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 №6

по дисциплине «Основы алгоритмизации и программирования»

**«РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ   
РАБОТЫ С ДАННЫМИ»**

Вариант 3

# Содержание

Содержание 2

Техническое задание 3

Цель задания 3

Задача 3

Ход работы 4

Написание программы 4

Алгоритм программы 5

Пример работы программы 6

Вывод 7

Листинг 8

AccessDB.h 8

AccessDB.cpp 8

MyForm.h 12

MyForm.cpp 19

GetData.h 22

GetData.cpp 27

# Техническое задание

## Цель задания

Используя среду разработки MS Visual Studio, создать приложение, с помощью которого можно модифицировать данные об объекте в файле.

## Задача

1. Разработать формат хранения данных об экземпляре объекта в файле, используя полученное задание.
2. Создать текстовый файл, в котором соответственно принятому формату хранится информация об экземпляре класса объектов.
3. Создать проект Windows Forms в среде VS.
4. Разработать форму диалога для редактирования свойств объекта по данным из выбранного файла.

Заданием моего варианта стал формат данных для хранения информации о книгах. Для его хранения будет использоваться Access Database (.accdb), которая через инструменты .NET Framework будет подключаться к приложению и редактироваться через написанный интерфейс.

# Ход работы

## Написание программы

Сначала я разработал формат данных, в котором буду хранить поступающую информацию. За основу я взял базу данных Access Database (.accdb), и разработал формат данных. В файле одна таблица – Books – содержащая следующие поля:

* Счётчик ID
* Короткий текст ISBN
* Длинный текст Title
* Длинный текст Author
* Дата и время PublishingDate
* Числовой PageCount

Для работы с созданной БД я воспользовался готовой библиотекой .NET Framework: «System.Data.OleDb», а также провайдером работы этой базой данных «Microsoft.ACE.OLEDB.12.0». С их помощью возможно подключение к .accdb файлу, и работе с содержащейся в ней базой данных.

Был написан класс AccessDB, содержащий методы для работы с БД – CRUD (Create – Read – Update – Delete). Он ответственен за открытие .accdb файла данных, проверки его корректности в контексте данной задачи, и совершение операций CRUD.

Также были написаны две формы WinForm – окно работы с базой данных, и окно ввода данных, содержащее текстовые поля.

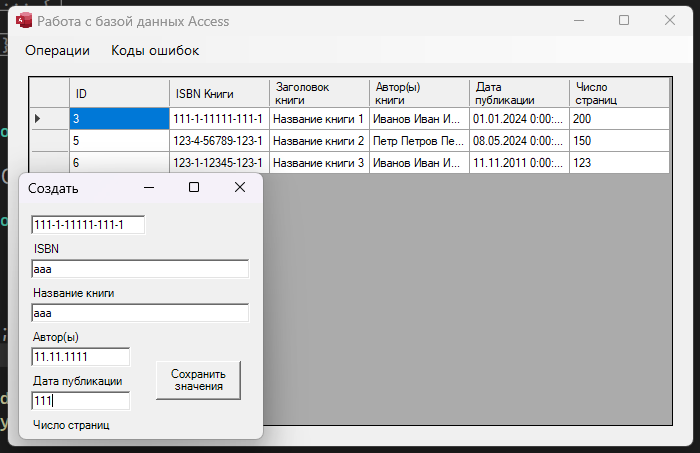
Первое окно (MyForm) содержит gridDataView (готовое решение Microsoft для отображения данных) и меню с кнопками menuStrip. Меню содержит кнопку «Операции», вызывающую подменю с кнопками «Открыть БД», «Создать запись», «Прочитать записи», «Обновить запись», «Удалить запись», а также кнопку «Коды ошибок» для технической отладки.

Второе окно (GetData) содержит пять текстовых полей для ввода данных пользователем (при создании/изменении записи в базе данных). Это окно изначально скрыто, и вызывается при вызове соответствующей операции. Некоторые поля маскированы, то есть идет проверка корректности ввода (например, в поле ввода даты стоит маска 00-00-0000, ограничивающая ввод цифрами, не более чем в маске).

## Алгоритм программы

1. Пользователь запускает программу. Его встречает интерфейс работы с БД.
2. Пользователь открывает БД, лежащую в .accdb файле. Для этого вызывается диалоговое окно выбора файла, с ограничением на формат открываемого файла. Если пользователь не выбирает файл, или файл не соответствует формату данных программы, высветится ошибка, иначе БД будет готова к работе.
3. Пользователь может совершить любую из операций CRUD, нажав на соответствующую кнопку в верхнем меню.
   1. При вызове операции «Прочитать записи», программа обращается к БД, совершает SQL запрос «SELECT \* FROM Books», и отображает на экран полученные записи.
   2. При вызове операции «Создать запись» программа открывает дополнительное окно «GetData», в котором пользователь вводит новые данные. Для сохранения введенных данных пользователь нажимает кнопку «Сохранить изменения». Программа совершает SQL запрос «INSERT INTO Books (ISBN, Title, Author, PublishingDate, PageCount) VALUES (?, ?, ?, ?, ?)», и отображает новую запись на исходном окне.
   3. При вызове операции «Обновить запись» (пользователь предварительно выбирает запись к редактированию, нажав на нее), вызывается то же самое окно GetData, с отличием в том, что текстовые поля уже заполнены исходными данными. Аналогично пользователь нажимает кнопку, программа выполняет SQL запрос «UPDATE Books SET ISBN = ?, PublishingDate = ?, Title = ?, Author = ?, PageCount = ? WHERE ID = ?», и на исходном окне в таблице данных изменяет выбранную запись новыми данными.
   4. При вызове операции «Удалить запись» (пользователь предварительно выбирает запись к редактированию, нажав на нее), программа получает ID записи, совершает удаление из БД по SQL запросу «DELETE FROM Books WHERE ID = ?», и убирает удаленную запись из таблицы данных.
   5. При нажатии кнопки «Коды ошибок» вызывается информационное окно с расшифровкой всех кодов ошибок. Добавлено в отладочных целях.
4. При завершении работы пользователь может закрыть окно. Все изменения автоматически сохраняются, и не требуют дополнительных действий.

## Пример работы программы



# Вывод

В ходе лабораторной работы был реализован интерфейс работы с базой данных на основе Access Database. Реализован класс AccessDB, содержащий методы CRUD, а также формы WinForms, в котором пользователь совершает операции над БД.

# Листинг

## AccessDB.h

1. #pragma once
2. using namespace System;
3. using namespace System::Data;
4. using namespace System::Data::OleDb;
5. using namespace System::Collections::Generic;
6. enum ErrorsDB {
7. DB\_OK = 0,
8. DB\_CANNOT\_CONNECT = 1,
9. DB\_WRONG\_TABLES = 2,
10. DB\_WRONG\_COLUMNS = 3,
11. DB\_READ\_ERROR = 4,
12. DB\_CREATE\_ERROR = 5,
13. DB\_UPDATE\_ERROR = 6,
14. DB\_DELETE\_ERROR = 7
15. };
16. ref struct BookNode {
17. Int32 ID;
18. String^ ISBN;
19. DateTime^ date;
20. String^ title;
21. String^ author;
22. Int32 pageCount;
23. };
24. ref class AccessDB {
25. private:
26. OleDbConnection^ dbConnect;
27. public:
28. ErrorsDB OpenDB(String^ path);
29. ErrorsDB CheckDB();
30. Void CloseDB();
31. ErrorsDB Create(BookNode^ node);
32. List<BookNode^>^ Read();
33. ErrorsDB Update(BookNode^ node);
34. ErrorsDB Delete(Int32 ID);
35. };

## AccessDB.cpp

1. #include "AccessDB.h"
2. ErrorsDB AccessDB::OpenDB(String^ path) {
3. try {
4. String^ configString = "provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + path + ";";
5. this->dbConnect = gcnew OleDbConnection(configString);
6. return CheckDB();
7. }
8. catch (Exception^ e) {
9. return DB\_CANNOT\_CONNECT;
10. }
11. }
12. ErrorsDB AccessDB::CheckDB() {
13. ErrorsDB status = DB\_OK;
14. try {
15. this->dbConnect->Open();
16. DataTable^ schemaTable = dbConnect->GetSchema("Tables", gcnew array<String^> {nullptr, nullptr, nullptr, "TABLE"});
17. if (schemaTable->Rows->Count == 1) {
18. DataRow^ table = schemaTable->Rows[0];
19. if (table["TABLE\_NAME"]->ToString() == "Books") {
20. DataTable^ columns = dbConnect->GetSchema("Columns", gcnew array<String^> {nullptr, nullptr, "Books", nullptr});
21. bool hasID = false, hasISBN = false, hasPublishingDate = false, hasTitle = false, hasAuthor = false, hasPageCount = false;
22. for each (DataRow ^ col in columns->Rows) {
23. String^ columnName = col["COLUMN\_NAME"]->ToString();
24. String^ dataType = col["DATA\_TYPE"]->ToString();
25. if (columnName == "ID" && dataType == "3")
26. hasID = true;
27. else if (columnName == "ISBN" && dataType == "130")
28. hasISBN = true;
29. else if (columnName == "Title" && dataType == "130")
30. hasTitle = true;
31. else if (columnName == "Author" && dataType == "130")
32. hasAuthor = true;
33. else if (columnName == "PublishingDate" && dataType == "7")
34. hasPublishingDate = true;
35. else if (columnName == "PageCount" && dataType == "3")
36. hasPageCount = true;
37. else {
38. status = DB\_WRONG\_COLUMNS;
39. break;
40. }
41. }
42. if (!(hasID && hasISBN && hasPublishingDate && hasTitle && hasAuthor && hasPageCount))
43. status = DB\_WRONG\_COLUMNS;
44. else
45. status = DB\_OK;
46. }
47. else
48. status = DB\_WRONG\_TABLES;
49. }
50. else
51. status = DB\_WRONG\_TABLES;
52. }
53. catch (Exception^ e) {
54. status = DB\_CANNOT\_CONNECT;
55. }
56. finally {
57. if (this->dbConnect->State == ConnectionState::Open)
58. this->dbConnect->Close();
59. }
60. return status;
61. }
62. Void AccessDB::CloseDB() {
63. delete this->dbConnect;
64. this->dbConnect = nullptr;
65. }
66. ErrorsDB AccessDB::Create(BookNode^ node) {
67. ErrorsDB status = DB\_OK;
68. try {
69. this->dbConnect->Open();
70. String^ query = "INSERT INTO Books (ISBN, Title, Author, PublishingDate, PageCount) VALUES (?, ?, ?, ?, ?)";
71. OleDbCommand^ command = gcnew OleDbCommand(query, this->dbConnect);
72. command->Parameters->AddWithValue("?", node->ISBN);
73. command->Parameters->AddWithValue("?", node->title);
74. command->Parameters->AddWithValue("?", node->author);
75. command->Parameters->AddWithValue("?", node->date);
76. command->Parameters->AddWithValue("?", node->pageCount);
77. command->ExecuteNonQuery();
78. status = DB\_OK;
79. }
80. catch (Exception^ e) {
81. status = DB\_CREATE\_ERROR;
82. }
83. finally {
84. if (this->dbConnect->State == ConnectionState::Open)
85. this->dbConnect->Close();
86. }
87. return status;
88. }
89. List<BookNode^>^ AccessDB::Read() {
90. String^ query = "SELECT \* FROM Books";
91. OleDbCommand^ command = gcnew OleDbCommand(query, this->dbConnect);
92. List<BookNode^>^ bookList = gcnew List<BookNode^>();
94. try {
95. this->dbConnect->Open();
96. OleDbDataReader^ reader = command->ExecuteReader();
97. while (reader->Read()) {
98. BookNode^ book = gcnew BookNode();
99. book->ID = Convert::ToInt32(reader["ID"]);
100. book->ISBN = reader["ISBN"]->ToString();
101. book->date = static\_cast<DateTime^>(reader["PublishingDate"]);
102. book->title = reader["Title"]->ToString();
103. book->author = reader["Author"]->ToString();
104. book->pageCount = Convert::ToInt32(reader["PageCount"]);
105. bookList->Add(book);
106. }
107. }
108. catch (Exception^ e) {
109. this->dbConnect->Close();
110. return nullptr;
111. }
112. finally {
113. if (this->dbConnect->State == ConnectionState::Open)
114. this->dbConnect->Close();
115. }
116. return bookList;
117. }
118. ErrorsDB AccessDB::Update(BookNode^ node) {
119. ErrorsDB status = DB\_OK;
120. try {
121. this->dbConnect->Open();
122. String^ query = "UPDATE Books SET ISBN = ?, PublishingDate = ?, Title = ?, Author = ?, PageCount = ? WHERE ID = ?";
123. OleDbCommand^ command = gcnew OleDbCommand(query, this->dbConnect);
124. command->Parameters->AddWithValue("?", node->ISBN);
125. command->Parameters->AddWithValue("?", node->date);
126. command->Parameters->AddWithValue("?", node->title);
127. command->Parameters->AddWithValue("?", node->author);
128. command->Parameters->AddWithValue("?", node->pageCount);
129. command->Parameters->AddWithValue("?", node->ID);
130. command->ExecuteNonQuery();
131. status = DB\_OK;
132. }
133. catch (Exception^ e) {
134. status = DB\_UPDATE\_ERROR;
135. }
136. finally {
137. if (this->dbConnect->State == ConnectionState::Open)
138. this->dbConnect->Close();
139. }
140. return status;
141. }
142. ErrorsDB AccessDB::Delete(Int32 ID) {
143. ErrorsDB status = DB\_OK;
144. try {
145. this->dbConnect->Open();
146. String^ query = "DELETE FROM Books WHERE ID = ?";
147. OleDbCommand^ command = gcnew OleDbCommand(query, this->dbConnect);
148. command->Parameters->AddWithValue("?", ID);
149. command->ExecuteNonQuery();
150. status = DB\_OK;
151. }
152. catch (Exception^ e) {
153. status = DB\_DELETE\_ERROR;
154. }
155. finally {
156. if (this->dbConnect->State == ConnectionState::Open)
157. this->dbConnect->Close();
158. }
159. return status;
160. }

## MyForm.h

1. #pragma once
2. #include "AccessDB.h"
3. namespace MumladzeLab6 {
4. using namespace System;
5. using namespace System::ComponentModel;
6. using namespace System::Collections;
7. using namespace System::Windows::Forms;
8. using namespace System::Data;
9. using namespace System::Drawing;
10. /// <summary>
11. /// Сводка для MyForm
12. /// </summary>
13. public ref class MyForm : public System::Windows::Forms::Form {
15. public:
16. MyForm(void);
17. protected:
18. ~MyForm();
19. private: System::Windows::Forms::MenuStrip^ menuStrip;
20. private: System::Windows::Forms::ToolStripMenuItem^ actionsToolStripMenuItem;
21. private: System::Windows::Forms::ToolStripMenuItem^ openDataBaseToolStripMenuItem;
22. private: System::Windows::Forms::ToolStripMenuItem^ createNodeToolStripMenuItem;
23. private: System::Windows::Forms::ToolStripMenuItem^ readNodeToolStripMenuItem;
24. private: System::Windows::Forms::ToolStripMenuItem^ updateNodeToolStripMenuItem;
25. private: System::Windows::Forms::ToolStripMenuItem^ deleteNodeToolStripMenuItem;
26. private: System::Windows::Forms::DataGridView^ dataGridView;
27. private: System::Windows::Forms::ToolStripMenuItem^ errorCodesToolStripMenuItem;
28. private: System::Windows::Forms::DataGridViewTextBoxColumn^ ID;
29. private: System::Windows::Forms::DataGridViewTextBoxColumn^ ISBN;
30. private: System::Windows::Forms::DataGridViewTextBoxColumn^ Title;
31. private: System::Windows::Forms::DataGridViewTextBoxColumn^ Author;
32. private: System::Windows::Forms::DataGridViewTextBoxColumn^ Date;
33. private: System::Windows::Forms::DataGridViewTextBoxColumn^ PageCount;
34. protected:

37. private:
38. System::ComponentModel::Container ^components;
39. #pragma region Windows Form Designer generated code
40. /// <summary>
41. /// Требуемый метод для поддержки конструктора — не изменяйте
42. /// содержимое этого метода с помощью редактора кода.
43. /// </summary>
44. void InitializeComponent(void) {
45. System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));
46. this->menuStrip = (gcnew System::Windows::Forms::MenuStrip());
47. this->actionsToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
48. this->openDataBaseToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
49. this->createNodeToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
50. this->readNodeToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
51. this->updateNodeToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
52. this->deleteNodeToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
53. this->errorCodesToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
54. this->dataGridView = (gcnew System::Windows::Forms::DataGridView());
55. this->ID = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
56. this->ISBN = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
57. this->Title = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
58. this->Author = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
59. this->Date = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
60. this->PageCount = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
61. this->menuStrip->SuspendLayout();
62. (cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView))->BeginInit();
63. this->SuspendLayout();
64. //
65. // menuStrip
66. //
67. this->menuStrip->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {
68. this->actionsToolStripMenuItem,
69. this->errorCodesToolStripMenuItem
70. });
71. this->menuStrip->Location = System::Drawing::Point(0, 0);
72. this->menuStrip->Name = L"menuStrip";
73. this->menuStrip->Size = System::Drawing::Size(684, 27);
74. this->menuStrip->TabIndex = 0;
75. //
76. // actionsToolStripMenuItem
77. //
78. this->actionsToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(5) {
79. this->openDataBaseToolStripMenuItem,
80. this->createNodeToolStripMenuItem, this->readNodeToolStripMenuItem, this->updateNodeToolStripMenuItem, this->deleteNodeToolStripMenuItem
81. });
82. this->actionsToolStripMenuItem->Name = L"actionsToolStripMenuItem";
83. this->actionsToolStripMenuItem->Size = System::Drawing::Size(86, 23);
84. this->actionsToolStripMenuItem->Text = L"Операции";
85. //
86. // openDataBaseToolStripMenuItem
87. //
88. this->openDataBaseToolStripMenuItem->Name = L"openDataBaseToolStripMenuItem";
89. this->openDataBaseToolStripMenuItem->Size = System::Drawing::Size(216, 24);
90. this->openDataBaseToolStripMenuItem->Text = L"Открыть базу данных";
91. this->openDataBaseToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::OpenDataBaseToolStripMenuItem\_Click);
92. //
93. // createNodeToolStripMenuItem
94. //
95. this->createNodeToolStripMenuItem->Name = L"createNodeToolStripMenuItem";
96. this->createNodeToolStripMenuItem->Size = System::Drawing::Size(216, 24);
97. this->createNodeToolStripMenuItem->Text = L"Создать запись";
98. this->createNodeToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::CreateNodeToolStripMenuItem\_Click);
99. //
100. // readNodeToolStripMenuItem
101. //
102. this->readNodeToolStripMenuItem->Name = L"readNodeToolStripMenuItem";
103. this->readNodeToolStripMenuItem->Size = System::Drawing::Size(216, 24);
104. this->readNodeToolStripMenuItem->Text = L"Прочитать записи";
105. this->readNodeToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::ReadNodeToolStripMenuItem\_Click);
106. //
107. // updateNodeToolStripMenuItem
108. //
109. this->updateNodeToolStripMenuItem->Name = L"updateNodeToolStripMenuItem";
110. this->updateNodeToolStripMenuItem->Size = System::Drawing::Size(216, 24);
111. this->updateNodeToolStripMenuItem->Text = L"Обновить запись";
112. this->updateNodeToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::UpdateNodeToolStripMenuItem\_Click);
113. //
114. // deleteNodeToolStripMenuItem
115. //
116. this->deleteNodeToolStripMenuItem->Name = L"deleteNodeToolStripMenuItem";
117. this->deleteNodeToolStripMenuItem->Size = System::Drawing::Size(216, 24);
118. this->deleteNodeToolStripMenuItem->Text = L"Удалить запись";
119. this->deleteNodeToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::DeleteNodeToolStripMenuItem\_Click);
120. //
121. // errorCodesToolStripMenuItem
122. //
123. this->errorCodesToolStripMenuItem->Name = L"errorCodesToolStripMenuItem";
124. this->errorCodesToolStripMenuItem->Size = System::Drawing::Size(109, 23);
125. this->errorCodesToolStripMenuItem->Text = L"Коды ошибок";
126. this->errorCodesToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::ErrorCodesToolStripMenuItem\_Click);
127. //
128. // dataGridView
129. //
130. this->dataGridView->AllowUserToAddRows = false;
131. this->dataGridView->AllowUserToDeleteRows = false;
132. this->dataGridView->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
133. this->dataGridView->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(6) {
134. this->ID, this->ISBN,
135. this->Title, this->Author, this->Date, this->PageCount
136. });
137. this->dataGridView->Location = System::Drawing::Point(20, 40);
138. this->dataGridView->Name = L"dataGridView";
139. this->dataGridView->ReadOnly = true;
140. this->dataGridView->Size = System::Drawing::Size(645, 350);
141. this->dataGridView->TabIndex = 1;
142. //
143. // ID
144. //
145. this->ID->HeaderText = L"ID";
146. this->ID->Name = L"ID";
147. this->ID->ReadOnly = true;
148. //
149. // ISBN
150. //
151. this->ISBN->HeaderText = L"ISBN Книги";
152. this->ISBN->Name = L"ISBN";
153. this->ISBN->ReadOnly = true;
154. //
155. // Title
156. //
157. this->Title->HeaderText = L"Заголовок книги";
158. this->Title->Name = L"Title";
159. this->Title->ReadOnly = true;
160. //
161. // Author
162. //
163. this->Author->HeaderText = L"Автор(ы) книги";
164. this->Author->Name = L"Author";
165. this->Author->ReadOnly = true;
166. //
167. // Date
168. //
169. this->Date->HeaderText = L"Дата публикации";
170. this->Date->Name = L"Date";
171. this->Date->ReadOnly = true;
172. //
173. // PageCount
174. //
175. this->PageCount->HeaderText = L"Число страниц";
176. this->PageCount->Name = L"PageCount";
177. this->PageCount->ReadOnly = true;
178. //
179. // MyForm
180. //
181. this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
182. this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
183. this->ClientSize = System::Drawing::Size(685, 410);
184. this->Controls->Add(this->dataGridView);
185. this->Controls->Add(this->menuStrip);
186. this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;
187. this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
188. this->MainMenuStrip = this->menuStrip;
189. this->Name = L"MyForm";
190. this->Text = L"Работа с базой данных Access";
191. this->menuStrip->ResumeLayout(false);
192. this->menuStrip->PerformLayout();
193. (cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView))->EndInit();
194. this->ResumeLayout(false);
195. this->PerformLayout();
196. }
197. #pragma endregion
199. private: AccessDB^ dataBase;
201. private: System::Void ErrorCodesToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);
202. private: System::Void OpenDataBaseToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);
203. private: System::Void CreateNodeToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);
204. private: System::Void ReadNodeToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);
205. private: System::Void UpdateNodeToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);
206. private: System::Void DeleteNodeToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);
208. };
209. }

## MyForm.cpp

1. #include "MyForm.h"
2. #include "AccessDB.h"
3. #include "GetData.h"
4. using namespace System;
5. using namespace System::Windows;
6. using namespace System::Windows::Forms;
7. using namespace MumladzeLab6;
8. using namespace System::Data::OleDb;
9. [STAThreadAttribute]
10. int main(array<String^>^ args) {
11. Application::SetCompatibleTextRenderingDefault;
12. Application::EnableVisualStyles;
14. MyForm app;
15. Application::Run(% app);
16. }
17. MyForm::MyForm(void) {
18. InitializeComponent();
19. this->dataBase = gcnew AccessDB();
20. }
21. MyForm::~MyForm() {
22. if (components) {
23. delete components;
24. }
25. }
26. Void MyForm::ErrorCodesToolStripMenuItem\_Click(Object^ sender, EventArgs^ e) {
27. String^ str = gcnew String("1 - невозможно соединиться с БД\n2 - БД не соответствует формату программы\n" +
28. "3 - таблица Books не соответствует формату программы\n4 - ошибка чтения БД\n" +
29. "5 - ошибка создания записи в БД\n6 - ошибка обновления записи в БД\n7 - ошибка удаления из БД");
30. MessageBox::Show(str, "Info", MessageBoxButtons::OK, MessageBoxIcon::Information);
31. }
32. Void MyForm::OpenDataBaseToolStripMenuItem\_Click(Object^ sender, EventArgs^ e) {
33. OpenFileDialog^ openFileDialog = gcnew OpenFileDialog();
34. openFileDialog->Filter = "Access Database files (\*.accdb)|\*.accdb";
35. String^ pathFile;
36. if (openFileDialog->ShowDialog() == Forms::DialogResult::OK) {
37. pathFile = openFileDialog->FileName;
38. ErrorsDB status = this->dataBase->OpenDB(pathFile);
39. if (status != DB\_OK) {
40. MessageBox::Show("Ошибка код " + (static\_cast<int>(status)).ToString(), "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);
41. this->dataBase->CloseDB();
42. }
43. }
44. else {
45. MessageBox::Show("Внимание: вы не выбрали базу данных. Для работы с БД необходимо ее открыть.", "Ошибка открытия БД", MessageBoxButtons::OK, MessageBoxIcon::Warning);
46. return;
47. }
49. }
50. Void MyForm::CreateNodeToolStripMenuItem\_Click(Object^ sender, EventArgs^ e) {
51. GetData^ getDataForm = gcnew GetData(nullptr);
52. if (getDataForm->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
53. BookNode^ node = getDataForm->GetDataFromBox();
54. ErrorsDB status = this->dataBase->Create(node);
55. if (status == DB\_OK) {
56. this->dataGridView->Rows->Add(node->ID, node->ISBN, node->title,
57. node->author, node->date, node->pageCount);
58. }
59. else {
60. MessageBox::Show("Ошибка код " + (static\_cast<int>(status)).ToString(), "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);
61. }
62. }
63. }
64. Void MyForm::ReadNodeToolStripMenuItem\_Click(Object^ sender, EventArgs^ e) {
65. this->dataGridView->Rows->Clear();
66. List<BookNode^>^ bookList = this->dataBase->Read();
67. if (bookList) {
68. for each (BookNode^ node in bookList) {
69. this->dataGridView->Rows->Add(node->ID, node->ISBN, node->title,
70. node->author, node->date, node->pageCount);
71. }
72. }
73. else {
74. ErrorsDB status = DB\_READ\_ERROR;
75. MessageBox::Show("Ошибка код " + (static\_cast<int>(status)).ToString(), "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);
76. }
77. }
78. Void MyForm::UpdateNodeToolStripMenuItem\_Click(Object^ sender, EventArgs^ e) {
79. if (dataGridView->SelectedRows->Count != 1) {
80. MessageBox::Show("Пожалуйста, выберите одну строку для обновления", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);
81. return;
82. }
83. DataGridViewRow^ selectedRow = dataGridView->SelectedRows[0];
84. BookNode^ node = gcnew BookNode();
85. node->ID = Convert::ToInt32(selectedRow->Cells["ID"]->Value);
86. node->ISBN = Convert::ToString(selectedRow->Cells["ISBN"]->Value);
87. node->title = Convert::ToString(selectedRow->Cells["Title"]->Value);
88. node->author = Convert::ToString(selectedRow->Cells["Author"]->Value);
89. node->date = Convert::ToDateTime(selectedRow->Cells["Date"]->Value);
90. node->pageCount = Convert::ToInt32(selectedRow->Cells["PageCount"]->Value);
91. GetData^ getDataForm = gcnew GetData(node);
92. if (getDataForm->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
93. BookNode^ updatedNode = getDataForm->GetDataFromBox();
94. ErrorsDB status = this->dataBase->Update(updatedNode);
95. if (status == DB\_OK) {
96. for each (DataGridViewRow ^ row in this->dataGridView->Rows) {
97. if (Convert::ToInt32(row->Cells["ID"]->Value) == updatedNode->ID) {
98. row->Cells["ISBN"]->Value = updatedNode->ISBN;
99. row->Cells["Title"]->Value = updatedNode->title;
100. row->Cells["Author"]->Value = updatedNode->author;
101. row->Cells["Date"]->Value = updatedNode->date->ToString("d");
102. row->Cells["PageCount"]->Value = updatedNode->pageCount;
103. break;
104. }
105. }
106. }
107. else {
108. MessageBox::Show("Ошибка код " + (static\_cast<int>(status)).ToString(), "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);
109. }
110. }
111. }
112. Void MyForm::DeleteNodeToolStripMenuItem\_Click(Object^ sender, EventArgs^ e) {
113. if (dataGridView->SelectedRows->Count != 1) {
114. MessageBox::Show("Пожалуйста, выберите одну строку для удаления", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);
115. return;
116. }
117. DataGridViewRow^ selectedRow = dataGridView->SelectedRows[0];
118. Int32 ID = Convert::ToInt32(selectedRow->Cells["ID"]->Value);
119. ErrorsDB status = this->dataBase->Delete(ID);
120. if (status == DB\_OK) {
121. for each (DataGridViewRow ^ row in this->dataGridView->Rows) {
122. if (Convert::ToInt32(row->Cells["ID"]->Value) == ID) {
123. this->dataGridView->Rows->Remove(row);
124. break;
125. }
126. }
127. }
128. else {
129. MessageBox::Show("Ошибка код " + (static\_cast<int>(status)).ToString(), "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);
130. }
131. }

## GetData.h

1. #pragma once
2. #include "AccessDB.h"
3. namespace MumladzeLab6 {
4. using namespace System;
5. using namespace System::ComponentModel;
6. using namespace System::Collections;
7. using namespace System::Windows::Forms;
8. using namespace System::Data;
9. using namespace System::Drawing;
10. /// <summary>
11. /// Сводка для GetData
12. /// </summary>
13. public ref class GetData : public System::Windows::Forms::Form {

16. public:
17. GetData(BookNode^ node);
18. protected:
19. ~GetData();
20. private: System::Windows::Forms::MaskedTextBox^ boxISBN;
21. private: System::Windows::Forms::Label^ labelISBN;
22. private: System::Windows::Forms::TextBox^ boxTitle;
23. private: System::Windows::Forms::Label^ labelTitle;
24. private: System::Windows::Forms::TextBox^ boxAuthor;
25. private: System::Windows::Forms::Label^ labelAuthor;
26. private: System::Windows::Forms::MaskedTextBox^ boxDate;
27. private: System::Windows::Forms::Label^ labelDate;
28. private: System::Windows::Forms::TextBox^ boxPages;
29. private: System::Windows::Forms::Label^ labelPages;
30. private: System::Windows::Forms::Button^ buttonSave;
31. private:
32. System::ComponentModel::Container ^components;
33. #pragma region Windows Form Designer generated code
34. /// <summary>
35. /// Требуемый метод для поддержки конструктора — не изменяйте
36. /// содержимое этого метода с помощью редактора кода.
37. /// </summary>
38. void InitializeComponent(void) {
39. this->boxISBN = (gcnew System::Windows::Forms::MaskedTextBox());
40. this->labelISBN = (gcnew System::Windows::Forms::Label());
41. this->boxTitle = (gcnew System::Windows::Forms::TextBox());
42. this->labelTitle = (gcnew System::Windows::Forms::Label());
43. this->boxAuthor = (gcnew System::Windows::Forms::TextBox());
44. this->labelAuthor = (gcnew System::Windows::Forms::Label());
45. this->boxDate = (gcnew System::Windows::Forms::MaskedTextBox());
46. this->labelDate = (gcnew System::Windows::Forms::Label());
47. this->boxPages = (gcnew System::Windows::Forms::TextBox());
48. this->labelPages = (gcnew System::Windows::Forms::Label());
49. this->buttonSave = (gcnew System::Windows::Forms::Button());
50. this->SuspendLayout();
51. //
52. // boxISBN
53. //
54. this->boxISBN->Location = System::Drawing::Point(12, 12);
55. this->boxISBN->Mask = L"000-0-00000-000-0";
56. this->boxISBN->Name = L"boxISBN";
57. this->boxISBN->Size = System::Drawing::Size(115, 20);
58. this->boxISBN->TabIndex = 2;
59. //
60. // labelISBN
61. //
62. this->labelISBN->AutoSize = true;
63. this->labelISBN->Location = System::Drawing::Point(12, 35);
64. this->labelISBN->Name = L"labelISBN";
65. this->labelISBN->Size = System::Drawing::Size(32, 13);
66. this->labelISBN->TabIndex = 3;
67. this->labelISBN->Text = L"ISBN";
68. //
69. // boxTitle
70. //
71. this->boxTitle->Location = System::Drawing::Point(12, 56);
72. this->boxTitle->Name = L"boxTitle";
73. this->boxTitle->Size = System::Drawing::Size(219, 20);
74. this->boxTitle->TabIndex = 4;
75. //
76. // labelTitle
77. //
78. this->labelTitle->AutoSize = true;
79. this->labelTitle->Location = System::Drawing::Point(12, 79);
80. this->labelTitle->Name = L"labelTitle";
81. this->labelTitle->Size = System::Drawing::Size(89, 13);
82. this->labelTitle->TabIndex = 5;
83. this->labelTitle->Text = L"Название книги";
84. //
85. // boxAuthor
86. //
87. this->boxAuthor->Location = System::Drawing::Point(12, 100);
88. this->boxAuthor->Name = L"boxAuthor";
89. this->boxAuthor->Size = System::Drawing::Size(219, 20);
90. this->boxAuthor->TabIndex = 6;
91. //
92. // labelAuthor
93. //
94. this->labelAuthor->AutoSize = true;
95. this->labelAuthor->Location = System::Drawing::Point(12, 123);
96. this->labelAuthor->Name = L"labelAuthor";
97. this->labelAuthor->Size = System::Drawing::Size(51, 13);
98. this->labelAuthor->TabIndex = 7;
99. this->labelAuthor->Text = L"Автор(ы)";
100. //
101. // boxDate
102. //
103. this->boxDate->Location = System::Drawing::Point(12, 144);
104. this->boxDate->Mask = L"00/00/0000";
105. this->boxDate->Name = L"boxDate";
106. this->boxDate->Size = System::Drawing::Size(100, 20);
107. this->boxDate->TabIndex = 8;
108. this->boxDate->ValidatingType = System::DateTime::typeid;
109. //
110. // labelDate
111. //
112. this->labelDate->AutoSize = true;
113. this->labelDate->Location = System::Drawing::Point(12, 167);
114. this->labelDate->Name = L"labelDate";
115. this->labelDate->Size = System::Drawing::Size(95, 13);
116. this->labelDate->TabIndex = 9;
117. this->labelDate->Text = L"Дата публикации";
118. //
119. // boxPages
120. //
121. this->boxPages->Location = System::Drawing::Point(12, 188);
122. this->boxPages->Name = L"boxPages";
123. this->boxPages->Size = System::Drawing::Size(100, 20);
124. this->boxPages->TabIndex = 10;
125. this->boxPages->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &GetData::BoxPages\_KeyPress);
126. //
127. // labelPages
128. //
129. this->labelPages->AutoSize = true;
130. this->labelPages->Location = System::Drawing::Point(12, 211);
131. this->labelPages->Name = L"labelPages";
132. this->labelPages->Size = System::Drawing::Size(83, 13);
133. this->labelPages->TabIndex = 11;
134. this->labelPages->Text = L"Число страниц";
135. //
136. // buttonSave
137. //
138. this->buttonSave->Location = System::Drawing::Point(137, 158);
139. this->buttonSave->Name = L"buttonSave";
140. this->buttonSave->Size = System::Drawing::Size(85, 39);
141. this->buttonSave->TabIndex = 12;
142. this->buttonSave->Text = L"Сохранить значения";
143. this->buttonSave->UseVisualStyleBackColor = true;
144. this->buttonSave->Click += gcnew System::EventHandler(this, &GetData::ButtonSave\_Click);
145. //
146. // GetData
147. //
148. this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
149. this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
150. this->ClientSize = System::Drawing::Size(244, 236);
151. this->Controls->Add(this->buttonSave);
152. this->Controls->Add(this->labelPages);
153. this->Controls->Add(this->boxPages);
154. this->Controls->Add(this->labelDate);
155. this->Controls->Add(this->boxDate);
156. this->Controls->Add(this->labelAuthor);
157. this->Controls->Add(this->boxAuthor);
158. this->Controls->Add(this->labelTitle);
159. this->Controls->Add(this->boxTitle);
160. this->Controls->Add(this->labelISBN);
161. this->Controls->Add(this->boxISBN);
162. this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;
163. this->Name = L"GetData";
164. this->Text = L"NewWindow";
165. this->ResumeLayout(false);
166. this->PerformLayout();
167. }
168. #pragma endregion
170. private: Int32 ID = -1;
171. private: System::Void BoxPages\_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e);
172. private: System::Void ButtonSave\_Click(System::Object^ sender, System::EventArgs^ e);
173. private: System::Boolean IsAnyFieldUncorrect();
174. public: BookNode^ GetDataFromBox();
176. };
177. }

## GetData.cpp

1. #include "GetData.h"
2. using namespace MumladzeLab6;
3. GetData::GetData(BookNode^ node) {
4. InitializeComponent();
5. if (node != nullptr) {
6. this->ID = node->ID;
7. this->boxISBN->Text = node->ISBN;
8. this->boxTitle->Text = node->title;
9. this->boxAuthor->Text = node->author;
10. this->boxDate->Text = node->date->ToString();
11. this->boxPages->Text = node->pageCount.ToString();
12. this->Text = "Изменить";
13. }
14. else {
15. this->Text = "Создать";
16. }
17. }
18. GetData::~GetData() {
19. if (components) {
20. delete components;
21. }
22. }
23. Void GetData::BoxPages\_KeyPress(Object^ sender, KeyPressEventArgs^ e) {
24. if ('0' <= e->KeyChar && e->KeyChar <= '9' || e->KeyChar == '\b') {
25. e->Handled = false;
26. }
27. else {
28. e->Handled = true;
29. }
30. }
31. Void GetData::ButtonSave\_Click(System::Object^ sender, System::EventArgs^ e) {
32. if (this->IsAnyFieldUncorrect()) {
33. MessageBox::Show("Найдено некорректное поле, перепроверьте ввод", "Внимание!", MessageBoxButtons::OK, MessageBoxIcon::Warning);
34. }
35. else {
36. this->DialogResult = System::Windows::Forms::DialogResult::OK;
37. this->Close();
38. }
39. }
40. Boolean GetData::IsAnyFieldUncorrect() {
41. if (this->boxISBN->Text->Length != 17)
42. return true;
43. if (String::IsNullOrWhiteSpace(this->boxTitle->Text))
44. return true;
45. if (String::IsNullOrWhiteSpace(this->boxAuthor->Text))
46. return true;
47. DateTime temp\_date;
48. if (!DateTime::TryParse(this->boxDate->Text, temp\_date))
49. return true;
50. Int32 temp\_int;
51. if (String::IsNullOrWhiteSpace(this->boxPages->Text))
52. return true;
53. return false;
54. }
55. BookNode^ GetData::GetDataFromBox() {
56. BookNode^ node = gcnew BookNode();
57. node->ID = this->ID;
58. node->ISBN = this->boxISBN->Text;
59. node->title = this->boxTitle->Text;
60. node->author = this->boxAuthor->Text;
61. node->date = DateTime::Parse(this->boxDate->Text);
62. node->pageCount = Int32::Parse(this->boxPages->Text);
63. return node;
64. }