```
/*Строка содержит римские цифры до 50(L)*/

/*Отчет:
Эта программа проверяет лексическую правильность строки с римским числом. Проход строчки осуществляется с конца.
Дальнейшие комментарии ниже по коду.
*/

#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <set>

using namespace std;
enum States { START, I, V, X, L, ERROR, FINISH, STOP, BLANK }; //список состояний

void addEdge(vector<int>[], int, int);

int main()
{
    setlocale(LC_ALL, "Russian");
    set<pair<int, int>> setPair; //матрица перехода
    ifstream f("file.txt");
    if (!f.is_open()) { cout << "Файл не открыт!" << endl; return 1; }
    string s;
    while (!f.eof()) {
        getline(f, s);
        States st = START;
        int currI = 0;
        int currX = 0;
        int i = s.length() - 1;
        while (st != STOP)
        {
            switch (st)
            {
            case START: {
                if (s.length() == 0) { st = BLANK;  setPair.insert({ START, BLANK }); }
                else
                    if (s[i] == 'I') {
                        st = I; setPair.insert({ START, I }); i--; currI++; // последняя цифра - I
                    }
                    else if (s[i] == 'L') {
                        st = L; setPair.insert({ START, L }); i--; // последняя цифра - L
                    }
                    else if (s[i] == 'X') {
                        st = X; setPair.insert({ START, X }); i--; currX++; // последняя цифра - X
                    }
                    else if (s[i] == 'V') {
                        st = V; setPair.insert({ START, V }); i--; // последняя цифра - V
                    }
                    else {
                    st = ERROR; setPair.insert({ START, ERROR }); // Если другой символ в конце - ошибка
                }
                break;
            }
            case I:
            {

                if (i == -1) {
                    st = FINISH; setPair.insert({ I, FINISH }); // Успешно дошли до конца строки - финиш.
```

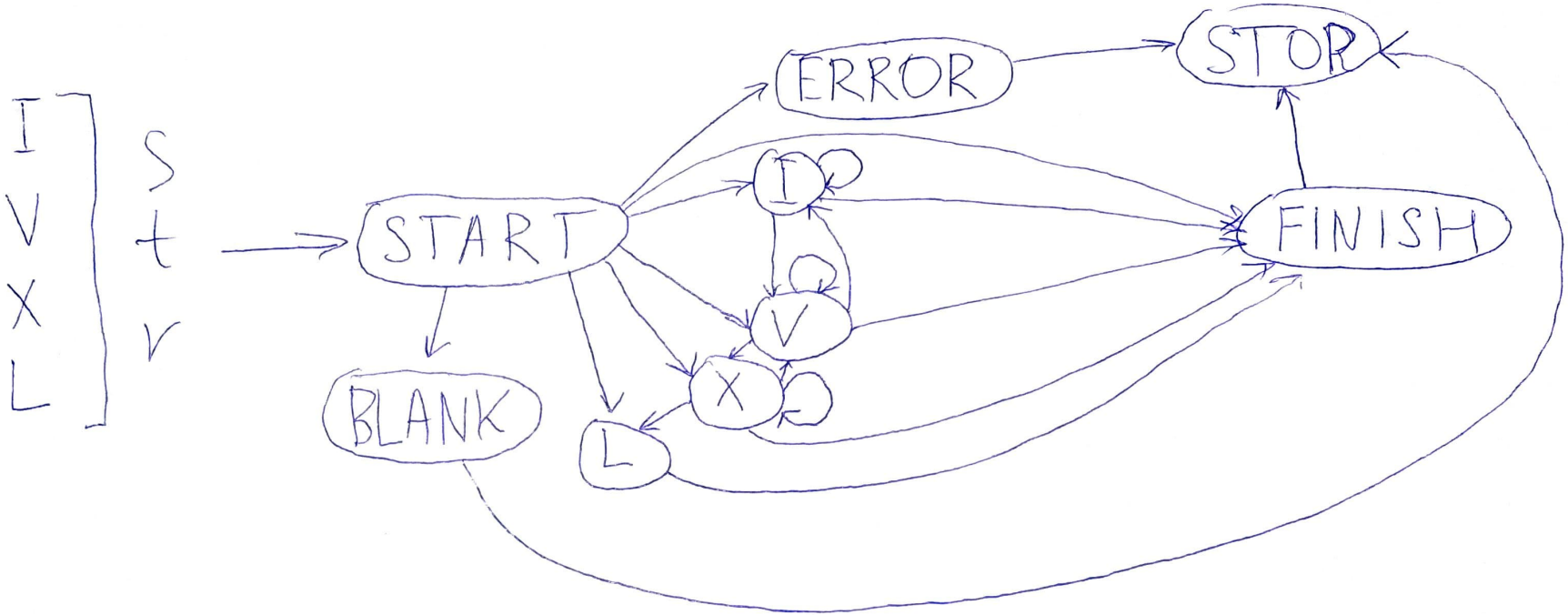
```
                }
                else
                    if (s[i] != 'X' && s[i] != 'V' && s[i] != 'L' && s[i] != 'I') {
                        st = ERROR; setPair.insert({ I, ERROR }); // проверка на другие
символы
                    }
                    else if (s.length() - i >= 2 && s[i] == 'I' && (s[i + 2] == 'V' ||
s[i + 2] == 'X')) { st = ERROR; setPair.insert({ I, ERROR }); }
                // проверка на то, чтобы перед V и X было не более одной I
                    else if (s[i] == 'V') { st = V; setPair.insert({ I, V }); i--; } //
предыдущая цифра - V
                    else if (s[i] == 'X') { st = X; setPair.insert({ I, X }); i--; } //
предыдущая цифра - X
                    else if (s[i] == 'L') { st = L; setPair.insert({ I, L }); i--; } //
предыдущая цифра - L
                    else if (s[i] == 'I' && currI > 2) { st = ERROR; setPair.insert({ I,
ERROR }); } // проверка на невозможность постановки более трех I подряд
                    else if (s[i] == 'I') { i--; currI++; } // предыдущая цифра - I
                    else { st = ERROR; setPair.insert({ I, ERROR }); } // ошибка в ином
случае
                break;
            }
            case V:
            {
                if (i == -1) {
                    st = FINISH; setPair.insert({ V, FINISH }); // Успешно дошли до конца
строки - финиш.
                }
                else
                    if (s[i] != 'X' && s[i] != 'L' && s[i] != 'I') {
                        st = ERROR; setPair.insert({ I, ERROR }); // проверка на другие
символы
                    }
                    else if (s[i] == 'I' && s[i + 2] == 'I') { st = ERROR;
setPair.insert({ V, ERROR }); } // предотвращение неправильной постановки I одновременно
до и после V
                    else if (s[i] == 'I') { st = I; setPair.insert({ V, I }); i--; } //
предыдущая цифра - I
                    else if (s[i] == 'X') { st = X; setPair.insert({ V, X }); i--; } //
предыдущая цифра - X
                    else if (s[i] == 'L') { st = L; setPair.insert({ V, L }); i--; } //
предыдущая цифра - L
                    else { st = ERROR; setPair.insert({ X, ERROR }); } // ошибка в ином
случае (даже при V, так как две V не могут идти подряд)
                break;
            }
            case X:
            {
                if (i == -1) {
                    st = FINISH; setPair.insert({ X, FINISH }); // Успешно дошли до конца
строки - финиш.
                }
                else
                    if (s[i] != 'X' && s[i] != 'V' && s[i] != 'L' && s[i] != 'I') {
                        st = ERROR; setPair.insert({ I, ERROR });
                    }
                    else if (s[i] == 'X' && currX > 2) { st = ERROR; setPair.insert({ X,
ERROR }); } // проверка на невозможность постановки более трех X подряд
                    else if (s[i] == 'X') { i--; currX++; } // предыдущая цифра - X
                    else if (s[i] == 'I' && s[i + 2] == 'I') { st = ERROR;
setPair.insert({ X, ERROR }); } // предотвращение неправильной постановки I одновременно
до и после X
                    else if (s[i] == 'I') { st = I; setPair.insert({ X, I }); i--; } //
предыдущая цифра - I
```

```
                    else if (s[i] == 'V') { st = V; setPair.insert({ X, V }); i--; } //
предыдущая цифра - V
                    else if (s[i] == 'L') { st = L; setPair.insert({ X, L }); i--; } //
предыдущая цифра - L
                    else { st = ERROR; setPair.insert({ X, ERROR }); } // ошибка в ином
случае
                break;
            }
            case L:
            {
                if (i == -1) {
                    st = FINISH; setPair.insert({ L, FINISH }); // Успешно дошли до конца
строки - финиш.
                }
                else
                    if (i != -1 && i != 0) { st = ERROR; setPair.insert({ L, ERROR }); }
//проверка на правильность постановки L в числе
                    else if (i == 0 && s[i] == 'X') { st = FINISH; setPair.insert({ L,
FINISH }); } // проверка на правильность отбавления X от L
                    else { st = ERROR; setPair.insert({ L, ERROR }); } // ошибка в ином
случае
                break;
            }

            case ERROR: {
                cout << s << " - число лексически НЕверное!" << "\n"; // вывод сообщения
об ошибке
                st = STOP; // переход в СТОП
                setPair.insert({ ERROR, STOP });
                break;
            }
            case FINISH: {
                cout << s << " - число лексически верное!" << "\n"; // вывод сообщения о
правильности
                st = STOP; setPair.insert({ FINISH, STOP }); // переход в СТОП
                break;
            }
            case BLANK: {
                cout << s << " - пустая строка!" << endl; // вывод состояния о пустоте
                st = STOP;
                setPair.insert({ BLANK, STOP });// переход в стоп
                break;
            }
            }
        }
        vector<int> adj[9];
        for (auto x : setPair)
        {
            addEdge(adj, x.first, x.second);
        }
        //printGraph(adj, 9);
    }


    f.close();
    system("pause");
    return 0;
}

void addEdge(vector<int>adj[], int u, int v)
{
    adj[u].push_back(v);
    adj[v].push_back(u);
}
```


I
V
X
L
Str
START
ERROR
STOR
I
V
X
L
BLANK
FINISH