Лабораторная работа №4

“Исследование алгоритма восходящего анализатора

языка грамматики простого предшествования”

1 ПОСТАНОВКА ЗАДАЧИ

Таблица 1 – Фрагмент программы для анализа



2 ХОД РАБОТЫ

1. Все расчеты в excel.

2. Код программы:

#include <iostream>

#include <iomanip>

#include <vector>

using namespace std;

const int MATRIX\_SIZE = 13;

const int matrix[MATRIX\_SIZE][MATRIX\_SIZE] = {

// 1 2 3 4 5 6 7 8 9 10 11 12 13

/\* 1 \*/ { 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0 },

/\* 2 \*/ { 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

/\* 3 \*/ { 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0 },

/\* 4 \*/ { 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1 },

/\* 5 \*/ { 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

/\* 6 \*/ { 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

/\* 7 \*/ { 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

/\* 8 \*/ { 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 },

/\* 9 \*/ { 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

/\* 10\*/ { 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

/\* 11\*/ { 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 },

/\* 12\*/ { 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

/\* 13\*/ { 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

};

void print\_error(int error\_code, int step, int last\_true);

string error\_code\_to\_message(int error\_code, int last\_true);

int numeric\_to\_code(int numeric);

string error\_code\_decryption(int error\_code);

string numeric\_to\_symbol(int numeric);

int main()

{

setlocale(LC\_ALL, "Russian");

vector<int> initial\_array = {

300, 508, 400, 507,

300, 508, 400, 505, 300, 507,

100, 503, 300, 504,

501,

200, 400, 509, 300, 508, 300, 505, 400, 507,

200, 400, 509, 300, 508, 300, 505, 400, 507,

502

};

cout << "> size = " << initial\_array.size() << endl;

vector<int> primary\_array = initial\_array;

for (int i = 0; i < primary\_array.size(); i++)

{

switch (primary\_array[i])

{

case 100: { primary\_array[i] = 1; break; }

case 200: { primary\_array[i] = 2; break; }

case 300: { primary\_array[i] = 3; break; }

case 400: { primary\_array[i] = 4; break; }

case 501: { primary\_array[i] = 5; break; }

case 502: { primary\_array[i] = 6; break; }

case 503: { primary\_array[i] = 7; break; }

case 504: { primary\_array[i] = 8; break; }

case 505: { primary\_array[i] = 9; break; }

case 506: { primary\_array[i] = 10; break; }

case 507: { primary\_array[i] = 11; break; }

case 508: { primary\_array[i] = 12; break; }

case 509: { primary\_array[i] = 13; break; }

case 510: { primary\_array[i] = 14; break; }

default: { break; }

}

}

for (int i = 0; i < primary\_array.size() - 1; i++)

{

int symbol = primary\_array[i] - 1;

int symbol\_next = primary\_array[i + 1] - 1;

if (matrix[symbol][symbol\_next] == 1)

{

continue;

}

else

{

print\_error(numeric\_to\_code(symbol\_next + 1), i, symbol + 1);

system("pause");

return -1;

}

}

cout << "Корректное завершение" << endl;

system("pause");

return 0;

}

void print\_error(int error\_code, int step, int last\_true)

{

string error\_message = error\_code\_to\_message(error\_code, last\_true);

string error\_desc = error\_code\_decryption(error\_code);

cout << "Обнаружена ошибка. Шаг выполнения "

<< step + 1 << endl

<< error\_desc << endl

<< error\_message << endl;

}

string error\_code\_to\_message(int error\_code, int last\_true)

{

string message;

int target\_number = last\_true;

switch (error\_code)

{

case 100: { message = "Встречено служебное слово - FUNC."; break; }

case 200: { message = "Встречено служебное слово - FINAL."; break; }

case 300: { message = "Встречен идентификатор."; break; }

case 400: { message = "Встречена константа."; break; }

case 501: { message = "Встречен символ ':'."; break; }

case 502: { message = "Встречен символ '\\'."; break; }

case 503: { message = "Встречен символ '!'."; break; }

case 504: { message = "Встречен символ '{'."; break; }

case 505: { message = "Встречен знак '}'."; break; }

case 506: { message = "Встречен знак '('."; break; }

case 507: { message = "Встречен символ ')'."; break; }

case 508: { message = "Встречен оператор присваивания '='."; break; }

case 509: { message = "Встречен символ '&&'."; break; }

case 510: { message = "Встречен символ '!!'."; break; }

default: { break; }

}

message = message + " Ожидалось: ";

bool flag = false;

for (int i = 0; i < MATRIX\_SIZE; i++)

{

if (matrix[target\_number - 1][i] == 1)

{

if (!flag)

{

message = message + numeric\_to\_symbol(i + 1);

flag = true;

}

else

{

message = message + " / " + numeric\_to\_symbol(i + 1);

}

}

}

message = message + ".";

return message;

}

int numeric\_to\_code(int numeric)

{

int code = -1;

switch (numeric)

{

case 1: { code = 100; break; }

case 2: { code = 200; break; }

case 3: { code = 300; break; }

case 4: { code = 400; break; }

case 5: { code = 501; break; }

case 6: { code = 502; break; }

case 7: { code = 503; break; }

case 8: { code = 504; break; }

case 9: { code = 505; break; }

case 10: { code = 506; break; }

case 11: { code = 507; break; }

case 12: { code = 508; break; }

case 13: { code = 509; break; }

case 14: { code = 510; break; }

default: { break; }

}

return code;

}

string error\_code\_decryption(int error\_code)

{

string sym = "null";

switch (error\_code)

{

case 100: case 200: case 509: case 503: case 504: { sym = "Недопустимая конструкция языка."; break; }

case 300: { sym = "Недопустимая комбинация идентификаторов."; break; }

case 400: { sym = "Недопустимая комбинация констант."; break; }

case 501: case 502: { sym = "Недопустимая комбинация операторных скобок."; break; }

case 505: case 506: { sym = "Недопустимая комбинация знаков."; break; }

case 507: { sym = "Недопустимое окончание выражения."; break; }

case 508: { sym = "Недопустимое присваивание."; break; }

default: { sym = "Неоределенная компановка кода."; break; }

}

return sym;

}

string numeric\_to\_symbol(int numeric)

{

string sym = "null";

switch (numeric)

{

case 1: { sym = "FUNC"; break; }

case 2: { sym = "FINAL"; break; }

case 3: { sym = "iden"; break; }

case 4: { sym = "data"; break; }

case 5: { sym = ":"; break; }

case 6: { sym = "\\"; break; }

case 7: { sym = "!"; break; }

case 8: { sym = "{"; break; }

case 9: { sym = "}"; break; }

case 10: { sym = "("; break; }

case 11: { sym = ")"; break; }

case 12: { sym = "="; break; }

case 13: { sym = "&&"; break; }

case 14: { sym = "!!"; break; }

default: { break; }

}

return sym;

}

3. Результаты выполнения

Для проверки возьмем те же последовательности что и в прошлой лабораторной работе.

|  |  |
| --- | --- |
| **Входная последовательность** | **Результат** |
| 300 501 100 506 400 507 502 504 300 508 506 400 509 400 | Верно |
| 507 502 300 508 503 506 400 510 400 507 502 505 200 | Верно |
| 300 501 100 506 507 | Неверно |
| 100 506 400 507 | Неверно |



Рисунок 1 – Первая цепочка – верно



Рисунок 2 – Вторая цепочка – верно

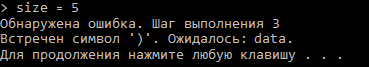


Рисунок 3 – Третья цепочка – неверно

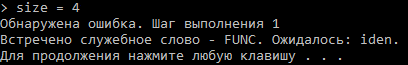


Рисунок 4 – Четвертая цепочка – неверно