**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи №12

На тему: «Алгоритм Бойєра-Мура»

З дисципліни: «Алгоритми та структури даних»

**Лектор** : доцент каф.ПЗ

Коротєєва Т.О.

**Виконала:** ст.гр.ПЗ-23

Кохман О.В.

**Прийняв:** асистент каф.ПЗ

Франко А.В.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2022 р.

\_ \_ \_ \_ \_ \_ \_ \_ .

Львів – 2022

**Тема:** Алгоритм Бойєра-Мура.

**Мета:** дізнатись про алгоритм Бойєра-Мура та реалізувати програму за допомогою цього алгоритму.

**Теоретичні відомості**

Алгоритм пошуку рядка Бойера - Мура - алгоритм загального призначення, призначений для пошуку підрядка в рядку. Розроблений Робертом Бойєром і Джеєм Муром 1977 року. Перевага цього алгоритму в тому, що ціною деякої кількості попередніх обчислень над шаблоном (але не над рядком, в якому ведеться пошук), шаблон порівнюється з вихідним текстом не у всіх позиціях - частина перевірок пропускається як результат, що не дає свідомо.

Загальна оцінка обчислювальної складності сучасного варіанта алгоритму Бойєра - Мура - O(n+m) , якщо не використовується таблиця стоп-символів. O(n+m+|Sigma |), якщо використовується таблиця стоп-символів, де n — довжина рядка, в якому виконується пошук, m - довжина шаблону пошуку, Sigma - алфавіт, на якому проводиться порівняння .

Покроковий опис алгоритму Бойера – Мура:

Алгоритм BM

Дано S[0..n] – стрічка, в якій відбувається пошук; P[0..m] – стрічка, входження якої у S необхідно найти; k –позиція стоп-символу в S; j –позиція стоп-символу в P; h-крок.

BM1. Повторювати кроки BM2, BM3, BM4.

BM2. Знайти стоп-символ у стрічці S.

BM3. Знайти стоп-символ у стрічці P. Якщо він відсутній, то h=m; якщо k>j, то h=|k-j|; якщо k<j, то h=1.

BM4. Перемістити позицію першого елемента стрічки P на h.

BM5. Кінець. Вихід.

**Індивідуальне завдання**

Виконати вказане для свого варіанту завдання. Виміряти час пошуку, а також час (окремо) додаткового опрацювання тексту та/чи взірця, якщо таке є.

10 варіант: Задано два тексти. В першому тексті знайти слово яке складається з найменшої кількості голосних літер замінити його на слово записане у зворотньому порядку і знайти його входження в другий текст відповідним алгоритмом пошуку.

**Код програми**

Назва файлу : MyForm.cpp

#include "MyForm.h"

using namespace Project1;

int main() {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Application::Run(gcnew MyForm());

return 0;

}

Назва файлу : MyForm.h

#pragma once

#include "standardString.h"

#include <string>

#include <chrono>

using namespace std;

using namespace std::chrono;

namespace Project1 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ button2;

protected:

private: System::Windows::Forms::RichTextBox^ richTextBox3;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::RichTextBox^ richTextBox2;

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->button2 = (gcnew System::Windows::Forms::Button());

this->richTextBox3 = (gcnew System::Windows::Forms::RichTextBox());

this->button1 = (gcnew System::Windows::Forms::Button());

this->richTextBox2 = (gcnew System::Windows::Forms::RichTextBox());

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->SuspendLayout();

//

// button2

//

this->button2->Location = System::Drawing::Point(353, 178);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(113, 33);

this->button2->TabIndex = 9;

this->button2->Text = L"find";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// richTextBox3

//

this->richTextBox3->Location = System::Drawing::Point(208, 217);

this->richTextBox3->Name = L"richTextBox3";

this->richTextBox3->Size = System::Drawing::Size(394, 195);

this->richTextBox3->TabIndex = 8;

this->richTextBox3->Text = L"";

//

// button1

//

this->button1->Location = System::Drawing::Point(353, 39);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(113, 37);

this->button1->TabIndex = 7;

this->button1->Text = L"generate";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// richTextBox2

//

this->richTextBox2->Location = System::Drawing::Point(446, 92);

this->richTextBox2->Name = L"richTextBox2";

this->richTextBox2->Size = System::Drawing::Size(299, 64);

this->richTextBox2->TabIndex = 6;

this->richTextBox2->Text = L"";

//

// richTextBox1

//

this->richTextBox1->Location = System::Drawing::Point(66, 92);

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->Size = System::Drawing::Size(299, 64);

this->richTextBox1->TabIndex = 5;

this->richTextBox1->Text = L"";

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(794, 521);

this->Controls->Add(this->button2);

this->Controls->Add(this->richTextBox3);

this->Controls->Add(this->button1);

this->Controls->Add(this->richTextBox2);

this->Controls->Add(this->richTextBox1);

this->Name = L"MyForm";

this->Text = L"MyForm";

this->ResumeLayout(false);

}

#pragma endregion

String^ text1 = "Enforcement only partially law understood enactment laws regulations. Instead examine power-infused, dynamic assemblages";

String^ text2 = "Enforcement only partially law understood wal lws enactment laws swal regulations. Instead examine power-infused, dynamic assemblages";

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

richTextBox1->Text = text1;

richTextBox2->Text = text2;

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

char str[] = "Enforcement only partially law understood wal lws enactment laws swal regulations.Instead examine power - infused, dynamic assemblages";

int i;

string s = "";

for (i = 0; i < 130; i++) {

s = s + str[i];

}

string my = mySearch(s);

int length = my.length();

string myNew = "";

for (int i = length - 1; i >= 0; i--) {

myNew += my[i];

}

richTextBox3->Text += "\nWord to search:" + (gcnew String(myNew.c\_str())) + "\n";

const char\* result = new char[length];

result = myNew.c\_str();

auto start = high\_resolution\_clock::now();

int pos = BMSearch(str, result, richTextBox3);

auto stop = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>(stop - start);

richTextBox3->Text += "shift - " + pos + "\n";

richTextBox3->Text += "Time -: " + duration.count() + " microseconds";

}

private:string mySearch(string text) {

char vowels[] = { 'a', 'o', 'i', 'e', 'u', 'y' };

int size = text.length();

char\* char\_array = new char[size + 1];

char\* char\_array2 = new char[size + 1];

strcpy(char\_array, text.c\_str());

strcpy(char\_array2, text.c\_str());

char\* token;

int arraySize = 0;

token = strtok(char\_array, " ");

while (token != NULL) {

token = strtok(NULL, " ");

arraySize++;

}

string\* array = new string[arraySize];

char\* token2;

token2 = strtok(char\_array2, " ");

int i = 0;

while (token2 != NULL) {

array[i] = token2;

token2 = strtok(NULL, " ");

i++;

}

int count = 0;

int min = 10;

string minS = "initial";

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

int length = array[i].length();

for (int j = 0; j < length; j++) {

for (int t = 0; t < 6; t++) {

if (array[i][j] == vowels[t]) {

count++;

}

}

}

if (count < min && count != 0) {

min = count;

minS = array[i];

}

count = 0;

}

return minS;

}

private: int BMSearch(char\* string, const char\* substring, System::Windows::Forms::RichTextBox^ richTextBox) {

int stringLength = 0;

int substringLength = 0;

int result = -1;

while (string[stringLength] != NULL) {

stringLength++;

}

while (substring[substringLength] != NULL) {

substringLength++;

}

if (stringLength == 0) {

printf("error\n");

}

else if (substringLength == 0) {

printf("error\n");

}

else {

int i, position;

int BMT[256];

for (i = 0; i < 256; i++) {

BMT[i] = substringLength;

}

for (i = substringLength - 1; i >= 0; i--) {

if (BMT[(short)(substring[i])] == substringLength) {

BMT[(short)(substring[i])] = substringLength - i - 1;

}

}

position = substringLength - 1;

while (position < stringLength)

if (substring[substringLength - 1] != string[position]) {

position = position + BMT[(short)(string[position])];

richTextBox->Text += "new position - " + position + "\n";

}

else

for (i = substringLength - 2; i >= 0; i--) {

if (substring[i] != string[position - substringLength + i + 1]) {

richTextBox->Text += "position " + position + ": " + gcnew String(std::string(1, (substring[i])).c\_str()) + " != " + gcnew String(std::string(1, string[position - substringLength + i + 1]).c\_str()) + "\n";

position += BMT[(short)(string[position - substringLength + i + 1])] - 1;

richTextBox->Text += "new position - " + position + "\n";

break;

}

else

if (i == 0)

return position - substringLength + 1;

}

}

printf("\n");

return result;

}

};

}

Назва файлу : standardString.h

#include <string>

static std::string toStandardString(System::String^ string) {

using System::Runtime::InteropServices::Marshal;

System::IntPtr pointer = Marshal::StringToHGlobalAnsi(string);

char\* charPointer = reinterpret\_cast<char\*>(pointer.ToPointer());

std::string returnString(charPointer, string->Length);

Marshal::FreeHGlobal(pointer);

return returnString;

}

**Протокол роботи**

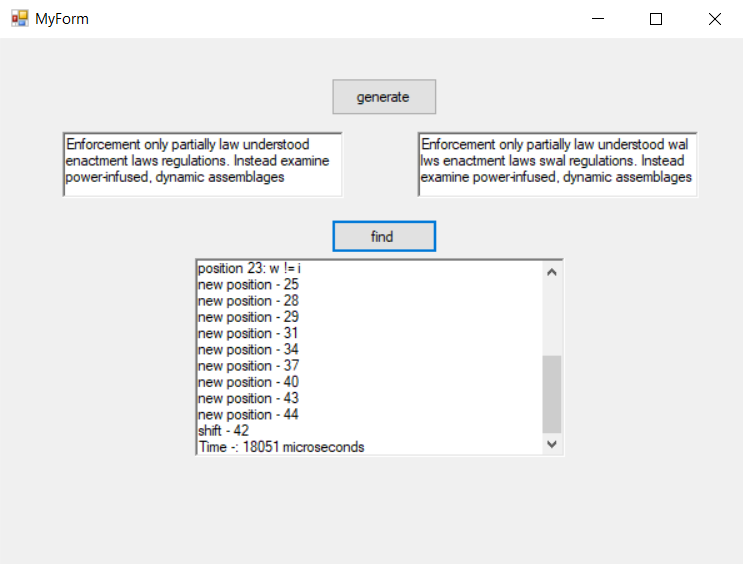
****

Рис. 1 Результат роботи програми.

**Висновок**

На цій лабораторній роботі я дізналась про алгоритм Бойєса-Мура , дізналась про його алгоритмічну складність та реалізувала програму за допомогою цього алгоритму та продемонструвала результат роботи програми на формі у Visual Studio 2022.