**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() //парсер входного потока

{

char ch=0;

int i = -1;

//если символ найден, match() возвращает true и мы запоминаем найденный символ

if (match(",")) return cursymb = SEPNUM;

if (match("]")) return cursymb = FINISH;

if (match("[")) return cursymb = START;

// если символ - число, то считываем его в массив

if (number(Number)) {

while (Number[++i])

{

curnumber = curnumber \* 10 + Number[i];

}

return cursymb = NUMBER;

}

if (match(";")) return cursymb=SEPROW;

if (match("\n") || !ch) return cursymb=NEXT;

//если ни один из символов не удовлетворяет условиям выше - некорректный символ на входе

throw Error("Wrong symbol!");

}

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

//передается ссылка на результирующий массив

void stroka(vector<int>& result)

{

//в конец вектора-массива чисел добавляется первое число

result.push\_back(chislo());

while(1)

{

//как только встречаем разделитель символов, спускаемся вниз, за следующим //символом

if(cursymb == SEPNUM)

{

result.push\_back(chislo());

} else return;

};

}

//функция обработки числа

double chislo()

{

symb();

//если считанный символ число – возвращаем его наверх и считываем след. //символ

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)**

//функция старта рекурсивного спуска

void matrix(vector<int>& result)

{

//если первый символ не '[' – не принимаем строку

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

//спускаемся вниз для анализа строк

stroki(result);

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

if(cursymb==FINISH) return;

//если последний символ не ']' – не принимаем строку

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

}

//функция анализа строк

void stroki(vector<int>& result)

{

rows++;

//спуск вниз для анализа отдельной строки

stroka(result);

while(1)

{

//если находим разделитель строк, то продолжаем анализировать

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{

//старт рекурсивного спуска

matrix(Matrix);

}

catch(Error err){

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

skip();

continue;

}

print(Matrix);

};

return 0;

}

Пример работы программы:

