УО «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра «Информационных систем и технологий»

Специальность 1-40 05 01 «Информационные Системы и Технологии»

**Реферат**

**На тему «Работа с XML в С++»**

|  |  |  |  |
| --- | --- | --- | --- |
| **Выполнил** |  |  |  |
| Студент 1 курса группы 1 |  |  | Д.И. Велютич |
|  | подпись, дата |  | инициалы и фамилия |
| **Проверил** |  |  | Н.И. Белодед |
|  | подпись, дата |  | инициалы и фамилия |

Минск 2023

Оглавление

[Введение и теория 2](#_Toc134322939)

[Объяснения работы программы 4](#_Toc134322940)

[Программный код 6](#_Toc134322941)

[Скриншоты и объяснения работы программы 7](#_Toc134322942)

[Выводы 9](#_Toc134322943)

# Введение и теория

**XML** (eXtensible Markup Language) – расширяемый язык разметки, который был разработан World Wide Web Consortium (W3C) для структурирования и хранения данных. XML является метаязыком, что позволяет пользователям определять свои собственные теги для описания данных. Этот язык используется в различных областях и приложениях, таких как веб-сервисы, настройки программ, обмен данными между системами и многое другое.

**Зачем он нужен:**

XML предназначен для структурирования, хранения и передачи данных. Он позволяет организовать информацию в иерархические структуры, которые легко читаются и понимаются как человеком, так и машиной. XML используется для разработки собственных языков разметки, что позволяет создать универсальный и гибкий инструмент для работы с данными.

**Где используется:**

XML используется в самых разных областях и приложениях:

1. Веб-сервисы: XML является основой для таких технологий, как SOAP (Simple Object Access Protocol) и REST (Representational State Transfer), которые позволяют обмениваться данными между различными приложениями и системами через интернет.

2. Конфигурация программ: XML часто используется для хранения настроек и конфигурации различных программ, что обеспечивает структурированный и удобный способ организации информации.

3. Документы: XML применяется для хранения и обработки документов в разных областях, таких как MathML (язык разметки математических формул) или DocBook (язык разметки для технической документации).

4. Базы данных: XML используется для представления данных в некоторых базах данных, таких как XML-базы данных или иерархические базы данных.

**Плюсы:**

1. Читаемость: XML легко читается и понимается как человеком, так и машиной, благодаря ясной иерархической структуре и текстовому представлению.

2. Расширяемость: XML позволяет пользователям создавать свои собственные теги и языки разметки, что делает его гибким и масштабируемым инструментом.

3. Стандартизация: XML является открытым стандартом, поддерживаемым множеством раз личных платформ и языков программирования. Это обеспечивает широкую совместимость и возможность использования XML в разнообразных сценариях.

4. Отделение данных от представления: XML позволяет разделить данные от их представления, что облегчает работу с данными и упрощает их обработку и обновление. Например, с помощью XSLT (Extensible Stylesheet Language Transformations) можно преобразовать XML-данные в HTML, PDF или другие форматы для отображения или печати.

5. Поддержка Unicode: XML поддерживает Unicode, что позволяет использовать разные языки и символы в данных, сохраняя их корректное представление.

**Минусы:**

1. Вербозность: XML-документы могут быть довольно громоздкими из-за большого количества тегов и атрибутов. Это может привести к увеличению объема данных, которые необходимо передавать и хранить, что снижает производительность и эффективность использования ресурсов.

2. Сложность: XML может быть сложным для изучения и использования, особенно когда речь идет о создании и применении собственных языков разметки или работы с продвинутыми технологиями, такими как XSLT и XML Schema.

3. Отсутствие нативной поддержки типов данных: XML не предоставляет нативной поддержки типов данных, таких как числа, даты и логические значения. Все данные в XML представлены в виде текста, что может усложнить обработку и валидацию данных.

4. Конкуренция с другими форматами: С развитием технологий появились другие форматы для структурирования и обмена данными, такие как JSON (JavaScript Object Notation), которые могут быть более простыми, компактными и быстрыми в некоторых случаях использования. Это может снизить актуальность XML в определенных областях.

# Объяснения работы программы

Данная программа использует библиотеку tinyxml2 для чтения и обработки XML-файлов. Работу программы можно разделить на несколько этапов:

1. Установка локали (русского языка) для корректного отображения кириллицы в консоли:

setlocale(0, "rus");

2. Создание объекта XML-документа:

tinyxml2::XMLDocument doc;

3. Загрузка XML-файла и проверка успешности загрузки:

tinyxml2::XMLError loadResult = doc.LoadFile("example.xml");

if (loadResult == tinyxml2::XML\_SUCCESS) {

// Если файл успешно загружен, продолжаем работу с ним

}

else {

// В случае ошибки выводим сообщение об ошибке

}

4. Получение корневого элемента XML-документа и проверка его наличия:

tinyxml2::XMLElement\* root = doc.FirstChildElement("RootElement");

if (root) {

// Если корневой элемент найден, продолжаем работу с дочерними элементами

}

else {

// В случае отсутствия корневого элемента выводим сообщение об ошибке

}

5. Обход дочерних элементов с именем "ChildElement" и их обработка:

tinyxml2::XMLElement\* child = root >FirstChildElement("ChildElement");

while (child) {

// Получение значения атрибута с именем "attribute\_name"

const char\* value = child->Attribute("attribute\_name");

// Проверка наличия атрибута и вывод его значения

if (value) {

// Вывод значения атрибута

}

else {

// В случае отсутствия атрибута выводим сообщение об ошибке

}

// Переход к следующему дочернему элементу с именем "ChildElement"

child = child->NextSiblingElement("ChildElement");

}

В результате выполнения программы на экран будут выведены сообщения о наличии корневого элемента и его дочерних элементов, а также значения атрибутов с именем "attribute\_name" для каждого дочернего элемента "ChildElement". Если атрибут отсутствует, программа выведет сообщение об ошибке.

# Программный код

#include <iostream>

#include "tinyxml2.h" // Подключение библиотеки TinyXML - 2

int main() {

setlocale(0, "rus"); // Установка локали для корректного отображения кириллицы

tinyxml2::XMLDocument doc; // Создание объекта XML-документа

tinyxml2::XMLError loadResult = doc.LoadFile("example.xml"); // Загрузка XML-файла

// Проверка успешности загрузки XML-файла

if (loadResult == tinyxml2::XML\_SUCCESS) {

std::cout << "XML-файл загружен успешно." << std::endl;

// Получение корневого элемента XML-документа

tinyxml2::XMLElement\* root = doc.FirstChildElement("RootElement");

// Проверка наличия корневого элемента

if (root) {

std::cout << "Корневой элемент найден." << std::endl;

// Получение первого дочернего элемента с именем "ChildElement"

tinyxml2::XMLElement\* child = root->FirstChildElement("ChildElement");

// Обход всех дочерних элементов с именем "ChildElement"

while (child) {

std::cout << "Дочерний элемент найден." << std::endl;

// Получение значения атрибута с именем "attribute\_name"

const char\* value = child->Attribute("attribute\_name");

// Проверка наличия атрибута и вывод его значения

if (value) {

std::cout << "Значение атрибута: " << value << std::endl;

}

else {

std::cout << "Атрибут 'имя\_атрибута' не найден." << std::endl;

}

// Переход к следующему дочернему элементу с именем "ChildElement"

child = child->NextSiblingElement("ChildElement");

}

}

else {

std::cout << "Корневой элемент не найден." << std::endl;

}

}

else {

std::cout << "Не удалось загрузить XML-файл." << std::endl;

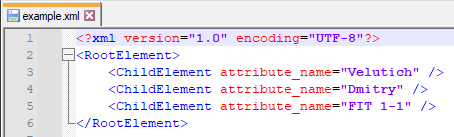
}

return 0;

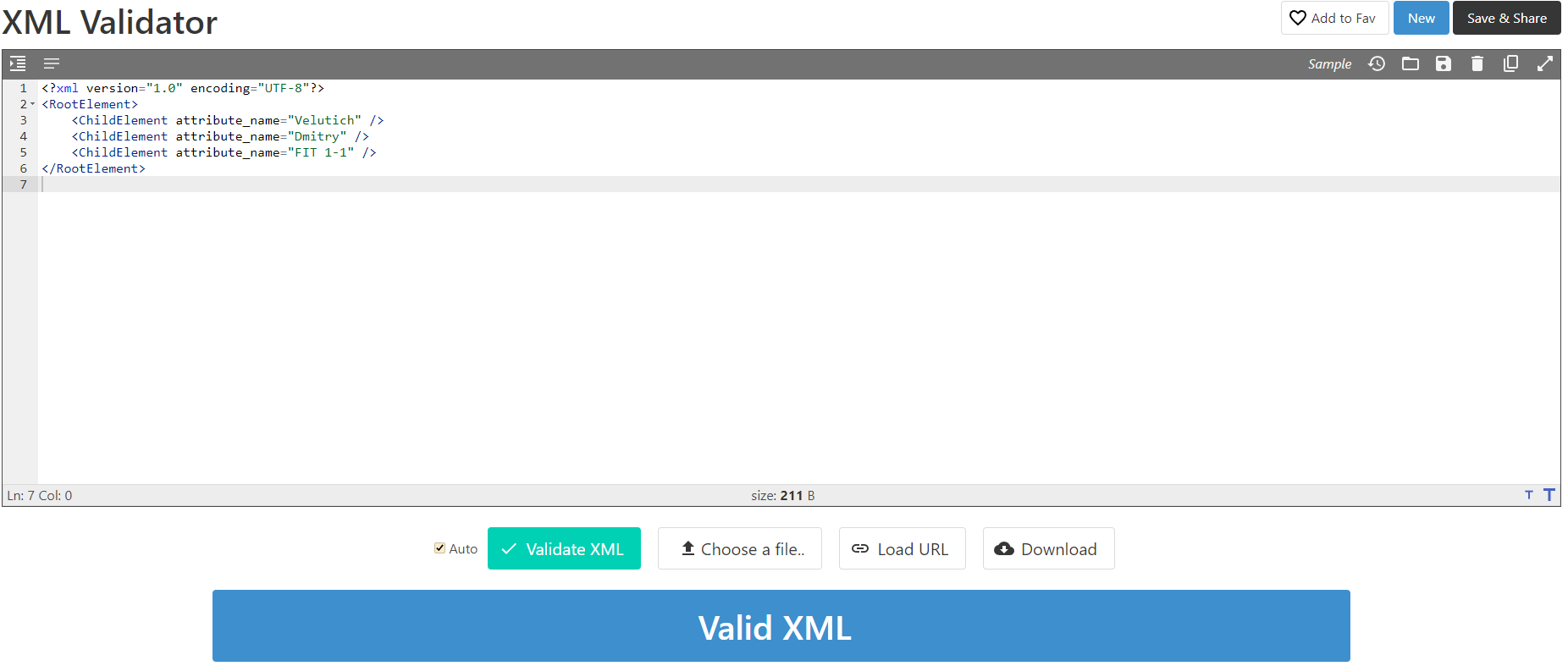
}

# Скриншоты и объяснения работы программы

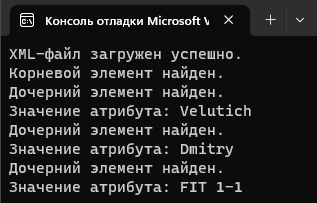
Первым делом создадим файл *example.xml*, который будет содержать какую-то информацию и проходить валидацию формата XML в целом.



Далее сразу проверим файл на валидность стандартам **W3C** на сайте <https://codebeautify.org/xmlvalidator> (валидатор одобрил код)



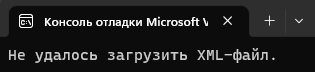
Теперь спокойно можем запускать нашу программу



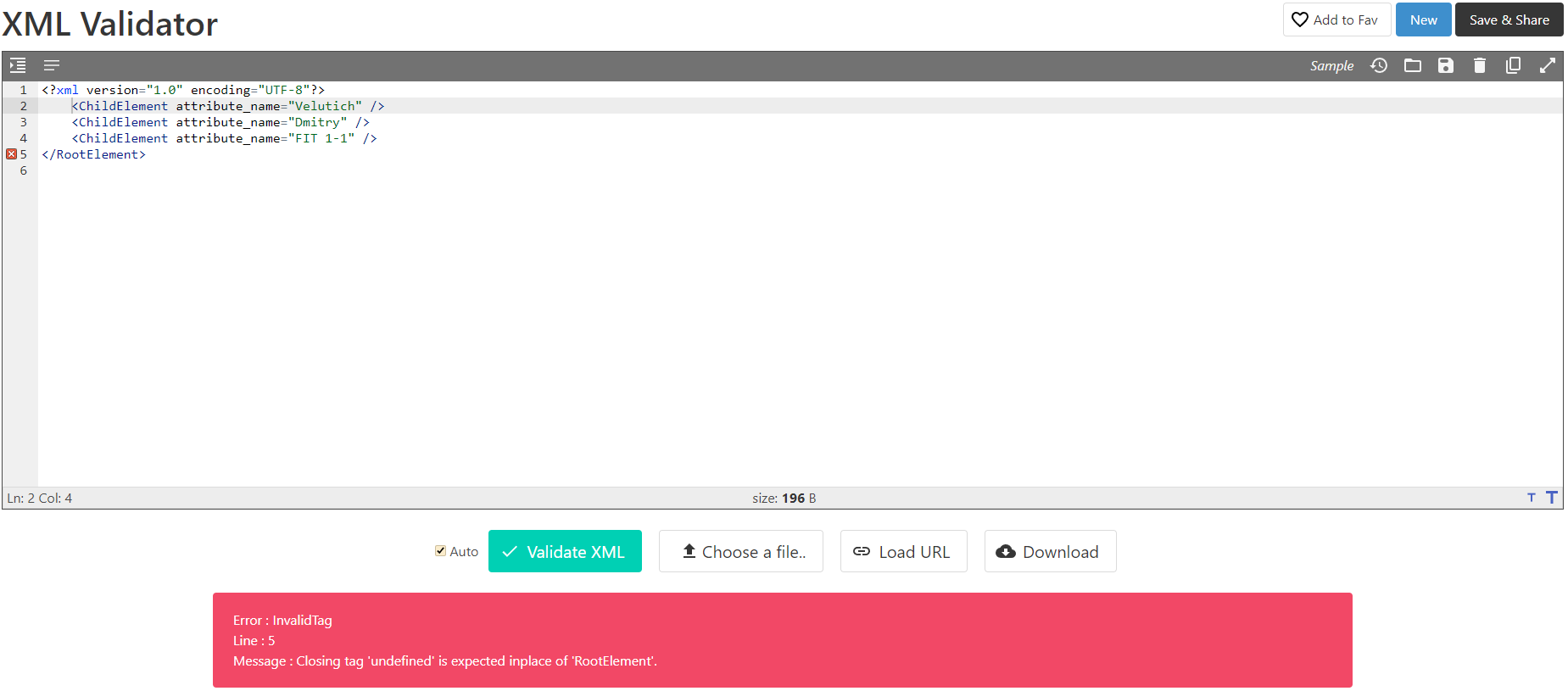
Мы видим следующие сообщения:

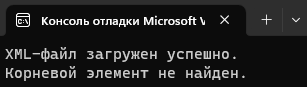
* XML-файл успешно загружен, что говорит нам о том, что программа в принципе нашла файл с именем example.xml и смогла его открыть
* Корневой элемент найден, ещё раз подтверждает то, что структура XML валидна и найден корневой элемент
* Дочерний элемент найден – идентично с корневым элементом, только это дочерний :)
* Программа последовательно перебирает дочерние элементы и выводит значения атрибутов в окно консоли.

Такой идеальный исход будет далеко не всегда, к примеру если программа не найдет файл example.xml, мы увидим следующее сообщение



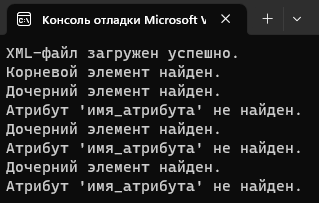
Теперь нарушим структуру XML документа и проверим, что скажет программа





В предыдущем примере мы нарушили структуру тем, что не открыли корневой элемент, соответственно это нарушает структуру XML и программа не может найти корневой элемент.

Теперь попробуем починить корневой элемент и удалить значения атрибутов всех дочерних элементов. Получим вот такой результат от программы.



# Выводы

В целом, XML является мощным и гибким инструментом для структурирования и обмена данными, который нашел применение во множестве областей и приложений. Однако, его использование может быть не всегда оправданным из-за некоторых недостатков, и в зависимости от конкретной задачи могут быть предпочтительными другие форматы данных.