Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №1

по дисциплине «Программные технологии проектирования ПО ВС»

на тему «Начало работы»

Выполнили студенты группы 22ВВП1

Хоссейни Нежад С.А.С.М.

Захаров А. С.

Приняли:

Деев М.В.

Патунин Д.В.

Пенза 2024

**Название**

Начало работы

**Цель работы**

Изучение объектно-ориентированного программирования и получение навыков написания программ, содержащих файловый ввод-вывод, контейнеры, итераторы и регулярные выражения на примере языка программирования C++.

**Лабораторное задание**

Написать программу, позволяющую пользователю извлекать из текстового документа, написанного с использованием языка разметки, определённое содержимое. Язык разметки и извлекаемое содержимое выбираются в соответствии с вариантом:



Имя файла передаётся как аргумент командной строки.

Для создания и управления коллекциями объектов использовать контейнеры. Для обхода контейнеров использовать итераторы.

Поиск производить посредством регулярных выражений.

**Описание программы**

Написали класс, который принимает в конструктор имя файла. В конструкторе мы сразу записываем весь файл в вектор lines. В классе написали методы getHeaders(), getURLs() и getParagraphs(), которые с помощью регулярных выражений обрабатывают строки файла и заносят нужные данные соответственно в вектора headers, urls, paragraphs. Написали в этом классе также методы printHeaders(), printURLs() и printParagraphs() для вызова соответственно методов getHeaders(), getURLs() и getParagraphs() и вывода их результата работы.

**Текст программы**

#include <iostream>

#include <regex>

#include <fstream>

#include <string>

#include <vector>

#include <iterator>

using namespace std;

class ProcessAskiiDoc

{

private:

vector<string> lines;

vector<string> headers;

vector<string> urls;

vector<string> paragraphs;

regex headerPattern{ R"(^=+\s+(.+)$)" };

regex urlPattern{ R"(https?://[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)+([/?#][^\s\[]\*)?)" };

regex paragraphPattern{ R"(^[^\s=\*+-.|>[].\*)" };

regex paragraphpPatternIV{ R"(^\s\*(?!image::|video::).\*)" };

regex codeBlockPattern{ R"(^(----|....)$)" };

bool codeBlock = false;

public:

ProcessAskiiDoc(const string& fileName)

{

ifstream inputFile(fileName);

if (!inputFile)

{

cout << "Error! Еhe file could not be opened!" << endl;

return;

}

string line;

while (getline(inputFile, line))

lines.push\_back(line);

inputFile.close();

}

vector<string> getHeaders()

{

smatch matches;

for (const auto& line : lines)

if (regex\_match(line, matches, headerPattern) && matches.size() > 1)

headers.push\_back(matches[1]);

return headers;

}

vector<string> getURLs()

{

smatch matches;

for (const auto& line : lines)

{

auto searchStart = line.cbegin();

while (regex\_search(searchStart, line.cend(), matches, urlPattern))

{

urls.push\_back(matches[0]);

searchStart = matches.suffix().first;

}

}

return urls;

}

vector<string> getParagraphs()

{

for (const auto& line : lines)

{

if (regex\_match(line, codeBlockPattern))

{

if (codeBlock == false)

codeBlock = true;

else

codeBlock = false;

continue;

}

if (codeBlock)

continue;

if (regex\_match(line, paragraphPattern) && regex\_match(line, paragraphpPatternIV))

paragraphs.push\_back(line);

}

return paragraphs;

}

void printHeaders()

{

getHeaders();

for (auto it = headers.begin(); it != headers.end(); ++it)

cout << "h: " << \*it << endl;

}

void printURLs()

{

getURLs();

for (auto it = urls.begin(); it != urls.end(); ++it)

cout << "URL: " << \*it << endl;

}

void printParagraphs()

{

getParagraphs();

for (auto it = paragraphs.begin(); it != paragraphs.end(); ++it)

cout << "p: " << \*it << endl;

}

};

int main(int argc, char\* argv[]) {

if (argc < 2)

{

cout << "Error: No filename provided!" << endl;

return 1;

}

ProcessAskiiDoc file(argv[1]);

file.printHeaders();

file.printURLs();

file.printParagraphs();

return 0;

}

**Текст файла**

== Introduction

AsciiDoc is a lightweight markup language used for writing articles, books, documentation, and presentations. It supports structured and unstructured text, links, tables, images, and more complex formatting like code blocks and quotes.

In this document, we will explore the different elements of AsciiDoc such as:

- Headings

- Paragraphs

- Code blocks

- Tables

- Quotes

- Lists

- Images

- Links

== Paragraphs

A paragraph in AsciiDoc is simply one or more lines of text. You can create paragraphs by writing your content in a continuous block. Here’s an example of a simple paragraph.

This is a paragraph in AsciiDoc. It is created by simply writing lines of text without any special markup. AsciiDoc will handle the formatting automatically.

== Lists

AsciiDoc supports various types of lists, including unordered, ordered, and description lists.

=== Unordered List

- Item one

- Item two

- Item three

=== Ordered List

. First item

. Second item

. Third item

== Links

You can include hyperlinks in AsciiDoc easily.

Here is a simple inline link to the AsciiDoc website: https://asciidoc.org[AsciiDoc Official Website].

== Images

Images can be embedded in AsciiDoc with the `image::` directive.

image::https://asciidoc.org/images/icons/asciidoc\_icon\_128.png[AsciiDoc Logo]

== Quotes

AsciiDoc allows the use of block quotes to emphasize certain content.

> AsciiDoc is a powerful language that can be used for a wide range of documentation needs.

== Code Blocks

To include source code or configuration files, use the `----` or `....` delimiters for code blocks.

[source,c++]

----

#include <iostream>

using namespace std;

int main() {

cout << "Hello, AsciiDoc!" << endl;

return 0;

}

----

Or, using the `....` delimiter:

....

print("This is a Python code block in AsciiDoc.")

....

== Tables

AsciiDoc also supports the creation of tables. Below is an example of a simple table:

[cols="1,2,3",options="header"]

|===

| Name | Age | Occupation

| John | 28 | Developer

| Sarah | 32 | Designer

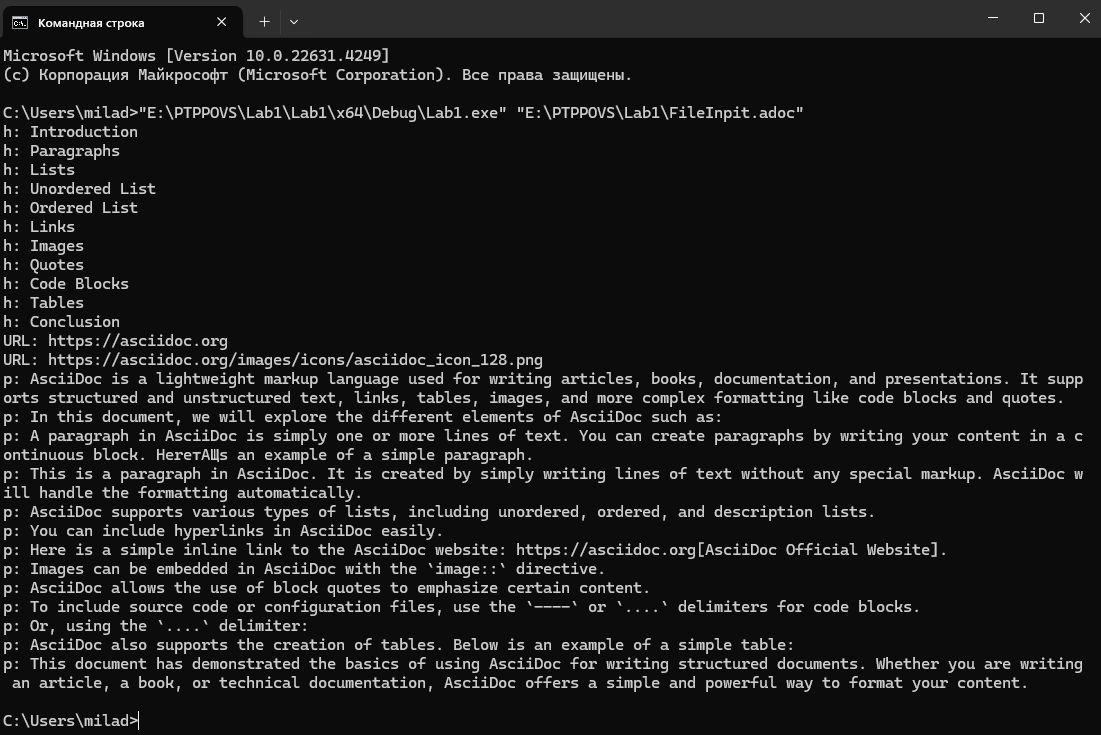
| Alex | 25 | Engineer

|===

== Conclusion

This document has demonstrated the basics of using AsciiDoc for writing structured documents. Whether you are writing an article, a book, or technical documentation, AsciiDoc offers a simple and powerful way to format your content.

**Результат работы программы**

****

**Вывод**

Изучили объектно-ориентированного программирование и получили навыки написания программ, содержащих файловый ввод-вывод, контейнеры, итераторы и регулярные выражения на примере языка программирования C++.