МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных Технологий

Кафедра МПО ЭВМ

Дисциплина «Теория автоматов и формальных языков»

Лабораторная работа №1-2

«Дополнительные функции лексического анализатора»

Выполнил:

студент группы 1ПИб-02-3оп-22

Маркелов Сергей Александрович

Проверил:

доцент, к.т.н.

Ганичева Оксана Георгиевна

Череповец, 2024 год

Задания

1. Написать программный код для своего варианта задания на 20-25 строк (допустимо до 30 строк)
2. Проверить его работоспособность. В качестве доказательства сделать скриншот программы и результатов. Результат должен выводиться в оформленном виде
3. Написать функцию лексического анализатора, выполняющую следующие действия:
   * удаление лишних пробелов во входном коде
   * удаление комментариев из текста программы
   * подсчет количества строк во входном тексте
4. Работу этой функции проверять на примере написанного в п. 1 рабочего кода
5. Составить блок-схему для этой функции.
6. Протестировать работу функции на других примерах.

Примечание: в отчет по работе включать все пункты и код программы с функцией. Код программы должен быть с комментариями.

|  |  |
| --- | --- |
| Вариант | Задание |
| 14 | Оператор switch и if языка С++ |

1. Программный код для тестирования функции

#include <iostream>

using namespace std;

void main(){

int a = 0, b = 0, c = 0;

cout << "Введите число от 0 до 5: ";

cin >> a;

cout << "Введите любое число: ";

cin >> b;

switch (a) { //оператор switch

case 0:

if (b != 0) c = a + b;

else c = a - b;

break;

case 1:

if (b >= 0) c = a++;

else c = b--;

break;

case 2:

if (b > 0) c = a \* b;

else c = a / b;

break;

case 3:

if (b < 0) a += b;

else b -= a;

c = a + b;

break;

case 4:

if (b <= 0) a \*= b;

else b /= a;

c = a + b;

break;

case 5:

if (b == 0) c = b % a;

else c = a / b;

break;

/\*case 6:

if (b == a) c = a;

else c = b;

break;\*/

default:

cout << "Ошибка!" << endl;

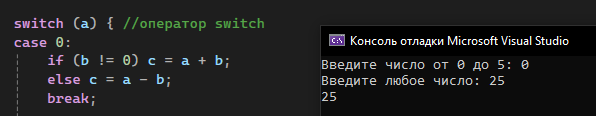
break;

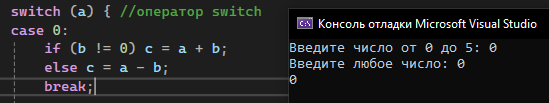
}

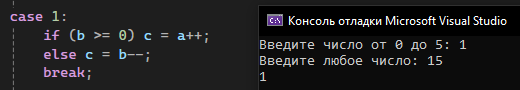
cout << c << endl;

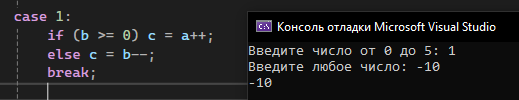
}

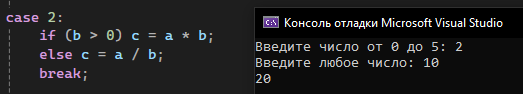
1. Результаты работы программы

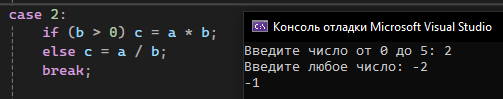


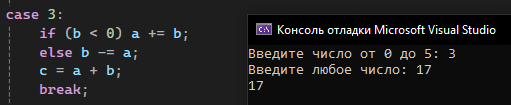


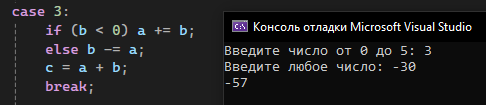


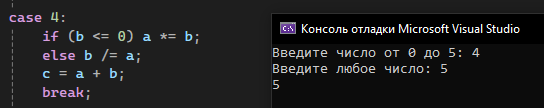


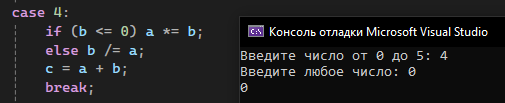


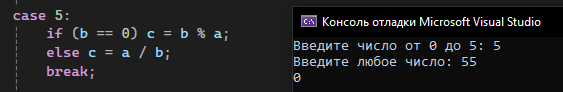


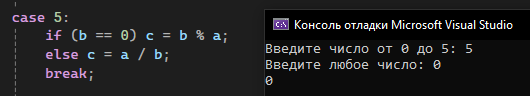


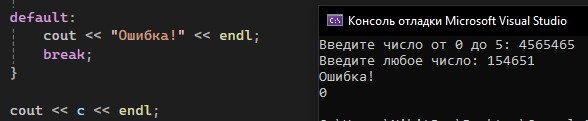












1. Функция лексического анализатора

А) Код программы

int Automat(char c, int s) {

switch (s) {

case 0: { //начало новой строки, увеличивается счетчик

if (c == '\n') return 12;

else if (c == '/') return 4;

else if (c == ' ') return 3;

else return 1;

}

break;

case 1: { //обнаружен другой символ кода

if (c == ' ') return 2;

else if (c == '\n') return 0;

else if (c == '/') return 4;

else return 1;

}

break;

case 2: { //обнаружен 1 пробел

if (c == ' ') return 3;

else if (c == '\n') return 0;

else if (c == '/') return 4;

else return 1;

}

break;

case 3: { //обнаружен 2 пробел подряд, он пропускается

if (c == ' ') return 3;

else if (c == '\n') return 0;

else if (c == '/') return 4;

else return 1;

}

break;

case 4: { //обнаружен 1 слеш

if (c == ' ') return 2;

else if (c == '\n') return 0;

else if (c == '/') return 5;

else if (c == '\*') return 7;

else return 1;

}

break;

case 5: { //обнаружен 2 слеш подряд, начался однострочный комментарий

if (c == '\n') return 0;

else return 6;

}

break;

case 6: { //пропуск символов однострочного комментария

if (c == '\n') return 0;

else return 6;

}

break;

case 7: { //обнаружена звездочка после слеша, начался многострочный комментарий

if (c == '\*') return 9;

else if (c == '\n') return 11;

else return 8;

}

break;

case 8: { //пропуск символов многострочного комментария

if (c == '\*') return 9;

else if (c == '\n') return 11;

else return 8;

}

break;

case 9: { //обнаружена звездочка в многострочном комментарии

if (c == '\*') return 9;

else if (c == '\n') return 11;

else if (c == '/') return 10;

else return 8;

}

break;

case 10: { //обнаружен слеш после звездочки в многострочном комментарии, комментарий закончен

if (c == '\n') return 0;

else if (c == '/') return 4;

else if (c == ' ') return 2;

else return 1;

}

break;

case 11: { //обнаружен перенос строки в многострочном комментарии, увеличивается счетчик

if (c == '\n') return 11;

else if (c == '\*') return 9;

else return 8;

}

break;

case 12: { //обнаружен 2 перенос строки подряд, он пропускается

if (c == '\n') return 12;

else if (c == '/') return 4;

else if (c == ' ') return 3;

else return 1;

}

}

}

#pragma endregion

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

setlocale(LC\_ALL, "Russian");

richTextBox2->Text = "";

if (richTextBox1->Text == "") richTextBox2->Text = "Код не введен!";

int n = 0, k1 = 0, k2 = 0, count = 0;

char c;

String^ str1 = "";

String^ str00 = richTextBox1->Text;

str00 += "\n";

string str = "";

string str0 = marshal\_as<string>(str00);

int s = 0;

for (int i = 0; i < str0.length(); i++) {

c = str0[i];

s = Automat(c, s);

if (s == 0 || s == 1 || s == 2 || s == 4) str += c; //запись символа кода в строку

if (s == 5 || s == 7) str.erase(str.length() - 1, 1); //стирание предыдущего символа (при // и /\*)

if (s == 0 || s == 11) {

if (str != "" && str != "\n") {

count++; //увеличение счетчика при переносе строки

str1 = count + "\t" + marshal\_as<String^>(str);

richTextBox2->Text += str1;

}

str = "";

}

}

str1 = count + "";

richTextBox3->Text = str1;

}

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

setlocale(LC\_ALL, "Russian");

OpenFileDialog^ openFile = gcnew OpenFileDialog();

openFile->Filter = "Текстовые файлы (\*.txt)|\*.txt";

String^ fileName;

if (System::Windows::Forms::DialogResult::OK == openFile->ShowDialog()) fileName = openFile->FileName;

try {

StreamReader^ file = gcnew StreamReader(fileName, Encoding::GetEncoding("windows-1251"));

richTextBox1->Text = file->ReadToEnd();

}

catch (Exception^ e)

{

richTextBox2->Text = "Ошибка открытия файла!";

}

}

Б) Граф конечного автомата



\* – звездочка, / – слеш, \_ – пробел, <- – перенос строки, С – другой символ

S0 – начало новой строки

S1 – обнаружение другого символа кода

S2 – обнаружение первого пробела

S3 – обнаружение второго пробела подряд

S4 – обнаружение первого слеша

S5 – обнаружение второго слеша подряд, начало однострочного комментария

S6 – пропуск символов однострочного комментария

S7 – обнаружение звездочки после слеша, начало многострочного комментария

S8 – пропуск символов многострочного комментария

S9 – обнаружение звездочки в многострочном комментарии

S10 – обнаружение слеша после звездочки в многострочном комментарии, конец многострочного комментария

S11 – обнаружение второго переноса строки подряд

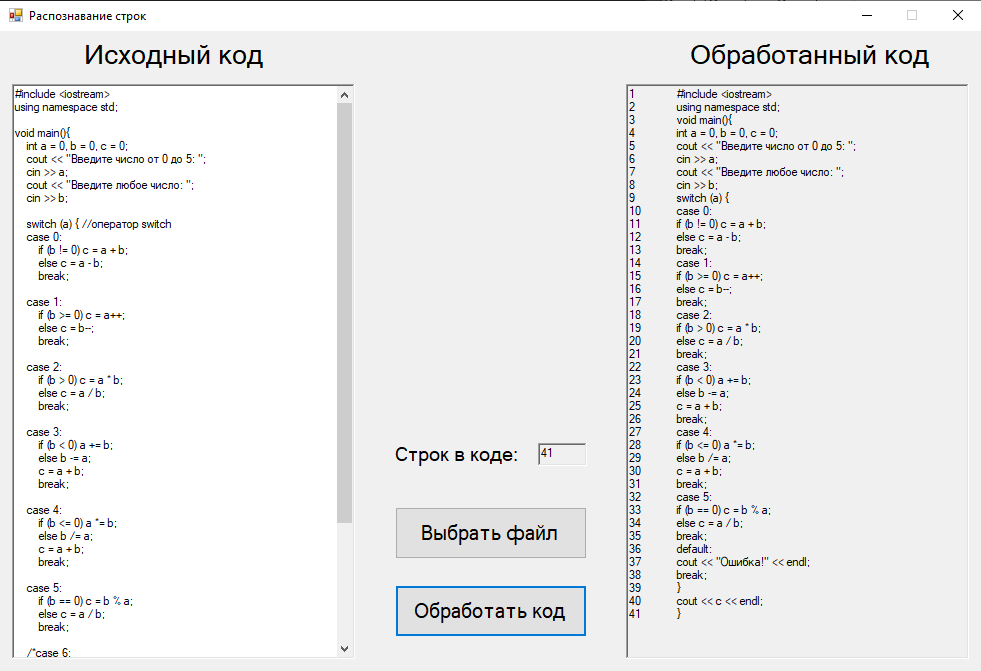
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (S0, c) → S1 | (S1, c) → S1 | (S2, c) → S1 | (S3, c) → S1 | (S4, c) → S1 | (S5, c) → S6 |
| (S0, /) → S4 | (S1, /) → S4 | (S2, /) → S4 | (S3, /) → S4 | (S4, /) → S5 | (S5, /) → S6 |
| (S0, \*) → S1 | (S1, \*) → S1 | (S2, \*) → S1 | (S3, \*) → S1 | (S4, \*) → S7 | (S5, \*) → S6 |
| (S0, \_) → S3 | (S1, \_) → S2 | (S2, \_) → S3 | (S3, \_) → S3 | (S4, \_) → S2 | (S5, \_) → S0 |
| (S0,<) → S11 | (S1,<) → S0 | (S2,<) → S0 | (S3,<) → S0 | (S4,<) → S0 | (S5,<) → S6 |

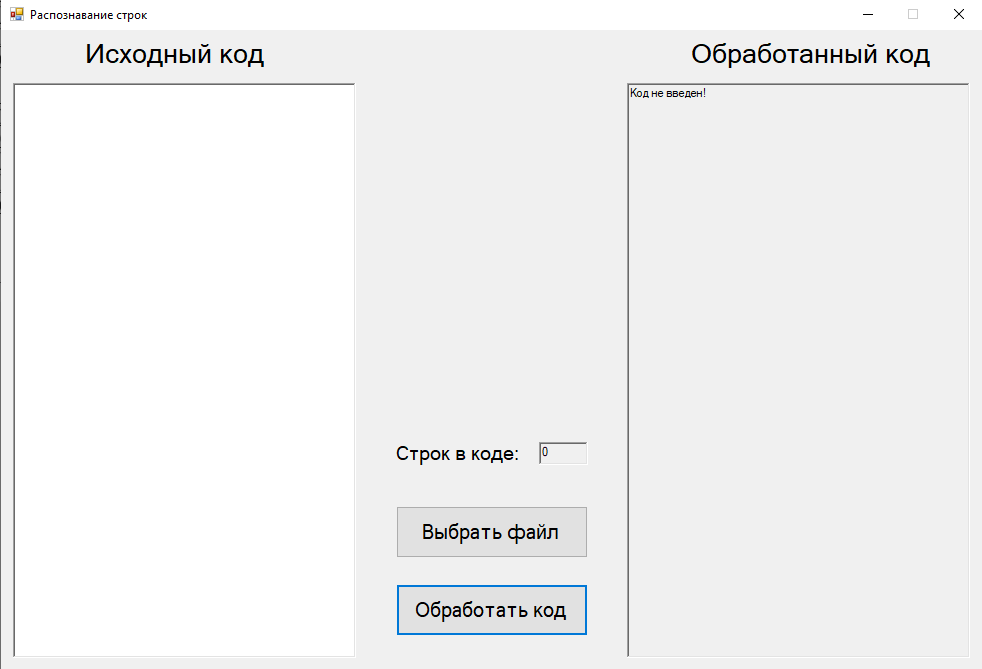
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (S6, c) → S6 | (S7, c) → S8 | (S8, c) → S8 | (S9, c) → S8 | (S10, c) → S1 | (S11, c) → S1 |
| (S6, /) → S6 | (S7, /) → S8 | (S8, /) → S8 | (S9, /) → S10 | (S10, /) → S4 | (S11, /) → S4 |
| (S6, \*) → S6 | (S7, \*) → S9 | (S8, \*) → S9 | (S9, \*) → S9 | (S10, \*) → S1 | (S11, \*) → S1 |
| (S6, \_) → S0 | (S7, \_) → S8 | (S8, \_) → S8 | (S9, \_) → S8 | (S10, \_) → S2 | (S11, \_) → S3 |
| (S6,<) → S6 | (S7,<) → S8 | (S8,<) → S8 | (S9,<) → S8 | (S10,<) → S0 | (S11,<) → S11 |

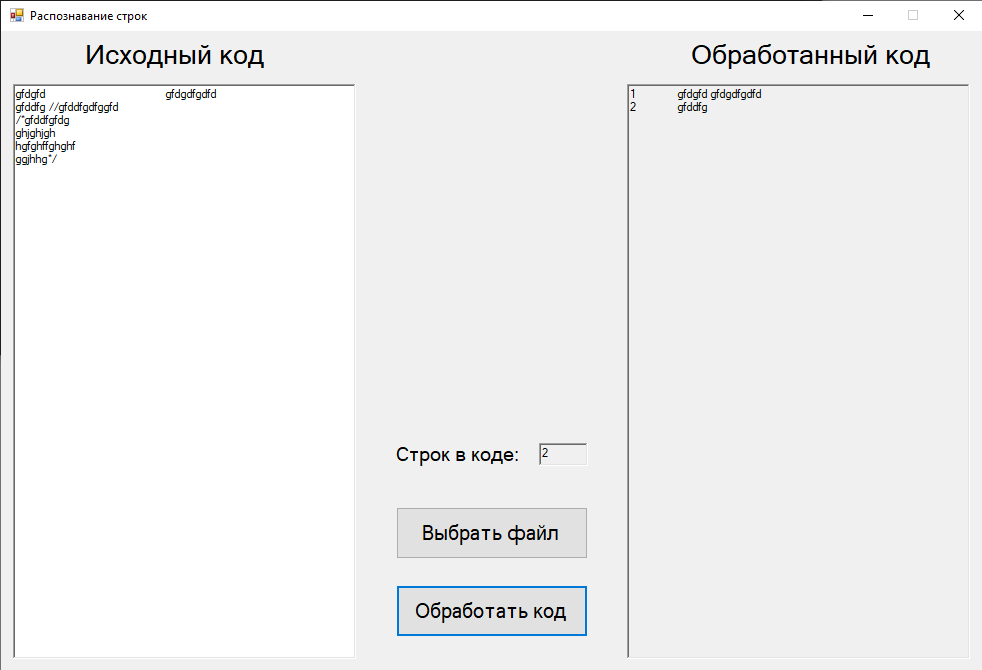
1. Блок-схема



1. Тестирование







Вывод

В ходе выполнения лабораторной работы был написан программный код на языке C++, содержащий операторы *switch* и *if*. Работоспособность кода была проверена на разных наборах данных.

Была написана функция лексического анализатора, которая удаляет лишние пробелы и комментарии во входном коде, а также подсчитывает количество строк в нем. Работа данной функции была проверена на ранее написанном коде и других примерах. Для функции была составлена блок-схема

Характеристики построенного автомата:

1. Детерменированный – следующее состояние определяется однозначно текущим состоянием и входным символом. Функция переходов имеет только одно результирующее состояние
2. Абстрактный – математическая модель дискретного устройства, которое в каждый момент времени находится в каком-либо одном состоянии из множества возможных
3. Конечный – число состояний конечно
4. Асинхронный – работает над последовательностью при ее поступлении на вход, что может произойти с различными интервалами времени
5. Распознаватель – автомат распознает строки и принимает либо отвергает их