УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №1

по предмету «Компиляторные технологии»

Вариант 3

Выполнил:

Заривняк В. С.

Гр. 451004

Проверил:

Болтак С. В

Минск 2025

**Расчет варианта:**

(451000) **37** % 7 + 1 = 3

**Задание:**

Для заданного вариантом вида текстовых данных проанализировать допустимые значения и разработать:

* регулярное выражение;
* НКА для этого регулярного выражения;
* эквивалентный ДКА и его таблицу переходов;
* программное средство, реализующее работу этого ДКА.

Программное средство должно поддерживать следующие возможности:

* проверка корректности произвольной строки;
* поиск всех подстрок, соответствующих требованиям, в произвольной строке.

16-чное число в синтаксисе C или Pascal.

**Подготовка лексической спецификации:**

HexDigit = [1-9a-fA-F]

PascalPrefix ::= '$'

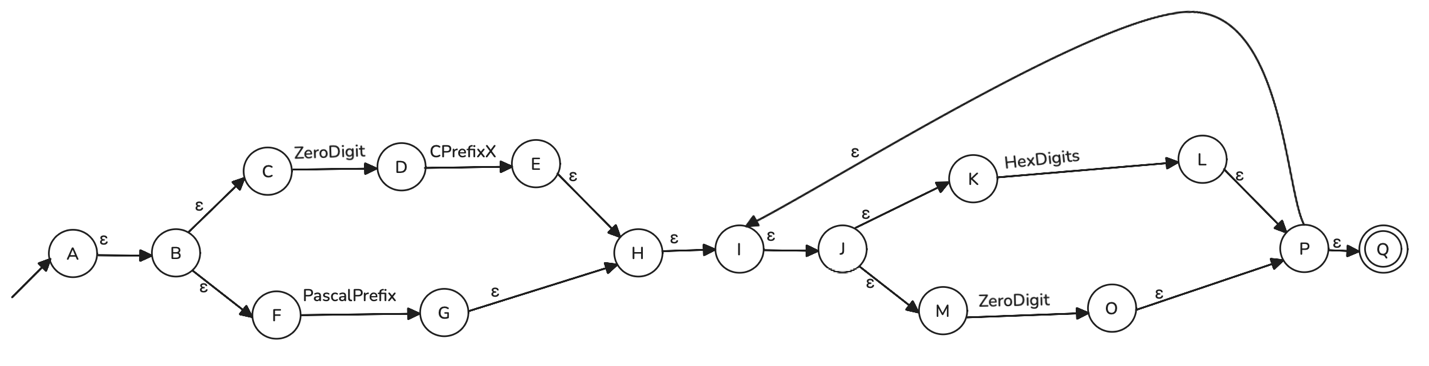
ZeroDigit ::= '0'

CPrefixX ::= 'x'|'X'

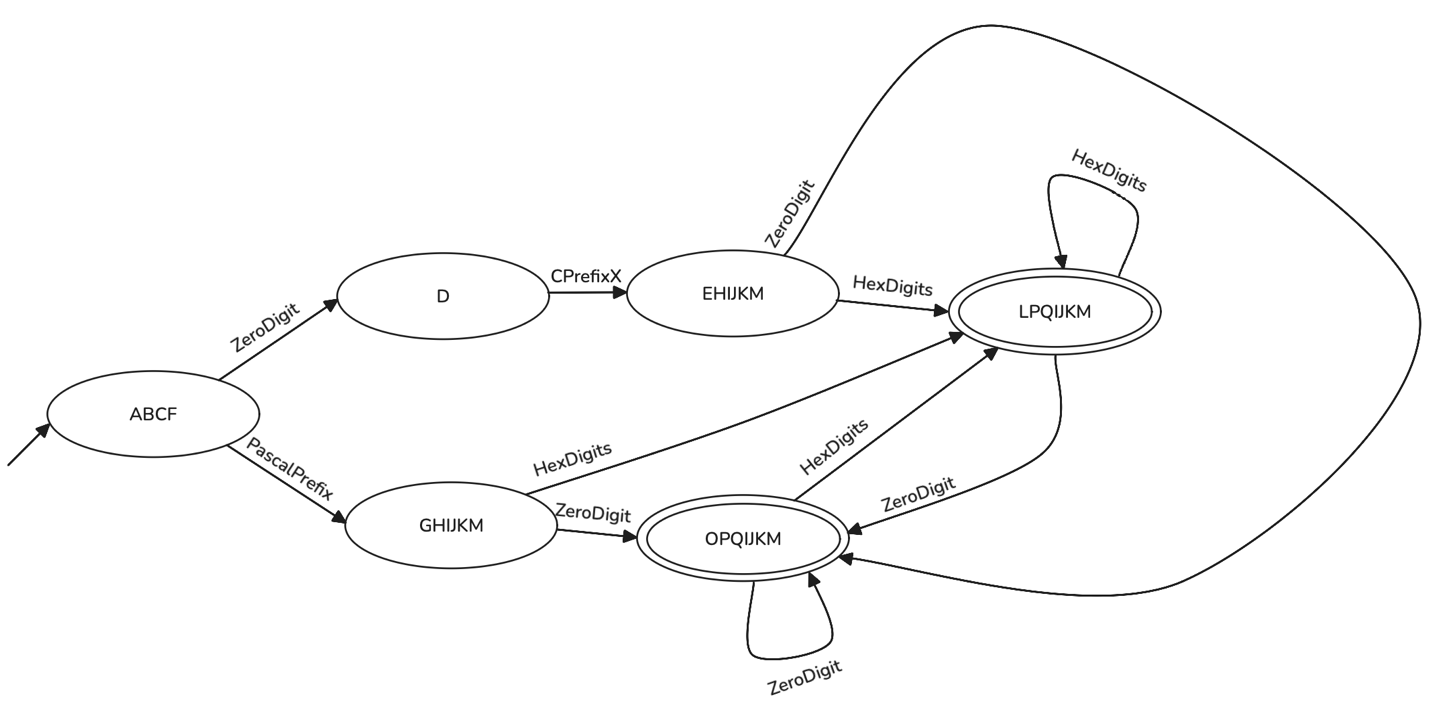
**Регулярное выражение:**

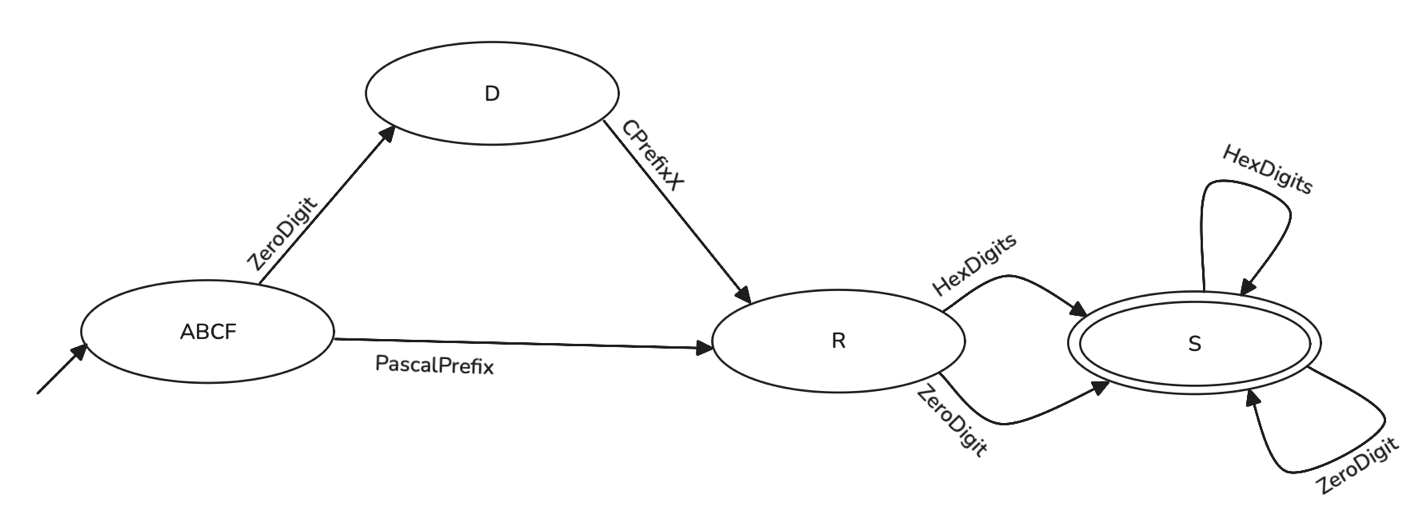
ValidHexNumber ::= (PascalPrefix | (ZeroDigit CPrefixX))(ZeroDigit | HexDigit)+

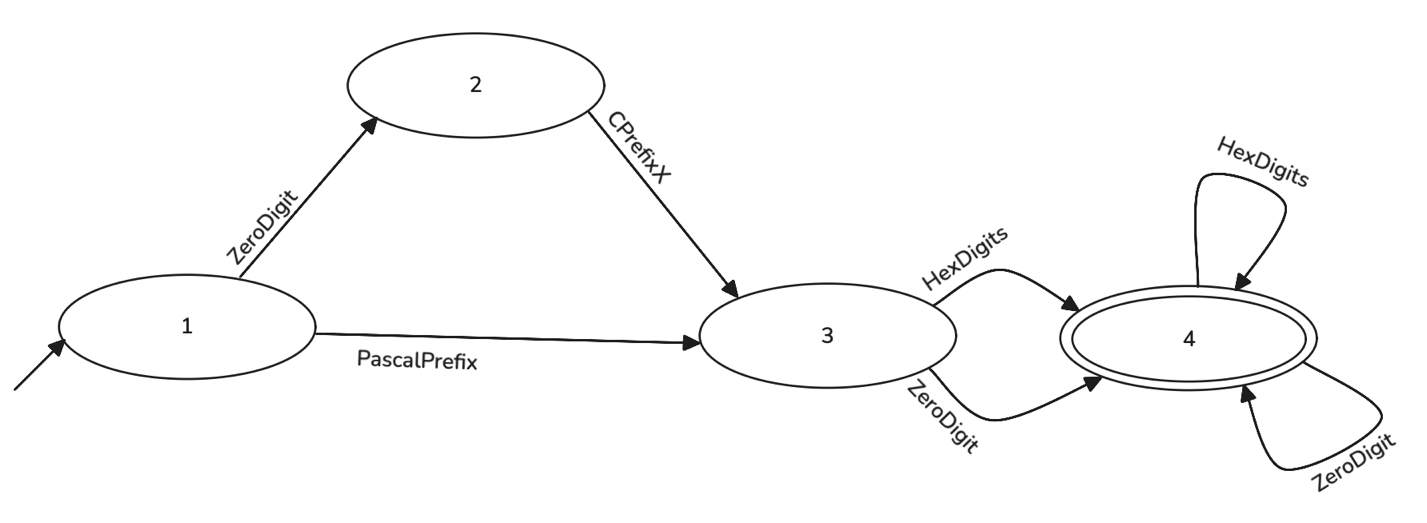
**НКА:**



**ДКА:**





****

**Таблица значений состояния полученного ДКА**

|  |  |
| --- | --- |
| **Номер состояния** | **Значение** |
| 1 | Прочитано: ничего.  Ожидается: “0” или “$”. |
| 2 | Прочитано: “0”.  Ожидается: “x” или “X”. |
| 3 | Прочитано: “x” или “X” или “$”.  Ожидается: шестнадцатеричная цифра. |
| 4 | Прочитано: шестнадцатеричная цифра.  Ожидается: шестнадцатеричная цифра или конец числа. |

**Таблица переходов**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Состояние/Классы символов | HexDigit | PascalPrefix | ZeroDigit | CPrefixX | Unknown |
| Ошибка | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 3 | 2 | 0 | 0 |
| 2 | 0 | 0 | 0 | 3 | 0 |
| 3 | 4 | 0 | 4 | 0 | 0 |
| 4 | 4 | 0 | 4 | 0 | 0 |

**Код программы C:**

#include <stdio.h>

typedef enum charType

{

HexDigit,

PascalPrefix,

ZeroDigit,

CPrefixX,

Unknown

} char\_t;

typedef enum state {

error,

expect\_zero\_or\_dollar,

expect\_x,

expect\_hex,

expect\_hex\_or\_end

} state\_t;

int transitions[5][5] = {{0, 0, 0, 0, 0},

{0, 3, 2, 0, 0},

{0, 0, 0, 3, 0},

{4, 0, 4, 0, 0},

{4, 0, 4, 0, 0}};

int finalState[5] = {0, 0, 0, 0, 1};

char\_t getCharType(const char c)

{

if (c == '0')

return ZeroDigit;

if (c == '$')

return PascalPrefix;

if (c == 'x' || c == 'X')

return CPrefixX;

if ( (c >= 'a' && c <= 'f') || (c >= 'A' && c <= 'F') || (c >= '1' && c <= '9') )

return HexDigit;

return Unknown;

}

int isValidNumber(const char \*str, const size\_t size)

{

state\_t state = 1;

for (int i = 0; i < size; ++i)

state = transitions[state][getCharType(\*(str + i))];

return finalState[state];

}

void findSubStrings(const char \*str, const int size) {

for (int k = 0; k < size; ++k) {

for (int j = k; j < size; ++j) {

state\_t state = 1;

for (int i = k; i <= j; ++i)

{

state = transitions[state][getCharType(\*(str + i))];

if (state == 0)

break;

}

if (finalState[state] == 1) {

for (int i = k; i <= j; ++i)

printf("%c", \*(str + i));

printf("\n");

}

}

}

}

int main(void) {

char str[30];

int i = 0;

printf("Enter your number: \n");

scanf("%30[^\n]", str);

while (str[i] != '\0')

i++;

if (isValidNumber(str, i))

printf("\033[1;32mCorrect\033[0m\n");

else

printf("\033[1;31mWrong!\033[0m\n");

findSubStrings(str, i);

return 0;

}

**Скриншоты:**

