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++   
В ОБЪЕКТНОМ СТИЛЕ»**

Вариант 3

# Содержание

Содержание 2

Техническое задание 3

Цель задания 3

Задача 3

Ход работы 4

Написание программы 4

Пример работы программы 4

Вывод 5

Листинг 6

app.cpp 6

datamanager.h 7

datamanager.cpp 9

# Техническое задание

## Цель задания

Цель задания: ознакомиться с возможностью создания классов в С++, которые можно в последствии использовать для разработки приложения под Windows (CLI-приложение).

## Задача

Задачи лабораторной работы:

1. Разработать объектную модель приложения.
2. Создать CLI-консольное приложение.
3. Создать в среде VS классы разработанной объектной модели.
4. Отладить методы классов.

Заданием моего варианта является работа с тремя файлами: список книг Books.txt, список дисциплин Subjects.txt, список связей между книгами и дисциплинами Links.txt.

# Ход работы

## Написание программы

Для работы с тремя файлами я сделал свой класс DataManager. Рассмотрю его поля и методы.

Класс имеет private поля String^ fileLinkPath, fileBookPath, fileSubjectPath. Они содержат адреса соответствующих текстовых файлов. Также у класса три private поля listLinks, listBooks, listSubjects. Они и будут хранить данные из трех файлов.

Рассмотрим public методы класса:

В конструкторе класса определяются пути к файлам Links.txt, Books.txt, Subjects.txt, а также списки через gcnew.

Функция ReadDataFiles читает данные из файлов при помощи System::IO::StreamReader в списки, и закрывает файлы.

Функция WriteDataFiles записывает в файлы данные из списков при помощи System::IO::StreamWriter. Строка для записи получается при помощи форматирование строк структуры данных в одну через String::Format.

Функция MakeNode принимает строку и формат данных, на ее основе создает соответствующую структуру данных и возвращает ее как Object^.

Функция AddNode принимает Object^ и формат данных и в соответствии с форматом записывает в нужный список объект.

Функция RemoveNode принимает Object^ и формат данных, ищет в списке данные по объекту, и удаляет вхождения. Если удаляется книга или дисциплина, также удаляются все связанные с ними связи.

Функция FindNodes принимает Object^ и формат данных, ищет в списке похожие данные (при помощи (String^) line->Contains()), собирает найденные объекты в List<Objects^>^, и возвращает его.

## Пример работы программы

Изображение выглядит как текст, снимок экрана, меню, черно-белый

Автоматически созданное описание

# Вывод

Итогом работы стали три класса, которые работают с тремя файлами с данными, могут оперировать ими (добавлять/удалять/искать).

# Листинг

## app.cpp

1. #include "DataManager.h"
2. using namespace System;
3. int main(array<String^>^ args) {
4. DataManager^ dm = gcnew DataManager();
5. dm->ReadDataFiles();
6. dm->AddNode(T\_BOOK, dm->MakeNode(T\_BOOK, "012345; Книга; Автор"));
7. dm->AddNode(T\_SUBJECT, dm->MakeNode(T\_SUBJECT, "111; Физра; Спорт"));
8. dm->AddNode(T\_LINK, dm->MakeNode(T\_LINK, "012345; 111"));
9. Object^ del\_book = dm->MakeNode(T\_BOOK, "012345; -; -");
10. dm->DeleteNode(T\_BOOK, del\_book);
11. Object^ del\_subj = dm->MakeNode(T\_SUBJECT, "111; -; -");
12. dm->DeleteNode(T\_SUBJECT, del\_subj);
13. dm->AddNode(T\_BOOK, dm->MakeNode(T\_BOOK, "012345; Книга; Автор"));
14. dm->WriteDataFiles();
15. for (int i = 0; i < 1; i++) {
16. i--;
17. };
18. }

## 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->Contains(line) || link->dis\_code->Contains(line)) {
204. list->Add(link);
205. }
206. }
207. break;
208. case T\_BOOK:
209. for each (Book^ book in (this->listBooks)) {
210. if (book->ISBN->Contains(line) || book->title->Contains(line) || book->author->Contains(line)) {
211. list->Add(book);
212. }
213. }
214. break;
215. case T\_SUBJECT:
216. for each (Subject^ subj in (this->listSubjects)) {
217. if (subj->dis\_code->Contains(line) || subj->name->Contains(line) || subj->description->Contains(line)) {
218. list->Add(subj);
219. }
220. }
221. break;
222. default:
223. Console::WriteLine("Ошибка, выбран некорректный формат данных");
224. break;
225. }
226. Console::WriteLine("Найдено " + find\_count + " записей по заданной строке");
227. return list;
228. }