

Лабораторная работа
по вычислительным методам алгебры на тему:

Решение систем линейных алгебраических уравнений с помощью
метода квадратного корня

Выполнил:
Архангельский И.А.

Проверил:
Кондратюк А.П.

Входные и выходные данные.

Входные данные

Входной файл в первой строке содержит число n - размерность матрицы коэффициентов, следующие n строк содержат матрицу $(A|f)$, где A - квадратная матрица коэффициентов СЛУ, f - вектор-столбец свободных членов.

Выходные данные

На выход в `stdout` подается решение системы если оно существует, в случае, если систему решить невозможно выводится `DivByZero`.

Блок-схема

Реализация

square.h

```
1  #include <cmath>
2  #include <stdio.h>
3
4  struct DivByZeroException {};
5
6  class Square
7  {
8      double** matrix_;
9      int sz;
10     double* f;
11     double sgn (double x);
12 public:
13     Square(char *filename);
14     double* solve ();
15     int getSize () {return sz;}
16     ~Square ();
17 };
```

square.cpp

```
1  #include "square.h"
2
3  Square::Square(char* filename)
4  {
5      FILE* fin = fopen(filename, "r");
6      fscanf (fin, "%d", &sz);
7      matrix_ = new double* [sz];
8      f = new double [sz];
9      for (int i=0; i<sz; i++)
10     {
11         matrix_[i] = new double [sz];
12         for (int j=0; j<sz; j++)
13         {
14             fscanf(fin, "%lf", &matrix_[i][j]);
15         }
16         fscanf (fin, "%lf", &f[i]);
17     }
18     fclose (fin);
19 }
20
21 Square::~~Square()
22 {
23     delete [] f;
24     for (int i=0; i<sz; i++)
25         delete [] matrix_[i];
26     delete [] matrix_;
27 }
28
29 double Square::sgn(double x)
30 {
31     if (x==0) return 0;
32     return x>0?1:-1;
33 }
34
35 double* Square::solve()
36 {
37     double** S = new double* [sz];
38     double* D = new double[sz];
39     for (int i=0; i<sz; i++)
40     {
41         S[i] = new double [sz];
42     }
43     for (int i=0; i<sz; i++)
44     {
45
46         for (int j=i; j<sz; j++)
47         {
48             if (i==j)
49             {
50                 double summ = 0;
51                 for (int k = 0; k<i; k++)
52                 {
53                     summ+=S[k][i]*S[k][i]*D[k];
54                 }
55             }
56         }
57     }
58 }
```

```

55         D[i] = sgn (matrix_[i][j]-summ);
56         S[i][j]=pow(fabs(summ-matrix_[i][j]), 0.5);
57     }
58     if (i<j)
59     {
60         double summ = 0;
61         for (int k=0;k<i;k++)
62         {
63             summ+=S[k][i]*D[k]*S[k][j];
64         }
65         if (S[i][i]*D[i]==0) throw DivByZeroException ();
66         S[i][j] = (matrix_[i][j] - summ)/(S[i][i]*D[i]);
67         S[j][i] = S[i][j];
68     }
69     if (i>j)
70     {
71         S[i][j] = 0;
72     }
73 }
74 }
75 }
76 double* y = new double [sz];
77 for (int i =0; i<sz; i++)
78 {
79     double summ = 0;
80     for (int k=0;k<i;k++)
81     {
82         summ+=S[i][k]*y[k];
83     }
84     y[i] = (f[i] - summ)/S[i][i];
85 }
86 double* x = new double [sz];
87 for (int i=sz-1;i>=0;i--)
88 {
89     double summ = 0;
90     for (int k = i+1; k<sz;k++)
91     {
92         summ += S[i][k]*D[i]*x[k];
93     }
94     x[i] = (y[i]-summ)/S[i][i]*D[i];
95 }
96 delete [] y;
97 for (int i=0;i<sz;i++)
98 {
99     delete [] S[i];
100 }
101 delete [] D;
102 delete [] S;
103 return x;
104 }

```

main.cpp

```

1  #include <iostream>
2  #include <square.h>
3
4  int main(int argc, char *argv[])
5  {
6      for (int i=1;i<argc;i++)
7      {
8          Square sq = Square(argv[i]);
9          double* res;
10         try
11         {
12             res = sq.solve();
13             for (int i=0;i<sq.getSize(); i++)
14             {
15                 printf ("%3.3lf ", res[i]);
16             }
17             printf ("\n");
18             delete [] res;
19         }
20         catch (DivByZeroException e)
21         {
22             printf ("DivByZero\n");
23         }
24     }
25     return 0;
26 }

```

Тестовые данные

<div>test01.in</div> <div>3 3.2 1 1 4 1 3.7 1 4.5 1 1 4.2 4</div>	<div>test01.out</div> <div>0.811 0.846 0.558</div>
<div>test02.in</div> <div>4 1 1 1 1 0 1 2 3 4 0 1 3 6 10 0 1 4 10 20 0</div>	<div>test02.out</div> <div>0.000 0.000 0.000 0.000</div>
<div>test03.in</div> <div>4 -1 1 1 1 5 1 2 1 1 2 1 1 -3 1 3 1 1 1 4 4</div>	<div>test03.out</div> <div>-2.474 1.947 -0.737 1.316</div>
<div>test04.in</div> <div>4 1 1 1 1 4 1 1 1 1 3 1 1 1 2 4 1 1 2 1 5</div>	<div>test04.out</div> <div>DivByZero</div>
<div>test05.in</div> <div>3 1 1 1 3 1 2 3 6 1 3 1 5</div>	<div>test05.out</div> <div>1.000 1.000 1.000</div>