Выполнил

студент группы КТбо1-2 А. С. Мумладзе

Принял

доцент кафедры САиТ В. С. Лапшин

Таганрог 2024

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

Кафедра системного анализа и телекоммуникаций

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине «Основы алгоритмизации и программирования»

**«РАЗРАБОТКА ПРИЛОЖЕНИЙ НА С++   
С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ»**

Вариант 3

# Содержание

Содержание 2

Техническое задание 3

Цель задания 3

Задача 3

Ход работы 4

Написание программы 4

Пример работы программы 5

Вывод 6

Листинг 7

DataManager.h 7

DataManager.cpp 8

MyForm.cpp 13

MyForm.h 13

MyFormFunc.cpp 20

# Техническое задание

## Цель задания

Цель задания: разработать приложение с графическим интерфейсом для Windows, используя библиотеку .NET (CLI-приложение).

## Задача

Задачи лабораторной работы:

1. Создать проект WinForms в среде Visual Studio;
2. Включить класс, разработанный в лабораторной работе №4, в созданный проект;
3. Разработать форму ввода данных, контроля введенной информации и вывода информации на экран. Функции работы с информацией взять из подключенного класса;
4. Отладить приложение.

# Ход работы

## Написание программы

Для создания WinForms приложения я воспользовался шаблоном CLR. Для этого я создал пустое приложение через «Решение проекта» - «Создать элемент» - «Среда CLR» - «Форма WinForms».

Далее через конструктор VS я добавил приложению кнопки Button, список ListView и текстовые поля TextBox. Visual Studio автоматически собрал необходимый код в MyForm.h, с которым можно работать далее.

Затем я добавил логику элементам интерфейса. Для этого я создал файл MyFormFunc.cpp, и в нем сделал определения всех функций-триггеров.

Логика программы следующая:

1. После запуска приложения пользователь должен выбрать категорию данных: книги, дисциплины, связи. Далее пользователь может делать следующее:
2. Пользователь может найти информацию. Для этого, в текстовое поле под кнопками выбора категорию он вводит искомую информацию. После нажатия кнопки «Поиск» программа ищет в списке совпадения, и в списке отображает найденное. Поиск происходит по всем полям данной категории.
   1. При первом нажатии кнопки «Поиск» происходит чтение файлов данных из папки Data. Если какой-то из файлов не может быть прочтен, пользователю в MessageBox высветится ошибка, что отсутствует(ют) файлы, и программа автоматически закроется.
3. Пользователь может добавить запись в список. Для этого он выбирает категорию, вписывает данные в текстовые поля справа, и нажимает кнопку «Добавить». Если не все требуемые поля будут заполнены, пользователь будет предупрежден об этом.
4. Пользователь может удалить данные из списка. Для этого, после совершения поиска (пункт 2) пользователь выбирает в списке записи для удаления (кликает на левую колонку с нужной(ыми) записью(ями)), и нажимает кнопку «Удалить»
   1. Если пользователь удаляет книгу/дисциплину, автоматически будут удалены и записи связей, которые содержат удаленные элементы. Пользователю не потребуется удалять вручную разорванные связи.
5. После завершения работы пользователь закрывает программу. В деструкторе приложения выполняется функция обновления текстовых файлов новыми данными.

## Пример работы программы

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

# Вывод

Итогом работы стало приложение WinForms с графическим интерфейсом, позволяющее управлять данными о книгах, дисциплинах и их взаимосвязях. Пользователь может найти данные, добавить/удалить запись.

В ходе работы я воспользовался реализованным классом DataManager из лабораторной работы №4, а также ознакомился с возможностями конструктора CLI приложения Visual Studio, и реализовал свое приложение с графическим интерфейсом.

# Листинг

## DataManager.h

1. #pragma once
2. using namespace System;
3. using namespace System::IO;
4. using namespace System::Collections;
5. using namespace System::Collections::Generic;
6. // Перечислений форматов данных (связь, книга, дисциплина)
7. enum DataType {
8. T\_LINK = 0,
9. T\_BOOK = 1,
10. T\_SUBJECT = 2
11. };
12. // Структура записи связи {книга, дисциплина}
13. ref struct Link {
14. String^ ISBN;
15. String^ dis\_code;
16. };
17. // Структура записи одной книги
18. ref struct Book {
19. String^ ISBN;
20. String^ title;
21. String^ author;
22. };
23. // Структура записи одной дисциплины
24. ref struct Subject {
25. String^ dis\_code;
26. String^ name;
27. String^ description;
28. };
29. // Класс, управляющий данными о книгах, дисциплинах и их связях
30. ref class DataManager {
31. private:
32. // Путь до файла Links.txt
33. String^ fileLinkPath;
34. // Путь до файла Books.txt
35. String^ fileBookPath;
36. // Путь до файла Subjects.txt
37. String^ fileSubjectPath;
38. // Список объектов Link
39. List<Object^>^ listLinks;
40. // Список объектов Book
41. List<Object^>^ listBooks;
42. // Список объектов Subject
43. List<Object^>^ listSubjects;
44. public:
45. // Конструктор класса
46. DataManager();
47. // Прочитать данные из файлов в списки
48. void ReadDataFiles();
49. // Записать данные из списков в файлы
50. void WriteDataFiles();
51. // Преобразовать строку данных в Object
52. Object^ MakeNode(DataType, String^);
53. // Добавить запись в список
54. void AddNode(DataType, Object^);
55. // Удалить запись из списка
56. void DeleteNode(DataType, Object^);
57. // Найти запись в списке
58. List<Object^>^ FindNodes(DataType, String^);
59. };

## DataManager.cpp

1. #include "DataManager.h"
2. using namespace System;
3. using namespace System::IO;
4. DataManager::DataManager() {
5. this->fileLinkPath = gcnew String("./Data/Links.txt");
6. this->fileBookPath = gcnew String("./Data/Books.txt");
7. this->fileSubjectPath = gcnew String("./Data/Subjects.txt");
8. this->listLinks = gcnew List<Object^>();
9. this->listBooks = gcnew List<Object^>();
10. this->listSubjects = gcnew List<Object^>();
11. Console::WriteLine("Создан объект класса DataManager");
12. }
13. void DataManager::ReadDataFiles() {
14. try {
15. StreamReader^ sr = gcnew StreamReader(this->fileLinkPath);
16. String^ line;
17. while ((line = sr->ReadLine()) != nullptr) {
18. this->AddNode(T\_LINK, this->MakeNode(T\_LINK, line));
19. }
20. sr->Close();
21. Console::WriteLine("Прочитан файл Links.txt");
22. }
23. catch (Exception^ e) {
24. Console::WriteLine("Ошибка, невозможно прочитать Links.txt\n" + e);
25. Environment::Exit(1);
26. }
27. try {
28. StreamReader^ sr = gcnew StreamReader(this->fileBookPath);
29. String^ line;
30. while ((line = sr->ReadLine()) != nullptr) {
31. this->AddNode(T\_BOOK, this->MakeNode(T\_BOOK, line));
32. }
33. sr->Close();
34. Console::WriteLine("Прочитан файл Books.txt");
35. }
36. catch (Exception^ e) {
37. Console::WriteLine("Ошибка, невозможно прочитать Books.txt\n" + e);
38. Environment::Exit(1);
39. }
40. try {
41. StreamReader^ sr = gcnew StreamReader(this->fileSubjectPath);
42. String^ line;
43. while ((line = sr->ReadLine()) != nullptr) {
44. this->AddNode(T\_SUBJECT, this->MakeNode(T\_SUBJECT, line));
45. }
46. sr->Close();
47. Console::WriteLine("Прочитан файл Subjects.txt");
48. }
49. catch (Exception^ e) {
50. Console::WriteLine("Ошибка, невозможно прочитать Subjects.txt\n" + e);
51. Environment::Exit(1);
52. }
53. }
54. void DataManager::WriteDataFiles() {
55. StreamWriter^ sw = nullptr;
56. sw = File::CreateText(this->fileLinkPath);
57. for each (Link^ link in (this->listLinks)) {
58. String^ text = String::Format("{0}; {1}", link->ISBN, link->dis\_code);
59. sw->WriteLine(text);
60. }
61. sw->Close();
62. Console::WriteLine("Файл Links.txt обновлен новыми данными");
63. sw = File::CreateText(this->fileBookPath);
64. for each (Book ^ book in (this->listBooks)) {
65. String^ text = String::Format("{0}; {1}; {2}", book->ISBN, book->title, book->author);
66. sw->WriteLine(text);
67. }
68. sw->Close();
69. Console::WriteLine("Файл Books.txt обновлен новыми данными");
70. sw = File::CreateText(this->fileSubjectPath);
71. for each (Subject ^ subject in (this->listSubjects)) {
72. String^ text = String::Format("{0}; {1}; {2}", subject->dis\_code, subject->name, subject->description);
73. sw->WriteLine(text);
74. }
75. sw->Close();
76. Console::WriteLine("Файл Books.txt обновлен новыми данными");
77. }
78. Object^ DataManager::MakeNode(DataType type, String^ line) {
79. Object^ node = nullptr;
80. array<String^>^ data = line->Split(';');
81. switch (type) {
82. case T\_LINK:
83. if (data->Length == 2) {
84. Link^ temp = gcnew Link();
85. temp->ISBN = data[0]->Trim();
86. temp->dis\_code = data[1]->Trim();
87. node = temp;
88. }
89. break;
90. case T\_BOOK:
91. if (data->Length == 3) {
92. Book^ temp = gcnew Book();
93. temp->ISBN = data[0]->Trim();
94. temp->title = data[1]->Trim();
95. temp->author = data[2]->Trim();
96. node = temp;
97. }
98. break;
99. case T\_SUBJECT:
100. if (data->Length == 3) {
101. Subject^ temp = gcnew Subject();
102. temp->dis\_code = data[0]->Trim();
103. temp->name = data[1]->Trim();
104. temp->description = data[2]->Trim();
105. node = temp;
106. }
107. break;
108. default:
109. Console::WriteLine("Ошибка, выбран некорректный формат данных");
110. break;
111. }
112. return node;
113. }
114. void DataManager::AddNode(DataType type, Object^ node) {
115. switch (type) {
116. case T\_LINK:
117. this->listLinks->Add(node);
118. Console::WriteLine("В список связей добавлена запись");
119. break;
120. case T\_BOOK:
121. this->listBooks->Add(node);
122. Console::WriteLine("В список книг добавлена запись");
123. break;
124. case T\_SUBJECT:
125. this->listSubjects->Add(node);
126. Console::WriteLine("В список дисциплин добавлена запись");
127. break;
128. default:
129. Console::WriteLine("Ошибка, выбран некорректный формат данных");
130. break;
131. }
132. }
133. void DataManager::DeleteNode(DataType type, Object^ node) {
134. Link^ t\_link = nullptr;
135. Book^ t\_book = nullptr;
136. Subject^ t\_subj = nullptr;
137. int del\_count = 0;
138. switch (type) {
139. case T\_LINK:
140. t\_link = dynamic\_cast <Link^> (node);
141. for (int i = 0; i < this->listLinks->Count; i++) {
142. Link^ link = dynamic\_cast <Link^> (this->listLinks[i]);
143. if (t\_link->ISBN == link->ISBN && t\_link->dis\_code == link->dis\_code) {
144. this->listLinks->RemoveAt(i);
145. break;
146. }
147. }
148. Console::WriteLine("Из списка связей удалена запись");
149. break;
150. case T\_BOOK:
151. t\_book = dynamic\_cast<Book^>(node);
152. for (int i = 0; i < this->listBooks->Count; i++) {
153. Book^ book = dynamic\_cast <Book^> (this->listBooks[i]);
154. if (t\_book->ISBN == book->ISBN) {
155. for (int j = 0; j < this->listLinks->Count; j++) {
156. Link^ link = dynamic\_cast <Link^> (this->listLinks[j]);
157. if (link->ISBN == t\_book->ISBN) {
158. this->listLinks->RemoveAt(j);
159. del\_count++;
160. }
161. }
162. this->listBooks->RemoveAt(i);
163. break;
164. }
165. }
166. Console::WriteLine("Из списка книг удалена запись");
167. if (del\_count > 0) {
168. Console::WriteLine("Вслед удалено " + del\_count + " записей из списка связей");
169. }
170. break;
171. case T\_SUBJECT:
172. t\_subj = dynamic\_cast<Subject^>(node);
173. for (int i = 0; i < this->listSubjects->Count; i++) {
174. Subject^ subj = dynamic\_cast <Subject^> (this->listSubjects[i]);
175. if (t\_subj->dis\_code == subj->dis\_code) {
176. for (int j = 0; j < this->listLinks->Count; j++) {
177. Link^ link = dynamic\_cast <Link^> (this->listLinks[j]);
178. if (link->dis\_code == t\_subj->dis\_code) {
179. this->listLinks->RemoveAt(j);
180. del\_count++;
181. }
182. }
183. this->listSubjects->RemoveAt(i);
184. break;
185. }
186. }
187. Console::WriteLine("Из списка дисциплин удалена запись");
188. if (del\_count > 0) {
189. Console::WriteLine("Вслед удалено " + del\_count + " записей из списка связей");
190. }
191. break;
192. default:
193. Console::WriteLine("Ошибка, выбран некорректный формат данных");
194. break;
195. }
196. }
197. List<Object^>^ DataManager::FindNodes(DataType type, String^ line) {
198. int find\_count = 0;
199. List<Object^>^ list = gcnew List<Object^>();
200. switch (type) {
201. case T\_LINK:
202. for each (Link^ link in (this->listLinks)) {
203. if (link->ISBN->ToLower()->Contains(line) || link->dis\_code->ToLower()->Contains(line)) {
204. list->Add(link);
205. find\_count++;
206. }
207. }
208. break;
209. case T\_BOOK:
210. for each (Book^ book in (this->listBooks)) {
211. if (book->ISBN->ToLower()->Contains(line) || book->title->ToLower()->Contains(line) || book->author->ToLower()->ToLower()->Contains(line)) {
212. list->Add(book);
213. find\_count++;
214. }
215. }
216. break;
217. case T\_SUBJECT:
218. for each (Subject^ subj in (this->listSubjects)) {
219. if (subj->dis\_code->ToLower()->Contains(line) || subj->name->ToLower()->Contains(line) || subj->description->ToLower()->Contains(line)) {
220. list->Add(subj);
221. find\_count++;
222. }
223. }
224. break;
225. default:
226. Console::WriteLine("Ошибка, выбран некорректный формат данных");
227. break;
228. }
229. Console::WriteLine("Найдено " + find\_count + " записей по заданной строке");
230. return list;
231. }

## MyForm.cpp

1. #include "MyForm.h"
2. using namespace System;
3. using namespace System::Windows::Forms;
4. [STAThreadAttribute]
5. int main(array<String^>^ args) {
6. Application::SetCompatibleTextRenderingDefault;
7. Application::EnableVisualStyles;
8. MumladzeLab5::MyForm app;
9. Application::Run(% app);
10. }

## MyForm.h

1. #pragma once
2. #include "DataManager.h"
3. enum SearchStatus {
4. S\_LINK = 0,
5. S\_BOOK = 1,
6. S\_SUBJECT = 2
7. };
8. namespace MumladzeLab5 {
9. using namespace System;
10. using namespace System::ComponentModel;
11. using namespace System::Collections;
12. using namespace System::Windows::Forms;
13. using namespace System::Data;
14. using namespace System::Drawing;
15. public ref class MyForm : public System::Windows::Forms::Form {
17. public: MyForm(void);
18. protected: ~MyForm();
19. private: DataManager^ dm;
20. private: System::Windows::Forms::Button^ buttonSearchBook;
21. private: System::Windows::Forms::Button^ buttonSearchSubject;
22. private: System::Windows::Forms::TextBox^ textBoxSearch;
23. private: System::Windows::Forms::Button^ buttonSearchStart;
24. private: System::Windows::Forms::ListView^ listViewData1;
25. private: System::Windows::Forms::ColumnHeader^ columnHeader1;
26. private: System::Windows::Forms::ColumnHeader^ columnHeader2;
27. private: System::Windows::Forms::ColumnHeader^ columnHeader3;
28. private: System::Windows::Forms::Label^ labelSearch;
29. private: System::Windows::Forms::TextBox^ textBoxInputData1;
30. private: System::Windows::Forms::Label^ labelInputData1;
31. private: System::Windows::Forms::TextBox^ textBoxInputData2;
32. private: System::Windows::Forms::Label^ labelInputData2;
33. private: System::Windows::Forms::TextBox^ textBoxInputData3;
34. private: System::Windows::Forms::Label^ labelInputData3;
35. private: System::Windows::Forms::Button^ buttonRemoveNode;
36. private: System::Windows::Forms::Button^ buttonAddNode;
37. private: System::Windows::Forms::Button^ buttonSearchLink;
38. /// <summary>
39. /// Обязательная переменная конструктора.
40. /// </summary>
41. private: System::ComponentModel::Container ^components;
42. #pragma region Windows Form Designer generated code
43. /// <summary>
44. /// Требуемый метод для поддержки конструктора — не изменяйте
45. /// содержимое этого метода с помощью редактора кода.
46. /// </summary>
47. void InitializeComponent(void) {
48. this->buttonSearchLink = (gcnew System::Windows::Forms::Button());
49. this->buttonSearchBook = (gcnew System::Windows::Forms::Button());
50. this->buttonSearchSubject = (gcnew System::Windows::Forms::Button());
51. this->textBoxSearch = (gcnew System::Windows::Forms::TextBox());
52. this->buttonSearchStart = (gcnew System::Windows::Forms::Button());
53. this->listViewData1 = (gcnew System::Windows::Forms::ListView());
54. this->columnHeader1 = (gcnew System::Windows::Forms::ColumnHeader());
55. this->columnHeader2 = (gcnew System::Windows::Forms::ColumnHeader());
56. this->columnHeader3 = (gcnew System::Windows::Forms::ColumnHeader());
57. this->labelSearch = (gcnew System::Windows::Forms::Label());
58. this->textBoxInputData1 = (gcnew System::Windows::Forms::TextBox());
59. this->labelInputData1 = (gcnew System::Windows::Forms::Label());
60. this->textBoxInputData2 = (gcnew System::Windows::Forms::TextBox());
61. this->labelInputData2 = (gcnew System::Windows::Forms::Label());
62. this->textBoxInputData3 = (gcnew System::Windows::Forms::TextBox());
63. this->labelInputData3 = (gcnew System::Windows::Forms::Label());
64. this->buttonRemoveNode = (gcnew System::Windows::Forms::Button());
65. this->buttonAddNode = (gcnew System::Windows::Forms::Button());
66. this->SuspendLayout();
67. //
68. // buttonSearchLink
69. //
70. this->buttonSearchLink->Location = System::Drawing::Point(260, 20);
71. this->buttonSearchLink->Name = L"buttonSearchLink";
72. this->buttonSearchLink->Size = System::Drawing::Size(100, 45);
73. this->buttonSearchLink->TabIndex = 2;
74. this->buttonSearchLink->Text = L"Связи";
75. this->buttonSearchLink->UseVisualStyleBackColor = true;
76. this->buttonSearchLink->Click += gcnew System::EventHandler(this, &MyForm::buttonSearchLink\_Click);
77. //
78. // buttonSearchBook
79. //
80. this->buttonSearchBook->Location = System::Drawing::Point(40, 20);
81. this->buttonSearchBook->Name = L"buttonSearchBook";
82. this->buttonSearchBook->Size = System::Drawing::Size(100, 45);
83. this->buttonSearchBook->TabIndex = 0;
84. this->buttonSearchBook->Text = L"Книги";
85. this->buttonSearchBook->UseVisualStyleBackColor = true;
86. this->buttonSearchBook->Click += gcnew System::EventHandler(this, &MyForm::buttonSearchBook\_Click);
87. //
88. // buttonSearchSubject
89. //
90. this->buttonSearchSubject->Location = System::Drawing::Point(150, 20);
91. this->buttonSearchSubject->Name = L"buttonSearchSubject";
92. this->buttonSearchSubject->Size = System::Drawing::Size(100, 45);
93. this->buttonSearchSubject->TabIndex = 1;
94. this->buttonSearchSubject->Text = L"Дисциплины";
95. this->buttonSearchSubject->UseVisualStyleBackColor = true;
96. this->buttonSearchSubject->Click += gcnew System::EventHandler(this, &MyForm::buttonSearchSubject\_Click);
97. //
98. // textBoxSearch
99. //
100. this->textBoxSearch->Enabled = false;
101. this->textBoxSearch->Location = System::Drawing::Point(40, 80);
102. this->textBoxSearch->Name = L"textBoxSearch";
103. this->textBoxSearch->Size = System::Drawing::Size(450, 22);
104. this->textBoxSearch->TabIndex = 3;
105. //
106. // buttonSearchStart
107. //
108. this->buttonSearchStart->Enabled = false;
109. this->buttonSearchStart->Location = System::Drawing::Point(370, 20);
110. this->buttonSearchStart->Name = L"buttonSearchStart";
111. this->buttonSearchStart->Size = System::Drawing::Size(120, 45);
112. this->buttonSearchStart->TabIndex = 4;
113. this->buttonSearchStart->Text = L"Поиск";
114. this->buttonSearchStart->UseVisualStyleBackColor = true;
115. this->buttonSearchStart->Click += gcnew System::EventHandler(this, &MyForm::buttonSearchStart\_Click);
116. //
117. // listViewData1
118. //
119. this->listViewData1->Columns->AddRange(gcnew cli::array< System::Windows::Forms::ColumnHeader^ >(3) {
120. this->columnHeader1,
121. this->columnHeader2, this->columnHeader3
122. });
123. this->listViewData1->HideSelection = false;
124. this->listViewData1->Location = System::Drawing::Point(40, 135);
125. this->listViewData1->Name = L"listViewData1";
126. this->listViewData1->Size = System::Drawing::Size(450, 250);
127. this->listViewData1->TabIndex = 5;
128. this->listViewData1->UseCompatibleStateImageBehavior = false;
129. this->listViewData1->View = System::Windows::Forms::View::Details;
130. this->listViewData1->ItemSelectionChanged += gcnew System::Windows::Forms::ListViewItemSelectionChangedEventHandler(this, &MyForm::listViewData1\_ItemSelectionChanged);
131. //
132. // columnHeader1
133. //
134. this->columnHeader1->Text = L"Поле 1";
135. this->columnHeader1->Width = 148;
136. //
137. // columnHeader2
138. //
139. this->columnHeader2->Text = L"Поле 2";
140. this->columnHeader2->Width = 148;
141. //
142. // columnHeader3
143. //
144. this->columnHeader3->Text = L"Поле 3";
145. this->columnHeader3->Width = 148;
146. //
147. // labelSearch
148. //
149. this->labelSearch->AutoSize = true;
150. this->labelSearch->Location = System::Drawing::Point(40, 104);
151. this->labelSearch->Name = L"labelSearch";
152. this->labelSearch->Size = System::Drawing::Size(47, 16);
153. this->labelSearch->TabIndex = 6;
154. this->labelSearch->Text = L"Поиск";
155. //
156. // textBoxInputData1
157. //
158. this->textBoxInputData1->Location = System::Drawing::Point(520, 135);
159. this->textBoxInputData1->Name = L"textBoxInputData1";
160. this->textBoxInputData1->Size = System::Drawing::Size(200, 22);
161. this->textBoxInputData1->TabIndex = 7;
162. this->textBoxInputData1->Enabled = false;
163. //
164. // labelInputData1
165. //
166. this->labelInputData1->AutoSize = true;
167. this->labelInputData1->Location = System::Drawing::Point(520, 160);
168. this->labelInputData1->Name = L"labelInputData1";
169. this->labelInputData1->Size = System::Drawing::Size(94, 16);
170. this->labelInputData1->TabIndex = 8;
171. this->labelInputData1->Text = L"Поле ввода 1";
172. //
173. // textBoxInputData2
174. //
175. this->textBoxInputData2->Location = System::Drawing::Point(520, 190);
176. this->textBoxInputData2->Name = L"textBoxInputData2";
177. this->textBoxInputData2->Size = System::Drawing::Size(200, 22);
178. this->textBoxInputData2->TabIndex = 9;
179. this->textBoxInputData2->Enabled = false;
180. //
181. // labelInputData2
182. //
183. this->labelInputData2->AutoSize = true;
184. this->labelInputData2->Location = System::Drawing::Point(520, 215);
185. this->labelInputData2->Name = L"labelInputData2";
186. this->labelInputData2->Size = System::Drawing::Size(94, 16);
187. this->labelInputData2->TabIndex = 10;
188. this->labelInputData2->Text = L"Поле ввода 2";
189. //
190. // textBoxInputData3
191. //
192. this->textBoxInputData3->Location = System::Drawing::Point(520, 245);
193. this->textBoxInputData3->Name = L"textBoxInputData3";
194. this->textBoxInputData3->Size = System::Drawing::Size(200, 22);
195. this->textBoxInputData3->TabIndex = 11;
196. this->textBoxInputData3->Enabled = false;
197. //
198. // labelInputData3
199. //
200. this->labelInputData3->AutoSize = true;
201. this->labelInputData3->Location = System::Drawing::Point(520, 275);
202. this->labelInputData3->Name = L"labelInputData3";
203. this->labelInputData3->Size = System::Drawing::Size(94, 16);
204. this->labelInputData3->TabIndex = 12;
205. this->labelInputData3->Text = L"Поле ввода 3";
206. //
207. // buttonRemoveNode
208. //
209. this->buttonRemoveNode->Enabled = false;
210. this->buttonRemoveNode->Location = System::Drawing::Point(520, 325);
211. this->buttonRemoveNode->Name = L"buttonRemoveNode";
212. this->buttonRemoveNode->Size = System::Drawing::Size(90, 40);
213. this->buttonRemoveNode->TabIndex = 13;
214. this->buttonRemoveNode->Text = L"Удалить";
215. this->buttonRemoveNode->UseVisualStyleBackColor = true;
216. this->buttonRemoveNode->Click += gcnew System::EventHandler(this, &MyForm::buttonRemoveNode\_Click);
217. //
218. // buttonAddNode
219. //
220. this->buttonAddNode->Enabled = false;
221. this->buttonAddNode->Location = System::Drawing::Point(630, 325);
222. this->buttonAddNode->Name = L"buttonAddNode";
223. this->buttonAddNode->Size = System::Drawing::Size(90, 40);
224. this->buttonAddNode->TabIndex = 14;
225. this->buttonAddNode->Text = L"Записать";
226. this->buttonAddNode->UseVisualStyleBackColor = true;
227. this->buttonAddNode->Click += gcnew System::EventHandler(this, &MyForm::buttonAddNode\_Click);
228. //
229. // MyForm
230. //
231. this->ClientSize = System::Drawing::Size(750, 400);
232. this->Controls->Add(this->buttonAddNode);
233. this->Controls->Add(this->buttonRemoveNode);
234. this->Controls->Add(this->labelInputData3);
235. this->Controls->Add(this->textBoxInputData3);
236. this->Controls->Add(this->labelInputData2);
237. this->Controls->Add(this->textBoxInputData2);
238. this->Controls->Add(this->labelInputData1);
239. this->Controls->Add(this->textBoxInputData1);
240. this->Controls->Add(this->labelSearch);
241. this->Controls->Add(this->listViewData1);
242. this->Controls->Add(this->buttonSearchStart);
243. this->Controls->Add(this->textBoxSearch);
244. this->Controls->Add(this->buttonSearchSubject);
245. this->Controls->Add(this->buttonSearchBook);
246. this->Controls->Add(this->buttonSearchLink);
247. this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::Fixed3D;
248. this->Name = L"MyForm";
249. this->Text = L"Библиотечная картотека";
250. this->ResumeLayout(false);
251. this->PerformLayout();
252. }
253. #pragma endregion
254. private: bool IsChangeSearchButtonClicked = false;
255. bool IsDataInitialized = false;
256. SearchStatus searchStatus = S\_BOOK;
257. private: Void initializationData();
258. private: Void buttonSearchLink\_Click(System::Object^ sender, System::EventArgs^ e);
259. private: Void buttonSearchBook\_Click(Object^ sender, EventArgs^ e);
260. private: Void buttonSearchSubject\_Click(Object^ sender, EventArgs^ e);
261. private: Void buttonSearchStart\_Click(Object^ sender, EventArgs^ e);
262. private: Void buttonRemoveNode\_Click(Object^ sender, EventArgs^ e);
263. private: Void buttonAddNode\_Click(Object^ sender, EventArgs^ e);
264. private: Void listViewData1\_ItemSelectionChanged(Object^ sender, ListViewItemSelectionChangedEventArgs^ e);
265. };
266. }

## MyFormFunc.cpp

1. #include "MyForm.h"
2. #include "DataManager.h"
3. using namespace MumladzeLab5;
4. MyForm::MyForm() {
5. InitializeComponent();
6. this->dm = gcnew DataManager();
7. }
8. MyForm::~MyForm() {
9. if (components) {
10. delete components;
11. }
12. if (IsDataInitialized) {
13. this->dm->WriteDataFiles();
14. delete this->dm;
15. dm = nullptr;
16. IsDataInitialized = false;
17. }
18. }
19. Void MyForm::initializationData() {
20. try {
21. this->dm->ReadDataFiles();
22. this->IsDataInitialized = true;
23. }
24. catch (Exception^ e) {
25. MessageBox::Show("Ошибка, не найдены файлы данных для чтения. Проверьте, что рядом "
26. "с программой находится папка Data с тремя текстовыми файлами данных.",
27. "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);
28. this->Close();
29. }
30. }
31. Void MyForm::buttonSearchLink\_Click(System::Object^ sender, System::EventArgs^ e) {
32. IsChangeSearchButtonClicked = true;
33. textBoxSearch->Enabled = true;
34. buttonSearchStart->Enabled = true;
35. buttonAddNode->Enabled = true;
36. textBoxInputData1->Enabled = true;
37. textBoxInputData2->Enabled = true;
38. textBoxInputData3->Enabled = false;
39. buttonSearchLink->FlatStyle = FlatStyle::Popup;
40. buttonSearchBook->FlatStyle = FlatStyle::Standard;
41. buttonSearchSubject->FlatStyle = FlatStyle::Standard;
42. searchStatus = S\_LINK;
43. textBoxSearch->Text = L"";
44. labelInputData1->Text = L"ISBN книги";
45. labelInputData2->Text = L"Код дисциплины";
46. labelInputData3->Text = L"";
47. labelSearch->Text = L"Введите информацию о связи";
48. listViewData1->Items->Clear();
49. columnHeader1->Text = "ISBN";
50. columnHeader1->Width = 220;
51. columnHeader2->Text = "Название книги";
52. columnHeader2->Width = 220;
53. columnHeader3->Text = "";
54. columnHeader3->Width = 0;
55. }
56. Void MyForm::buttonSearchBook\_Click(Object^ sender, EventArgs^ e) {
57. IsChangeSearchButtonClicked = true;
58. textBoxSearch->Enabled = true;
59. buttonSearchStart->Enabled = true;
60. buttonAddNode->Enabled = true;
61. textBoxInputData1->Enabled = true;
62. textBoxInputData2->Enabled = true;
63. textBoxInputData3->Enabled = true;
64. buttonSearchLink->FlatStyle = FlatStyle::Standard;
65. buttonSearchBook->FlatStyle = FlatStyle::Popup;
66. buttonSearchSubject->FlatStyle = FlatStyle::Standard;
67. searchStatus = S\_BOOK;
68. textBoxSearch->Text = L"";
69. labelInputData1->Text = L"ISBN книги";
70. labelInputData2->Text = L"Название книги";
71. labelInputData3->Text = L"Автор";
72. labelSearch->Text = L"Введите информацию о книге";
73. listViewData1->Items->Clear();
74. columnHeader1->Text = "ISBN";
75. columnHeader1->Width = 100;
76. columnHeader2->Text = "Название книги";
77. columnHeader2->Width = 230;
78. columnHeader3->Text = "Автор";
79. columnHeader3->Width = 115;
80. }
81. Void MyForm::buttonSearchSubject\_Click(Object^ sender, EventArgs^ e) {
82. IsChangeSearchButtonClicked = true;
83. textBoxSearch->Enabled = true;
84. buttonSearchStart->Enabled = true;
85. buttonAddNode->Enabled = true;
86. textBoxInputData1->Enabled = true;
87. textBoxInputData2->Enabled = true;
88. textBoxInputData3->Enabled = true;
89. buttonSearchLink->FlatStyle = FlatStyle::Standard;
90. buttonSearchBook->FlatStyle = FlatStyle::Standard;
91. buttonSearchSubject->FlatStyle = FlatStyle::Popup;
92. searchStatus = S\_SUBJECT;
93. textBoxSearch->Text = L"";
94. labelInputData1->Text = L"Код дисциплины";
95. labelInputData2->Text = L"Название дисциплины";
96. labelInputData3->Text = L"Описание дисциплины";
97. labelSearch->Text = L"Введите информацию о дисциплине";
98. listViewData1->Items->Clear();
99. columnHeader1->Text = "Код";
100. columnHeader1->Width = 75;
101. columnHeader2->Text = "Название дисциплины";
102. columnHeader2->Width = 145;
103. columnHeader3->Text = "Описание";
104. columnHeader3->Width = 225;
105. }
106. Void MyForm::buttonSearchStart\_Click(System::Object^ sender, System::EventArgs^ e) {
107. if (!IsDataInitialized) {
108. initializationData();
109. IsDataInitialized = true;
110. }
111. listViewData1->Items->Clear();
112. List<Object^>^ finded = nullptr;
113. String^ text = textBoxSearch->Text->ToLower();
114. switch (searchStatus) {
115. case S\_LINK:
116. finded = dm->FindNodes(T\_LINK, text);
117. for each (Link^ link in finded) {
118. ListViewItem^ item = gcnew ListViewItem();
119. item->Text = link->ISBN;
120. item->SubItems->Add(link->dis\_code);
121. listViewData1->Items->Add(item);
122. }
123. break;
124. case S\_BOOK:
125. finded = dm->FindNodes(T\_BOOK, text);
126. for each (Book^ book in finded) {
127. ListViewItem^ item = gcnew ListViewItem();
128. item->Text = book->ISBN;
129. item->SubItems->Add(book->title);
130. item->SubItems->Add(book->author);
131. listViewData1->Items->Add(item);
132. }
133. break;
134. case S\_SUBJECT:
135. finded = dm->FindNodes(T\_SUBJECT, text);
136. for each (Subject^ subject in finded) {
137. ListViewItem^ item = gcnew ListViewItem();
138. item->Text = subject->dis\_code;
139. item->SubItems->Add(subject->name);
140. item->SubItems->Add(subject->description);
141. listViewData1->Items->Add(item);
142. }
143. break;
144. default:
145. MessageBox::Show("Ошибка - не выбрана категория поиска", "Ошибка",
146. MessageBoxButtons::OK, MessageBoxIcon::Error);
147. break;
148. }
149. listViewData1->Scrollable = true;
150. }
151. Void MyForm::buttonAddNode\_Click(System::Object^ sender, System::EventArgs^ e) {
152. String^ text1 = textBoxInputData1->Text;
153. String^ text2 = textBoxInputData2->Text;
154. String^ text3 = textBoxInputData3->Text;
155. if (String::IsNullOrWhiteSpace(text1) || String::IsNullOrWhiteSpace(text2)) {
156. MessageBox::Show("Ошибка. Вы ввели не все данные, необходимые для записи в список.",
157. "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);
158. }
159. else {
160. ListViewItem^ item = gcnew ListViewItem();
161. item->Text = text1;
162. item->SubItems->Add(text2);
163. Link^ link = gcnew Link();
164. Book^ book = gcnew Book();
165. Subject^ subject = gcnew Subject();
166. Object^ node = nullptr;
167. switch (searchStatus) {
168. case S\_LINK:
169. link->ISBN = text1;
170. link->dis\_code = text2;
171. node = link;
172. dm->AddNode(T\_LINK, node);
173. break;
174. case S\_BOOK:
175. if (String::IsNullOrWhiteSpace(text3)) {
176. MessageBox::Show("Ошибка. Вы ввели не все данные, необходимые для записи в список.",
177. "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);
178. }
179. else {
180. book->ISBN = text1;
181. book->title = text2;
182. book->author = text3;
183. node = book;
184. dm->AddNode(T\_BOOK, node);
185. item->SubItems->Add(text3);
186. }
187. break;
188. case S\_SUBJECT:
189. if (String::IsNullOrWhiteSpace(text3)) {
190. MessageBox::Show("Ошибка. Вы ввели не все данные, необходимые для записи в список.",
191. "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);
192. }
193. else {
194. subject->dis\_code = text1;
195. subject->name = text2;
196. subject->description = text3;
197. node = subject;
198. dm->AddNode(T\_SUBJECT, node);
199. item->SubItems->Add(text3);
200. }
201. break;
202. default:
203. MessageBox::Show("Ошибка - не выбрана категория записи", "Ошибка",
204. MessageBoxButtons::OK, MessageBoxIcon::Error);
205. return;
206. }
207. if (node) {
208. listViewData1->Items->Add(item);
209. textBoxInputData1->Text = "";
210. textBoxInputData2->Text = "";
211. textBoxInputData3->Text = "";
212. }
213. }
214. }
215. Void MyForm::listViewData1\_ItemSelectionChanged(Object^ sender, ListViewItemSelectionChangedEventArgs^ e) {
216. buttonRemoveNode->Enabled = listViewData1->SelectedItems->Count > 0;
217. }
218. Void MyForm::buttonRemoveNode\_Click(System::Object^ sender, System::EventArgs^ e) {
219. if (listViewData1->SelectedItems->Count > 0) {
220. Link^ link = gcnew Link();
221. Book^ book = gcnew Book();
222. Subject^ subject = gcnew Subject();
223. Object^ node = nullptr;
224. List<ListViewItem^>^ itemsToRemove = gcnew List<ListViewItem^>();
225. for each (ListViewItem^ item in listViewData1->SelectedItems) {
226. itemsToRemove->Add(item);
227. }
228. for each (ListViewItem^ item in itemsToRemove) {
229. listViewData1->Items->Remove(item);
230. switch (searchStatus) {
231. case S\_LINK:
232. link->ISBN = item->Text;
233. link->dis\_code = item->SubItems[0]->Text;
234. node = link;
235. dm->DeleteNode(T\_LINK, node);
236. break;
237. case S\_BOOK:
238. book->ISBN = item->Text;
239. node = book;
240. dm->DeleteNode(T\_BOOK, node);
241. break;
242. case S\_SUBJECT:
243. subject->dis\_code = item->Text;
244. node = book;
245. dm->DeleteNode(T\_BOOK, node);
246. break;
247. default:
248. MessageBox::Show("Ошибка - не выбрана категория удаления", "Ошибка",
249. MessageBoxButtons::OK, MessageBoxIcon::Error);
250. break;
251. }
252. }
253. }
254. buttonRemoveNode->Enabled = false;
255. }