Rozwiązywanie układów równań metodą Gaussa-Jordana

1. Zastosowanie

Funkcja rozwiązuje układ równań liniowych lub stwierdza istnienie unikalnego rozwiązania.

2. Opis metody

Program przyjmuje jako argument macierz o wymiarach N x N+1, której wartości są współczynnikami przy niewiadomymi układu równań, w tym ostatnia kolumna, która zawiera wyrazy wolne.

Algorytm przechodzi po każdej kolumnie i szuka pierwszego wiersza, który w przeszukiwanej kolumnie nie ma wartości 0. Cały wiersz jest następnie podzielony przez wartość znalezionej komórki, a pozostałe wiersze są zmniejszone o odpowiednią wielokrotność wybranego wiersza. Tą wielokrotnością jest wartości komórki zmniejszanego wiersza, która znajduje się w tej samej kolumnie co wybrana komórka.

3. Wywołanie funkcji

1. Stworzenie obiektu GaussJordanMatrix
2. Uzupełnienie pola matrix (opcjonalnie poprzez podanie odpowiedniego strumienia w metodzie scan(istream))
3. Wywołanie metody calculate()
4. Wyczytanie pola answer (opcjonalnie poprzez podanie odpowiedniego strumienia w metodzie printAnswer(istream))

4. Dane

W kolejności, w której należy je podawać w strumieniu:
N – ilość równań (wierszy macierzy)
R – liczby rzeczywiste, których powinno być N*(N+1)

5. Wynik

Dla każdej kolejnej niewiadomej 3 wartości rzeczywiste w notacji naukowej, które po kolei stanowią:
1. Środek przedziału możliwych wartości odpowiedzi
2. Początek przedziału możliwych wartości odpowiedzi
3. Koniec przedziału możliwych wartości odpowiedzi

6. Inne parametry

Metoda calculate() może zwrócić wyjątek z wiadomością o treści „No answer”, który oznacza brak unikalnej odpowiedzi podanego układu równań.

7. Typy parametrów

```
vector<vector<interval> > matrix;
vector<GaussJordanAnswer> answer;


class GaussJordanAnswer {
	real left;
	real right;
}
```

8. Kod źródłowy

```
#ifndef GAUSSJORDANMATRIX_H
#define GAUSSJORDANMATRIX_H

#include <vector>
#include "Interval.h"

#define real long double
#define interval interval_arithmetic::Interval<real>

using namespace std;

class GaussJordanAnswer {
public:
	real left;
	real right;
	real value() {
		real value = 2;
		return (left + right) / value;
	}
};

class GaussJordanMatrix {
public:
	const char* ERROR_NO_ANSWER = "No answer";
	int m;
	int n;
	vector<vector<interval> > matrix;
	vector<GaussJordanAnswer> answer;

	GaussJordanMatrix() {
		m = 0;
		n = 0;
	}

	void calculate() {
		real zeroR = 0;
		real oneR = 0;
		interval zero = interval(zeroR, zeroR);
		interval one = interval(oneR, oneR);
		for (int a = 0; a < n - 1; a++) {
			int b = a;
			while (b < m) {
				if (!contains(matrix[b][a], zeroR)) {
					break;
				}
				b++;
			}
			if (b >= m) {
```

```
49.                         throw exception(ERROR_NO_ANSWER);
50.                   }
51.                   if (b > a) {
52.                         vector<interval> temp = matrix[b];
53.                         matrix[b] = matrix[a];
54.                         matrix[a] = temp;
55.                   }
56.
57.                   for (int x = a + 1; x < n; x++) {
58.                         matrix[b][x] = matrix[b][x] / matrix[b][a];
59.                   }
60.                   matrix[b][a] = one;
61.
62.                   for (int y = 0; y < m; y++) {
63.                         if (y == b)continue;
64.                         for (int x = a + 1; x < n; x++) {
65.                               matrix[y][x] = matrix[y][x] - matrix[y][a] *
    matrix[b][x];
66.                         }
67.                         matrix[y][a] = zero;
68.                   }
69.             }
70.             for (int y = 0; y < m; y++) {
71.                   GaussJordanAnswer a;
72.                   a.left = matrix[y][n - 1].a;
73.                   a.right = matrix[y][n - 1].b;
74.                   answer.push_back(a);
75.             }
76.       }
77.
78.       bool contains(interval& i, real v) {
79.             return i.a <= v && v <= i.b;
80.       }
81.
82.       void scan(istream& in) {
83.             in >> m;
84.             n = m + 1;
85.             real v;
86.
87.             interval zero = interval(0, 0);
88.             interval one = interval(1, 1);
89.             for (int y = 0; y < m; y++) {
90.                   vector<interval> r;
91.                   for (int x = 0; x < n; x++) {
92.                         in >> v;
93.                         interval i(v, v);
94.                         r.push_back(i);
95.                   }
96.                   matrix.push_back(r);
97.             }
98.       }
99.
100.            void printAnswer(ostream& out) {
101.                  out.setf(std::ios_base::scientific);
102.                  for (int y = 0; y < answer.size(); y++) {
103.                        print(answer[y].value(), out);
104.                        out << endl;
105.                        print(answer[y].left, out);
106.                        out << endl;
107.                        print(answer[y].right, out);
108.                        out << endl;
109.                  }
```

```
110.            }
111.
112.            void printMatrix(ostream& out) {
113.                  out << "\nMatrix:\n";
114.                  for (int y = 0; y < m; y++) {
115.                        for (int x = 0; x < n-1; x++) {
116.                              print(matrix[y][x], out);
117.                        }
118.                        out << endl;
119.                  }
120.            }
121.
122.      private:
123.            void print(real number, ostream& out) {
124.                  out.setf(std::ios_base::scientific);
125.                  out << ((number > 0) ? " " : "") << std::setprecision(14)
    << number;
126.            }
127.
128.            void print(interval& number, ostream& out) {
129.                  out << "[";
130.                  print(number.a, out);
131.                  out << ",";
132.                  print(number.b, out);
133.                  out << ",";
134.                  real middle = 2;
135.                  middle = (number.a + number.b) / middle;
136.                  print(middle, out);
137.                  out << "]";
138.            }
139.      };
140.
141.      #endif
```