минобрнауки россии

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| наименование института (факультета) |
| Кафедра математического и программного обеспечения ЭВМ |
| наименование кафедры |
| Теория автоматов и формальных языков |
| наименование дисциплины в соответствии с учебным планом |

ЛАБОРАТОРНАЯ РАБОТА №5\_6

Построение таблиц идентификаторов

|  |  |
| --- | --- |
| Исполнитель |  |
| студент | 1ПИб-02-2оп-22 |
|  | группа |
|  | Тихомиров В.В. |
|  | Фамилия, имя, отчество |
| Руководитель | Ганичева О.Г. |
|  | Ф.И.О. преподавателя |
| Оценка |  |
| Подпись |  |

2024 год

Задание

Написать часть/функцию лексического анализатора, обрабатывающую входную строку по конечному автомату на основе оператора выбора (это часть из предыдущей работы) и дополнить её формированием таблиц с классами лексем.

Результат работы программы:

1. Считываемая входная строка (код в соответствие со своим вариантом).
2. Таблицы с выделенными классами лексем (в которых содержится информация для формирования дескрипторного и псевдокодов).
3. Результат тестирования этой функции (проверка на повторное вхождение лексемы в заданный класс, отнесение к правильному классу лексем, выдача ошибки в случае неверной лексемы и др.).

В отчёте должна быть отражена следующая информация:

1. Код программы с комментариями
2. Скриншоты с результатами работы (сделать их читаемыми по размерам). Обратить внимание на демонстрацию результатов по п.3

Входной код

#include <iostream>

using namespace std;

void main() {

// инициализируем переменные

int d = 21;

int ds = 13;

double s = 54.32;

double ss = 2.1;

float a = 2.3e+1;

float f = 4.3e-1;

// выводим их значение на экран

for (int i = 0; i < 3; i++) {

switch (i)

{

case 0:

cout << d << endl;

cout << d + ds << endl;

break;

case 1:

cout << s << endl;

cout << s + ss << endl;

break;

case 2:

cout << a << endl;

cout << a + f << endl;

break;

default:

cout << "Error" << endl;

break;

}

}

}

Тесты программы

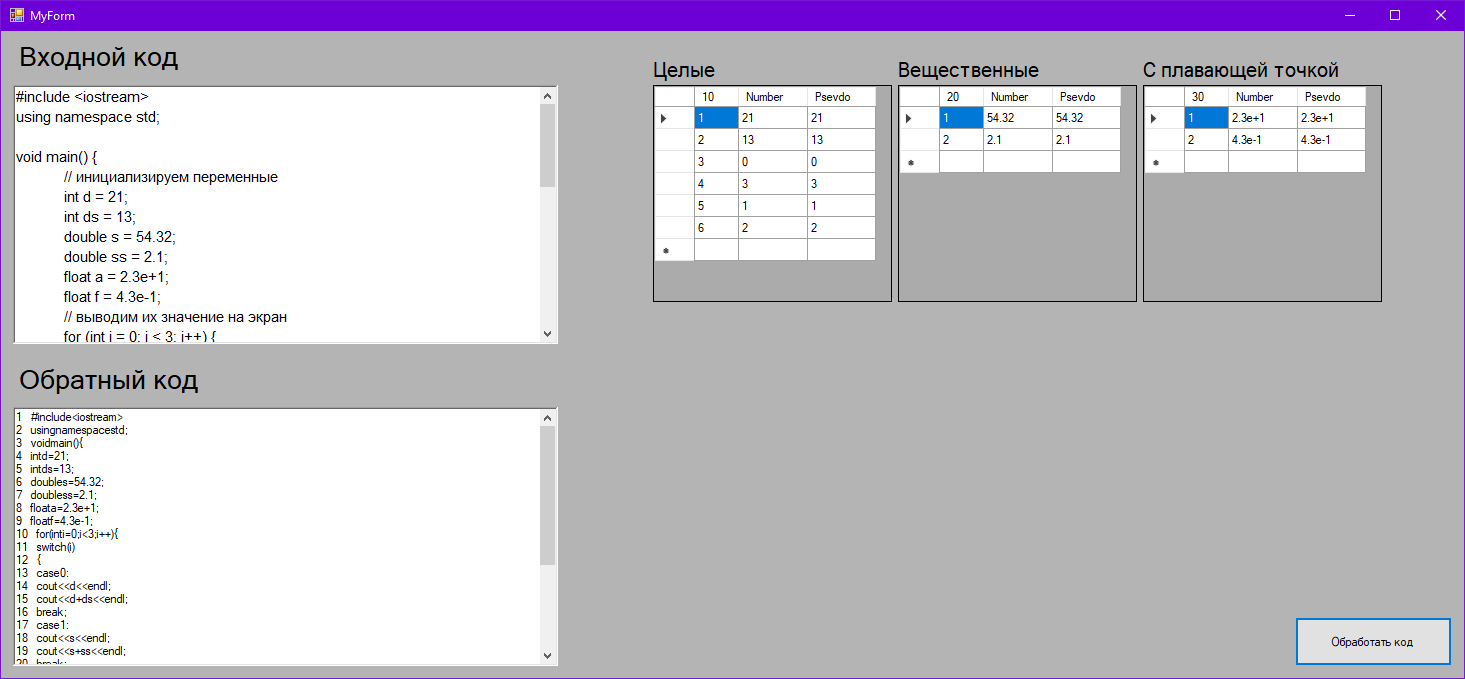


Рис. 1. Тест работы программы

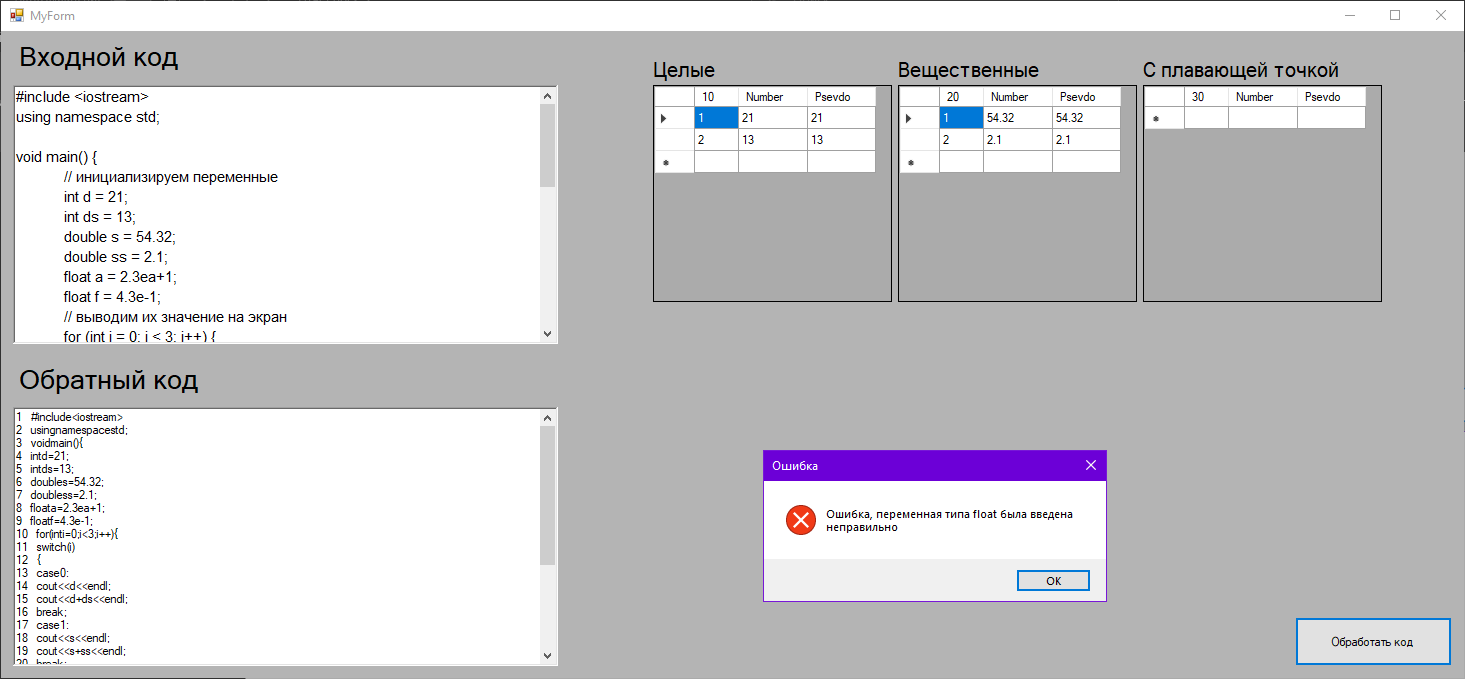


Рис. 2. Тест обработчика ошибок

Вывод

В ходе выполнения лабораторной работы была написана часть/функция лексического анализатора, обрабатывающая входную строку по конечному автомату на основе оператора выбора (эта часть из предыдущей работы), а также была дополнена **формированием таблиц с классами лексем. Были созданы таблицы с выделенными классами лексем (в которых содержится информация для формирования дескрипторного и псевдокодов). Была реализована проверка на повторное вхождение лексемы в заданный класс, а также отнесение к правильному классу лексем и обработка неправильных лексем.**

Код программы

IntVal->Rows->Clear();

if (IntVal->Columns->Count == 0) {

DataGridViewTextBoxColumn^ column1 = gcnew DataGridViewTextBoxColumn();

column1->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column1->Name = "10";

IntVal->Columns->Add(column1);

DataGridViewTextBoxColumn^ column2 = gcnew DataGridViewTextBoxColumn();

column2->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column2->Name = "Number";

IntVal->Columns->Add(column2);

DataGridViewTextBoxColumn^ column3 = gcnew DataGridViewTextBoxColumn();

column3->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column3->Name = "Psevdo";

IntVal->Columns->Add(column3);

}

doubleVal->Rows->Clear();

if (doubleVal->Columns->Count == 0) {

DataGridViewTextBoxColumn^ column1 = gcnew DataGridViewTextBoxColumn();

column1->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column1->Name = "20";

doubleVal->Columns->Add(column1);

DataGridViewTextBoxColumn^ column2 = gcnew DataGridViewTextBoxColumn();

column2->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column2->Name = "Number";

doubleVal->Columns->Add(column2);

DataGridViewTextBoxColumn^ column3 = gcnew DataGridViewTextBoxColumn();

column3->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column3->Name = "Psevdo";

doubleVal->Columns->Add(column3);

}

floatVal->Rows->Clear();

if (floatVal->Columns->Count == 0) {

DataGridViewTextBoxColumn^ column1 = gcnew DataGridViewTextBoxColumn();

column1->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column1->Name = "30";

floatVal->Columns->Add(column1);

DataGridViewTextBoxColumn^ column2 = gcnew DataGridViewTextBoxColumn();

column2->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column2->Name = "Number";

floatVal->Columns->Add(column2);

DataGridViewTextBoxColumn^ column3 = gcnew DataGridViewTextBoxColumn();

column3->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column3->Name = "Psevdo";

floatVal->Columns->Add(column3);

}

String^ str = input->Text + " ";

int s1 = 0;

int i = 0;

int in = 1, dou = 1, fl = 1;

int len = str->Length;

String^ st = "";

while (i < len) {

switch (s1)

{

case 0:

if (i >= str->Length) i--;

if (i != 0 && i < str->Length && (str[i - 1] != '0' || str[i - 1] != '1' || str[i - 1] != '2' || str[i - 1] != '3' ||

str[i - 1] != '4' || str[i - 1] != '5' || str[i - 1] != '6' || str[i - 1] != '7' || str[i - 1] != '8' || str[i - 1] != '9') && str[i] == '-' && (str[i + 1] == '0' || str[i + 1] == '1' || str[i + 1] == '2' || str[i + 1] == '3' ||

str[i + 1] == '4' || str[i + 1] == '5' || str[i + 1] == '6' || str[i + 1] == '7' || str[i + 1] == '8' || str[i + 1] == '9')) {

s1 = 1;

st = st + str[i];

i++;

}

if (i == 0 && i < str->Length && str[i] == '-' && (str[i + 1] == '0' || str[i + 1] == '1' || str[i + 1] == '2' || str[i + 1] == '3' ||

str[i + 1] == '4' || str[i + 1] == '5' || str[i + 1] == '6' || str[i + 1] == '7' || str[i + 1] == '8' || str[i + 1] == '9')) {

s1 = 1;

st = st + str[i];

i++;

}

if (str[i] == '0' || str[i] == '1' || str[i] == '2' || str[i] == '3' ||

str[i] == '4' || str[i] == '5' || str[i] == '6' || str[i] == '7' ||

str[i] == '8' || str[i] == '9') {

s1 = 1;

st = st + str[i];

i++;

}

else {

s1 = 0;

i++;

}

break;

case 1:

if ((str[i] == '0' || str[i] == '1' || str[i] == '2' || str[i] == '3' ||

str[i] == '4' || str[i] == '5' || str[i] == '6' || str[i] == '7' || str[i] == '8' || str[i] == '9')) {

s1 = 1;

st = st + str[i];

i++;

}

else if (str[i] == '.') {

s1 = 2;

i++;

}

else {

bool flag = false;

String^ searchValue = st->ToString();

for (int i = 0; i < IntVal->Rows->Count; i++)

{

if (IntVal->Rows[i]->Cells["Number"]->Value != nullptr &&

IntVal->Rows[i]->Cells["Number"]->Value->ToString()->Equals(searchValue))

{

flag = true;

break;

}

}

if (!flag)

{

IntVal->Rows->Add(in.ToString(), st, st);

in++;

}

s1 = 0;

st = "";

}

break;

case 2:

if ((str[i] == '0' || str[i] == '1' || str[i] == '2' || str[i] == '3' ||

str[i] == '4' || str[i] == '5' || str[i] == '6' || str[i] == '7' || str[i] == '8' || str[i] == '9')) {

s1 = 3;

st = st + ".";

st = st + str[i];

i++;

}

else {

bool flag = false;

String^ searchValue = st->ToString();

for (int i = 0; i < IntVal->Rows->Count; i++)

{

if (IntVal->Rows[i]->Cells["Number"]->Value != nullptr &&

IntVal->Rows[i]->Cells["Number"]->Value->ToString()->Equals(searchValue))

{

flag = true;

break;

}

}

if (!flag)

{

IntVal->Rows->Add(in.ToString(), st, st);

in++;

}

s1 = 0;

st = "";

}

break;

case 3:

if ((str[i] == '0' || str[i] == '1' || str[i] == '2' || str[i] == '3' ||

str[i] == '4' || str[i] == '5' || str[i] == '6' || str[i] == '7' || str[i] == '8' || str[i] == '9')) {

s1 = 3;

st = st + str[i];

i++;

}

else if (i < str->Length && str[i] == 'e' || str[i] == 'E') {

st = st + str[i];

s1 = 4;

i++;

}

else {

bool flag = false;

String^ searchValue = st->ToString();

for (int i = 0; i < doubleVal->Rows->Count; i++)

{

if (doubleVal->Rows[i]->Cells["Number"]->Value != nullptr &&

doubleVal->Rows[i]->Cells["Number"]->Value->ToString()->Equals(searchValue))

{

flag = true;

break;

}

}

if (!flag)

{

doubleVal->Rows->Add(dou.ToString(), st, st);

dou++;

}

s1 = 0;

st = "";

}

break;

case 4:

if ((str[i] == '0' || str[i] == '1' || str[i] == '2' || str[i] == '3' ||

str[i] == '4' || str[i] == '5' || str[i] == '6' || str[i] == '7' || str[i] == '8' || str[i] == '9')) {

st = st + str[i];

s1 = 6;

i++;

}

else if (str[i] == '+' || str[i] == '-') {

s1 = 5;

st = st + str[i];

i++;

}

else {

s1 = 7;

i++;

}

break;

case 5:

if ((str[i] == '0' || str[i] == '1' || str[i] == '2' || str[i] == '3' ||

str[i] == '4' || str[i] == '5' || str[i] == '6' || str[i] == '7' || str[i] == '8' || str[i] == '9')) {

st = st + str[i];

s1 = 6;

i++;

}

else {

s1 = 7;

i++;

}

break;

case 6:

if ((str[i] == '0' || str[i] == '1' || str[i] == '2' || str[i] == '3' ||

str[i] == '4' || str[i] == '5' || str[i] == '6' || str[i] == '7' || str[i] == '8' || str[i] == '9')) {

st = st + str[i];

i++;

s1 = 6;

}

else {

bool flag = false;

String^ searchValue = st->ToString();

for (int i = 0; i < floatVal->Rows->Count; i++)

{

if (floatVal->Rows[i]->Cells["Number"]->Value != nullptr &&

floatVal->Rows[i]->Cells["Number"]->Value->ToString()->Equals(searchValue))

{

flag = true;

break;

}

}

if (!flag)

{

floatVal->Rows->Add(fl.ToString(), st, st);

fl++;

}

s1 = 0;

st = "";

}

break;

case 7:

MessageBox::Show("Ошибка, переменная типа float была введена неправильно", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

s1 = 0;

st = "";

break;

default:

break;

}

}

}