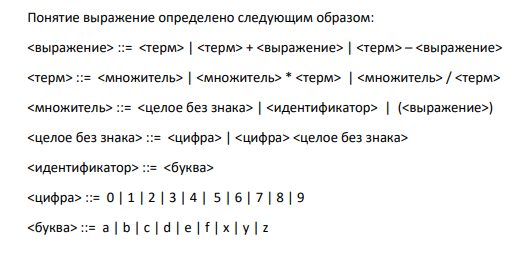
**Лабораторная работа № 9**

**Задание № 3**

1. **Общая постановка задачи**

Напишите функцию, выполняющую проверку соответствия строки, заданной параметром, правилу, указанному в Вашем варианте. Функция должна возвращать значение TRUE, если строка соответствует правилу, и FALSE, если строка НЕ соответствует правилу.

****

1. **Таблица с детальными требованиями и тест-планов**

|  |  |  |  |
| --- | --- | --- | --- |
| **Требование** | **Детальное требование** | **Данные** | **Ожидаемый результат** |
| 1. **Программа выполняется** | | | |
| 1. Вводится выражение | Если выражение соответствует шаблону, то выводится 1, если нет – выводится 0. | 5+(c-(5))  3с | 1  0 |

**3.Код программы**

#include <iostream>

#include <ctype.h>

#include <locale>

int bracketIsOpen = 0;

bool isExpression(const char\*& line);

bool isTerm(const char\*& line);

bool isMultiplier(const char\*& line);

bool isInt(const char\*& line);

bool isIdentifier(const char\*& line);

bool isNumber(const char\*& line);

bool isLetter(const char\*& line);

bool isLetter(const char\*& line)

{

if ((\*line >= 'a' && \*line <= 'f') || (\*line >= 'x' && \*line <= 'z'))

{

line++;

return true;

}

return false;

}

bool isNumber(const char\*&line)

{

if (std::isdigit(\*line))

{

line++;

return true;

}

return false;

}

bool isIdentifier(const char\*& line)

{

return isLetter(line);

}

bool isInt(const char\*& line)

{

static bool wasFirstDigit = false;

if (isNumber(line))

{

wasFirstDigit = true;

if (\*line != '\0')

{

return isInt(line);

}

}

if (wasFirstDigit)

{

wasFirstDigit = false;

return true;

}

else

{

return false;

}

}

bool isMultiplier(const char\*& line)

{

if (isInt(line))

{

return true;

}

else if (isIdentifier(line))

{

return true;

}

else if (\*line == '(')

{

++bracketIsOpen;

if (isExpression(++line) && \*line++ == ')')

{

return true;

}

}

return false;

}

bool isTerm(const char\*& line)

{

if (isMultiplier(line)) {

if ((\*line == '\*' || \*line == '/') &&

\*(line + 1) != '\0' &&

\*(line + 1) != '-' &&

\*(line + 1) != '+' &&

isTerm(++line))

{

return true;

}

return true;

}

return false;

}

bool isExpression(const char\*& line)

{

if (isTerm(line))

{

if (\*line == '\0')

{

return true;

}

else if (\*line == '+' || \*line == '-')

{

return isExpression(++line);

}

else if (\*line == ')' && bracketIsOpen > 0)

{

--bracketIsOpen;

return true;

}

}

return false;

}

int main()

{

setlocale(LC\_ALL, "ru");

const char\* expressions[] = { //тесты

"156",

"38+56",

"57-23",

"6\*7+39",

"5\*373-238",

"6/7+39",

"5/373-238",

"6\*5\*7\*7+1",

"6\*4\*2\*6-6\*3\*1\*3"

"a",

"1",

"6\*4\*a\*6-6\*3\*1\*a",

"a\*a\*c+a+a\*c",

"3c",

"3\*c",

"+",

"9,4+a",

"a\*c+(a\*v+1\*2)-(s)",

"a\*c/(a\*v+1\*2)+(s)",

"5+(c-(5))",

"5+(c-(5)",

"((((()))))",

"(5\*6)"

};

for (const char\* expression : expressions)

{

std::cout << "Выражение: " << expression << "\n";

std::cout << "Результат: " << (isExpression(expression)) << "\n\n";

}

return 0;

}