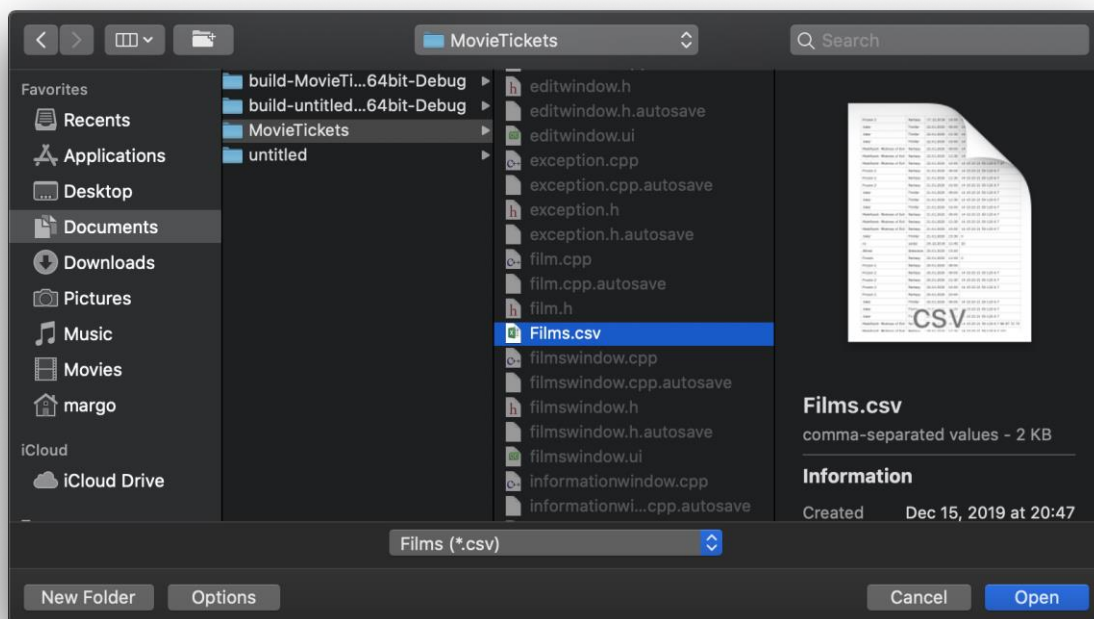


Рисунок 7.2 – Окно выбора файла

После нажатия на кнопку «Open», открывается окно с календарём, представленное на рисунке 7.3, в котором реализована возможность выбора даты. При нажатии на приведенные ниже кнопки, выполняются следующие действия:

- «Choose film» – открывается окно с таблицей сеансов на определенный день;
- «List of tickets sold» – открывается окно с ведомостью проданных билетов на определенный день;
- «Session schedule report» – открывается окно с отчетом о расписании сеансов и стоимостью билетов на определенный день.

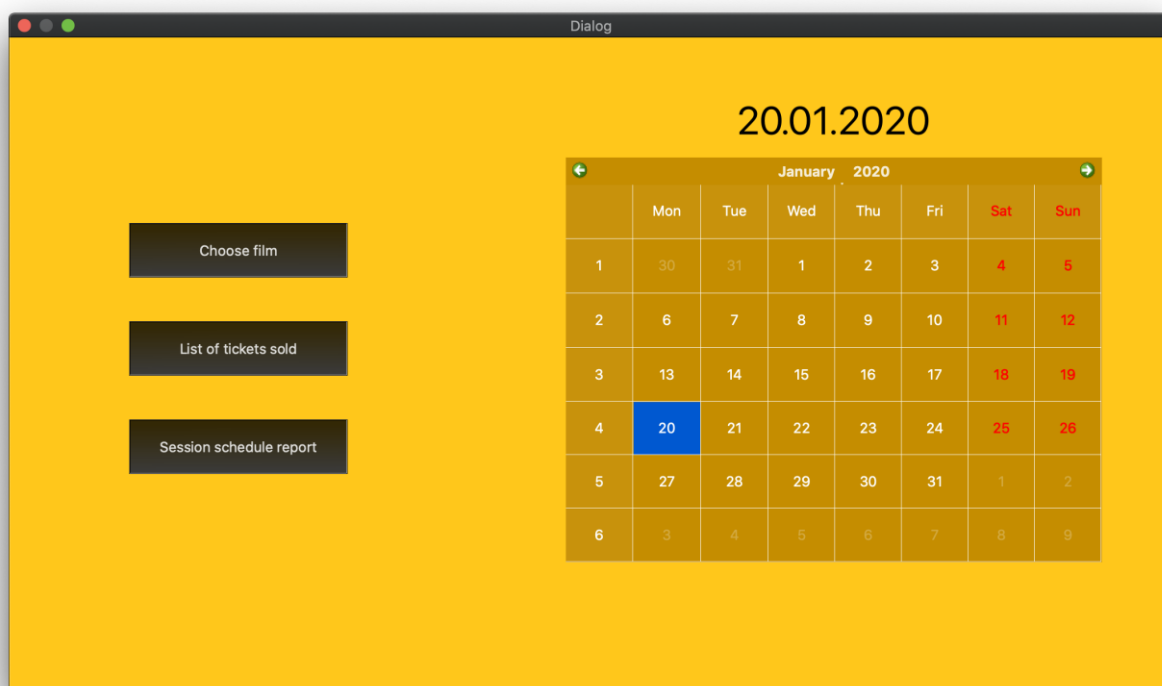


Рисунок 7.3 – Окно выбора даты

Окно с ведомостью проданных билетов (рисунок 7.4) и окно с отчетом о расписании сеансов (рисунок 7.5) на определенный день практически идентичны. Данные окна имеют возможность возврата на предыдущий экран, при нажатии на кнопку возврата со стрелкой.

	Film's name	Genre	Time	Number of sold tickets
1	Alfred	Detective	15:00	0
2	Frozen	Fantasy	12:00	1
3	Frozen 2	Fantasy	08:00	0
4	Frozen 2	Fantasy	09:00	8
5	Frozen 2	Fantasy	12:30	8
6	Frozen 2	Fantasy	16:00	8
7	Frozen 2	Fantasy	20:00	0
8	Joker	Thriller	09:00	8
9	Joker	Thriller	12:00	8
10	Joker	Thriller	16:00	8
11	Maleficent: Mistress of Evil	Fantasy	09:00	12
12	Maleficent: Mistress of Evil	Fantasy	12:30	9
13	Maleficent: Mistress of Evil	Fantasy	16:00	12

Рисунок 7.4 – Окно с ведомостью проданных билетов

	Film's name	Genre	Time	Number of sold tickets
1	Alfred	Detective	15:00	10 - 20 BYN
2	Frozen	Fantasy	12:00	10 - 20 BYN
3	Frozen 2	Fantasy	08:00	10 - 20 BYN
4	Frozen 2	Fantasy	09:00	10 - 20 BYN
5	Frozen 2	Fantasy	12:30	10 - 20 BYN
6	Frozen 2	Fantasy	16:00	10 - 20 BYN
7	Frozen 2	Fantasy	20:00	10 - 20 BYN
8	Joker	Thriller	09:00	10 - 20 BYN
9	Joker	Thriller	12:00	10 - 20 BYN
10	Joker	Thriller	16:00	10 - 20 BYN
11	Maleficent: Mistress of Evil	Fantasy	09:00	10 - 20 BYN
12	Maleficent: Mistress of Evil	Fantasy	12:30	10 - 20 BYN
13	Maleficent: Mistress of Evil	Fantasy	16:00	10 - 20 BYN

Рисунок 7.5 – Окно с отчетом о расписании сеансов

Окно с таблицей сеансов на определенный день представлено на рисунке 7.6. Таблица отсортирована по названиям фильмов в алфавитном порядке, а также сеансы на один и тот же фильм отсортированы по времени (от раннего к позднему).

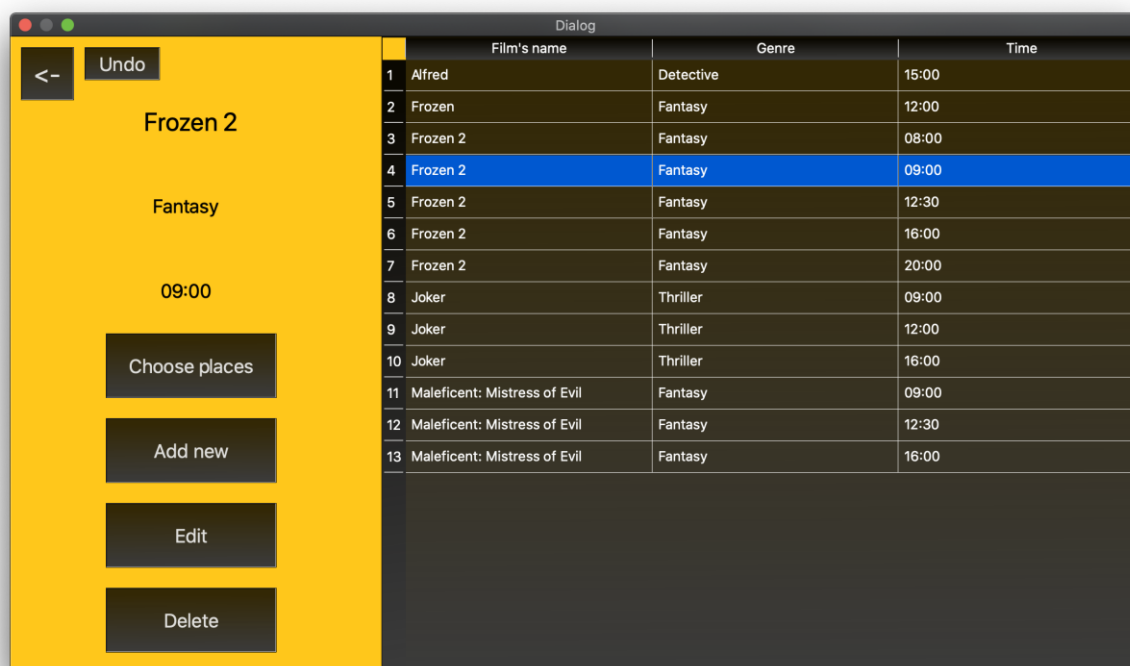


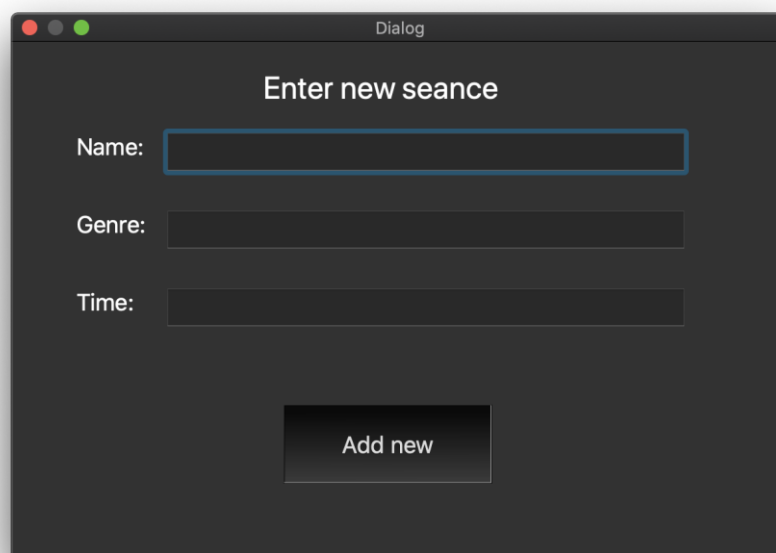
Рисунок 7.6 – Окно с таблицей сеансов на определенный день

Пользователь может выбрать сеанс в таблице, щёлкнув по нему левой кнопкой мыши. Также на данном окне расположены кнопки, благодаря которым у пользователя имеется возможность редактировать выбранный сеанс, удалять, а также добавлять новый. Описание всех кнопок окна представлено ниже:

- Кнопка со стрелкой – возврат на предыдущее окно;
- «Undo» – отмена последнего действия;
- «Choose places» – открытие окна со схемой зрительного зала и возможностью выбрать места;
- «Add new» – открытие окна добавления нового сеанса;

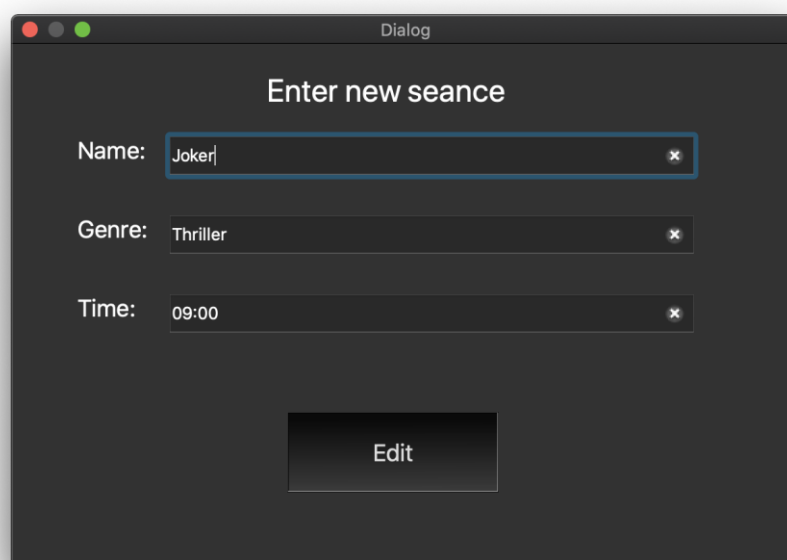
- «Edit» – открытие окна редактирования нового сеанса;
- «Delete» – удаление выбранного сеанса.

Окна добавления и редактирования сеанса представлены на рисунках 7.7 и 7.8.



The image shows a dark-themed dialog box titled "Dialog" with the subtitle "Enter new seance". It contains three input fields: "Name:", "Genre:", and "Time:". The "Name:" field is currently selected with a blue border. At the bottom center, there is a button labeled "Add new".

Рисунок 7.7 – Окно добавления нового сеанса



The image shows the same dark-themed dialog box titled "Dialog" with the subtitle "Enter new seance". The input fields are now filled with text: "Name:" contains "Joker", "Genre:" contains "Thriller", and "Time:" contains "09:00". Each field has a small 'x' icon on the right side. At the bottom center, there is a button labeled "Edit".

Рисунок 7.8 – Окно редактирования выбранного сеанса

Данные окна отличаются лишь названием кнопок «Add new» и «Edit», при нажатие на которые происходит переход к предыдущему окну с добавлением или редактированием сеанса соответственно. Информация о сеансе вводится в поля «Name», «Genre» и «Time» с клавиатуры. Для выбора определенного поля, необходимо нажать на него левой клавишей мыши.

Окно со схемой зрительного зала представлено на рисунке 7.9. В данном окне реализована возможность выбирать места в зале, причем выбранные места выделяются серым цветом, а занятые места – красным.

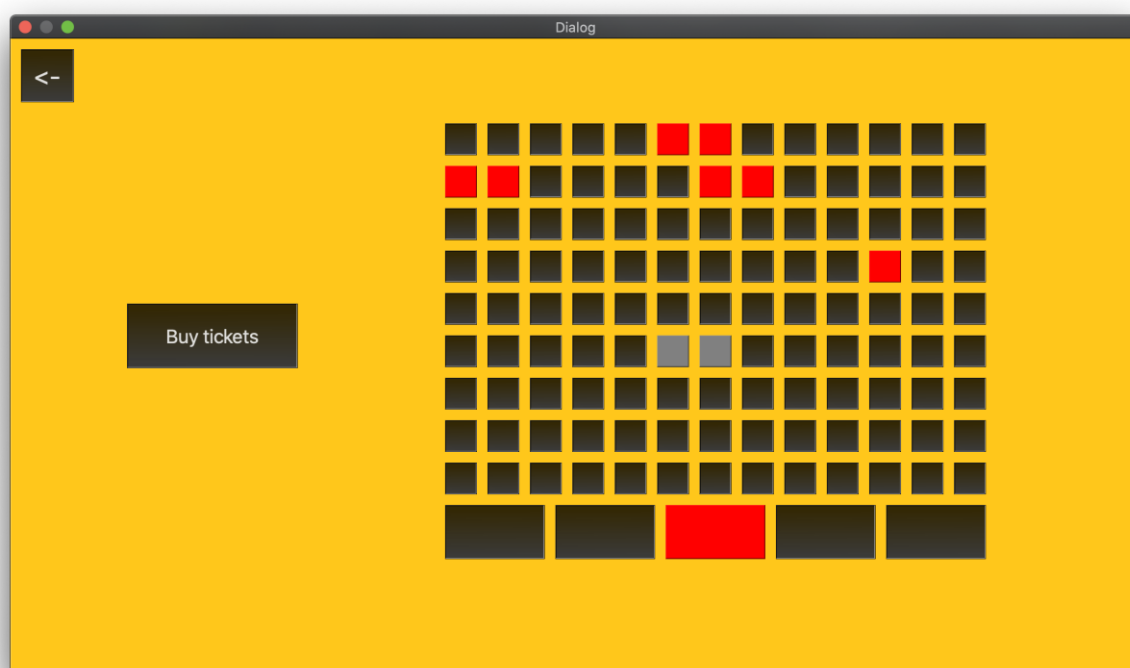


Рисунок 7.8 – Окно со схемой зрительного зала

На данном окне имеется кнопка со стрелкой, при нажатии на которую происходит переход к предыдущему окну. И кнопка «Buy tickets», открывающая окно с информацией о приобретенных билетах, представленное на рисунке 7.9.

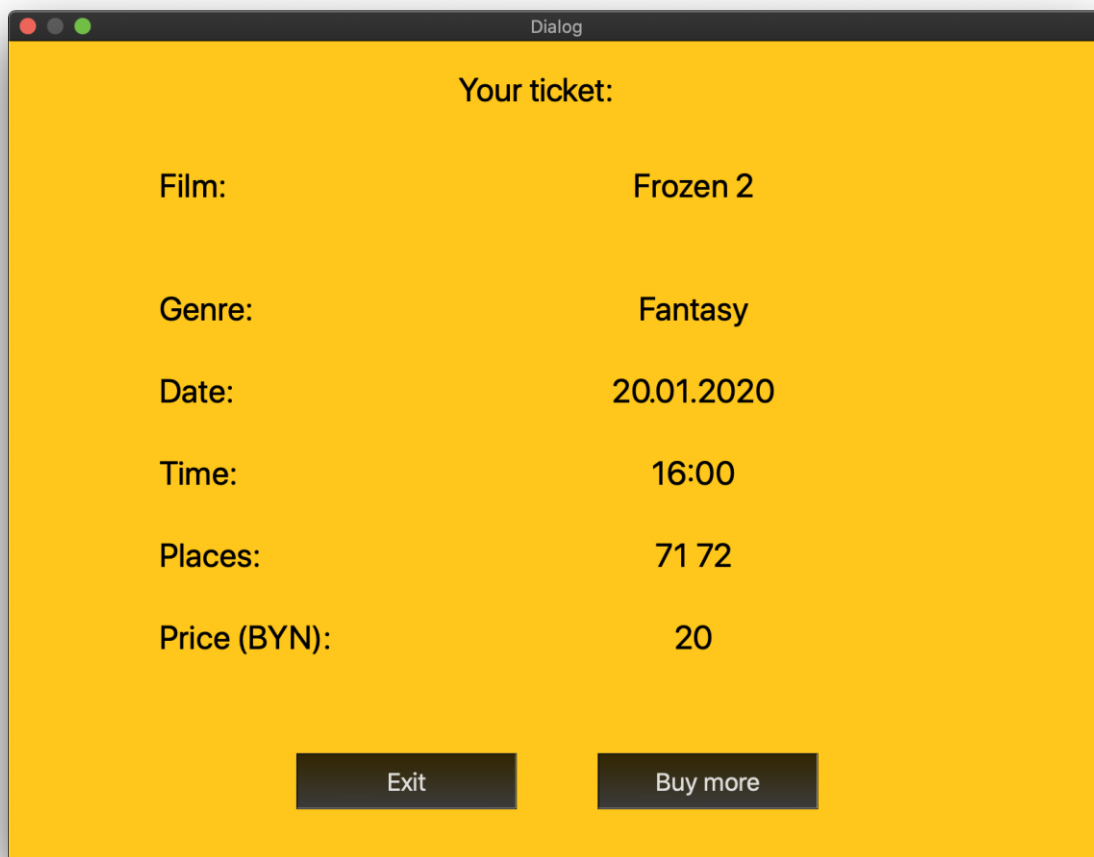


Рисунок 7.9 – Окно с информацией о приобретенных билетах

Данное окно имеет информацию о приобретенных билетах: название фильма, жанр, дата, время, места и цена билета. Также здесь расположены две кнопки, выполняющие следующие действия:

- «Exit» – закрытие окна и выход из программы;
- «Buy more» – переход обратно к окну выбора даты (рисунок 7.3).

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта была разработана программа оператора кинотеатра позволяющая хранить данные о сеансах фильмов, а также покупать билеты в кинотеатр. В программе реализованы операции добавления, удаления, редактирования информации, отмена последнего действия. Данная система поддерживает англоязычный интерфейс.

Были закреплены знания в области объектно-ориентированного программирования, получены навыки разработки в среде Qt Creator. Для разработки использовалась среда Qt Creator 4.10.2 на базе операционной системы macOS 10.15 Catalina.

К достоинствам программы можно отнести простой и понятный пользовательский интерфейс, возможность отмены последнего действия и схему зрительного зала для более удобного выбора мест. В дальнейшем планируется усовершенствование программы путём добавления поддержки русского языка, более подробной информации о фильмах, новых функций и улучшение пользовательского интерфейса.

Минимальные системные требования:

- Операционная система macOS 10.12 Sierra;
- Процессор Intel Core i5;
- Оперативная память 500 MB;
- Свободное место на жестком диске: 5 Мб;

ЛИТЕРАТУРА

[1] Шилдт, Герберт C++: базовый курс, 3-е издание. : Пер. с англ. — М.: Издательский дом "Вильямс", 2014. — 624 с. : ил. Парал. тит. англ.

[2] Луцик, Ю. А. Объектно-ориентированное программирование на языке C++ : учеб. пособие / Ю. А. Луцик, В. Н. Комличенко. — Минск : БГУИР, 2008. — 266 с. : ил.

[3] Лафоре, Роберт Объектно-ориентированное программирование в C++ / Роберт Лафоре ; пер.: А. Кузнецов, М. Назаров, В. Шрага. - 4-е изд. - Санкт-Петербург [и др.] : Питер, 2017. - 923 с.

[4] Шлее М. Qt 4.8. Профессиональное программирование на C++. — СПб.: БХВ-Петербург, 2012. — 912 с.: ил. — (В подлиннике)

ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема алгоритма добавления элемента в конец очереди

ПРИЛОЖЕНИЕ В

(обязательное)

Схема алгоритма удаления элемента очереди по индексу

ПРИЛОЖЕНИЕ Г

(обязательное)

Код программы

ПРИЛОЖЕНИЕ Д
(обязательное)
Ведомость документов