УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №1

по предмету «Компиляторные технологии»

Вариант 2

Выполнил:

Бражалович А. И.

Проверил:

Болтак С.В.

Группа 351004

Минск 2024

**Задание**

Строковый литерал языка Pascal (апостроф записывается, как два апострофа подряд).

**Регулярное выражение**

Классы символов:

Quote ::= ‘ ‘ ’

Char ::= [A-Z a-m o-s u-z 0-9 !#$%&’()\*+,-./:;<=>?@[]^\_`{}\~]

Регулярное выражение:

StringLiteral ::= quotes (char)\*quotes

**НКА (для регулярного выражения)**



**ДКА (для регулярного выражения)**



**ДКА (готовый к программной реализации)**



**Таблица переходов**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Error** | **Char** | **Quotes** |
| **0** | 0 | 0 | 0 |
| **1** | 0 | 0 | 2 |
| **2** | 0 | 2 | 3 |
| **3** | 0 | 0 | 2 |

**Код программы, реализующий работу ДКА**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

enum charType {

ctUnknoun,

ctQuote,

ctSymbol,

};

enum menuType {

mtNewLine = 1,

mtCheckLine,

mtCheckSubstr,

mtExit,

};

const int transitions[4][3] = {

{0, 0, 0},

{0, 2, 0},

{0, 3, 2},

{0, 2, 0}

};

const int isFinalState[4] = {0, 0, 0, 1};

const char \*STRING = " 1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%^&\*()\_+?:;,.}{][|\\/<>~";

char\* readStr() {

int size = 0;

char\* str = NULL;

int ch;

while ((ch = getchar()) != '\n' && ch != EOF) {

str = realloc(str, size + 2);

if (!str) {

fprintf(stderr, "Memory allocation failed\n");

return 1;

}

str[size++] = ch;

}

str[size] = '\0';

return str;

}

char\* readStrFromFile() {

int size = 0;

char\* str = NULL;

int ch;

FILE \*file = NULL;

char\* fileName;

do

{

printf("Enter file name: ");

fileName = readStr();

file = fopen(fileName, "r");

if (!file) {

printf("||ERROR FILE NOT EXIST||");

}

} while (!file);

while ((ch = fgetc(file)) != '\n' && ch != EOF) {

str = realloc(str, size + 2);

if (!str) {

fprintf(stderr, "Memory allocation failed\n");

return NULL;

}

str[size++] = ch;

}

str[size] = '\0';

fclose(file);

return str;

}

enum charType getCharType(char a) {

enum charType currType = ctUnknoun;

if (strchr(STRING, a) != NULL) {

currType = ctSymbol;

}

else if (a == '\'') {

currType = ctQuote;

}

return currType;

}

void checkAllStr(char\* str) {

int len = strlen(str);

int state = 1;

printf("\n[- %s -]\n\n", str);

for (char\* i = str; i < str + len; i++) {

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

}

if (isFinalState[state]) {

printf("Correct\n");

}

else {

printf("NOT correct\n");

}

char enter;

printf("\n\nPress enter... ");

scanf("%c", &enter);

}

void findAllSubstr(char\* str) {

int len = strlen(str);

int count = 0;

char\* startIndex = str;

int state = 1;

int newState;

printf("\n[- %s -]\n\n", str);

for (char\* i = str; i < str + len; i++) {

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

if (newState == 0) {

if (state == 3) {

count++;

printf("%d) ", count);

while (startIndex < i) {

printf("%c", \*startIndex++);

}

printf("\n");

}

state = 1;

startIndex = i+1;

}

else {

state = newState;

}

}

if (isFinalState[state]) {

count++;

printf("%d) ", count);

while (startIndex < str + len) {

printf("%c", \*startIndex++);

}

printf("\n");

}

printf("TOTAL: %d\n", count);

char enter;

printf("\n\nPress enter... ");

scanf("%c", &enter);

}

enum menuType getMenuChoice(char\* str) {

enum menuType MIN\_CHOICE = mtNewLine, MAX\_CHOICE = mtExit;

printf("\n--------------------------------------------------------\n\n");

if (str) {

printf ("[- %s -]\n\n", str);

printf("1) Enter new line\n");

printf("2) Check all line\n");

printf("3) Find all substring\n");

printf("4) Exit\n");

}

else {

MAX\_CHOICE = mtNewLine;

printf("1) Enter line\n");

printf("4) Exit\n");

}

printf("\n--------------------------------------------------------\n\n");

enum menuType mtChoice;

int choice = 0, isNotCorrect = 1;

char buff = '1';

do {

scanf("%d", &choice);

isNotCorrect = ((choice < MIN\_CHOICE || choice > MAX\_CHOICE) && choice != mtExit) || (buff = getchar()) != '\n';

if (isNotCorrect) {

printf("||Error, enter again||\n\n");

while ((buff = getchar()) != '\n' && buff != EOF);

}

} while(isNotCorrect);

return choice;

}

char\* enterNewLine() {

printf("\n--------------------------------------------------------\n\n");

printf(" Choose way to enter\n");

printf("1) Console\n");

printf("2) File\n");

printf("\n--------------------------------------------------------\n\n");

int choice = 0, isNotCorrect = 1;

char buff = '1';

do {

scanf("%d", &choice);

isNotCorrect = (choice < 1 || choice > 2) || (buff = getchar()) != '\n';

if (isNotCorrect) {

printf("||Error, enter again||\n\n");

while ((buff = getchar()) != '\n' && buff != EOF);

}

} while (isNotCorrect);

switch (choice)

{

case 1:

printf("Enter a string: ");

return readStr();

case 2:

return readStrFromFile();

}

}

int check(char\* str, int end) {

int state = 1;

for (char\* i = str; i <= str + end; i++) {

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

}

if (isFinalState[state]) {

char\* endPtr = str + end;

while (str <= endPtr) {

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

}

printf("\n");

return 1;

}

return 0;

}

void checkAll(char\* str) {

int\* array = NULL;

int size = 0;

int count = 0;

for (char\* i = str; \*i != '\0'; i++) {

if (\*i == '\'') {

array = (int\*) realloc(array, (size+1) \* sizeof(int));

array[size++] = i - str;

}

}

for (int i = 0; i < size-1; i++) {

for (int j = i + 1; j < size; j++) {

count += check(str + array[i], array[j]-array[i]);

}

}

printf("TOTAL: %d\n", count);

char enter;

printf("\n\nPress enter... ");

scanf("%c", &enter);

}

int main(void) {

char\* str = NULL;

enum menuType choice = mtExit;

do

{

choice = getMenuChoice(str);

switch (choice)

{

case mtNewLine:

str = enterNewLine();

break;

case mtCheckLine:

checkAllStr(str);

break;

case mtCheckSubstr:

checkAll(str);

break;

}

} while (choice != mtExit);

return 0;

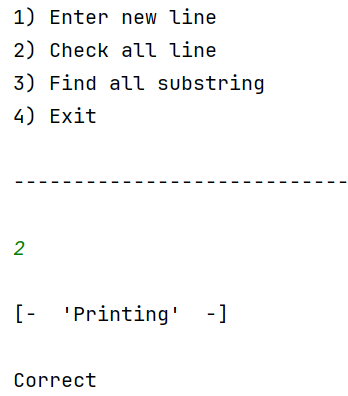
}

**Тестирование**

**Тест 1:**Исходная строка: 'Printing'

Ожидаемый результат: вся строка - литерал

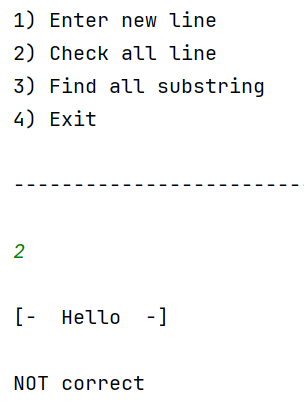
Полученный результат:



**Тест 2:**Исходная строка: Hello

Ожидаемый результат: Литералы отсутствуют

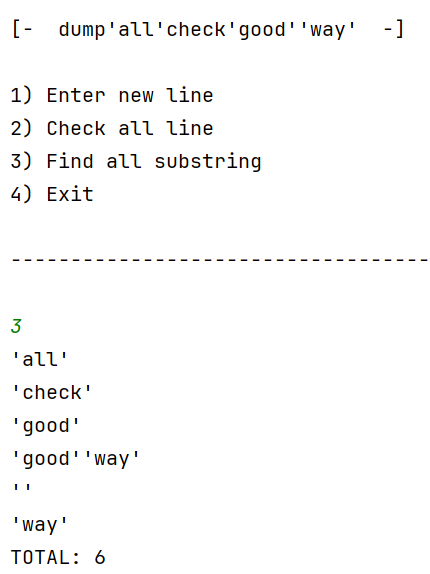
Полученный результат:



**Тест 3:**Исходная строка: dump'all'check'good''way'

Ожидаемый результат: строка содержит 6 литералов

Полученный результат:



**Тест 3:**Исходная строка: asd'45g''gh'5g

Ожидаемый результат: строка содержит 6 литералов

Полученный результат:

