Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компиляторные технологии (КТ)

ОТЧЕТ

по лабораторной работе № 1

Тема работы: Распознавание и поиск лексем

Выполнили Матошко И.В.,

студенты: гр. 151003 Барановский Р.А.

Проверил: Шостак Е.В.

Минск 2022

**Регулярное выражение**

Константы:

HEX\_LETTER = A..F

X = x

ZERO = 0

DIGITS = 0..9

ZERO\*X\*

**ДКА для регулярного выражения**



**Таблица переходов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Входной символ  Состояние | ZERO | X | HEX\_LETTER | DIGITS |
| 0 | 1 | -1 | -1 | -1 |
| 1 | -1 | 2 | -1 | -1 |
| 2 | -1 | -1 | 3 | 4 |
| 3 | -1 | -1 | 3 | 4 |
| 4 | -1 | -1 | 3 | 4 |

**НКА для регулярного выражения**



**Программный код, реализующий работу ДКА**

#include <stdio.h>

#include <vector>

#include <string>

#include <iostream>

#include <cstdlib>

enum class Types

{

ZERO,

X,

DIGITS,

HEX\_LETTER,

UNKNOWN,

MAX

};

Types recognizeType(const char symb, bool& state)

{

if (symb >= 'A' && symb <= 'F')

return Types::HEX\_LETTER;

else if (symb == 'x')

return Types::X;

else if (symb == '0' && !state)

{

state = true;

return Types::ZERO;

}

else if (symb >= '0' && symb <= '9')

return Types::DIGITS;

return Types::UNKNOWN;

}

enum class States

{

ACCEPT,

REJECT,

CONTINUE,

MAX

};

States auto\_hexDigit(const std::string& str)

{

int table\_Identificator[5][4] = { {1, -1, -1, -1},

{-1, 2, -1, -1},

{-1, -1, 3, 4},

{-1, -1, 3, 4},

{-1, -1, 3, 4} };

int i = 0;

int state = 0;

bool isZero = false;

while (str[i] && state != -1)

{

int type = static\_cast<int>(recognizeType(str[i++], isZero));

if (type == 4)

return States::REJECT;

state = table\_Identificator[state][type];

}

if (3 == state || 4 == state)

return States::ACCEPT;

else if (-1 == state)

return States::REJECT;

else

return States::CONTINUE;

}

struct Lexem

{

int startIndex;

int finishIndex;

};

std::vector<Lexem> extractLexems(const std::string& str)

{

std::vector<Lexem> lexems;

std::string subStr = "";

int i = 0, startIndex = 0, finishIndex = 0;

bool isValid;

States state = States::REJECT;

while (str[i])

{

subStr += str[i];

state = auto\_hexDigit(subStr);

if (state == States::CONTINUE)

{

finishIndex = i;

isValid = false;

}

else if (state == States::REJECT)

{

if (startIndex < finishIndex && isValid)

{

lexems.push\_back({ startIndex, finishIndex });

startIndex = i;

finishIndex = i;

--i;

}

else

{

startIndex = i + 1;

finishIndex = i + 1;

}

subStr = "";

isValid = false;

}

else if (state == States::ACCEPT)

{

finishIndex = i;

isValid = true;

}

++i;

}

if (state == States::ACCEPT)

{

lexems.push\_back({ startIndex, finishIndex });

}

return lexems;

}

int main()

{

std::string str;

//Check validity of inputed string

printf("Input string to check validity: ");

getline(std::cin, str);

if (auto\_hexDigit(str) == States::ACCEPT)

printf("ACCEPT");

else if (auto\_hexDigit(str) == States::REJECT)

printf("REJECT");

else if (auto\_hexDigit(str) == States::CONTINUE)

printf("CONTINUE");

//Extract all lexems(C-hexDigits)

printf("\n\nInput string to extract all lexems(C- hexDigits): ");

getline(std::cin, str);

std::vector<Lexem> lexems = extractLexems(str);

printf("All C-hexDigits:\n");

int i = 1;

for (const Lexem& lexem : lexems)

{

printf("%d) begin: %d end: %d ", i++, lexem.startIndex, lexem.finishIndex);

std::cout << "Lexem: " << str.substr(lexem.startIndex, lexem.finishIndex - lexem.startIndex + 1) << "\n";

}

}