МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное   
учреждение высшего образования

**"Южно-Уральский государственный университет**

**(национальный исследовательский университет)"**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

**ОТЧЕТ   
по учебной практике**

бакалавра направления 09.03.04 "Программная инженерия"

|  |  |
| --- | --- |
|  | Выполнил:\_\_\_\_\_\_\_\_\_\_\_\_ студент группы КЭ-203  Землянкина Я.А.  Проверил:\_\_\_\_\_\_\_\_\_\_\_\_\_  Преподаватель кафедры СП  Глизница М.Н.  Дата:\_\_\_\_\_\_\_\_, Оценка:\_\_\_\_\_\_\_\_\_\_ |

Челябинск-2022

Министерство науки и высшего образования Российской Федерации

Южно-Уральский государственный университет

Кафедра системного программирования

УТВЕРЖДАЮ

Зав. кафедрой

системного программирования

\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский

**ЗАДАНИЕ**

**по** **учебной практике**

1. **Цель работы**

Разработать GUI-приложение, работающее с входной информацией, вводимой пользователем с помощью управляемых элементов формы, либо из текстового файла.

1. **Исходные данные к работе**
2. База данных турагенств, содержащая поля: название турагенства, страна, город, условия проживания и проезда, сервис принимающей стороны, стоимость путевки.
3. Данные хранятся в файле в формате JSON.
4. Данные, которые хранятся во входном файле: id, Name, Country, City, Accommodation, Passage, Service, Cost.
5. **Перечень подлежащих разработке вопросов**
6. Определение структуры приложения (по модулям), структур данных, используемых для хранения основной пользовательской информации.
7. Дизайн оконного интерфейса, анализ структуры входных данных и их защита от некорректного ввода информации.
8. Разработка основного функционала приложения: основных форм и механизмов получения информации из их компонентов и их файлов; основного алгоритма функционирования приложения; тестирование приложения.
9. Подготовка руководства пользователя и документации для программиста.
10. **Сроки**

Дата выдачи задания: "27" июня 2022 г.

Срок сдачи законченной работы: "23" июля 2022 г.

**Руководитель:**

Преподаватель. кафедры СП \_\_\_\_\_\_\_\_\_\_\_\_\_ Глизница М.Н.

должность, ученая степень подпись ФИО руководителя

**Задание принял к исполнению:**

\_\_\_\_\_\_\_\_\_\_\_\_\_ Землянкина Я.А.

подпись ФИО студента

**ОГЛАВЛЕНИЕ**

[1. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc108979076)

[2. ДИЗАЙН ОКОННОГО ИНТЕРФЕЙСА 7](#_Toc108979077)

[3. РАЗРАБОТКА ФУНКЦИОНАЛА ОСНОВНЫХ ФОРМ И МЕХАНИЗМОВ ПОЛУЧЕНИЯ ИНФОРМАЦИИ 13](#_Toc108979078)

[3.1. Функционал стартового окна 13](#_Toc108979079)

[3.2. Функционал окна для поиска 13](#_Toc108979080)

[3.3. Функционал окна для редактирования 14](#_Toc108979081)

[4. РАЗРАБОТКА ОСНОВНОГО МЕХАНИЗМА ФУНКЦИОНИРОВАНИЯ ПРИЛОЖЕНИЯ 16](#_Toc108979082)

[5. ТЕСТИРОВАНИЕ 19](#_Toc108979083)

[5.1. Автономное тестирование 19](#_Toc108979084)

[5.2. Комплексное тестирование 20](#_Toc108979085)

[6. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 22](#_Toc108979086)

[6.1. Минимальные требования к системе 22](#_Toc108979087)

[6.2. Запуск приложения 22](#_Toc108979088)

[6.3. Способы ввода и вывода информации 22](#_Toc108979089)

[6.4. Описание всех возможностей программы 23](#_Toc108979090)

[ЗАКЛЮЧЕНИЕ 26](#_Toc108979091)

[ЛИТЕРАТУРА 27](#_Toc108979092)

# ПОСТАНОВКА ЗАДАЧИ

Требуется написать программу для работы с базой турагенств, в которую занесена стоимость каждого тура. Формат записи базы данных: страна и город отдыха, условие проживания и проезда, сервис принимающей стороны, стоимость путевки. Программа должна обеспечивать поиск по базе по заданным критериям, позволять редактировать и дополнять базу.

Записи базы данных будут иметь следующие типы (табл.1).

Табл. 1. Переменные, используемые для хранения записей базы данных

|  |  |  |
| --- | --- | --- |
| Переменная | Тип переменной | Содержательный смысл |
| id | int | Индекс записи |
| TravelAgencyName | string | Название турагентсва |
| Country | string | Страна |
| City | string | Город |
| Accommodation | string | Условие проживания |
| Passage | string | Условие проезда |
| HostService | string | Cервис принимающей стороны |
| Cost | float | Cтоимость путевки |

Можно реализовать решение данной задачи с помощью односвязного списка. Каждый его элемент будет включать информационную часть (переменные из табл. 1) и указатель на следующий элемент. Признаком конца списка служит нулевой указатель.

В модуле ListTravelAgency содержится структура данных SData (Листинг 1) и класс списка ListTravelAgency (Листинг 2)

Листинг 1. Структура данных

struct SData {

int id;

string name;

string contry;

string city;

string accommodation;

string passage;

string service;

float cost;

};

Листинг 2. Структура данных

class ListTravelAgency

Node\* head; // Корень списка

int count = 0; // Количество узлов списка

Node\* Prev(Node\*); // Переход к предыдущему узлу

string Split(string); // Убрать лишнее у искомой строки

public:

ListTravelAgency() { head = NULL; }

~ListTravelAgency();

int Count() { return count; } // Получение количества узлов списка

bool IsEmpty() { return head == NULL; } // Проверка, пуст ли список

Node\* GetNode(int); // Получение значения узла списка

Node\* GetFirst() { return head; } // Получение корневого узла списка

Node\* GetLast(); // Получение последнего узла списка

Node\* Next(Node\*); // Переход к следующему узлу

Node\* Add(SData val, Node\* node = NULL); // Добавление узла списка

void Edit(int i, SData val); // Изменение значения узла списка

Node\* Delete(int); // Удаление узла списка

int GetLastId(); // вернуть последний занятый id

bool SaveByJSON(); // Сохранение с помощъю JSON

bool LoadByJSON(); // Загрузка с помощъю JSON

json ToJSON(SData); // Преобразовать в JSON

SData FromJSON(json const); // Преобразовать из JSON

};

Разрабатываемое приложение состоит из четырех оконных форм:

1. Стартовая окно (окно с заставкой приложения).
2. Окно редактирования (редактирование пользователем полей базы данных).
3. Окно для поиска по базе данных.
4. Окно для вывода отчета

Каждой из разработанных оконных форм соответствует пара файлов (\*.h и \*.cpp), приведенные в Таблице 2. Там же приведена информация о файле ListTravelAgency.cpp, в котором содержится информация о классе, реализующем работу однонаправленного списка в рассмотренной задаче.

Табл. 2. Модули создаваемого проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Имя файла | Описание информации, содержащейся в нем | Функциональное назначение | Файлы проекта, подключенные к текущему файлу посредством директивы #include |
| ListTravelAgency.cpp | Класс списка, определение методов класса | Работа с данными списка | ListTravelAgency.h |
| FFind.h | Окно для поиска по заданным параметрам | Поиск по базе данных | FMainMenu.h  FReport.h |
| FMainMenu.h | Стартовое окно | Отображение стартового окна программы | FEdit.h  FFind.h |
| FEdit.h | Окно редактирования | Отображение информации по запросу пользователя | FMainMenu.h  ListTravelAgency.h |
| FReport.h | Окно отчета | Отображение искомых записей | ListTravelAgency.h |

На Рисунке 1 представлена иллюстрация структуры программы.



Рис. 1. Структура программы

# ДИЗАЙН ОКОННОГО ИНТЕРФЕЙСА

Для реализации стартового окна используется можно использовать следующую форму (рис.2).

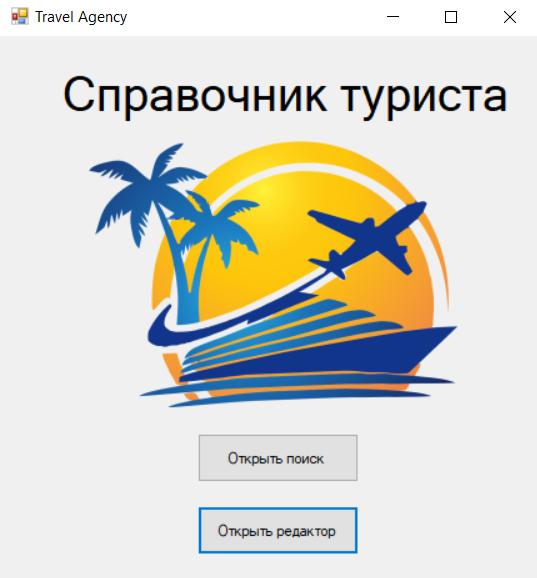


Рис. 2. Оформление формы стартового окна приложения

Размещенные на форме компоненты и перечень методов и событий, которые необходимо реализовать приведен в табл. 3.

Табл. 3. Компоненты стартового окна приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя компоненты | Тип | Ограничения для ввода информации | Реализованные события | Функциональное назначение |
| FindButtom | button | Инициализация открытия формы для поиска | Click | При нажатии на кнопку открывается форма для поиска |
| EditorButton | button | Инициализация открытия формы для редактирования базы данных | Click | При нажатии на кнопку открывается форма для редактирования базы данных |

Для реализации поиска в базе данных турагенств можно воспользоваться следующей формой (рис. 3).

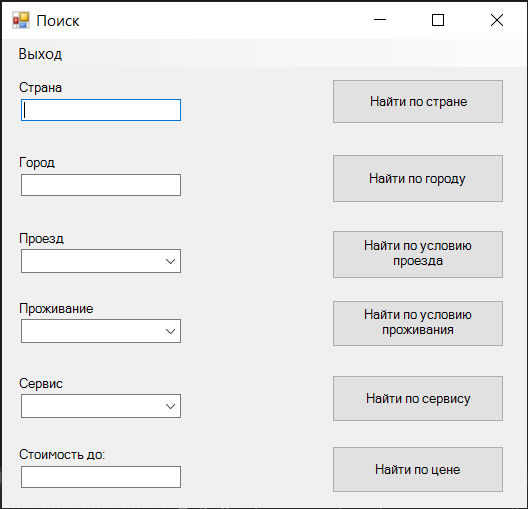


Рис. 3. Оформление окна для поиска

Размещенные на форме компоненты и перечень методов и событий, которые необходимо реализовать приведен в табл. 4.

Табл. 4. Компоненты окна для поиска по базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя компоненты | Тип | Ограничения для ввода информации | Реализованные события | Функциональное назначение |
| CountryText | textBox | Строка, вводимая пользователем | нет | Пользователь задает желаемую страну тура |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CityText | textBox | Строка, вводимая пользователем | нет | Пользователь задает желаемый город тура |
| PassageBox | comboBox | Значение из списка в свойстве Items | SelectedIndexChanged | Выбор условия проезда из списка и сохранение значения во временной переменной |
| AccommodationBox | comboBox | Значение из списка в свойстве Items | SelectedIndexChanged | Выбор условия проживания из списка и сохранение значения во временной переменной |
| ServiceBox | comboBox | Значение из списка в свойстве Items | SelectedIndexChanged | Выбор сервиса из списка и сохранение значения во временной переменной |
| PriceText | textBox | Ввод с клавиатуры только цифр | KeyPress (для проверки вводимой информации) | Пользователь задает желаемый стоимость тура |
| CountryButton | button | Инициализация поиска тура с заданной страной | Click | При нажатии на кнопку предлагается выбрать файл, в котором хранятся данные базы |
| CityButton | button | Инициализация поиска тура с заданным городом | Click | При нажатии на кнопку осуществляется поиск данных в списке |
| PassageButton | button | Инициализация поиска тура с заданной условием проезда | Click | При нажатии на кнопку осуществляется поиск данных в списке |
| AccommodationButton | button | Инициализация поиска тура с заданным условием проживания | Click | При нажатии на кнопку осуществляется поиск данных в списке |
| ServiceButton | button | Инициализация поиска тура с заданным сервисом | Click | При нажатии на кнопку осуществляется поиск данных в списке |
| PriceButton | button | Инициализация поиска тура с заданной ценой | Click | При нажатии на кнопку осуществляется поиск данных в списке |

Для реализации работы с базой турагенств можно воспользоваться следующей формой. На ней расположены две вкладки – «Просмотр» и «Редактирование» (рис. 4-5).

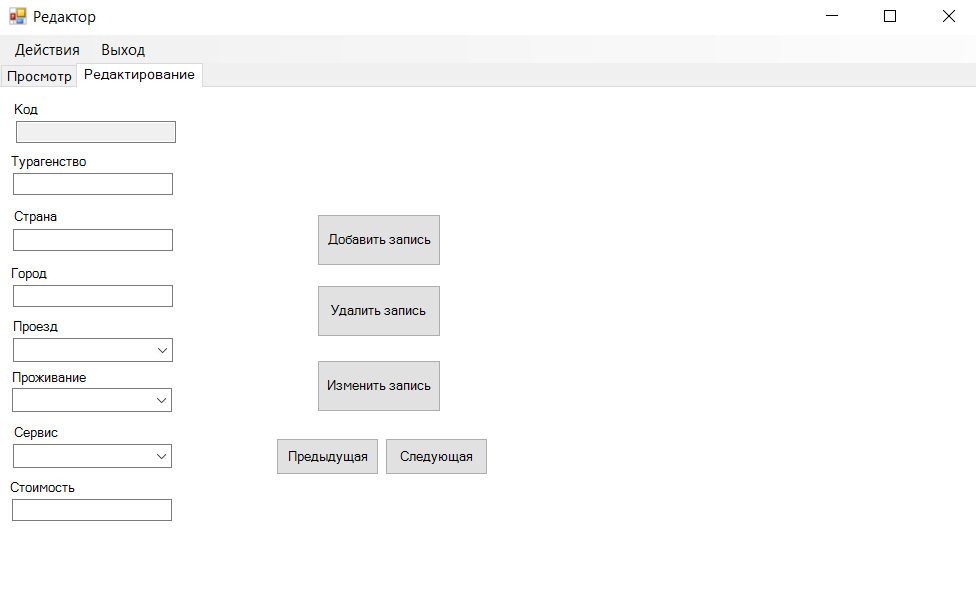


Рис. 4. Редактирование

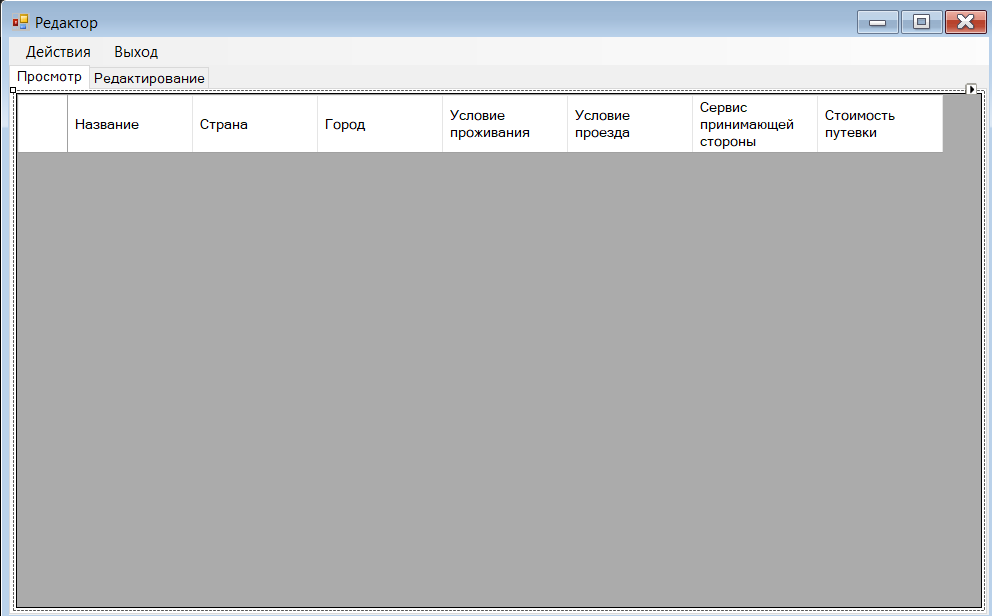


Рис. 5. Просмотр

Размещенные на форме компоненты и перечень методов и событий, которые необходимо реализовать приведен в табл.5.

Табл. 5. Компоненты окна редактирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя компоненты | Тип | Ограничения для ввода информации | Реализованные события | Функциональное назначение |
| textBoxId | textBox | Ввод запрещен | нет | Вывод кода узла списка |
| textBoxName | textBox | Строка, вводимая пользователем | нет | Ввод и вывод названия турагенства |
| textBoxCountry | textBox | Строка, вводимая пользователем | нет | Ввод и вывод страны |
| textBoxCity | textBox | Строка, вводимая пользователем | нет | Ввод и вывод города |
| comboBoxTravel | comboBox | Строка, вводимая пользователем | нет | Ввод и вывод условия проезда |
| comboBoxAccommodation | comboBox | Строка, вводимая пользователем | нет | Ввод и вывод условия проживания |
| comboBoxService | comboBox | Строка, вводимая пользователем | нет | Ввод и вывод сервиса |
| textBoxCost | textBox | Ввод с клавиатуры только цифр | KeyPress (для проверки вводимой информации) | Ввод и вывод стоимости путевки |
| buttonAdd | button | Инициализация добавления нового поля в базу данных | Click | При нажатии на кнопку осуществляется добавление новой записи |
| buttonDelete | button | Инициализация удаления поля из базы данных | Click | При нажатии на кнопку осуществляется удаление поля |
| buttonSave | button | Инициализация сохранения поля в базу данных | Click | При нажатии на кнопку осуществляется сохранение поля |
| buttonPrevious | button | Переход к предыдущей записи | Click | При нажатии на кнопку осуществляется отображение предыдущей записи в базе данных |
| buttonNext | button | Переход к следующей записи | Click | При нажатии на кнопку осуществляется отображение следующей записи в базе данных |
| dataGridView | DataGridView | Данные из файла | Нет | Отображение данных в табличной форме |

Алгоритмы и программная реализация приведенных в табл. 3-5 событий и методов приведены в следующем разделе.

# РАЗРАБОТКА ФУНКЦИОНАЛА ОСНОВНЫХ ФОРМ И МЕХАНИЗМОВ ПОЛУЧЕНИЯ ИНФОРМАЦИИ

В данном разделе приводится структурная схема, указывающая связь между вводимыми/выводимыми данными (на форме) и объектами, используемыми для их хранения.

## Функционал стартового окна

При нажатии на кнопку «Открыть поиск» вызывается метод Click, который открывает форму для поиска. Исходный код функции приведен в Листинге 3.

Листинг 3. Реализация события Click кнопки FindButton

private: System::Void FindButton\_Click(System::Object^ sender, System::EventArgs^ e) {

Find^ start = gcnew Find();

start->Show();

}

Аналогичным образом обрабатывается событие кнопки «Открыть редактор», открывая форму для редактирования.

## Функционал окна для поиска

Пользователь выбирает из выпадающего списка PassageBox условие проезда и нажимает на кнопку PassageButton, которая вызывает событие Click. Соответствующее значение типа String^ передается в метод Search(). Исходный код функции приведен в Листинге 4.

Листинг 4. Реализация события Click кнопки PassageButton

private: System::Void PassageButton\_Click(System::Object^ sender, System::EventArgs^ e) {

if (textBoxCountry->Text == "") {

MessageBox::Show("Не все поля введены!", "Внимание!");

return;

}

FReport^ form = gcnew FReport();

form->Show();

form->Search(1, textBoxPassage->Text);

}

Событие Click элементов CountryButton, CityButton, AccommodationButton, ServiceButton, PriceButton реализовано аналогичным способом.

При заполнении стоимости в текстовом поле texyBoxCost вызывается метод KeyPress, который проверяет вводимую информацию и запрещает вводить все символы, кроме цифр и служебных клавиш. Исходный код функции приведен в Листинге 5.

Листинг 5. PriceText\_KeyPress

private: System::Void texyBoxCost \_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

if (!Char::IsControl(e->KeyChar) && !Char::IsDigit(e->KeyChar))

{

e->Handled = true;

return;

}

## Функционал окна для редактирования

Пользователь заполняет все поля о турагенстве и нажимает на кнопку buttonAdd, которая вызывает событие Click. Текст из полей передается в функцию Add(), которая добавляет новую запись в базу данных. Исходный код представлен на Листинге 6.

Листинг 6. battonAdd\_Click

System::Void TravelAgency::FEdit::buttonAdd\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (textBoxCity->Text == "" || textBoxName->Text == "" || textBoxCountry->Text == "" || textBoxCost->Text == "" ||

comboBoxAccommodation->SelectedIndex == -1 || comboBoxService->SelectedIndex == -1 || comboBoxTravel->SelectedIndex == -1) {

MessageBox::Show("Не все поля введены!","Внимание!");

return;

}

SData d;

d.accommodation = ConvertToString(comboBoxAccommodation->SelectedItem->ToString());

d.service = ConvertToString(comboBoxService->SelectedItem->ToString());

d.passage = ConvertToString(comboBoxTravel->SelectedItem->ToString());

d.name = ConvertToString(textBoxName->Text);

d.city = ConvertToString(textBoxCity->Text);

d.contry = ConvertToString(textBoxCountry->Text);

d.cost = Convert::ToSingle(textBoxCost->Text);

listTA.Add(d);

MessageBox::Show("Данные добавлены!", "Внимание!");

PrintData();

}

Пользователь выбирает из списка узел, который требуется удалить, и нажимает кнопку battonDelete. Событие вызывает метод Delete() и удаляет элемент по передаваемому индексу. Исходный код представлен на Листинге 7.

Листинг 7. battonDelete\_Click

System::Void TravelAgency::FEdit::buttonDelete\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (MessageBox::Show("Продолжить удаление?", "Внимание!", MessageBoxButtons::YesNo) == Windows::Forms::DialogResult::Yes) {

listTA.Delete(selectedIndexData);

MessageBox::Show("Данные удалены!", "Внимание!");

PrintData();

}

}

Пользователь выбирает из списка узел, который требуется корректировать, и нажимает на кнопку battonEdit. Данные, которые были введены в текстовые поля, из типа String^ конвертируются в тип string и передаются в метод Edit(). Исходный код представлен на Листинге 8.

Листинг 8. battonEdit\_Click

System::Void TravelAgency::FEdit::buttonEdit\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (textBoxCity->Text == "" || textBoxName->Text == "" || textBoxCountry->Text == "" || textBoxCost->Text == "" ||

comboBoxAccommodation->SelectedIndex == -1 || comboBoxService->SelectedIndex == -1 || comboBoxTravel->SelectedIndex == -1) {

MessageBox::Show("Не все поля введены!", "Внимание!");

return;

}

SData d;

d.accommodation = ConvertToString(comboBoxAccommodation->SelectedItem->ToString());

d.service = ConvertToString(comboBoxService->SelectedItem->ToString());

d.passage = ConvertToString(comboBoxTravel->SelectedItem->ToString());

d.name = ConvertToString(textBoxName->Text);

d.city = ConvertToString(textBoxCity->Text);

d.contry = ConvertToString(textBoxCountry->Text);

d.cost = Convert::ToSingle(textBoxCost->Text);

listTA.Edit(selectedIndexData, d);

MessageBox::Show("Данные изменены!", "Внимание!");

PrintData();

}

# РАЗРАБОТКА ОСНОВНОГО МЕХАНИЗМА ФУНКЦИОНИРОВАНИЯ ПРИЛОЖЕНИЯ

Для загрузки JSON файла используется метод LoadByJSON, представленный в Листинге 9. Метод открывает файл и записывает данные в поля класса ListTravelAgency, меняя их тип на string.

Листинг 9. LoadByJSON

bool ListTravelAgency::LoadByJSON()

{

ifstream load("data.json");

json j;

load >> j;

if (!IsEmpty()) {

Clear();

}

int countData = j["count"];

for (int i = 1; i <= countData; i++) {

SData d;

d = FromJSON(j[to\_string(i)]);

Add(d);

}

return !IsEmpty();

}

Далее описывается работа с экземплярами класса ListTravelAgency, являющимися элементами односвязного списка.

В Листинге 10 описана функция добавления элемента в список. Добавление подразумевает проверку наличия элементов в списке, и в случае их отсутствия добавляемый элемент назначается первым, иначе он добавляется в конец.

Листинг 10. Add

Node\* ListTravelAgency::Add(SData val, Node\* node)

{

val.id = GetLastId() + 1;

Node\* elem = new Node();

elem->value = val;

elem->ptr = NULL;

count++;

if (node == NULL) // Добавление нового корня

{

// В начало, если нет

if (head == NULL) {

elem->ptr = NULL;

head = elem;

}

else { // иначе в конец

GetLast()->ptr = elem;

}

return elem;

}

// Добавление узла после текущего

elem->ptr = node->ptr;

node->ptr = elem;

return elem;

}

Листинг 11 представляет собой метод удаления элемента по заданному значению. В метод передается индекс значения, удаляется элемент и возвращается указатель на узел, следующий за удаляемым. Если удаляемый элемент находится в начале списка, то возвращаемым значением будет новый корень списка — тот узел, на который указывает удаляемый элемент.

Листинг 11. Delete

Node\* ListTravelAgency::Delete(int i)

{

if (i < 0 || i >= count) {

throw "Индекс вне диапазона!";

}

Node\* node = head;

while (node != NULL)

{

if (i == 0)

break;

node = node->ptr;

i--;

}

// В списке нет узлов

if (node == NULL) {

return NULL;

}

count--;

// Удаление корневого узла

if (node == head)

{

head = node->ptr;

delete node;

return head;

}

// Удаление промежуточного узла

Node\* prev = Prev(node);

prev->ptr = node->ptr;

delete node;

return prev;

}

В Листинге 12 описана функция редактирования записи. В метод передается индекс и значение из поля ввода. Функция производит поиск элемента из списка и заменяет его.

Листинг 12. Edit

void ListTravelAgency::Edit(int i, SData val)

{

if (i < 0 || i >= count) {

throw "Индекс вне диапазона!";

}

int id = GetNode(i)->value.id;

Node\* node = head;

while (node != NULL)

{

if (i == 0)

break;

node = node->ptr;

i--;

}

node->value = val;

node->value.id = id;

}

Листинг 13 демонстрирует поиск по заданному значению. Данный метод является перегруженным т.к. в коде реализован в двух вариантах. В листинге приведен метод, в который передается значения типа float. Второй метод имеет схожую реализацию.

Листинг 13. Search

void TravelAgency::FReport::Search(float cost)

{

Node\* p = listBySearch.GetFirst();

int q = 0;

bool found = false;

while (p != NULL) {

if (p->value.cost <= cost) {

found = true;

dataGridView->Rows->Add(ConvertToString(p->value.name), ConvertToString(p->value.contry), ConvertToString(p->value.city),

ConvertToString(p->value.accommodation), ConvertToString(p->value.travel), ConvertToString(p->value.service), p->value.cost.ToString());

dataGridView->Rows[q++]->HeaderCell->Value = p->value.id.ToString();

}

p = p->ptr;

}

}

# ТЕСТИРОВАНИЕ

Проведем автономное и комплексное тестирование разработанной программы.

## Автономное тестирование

Автономное тестирование показывает, как работают функции, приведенные в главе 4.

Тестирование операции чтения файла представлено в Табл. 6.

Табл. 6. Тестирование операции чтения файла

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Верно ли? |
| data.json | true | Да |
| Файл с некорректным вводом данных | Скриншот ошибки представлен в рис. 6. | Да |

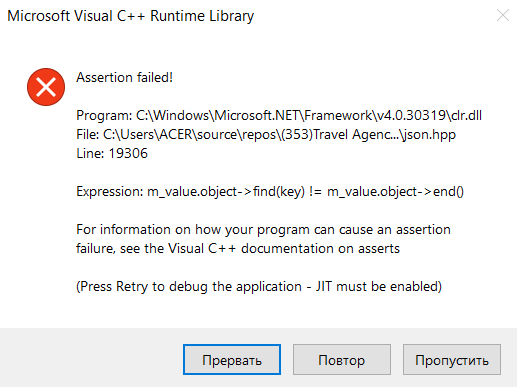


Рис. 6. Ошибка

Тестирование операции редактирования представлено в Табл. 7.

Табл. 7. Тестирование операции редактирования

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Верно ли? |
| id | Производится изменение узла списка | Да |
| textBoxPrice->Text = «123a» | System.FormatException: "Входная строка имела неверный формат." | Да |

Тестирование операции удаления представлено в Табл. 8.

Табл. 8. Тестирование операции удаления

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Верно ли? |
| id | head | Да |
| id = -1 | Индекс вне диапазона! | Да |

Тестирование операции вставки представлено в Табл. 9.

Табл. 9. Тестирование операции вставки

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Верно ли? |
| SData val | elem | Да |
| textBoxPrice->Text = «123a» | System.FormatException: "Входная строка имела неверный формат." | Да |

Тестирование операции поиска представлено в Табл. 10.

Табл. 10. Тестирование операции поиска

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Верно ли? |
| textBoxCountry->Text = «Турция» | Искомое значение найдено и помещено в отчет. | Да |
| textBoxCountry->Text = «Доминикана» | Выводится окно с предупреждением о том, что совпадения не найдены. | Да |

## Комплексное тестирование

В данном разделе приводятся протоколы тестирования приложения (табл. 11)

Табл. 11. Тестирование головного модуля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Описание ситуации | Входные данные | Выходные данные | Тест пройден? |
| 1. | Программа работает корректно | Корректные данные в текстовых полях | Функции отработали правильно | Да |
| 2. | Файл загрузился | data.json | Файл загрузился в таблицу просмотра | Да |
| 3. | Пользователь ввел значения не во все поля | SData val | Выводится окно с текстом «Не все поля введены!» | Да |
| 4. | Невозможно найти ключ по значению | textBox | Выводится окно с текстом «Совпадений не найдено!» | Да |
| 5. | Входной файл не открывается | data.json с некорректным вводом данных | Assertion failed! | Да |
| 6. | Неправильный тип значения | textBox | System.FormatException: "Входная строка имела неверный формат." | Да |
| 7. | Пользователь не сохранил данные | data.json | Данные в файле не сохранились. | Да |
| 8. | Пользователь пытается удалить несуществующий элемент | id | Индекс вне диапазона! | Да |
| 9. | Невозможно открыть заставку приложения | logotip.png | Окно без картинки | Да |
| 10. | Невозможно закрыть программу | Клик по кнопке закрытия | Программа не закрылась | Да |

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Минимальные требования к системе

* Windows XP;
* Процессор с тактовой частотой не ниже 1,8 ГГц;
* 2 ГБ ОЗУ;
* Свободное место на жестком диске: 100 МБ.

## Запуск приложения

Для запуска приложения необходимо открыть файл Travel\_Agency.exe. После этого загрузится начальное окно «Travel Agency» (рис. 7), с которого можно начать дальнейшую работу.

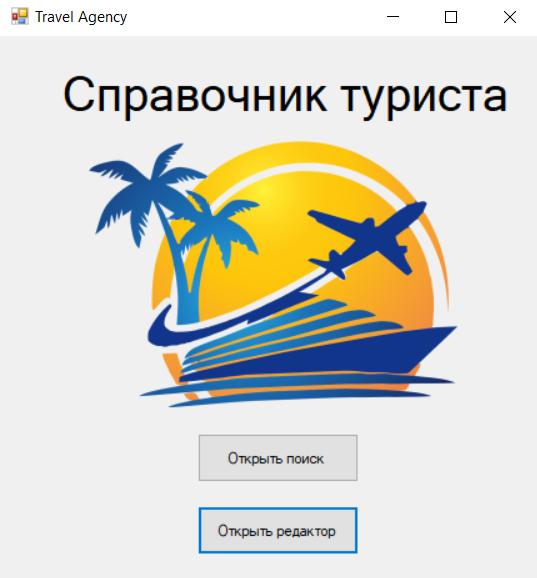


Рис. 7. Начальное окно

## Способы ввода и вывода информации

База данных содержится в файле data.json.

Чтобы загрузить базу данных, нужно в окне «Редактор» выбрать в меню кнопку «Действие», в выпадающем списке выбрать «Загрузить»

После работы с базой данных требуется сохранить все изменения. Для этого нужно в окне «Редактор» выбрать в меню пункт «Действие», в выпадающем списке выбрать «Сохранить». Данные сохранятся в исходный файл.

## Описание всех возможностей программы

На начальном окне (рис. 7) можно выбрать поиск по базе данных и редактор.

Нажимая на кнопку «Открыть поиск», открывается окно для поиска по ключевому слову (рис. 8).

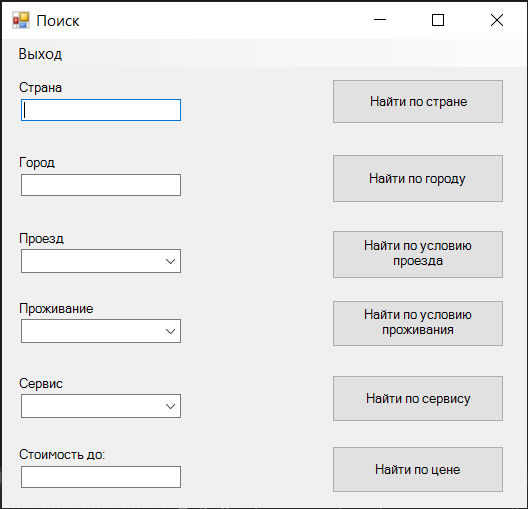


Рис. 8. Окно для поиска

Для того, чтобы найти необходимые записи в базе данных, требуется написать слово-ключ в соответствующее поле ввода и нажать на кнопку рядом. В полях «Проезд», «Проживание», «Сервис» есть выпадающий список, откуда выбираются значения. После нажатия кнопки появляется окно с таблицей искомых турагенств.

Для того, чтобы закрыть окно, используется пункт «Выход» в меню. Чтобы закрыть программу необходимо нажать на кнопку «Выход из программы». Для перехода на начальное окно соответствует кнопка «Назад». Для подтверждения выхода в всплывающем окне нужно нажать «Да».

Нажимая на кнопку «Открыть редактор», открывается окно для редактирования базы данных (рис. 9). На этом окне можно осуществить добавление новой записи, редактирование существующей записи и удаление.

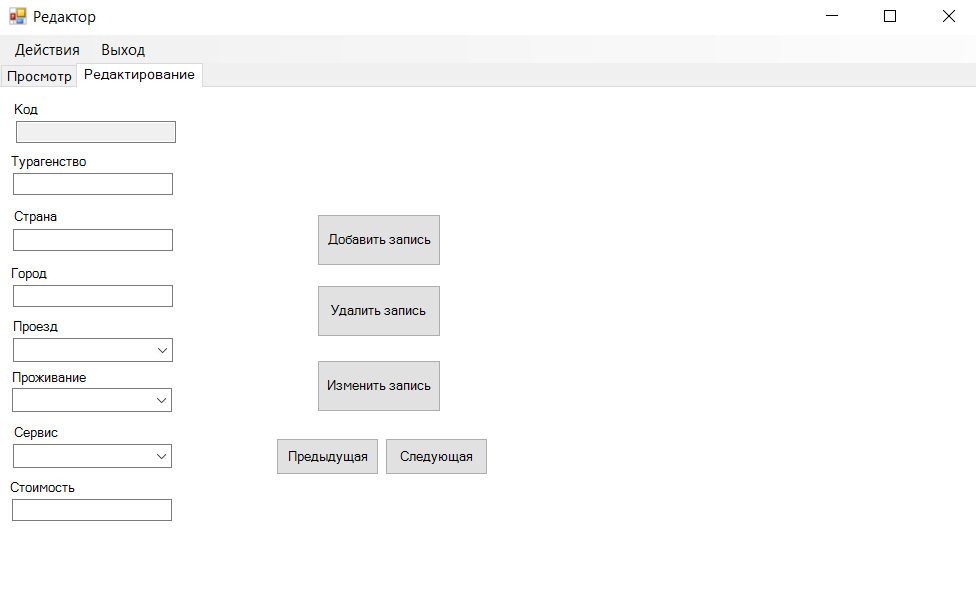


Рис. 9. Окно редактирования

Чтобы загрузить базу данных, нужно выбрать в меню кнопку «Действие», в выпадающем списке выбрать «Загрузить»

Для того, чтобы добавить новую запись, необходимо заполнить все поля и нажать на кнопку «Добавить запись»

Для перемещения по записям данных используются кнопки управления «Предыдущая» и «Следующая».

Чтобы удалить запись, необходимо выбрать нужную и нажать на кнопку «Удалить запись».

Для редактирования записи базы данных нужно выбрать желаемую запись, изменить значения и нажать на кнопку «Изменить запись»

После работы с базой данных требуется сохранить все изменения. Для этого нужно выбрать в меню пункт «Действие», в выпадающем списке выбрать «Сохранить». Данные сохранятся в исходный файл.

Для того, чтобы закрыть окно, используется пункт «Выход» в меню. Чтобы закрыть программу необходимо нажать на кнопку «Выход из программы». Для перехода на начальное окно соответствует кнопка «Назад». Для подтверждения выхода в всплывающем окне нужно нажать «Да»

## Перечень ошибок и их устранения

Не все поля введены (рис. 10):

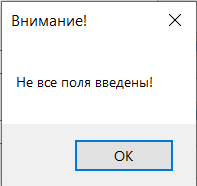


Рис. 10. Окно с предупреждением

Чтобы корректно работать с программой, необходимо заполнять все поля ввода.

Вписывание своих значений в поля с выпадающим списком:

В полях с выпадающим списком нужно выбирать только те значения, которые там имеются

Изменения не сохранены в файле с базой данных:

Чтобы изменения сохранились, нужно обязательно сохранить файл в соответствующем пункте меню.

# ЗАКЛЮЧЕНИЕ

Данная работа посвящена разработке оконного приложения для работы с базой данных туристических агенств с использованием языка C++. Для хранения данных использованы объекты класса ListTravelAgency, являющегося односвязным списком, приведенного в задании. При организации функционирования класса была создана перегруженная функция Search, используемая для поиска элемента списка с ключами разных типов, а также с различными наборами ключевых полей.

В ходе выполнения работы были пройдены все основные этапы разработки программного обеспечения: анализ, написание спецификации, проектирование, разработка алгоритмов, кодирование, тестирование и сопровождение. Проведено автономное тестирование основных методов разработанного класса и комплексное тестирование программы в целом. В результате тестирования ошибок не обнаружено. В данных пунктах описана проделанная в каждом случае работа и ее результаты.

В дальнейшей разработке могут быть добавлены новые элементы базы данных, также может быть улучшен интерфейс приложения для более удобной работы.

# ЛИТЕРАТУРА

1. Панюкова Т.А., Панюков А.В. Языки и методы программирования: Путеводитель по языку С++. Учебное пособие. – М.: Книжный дом «Либроком», 2013. – 216 с.
2. Макаровских Т.А., Панюков А.В. Языки и методы программирования: Создание простых GUI-приложений с помощью Visual C++: Учебное пособие. Изд. 2-е. – М.: «Ленанд», 2018. – 144 с.
3. Панюкова Т.А. Документирование программного обеспечения: В помощь техническому писателю: Учебное пособие. – М.: Книжный дом «Либроком», 2012. – 264 с.
4. Рязанова, Н. Ю. Программирование на языке С++ в среде Visual Studio CLR Windows Forms : учебное пособие / Н. Ю. Рязанова, К. Л. Тассов, М. В. Филиппов. — Москва : МГТУ им. Н.Э. Баумана, 2017. — 64 с. — ISBN 978-5-7038-4563-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/103509 (дата обращения: 28.05.2021). — Режим доступа: для авториз. пользователей.
5. Пахомов Б. И. C/C++ и MS Visual C++ 2012 для начинающих. — СПб.: БХВ-Петербург, 2013. – 512 с.   
   <https://karsu.uz/wp-content/uploads/2019/07/7.pdf>