**Отчёт по лабораторной работе №1. Вариант №2.**

**Задание:**

Обеспечить поддержку матриц, состоящих из целых чисел.

Матрицы вводить в нижеследующем формате:

<матрица>::=[<строки>]

<строки>::=<строка> | <строка> ; <строки>

<строка>::=<число> | <число> , <строка>

**Решение:**

1. Каждому порождающему правилу грамматики сопоставляется своя функция, вызывающая другие функции. Терминальные символы распознаются лексическим анализатором *get\_token():*

Token\_value get\_token()

{

char ch=0;

do{if(!input->get(ch))return curr\_tok=END;}

while(ch!='\n'&&isspace(ch));

switch(ch)

{

case 0 : return curr\_tok=END;

case ',':case ']' :case '[':

return curr\_tok=Token\_value(ch);

case '0': case '1': case '2': case '3':

case '4': case '5': case '6': case '7':

case '8': case '9':

case '.' :

input->putback(ch);

\*input>>value;

if(firstrow) colums++;

return curr\_tok=NUMBER;

case ';' :

rows++;

firstrow = false;

return curr\_tok=SEPARATE\_ROWS;

case '\n' :

return curr\_tok=PRINT;

default :

error("INCORECT WORD");

return curr\_tok=PRINT;

};

};

Самый последний возврат функции хранится в глобальной переменной curr\_tok, имеющей тип перечисления Token\_value:

enum Token\_value

{NUMBER,END,PRINT,SEPARATE\_ROWS=';',CLOSE=']',SEPARATE\_TOKENS=',',OPEN='['};

1. Нетерминальные символы распознаются функциями синтаксического анализа matrix(), prim() и row():

void matrix()

{

row();

for(;;)

{

switch(curr\_tok)

{

case SEPARATE\_ROWS :

if(!firstrow && curcolumn!=colums)

{error("ROWS SIZE ARE DIFFERENT"); return;}

row(); break;

case CLOSE : if(!firstrow && curcolumn!=colums) error("ROWS SIZE ARE DIFFERENT");

return;

default : error("] OR ; EXPECTED");return;

};

};

};

void row()

{

curcolumn = 0;

arr[currindex++]=prim(); curcolumn++;

for(;;)

{

switch(curr\_tok)

{

case SEPARATE\_TOKENS : arr[currindex++]=prim(); curcolumn++;break;

default : return;

};

};

};

double prim()

{

get\_token();

switch(curr\_tok)

{

case NUMBER :

{

double v = value; get\_token(); return v;

};

default : return error("FAIL EXPRESSION EXPECTED");

};

};

1. В главной функции производится анализ способа ввода данных (через аргументы командной строки или стандартный поток ввода std::cin), запускается алгоритм рекурсивного спуска, если ошибок в результате анализа не обнаружено, то принимаем строку:

int main(int argc, char\* argv[])

{

switch(argc)

{

case 1:

input=&cin;

break;

case 2:

input=new istringstream(argv[1]);

break;

default:

error("To much parameters!!!");

return 1;

};

while(input)

{

currindex = 0;

memset(arr,0,100);

get\_token();

if(curr\_tok==END)break;

if(curr\_tok==PRINT)continue;

if(curr\_tok!='[' && curr\_tok!='\n') { error("[ EXPECTED"); continue;}

matrix();

if(!no\_of\_errors) {

print(arr);

}

no\_of\_errors = rows = colums = curcolumn = 0;

firstrow =true;

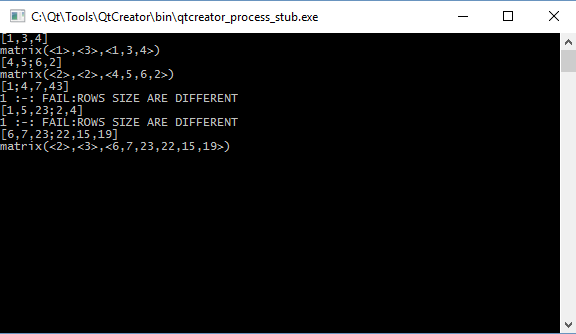
};

if(input!=&cin)delete input;

return no\_of\_errors;

};

Пример работы программы с вводом данных через std::cin:



Пример работы программы с входными параметрами – аргументами командной строки:

