Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Регулярные выражения в С++»

Выполнила:

Студент(ка) 1 курса 7 группы

Подшиваленко Диана Игоревна

Проверил:

Белодед Николай Иванович

2023, Минск

**Содержание**

1. Предисловие … 3
2. Что такое регулярные выражения … 4
3. Специальные символы в регулярных выражениях … 4
4. Примеры использования … 5

4.1 Проверка корректности ввода времени … 5

4.2 Проверка корректности ввода номера телефона … 6

4.3 Проверка корректности ввода gmail … 7

4.4 Поиск подстроки в строке … 8

4.5 Замена подстроки в строке … 9

5. Вывод … 10

**Предисловие**

В данном реферате будет рассмотрен такой инструмент проверки вводимой информации как регулярные выражения, а также будут приведены примеры их использования.

**1. Что такое регулярные выражения**

Регулярные выражения (Regular Expressions) — это мощный инструмент для работы с текстом. Они представляют собой последовательность символов, которые задают шаблон поиска и манипуляции с текстовыми данными. Их используют разработчики в коде приложения, тестировщики в тестах, даже просто при работе в командной строке.

Результатом работы с регулярным выражением может быть:

- проверка наличия искомого образца в заданном тексте;

- определение подстроки текста, которая сопоставляется образцу;

- определение групп символов, соответствующих отдельным частям образца.

Регулярные выражения используются для следующих задач:

1) **Поиск и сопоставление**: Регулярные выражения позволяют находить текст, соответствующий определенным шаблонам.

2) **Замена**: Регулярные выражения позволяют заменять текст, соответствующий определенным шаблонам, на другой текст.

3) **Валидация**: Регулярные выражения позволяют проверять, соответствует ли строка определенным правилам или формату.

Регулярные выражения имеют свой синтаксис, который позволяет создавать собственные шаблоны для поиска и работы с текстом. Многие языки программирования и текстовые редакторы предоставляют поддержку работы с регулярными выражениями, обеспечивают функции для выполнения операций поиска, замены и проверки соответствия.

**2. Специальные символы в регулярных выражениях**

В регулярных выражения есть свои специальные обозначения:

1) . – показывает, что это может быть один любой символ

2) \ – символ экранирования

3) \* – символ повторяется от 0 до бесконечности раз

4) + – повторяется от 1 до бесконечности раз

5) ? – повторяется 0 или 1 раз

6) ^ – строка должна начинаться с такого символа или нескольких символов, которые следуют после этого знака

7) [^] – все, что идет после уголка, не должно встречаться в строке

8) [] – означает, что в строке должно быть то, то расположено внутри квадратных скобок

9) ^[^] – строка не должна начинаться на то, что находится внутри квадратных скобок после уголка

10) $ – строка должна заканчиваться на символ или символы, которые идут до знака $

11) \w – этот символ заменяет любую латинскую букву, цифру, а также символ подчеркивания

12) \W – это любой символ, кроме символов, которые включает в себя \w

13) \d – любая цифра

14) \D – любой символ, кроме цифры

15) \s – символ пробела, табуляции, отступа или перехода на новую строку

16) \S – все, кроме символов, которые включает в себя \s

17) - – специальный символ, который, если находится в квадратных скобках, то говорит о диапазоне

18) {} – фигурными скобками указывается, сколько раз нужно, чтобы встречался символ или символы.

19) () – круглые скобки позволяют указывать варианты через вертикальную черту |

20) (?<=) – поиск окружающего текста слева от нужного слова или символа.

21) (?=) – поиск окружающего текста справа от нужного слова или символа

**3. Примеры использования**

**3.1 Проверка корректности ввода времени**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <string>

#include <regex>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n;

cout << "Введите количество записей\n";

cin >> n;

string a;

regex regular("([0-1]{1}[0-9]{1}|[2]{1}[0-3]{1})(:)([0-5]{1}[0-9]{1})(:)([0-5]{1}[0-9]{1})"); // регулярное выражение

SetConsoleOutputCP(1251);

for (int i = 0; i < n; i++) {

cout << "Введите время в формате ЧЧ:ММ:СС\n";

cin >> a;

if (regex\_match(a.c\_str(), regular)) {

cout << "Корректный ввод\n";

}

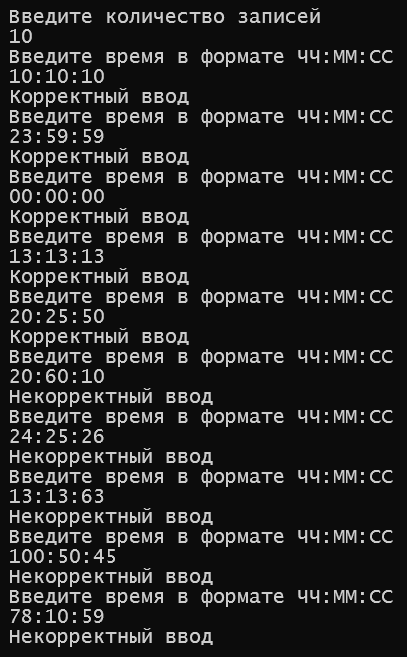
else {

cout << "Некорректный ввод\n";

}

}

}

Результат выполнения:

**3.2 Проверка корректности ввода номера телефона**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <string>

#include <regex>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n;

cout << "Введите количество записей\n";

cin >> n;

string a;

regex regular("(^\\+375)([0-9]{9})"); // регулярное выражение

SetConsoleOutputCP(1251);

for (int i = 0; i < n; i++) {

cout << "Введите номер телефона с кодом +375\n";

cin >> a;

if (regex\_match(a.c\_str(), regular)) {

cout << "Корректный ввод\n";

}

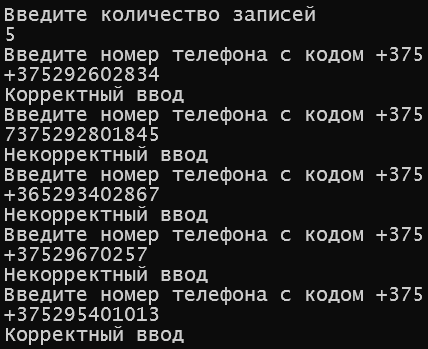
else {

cout << "Некорректный ввод\n";

}

}

}

Результат выполнения:

**3.3 Проверка корректности ввода gmail**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <string>

#include <regex>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n;

cout << "Введите количество записей\n";

cin >> n;

string a;

regex regular("((^[\\w]+)(@){1}(gmail){1}(\\.)(com$))"); // регулярное выражение

SetConsoleOutputCP(1251);

for (int i = 0; i < n; i++) {

cout << "Введите номер телефона с кодом +375\n";

cin >> a;

if (regex\_match(a.c\_str(), regular)) {

cout << "Корректный ввод\n";

}

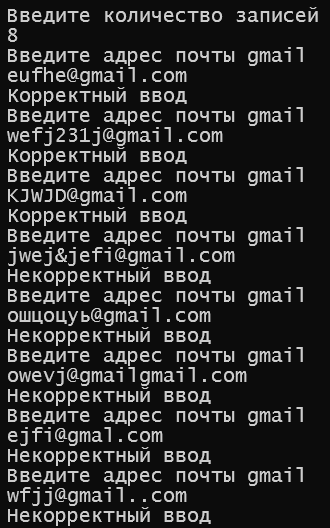
else {

cout << "Некорректный ввод\n";

}

}

}

Результат выполнения:

**3.4 Поиск подстроки в строке**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <string>

#include <regex>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Введите названия языков программирования на английском языке\n";

string a;

getline(cin, a);

regex regular1("(([pP]+)([yY]+)([Tt]+)([Hh]+)([oO]+)([Nn]+))"); // регулярное выражение

regex regular2("([cC]+\\++\\++)"); // регулярное выражение

regex regular3("([Gg]+[Oo]+)");

if (regex\_search(a.c\_str(), regular2)) {

cout << "C++ найден\n";

}

else {

cout << "C++ не найден\n";

}

if (regex\_search(a.c\_str(), regular1)) {

cout << "Python найден\n";

}

else {

cout << "Python не найден\n";

}

if (regex\_search(a.c\_str(), regular3)) {

cout << "GO найден\n";

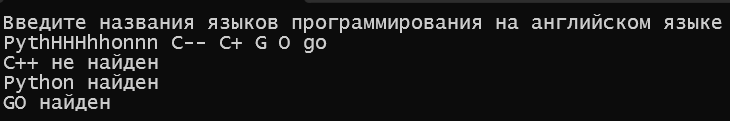
}

else {

cout << "Go не найден\n";

}

}

Результат выполнения:

**3.5 Замена подстроки в строке**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <string>

#include <regex>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string a = "Hello, World! Hello, people!";

cout << "Текущая строка: " << a << '\n';

regex regular1("(Hello)"); // регулярное выражение

a = regex\_replace(a, regular1, "Hi");

cout << "Полученная строка: " << a;

}

Результат выполнения:

**Вывод**

Использование регулярных выражений в программировании может быть очень мощным инструментом для работы со строками. Они позволяют выполнить различные операции, такие как поиск, замена. Регулярные выражения предоставляют хороший способ описания шаблонов, позволяют находить и извлекать подстроки из текста, заменять найденные совпадения подстрок на другие строки или выражения, могут использоваться для проверки соответствия строки определенному формату. Однако регулярные выражения могут быть сложными для понимания и написания, особенно для сложных шаблонов, могут иметь свои ограничения в том, что они могут сделать. Поэтому их надо использовать с осторожностью и только там, где это по-настоящему упростит вам работу.