**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:**

Программа состоит из следующих функциональных модулей:

- функция обработки первичных символов (принимает символы, принадлежащие входному алфавиту, иначе возникает исключение);

- функция обработки числа (лексический анализ входных символов);

- функция обработки строки (синтаксический анализ);

- функция обработки строк (синтаксический анализ);

- вспомогательные модули для диагностики ошибок и очистки потока ввода.

1. **Обработка первичных символов Symbol symb()**

enum Symbol //допустимый алфавит спец. символов

{

NUMBER,END,NEXT,SEPROW=';',FINISH=']',SEPNUM=',',START='['

};

Symbol symb()

{

char ch=0;

do //для пропуска пробельных символов и символов табуляции

{

if(!cin.get(ch)) return cursymb=END;

}

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

switch(ch)

{

case 0 :

return cursymb=END; //конец ввода

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

return cursymb=Symbol(ch);

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

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

case '8': case '9':

cin.putback(ch);

cin>>number;

return cursymb=NUMBER; //вернули число

case ';' :

return cursymb=SEPROW; //вернули разделитель строк

case '\n' :

return cursymb=NEXT; //вернули конец ввода текущей последов.

default :

throw Error("Wrong symbol!"); //символ не принадлежит алфавиту

};

}

1. **Обработка строки void stroka(vector<int>& result)**

//передается ссылка на результирующий массив

void stroka(vector<int>& result)

{

result.push\_back(chislo());

while(1)

{

if(cursymb == SEPNUM)

{

result.push\_back(chislo());

} else return;

};

}

//функция обработки числа

double chislo()

{

symb();

if(cursymb == NUMBER)

{

columns++;

int i = number;

symb();

return i;

} else {

//если после числа нет разделителя

throw Error("Wrong query of symbols");

};

}

1. **Обработка строк void stroki(vector<int>& result) и матрицы в целом void matrix(vector<int>& result)**

void matrix(vector<int>& result)

{

//последний и начальный символ ввода должны быть скобки

if(cursymb!='[' && cursymb!=NEXT) { throw Error("Where is [ ?");}

stroki(result);

//если в строках различное число элементов, то не принимаем матрицу

if(columns!=matrixsize && !firstrow) throw Error("different rows size");

if(cursymb==FINISH) return;

if(cursymb==NEXT) { throw Error("Where is ] ?");}

}

void stroki(vector<int>& result)

{

rows++;

stroka(result);

while(1)

{

if(cursymb == SEPROW)

{

if(firstrow) {

matrixsize = result.size();

firstrow = false;

}

//если в строках различное число элементов, то не принимаем матрицу

else if(columns!=matrixsize) throw Error("different rows size");

columns=0;

rows++;

stroka(result);

}

else return;

};

}

1. **Вспомогательные модули**

//вывод матрицы на экран

void print(vector<int>& result)

{

vector<int>::iterator it = result.begin();

cout << "matrix(<" << rows << ">,<" << columns << ">,<";

while(it!=result.end())

cout << \*(it++) << ",";

cout << '\b';

cout << ">)" << endl;

}

//функция восстанавливает поток при ошибке ввода (лексической и/или //синтаксической)

void skip()

{

while(cin)

{

char ch;

cin.get(ch);

if(ch==';' || ch=='\n') return;

}

}

//инициализация

//функция подготавливает глобальные переменные перед началом анализа

void init()

{

no\_of\_errors = rows = columns = matrixsize = 0;

firstrow =true;

}

//класс для обработки ошибок

struct Error{

string data;

Error(const string& s):data(s){no\_of\_errors++;}

};

1. **Вызов основной функции main()**

int main()

{

while(cin)

{

std::vector<int> Matrix;

Matrix.reserve(100);

init();

symb();

if(cursymb==END)break;

if(cursymb==NEXT)continue;

try{

matrix(Matrix);

}

catch(Error err){

cerr << "FAIL: " << err.data;

skip();

continue;

}

print(Matrix);

};

return 0;

}

Пример работы программы:

