БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа
по вычислительным методам алгебры на тему:

Метод Гаусса с выбором главного элемента

Выполнил:
Архангельский И.А.

Проверил:
Кондратюк А.П.

Минск, 2012

Часть I

Входные и выходные данные.

Входные данные

На вход программа принимает текстовый файл в котором первой строкой стоит целое неотрицательное число n, показывающее размерность матрицы A. Следующие n строк содержат расширенную матрицу $[A|B]$. Где B - столбец свободных членов.

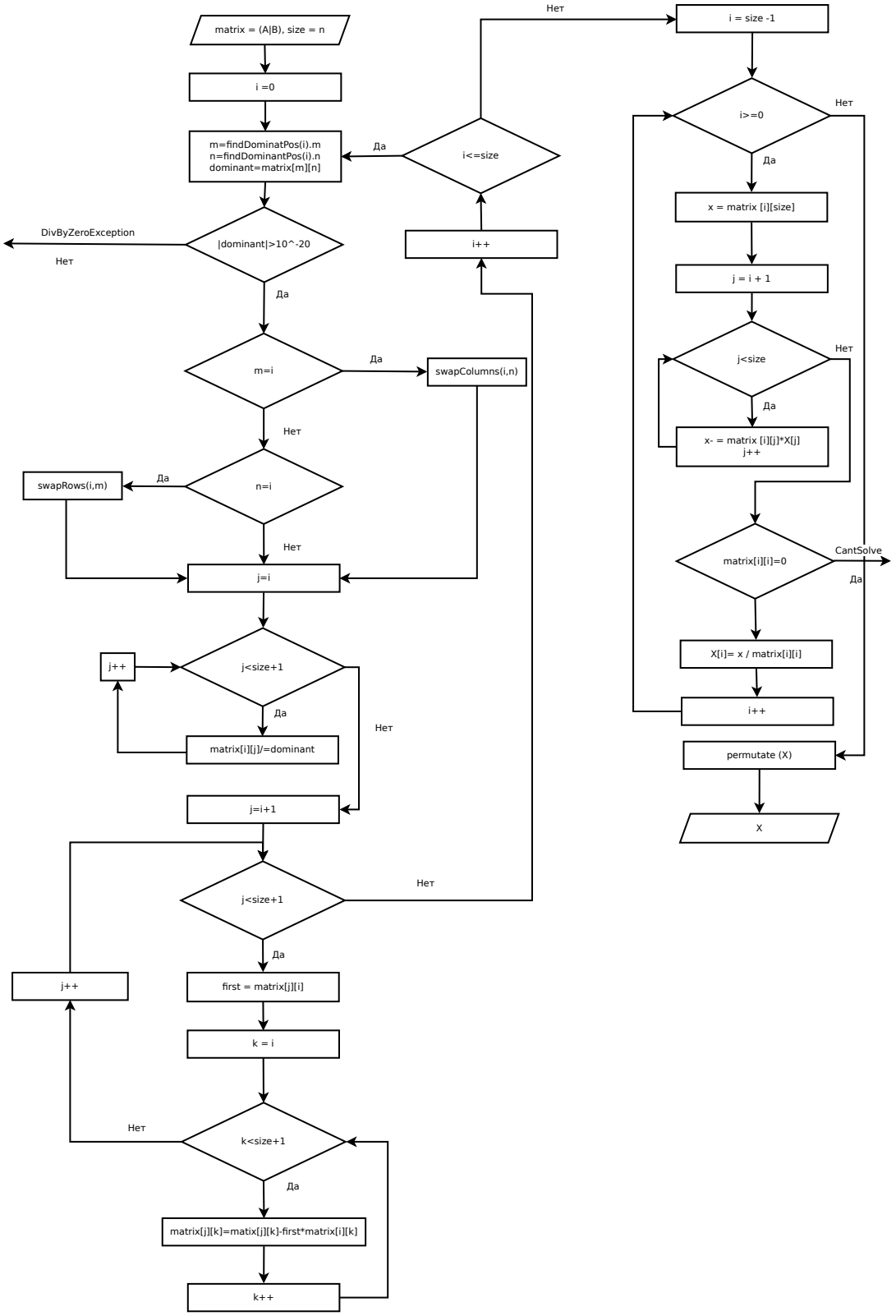
Выходные данные

На выход, в случае удачного решения СЛУ, в stdout выводится x^T, где x - вектор-столбец, являющийся решением системы. В случае, если систему решить невозможно, в stdout подается соответсвующая ошибка.

Часть II

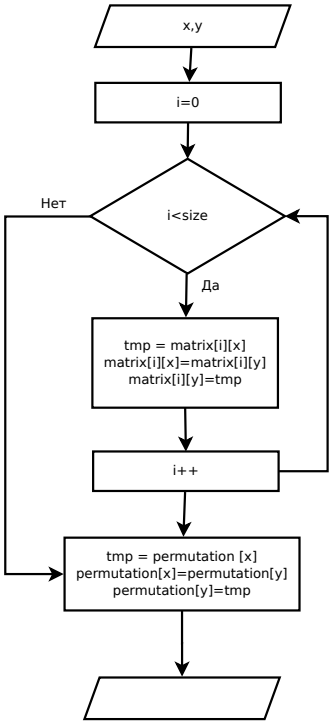
Блок-схема

Метод solve()

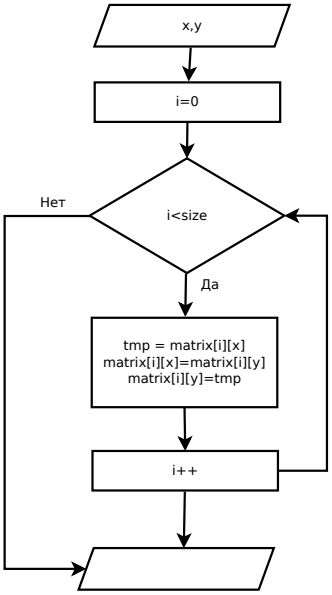


Вспомогательные методы

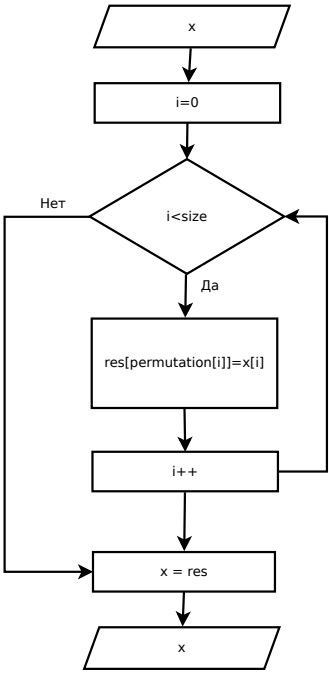
swapColumns (int x, int y)



swapRows (int x, int y)



permutate (double* &x)



Часть III

Реализация

gauss.h

```
1  #ifndef GAUSS_H
2  #define GAUSS_H
3
4  #include <cstdlib>
5  #include <stdio.h>
6  #include <math.h>
7  #include <iostream>
8
9  using namespace std;
10
11 struct DivByZeroException {};
12 struct CannotSolve {};
13 class Gauss
14 {
15     double ** matrix;
16     unsigned int size;
17     int *permutation;
18
19     struct Position
20     {
21         unsigned int n;
22         unsigned int m;
23     };
24
25     void permutate (double* &x);
26     Position findDominant (int k);
27     void swapRows (int x, int y);
28     void swapColumns (int x, int y);
29 public:
30     Gauss(char* filename);
31     double* solve ();
32     int getSize() {return size;}
33     ~Gauss ();
34 };
35
36 #endif // GAUSS_H
```

gauss.cpp

```
#include "gauss.h"

Gauss::Gauss (char *filename)
{
    FILE* fin = fopen (filename,"r");
    fscanf (fin,"%d\n",&size);
    matrix = new double* [size];
    for (int i=0;i<size;i++)
    {
        matrix[i]= new double [size+1];
        for (int j=0;j<size;j++)
        {
            fscanf(fin,"%lf",&matrix[i][j]);
        }
        fscanf (fin,"%lf\n",&matrix[i][size]);
    }
    permutation = new int [size];
    for (int i=0;i<size;i++)
    {
        permutation [i]= i;
    }
    fclose (fin);
}


Gauss::Position Gauss::findDominant(int k)
{
    Position pos;
    pos.m = k;
    pos.n = k;
    for (int i=k;k<size;k++)
    {
        if (fabs(matrix[i][k])>fabs(matrix[pos.m][pos.n]))
        {
            pos.m = i;
            pos.n = k;
        }
        if (fabs(matrix[k][i])>fabs(matrix[pos.m][pos.n]))
        {
            pos.m = k;
            pos.n = i;
        }
    }
    return pos;
}

void Gauss::swapRows(int x, int y)
{
    for (int i=1;i<size+1;i++)
    {
        double tmp = matrix[x][i];
        matrix [x][i] = matrix[y][i];
        matrix [y][i] = tmp;
    }
}

void Gauss::swapColumns(int x, int y)
{
    for (int i=0;i<size;i++)
    {
        double tmp = matrix[i][x];
```

```
62          matrix [i][x] = matrix[i][y];
63          matrix [i][y] = tmp;
64      }
65      int tmp = permutation[x];
66      permutation [x] = permutation[y];
67      permutation [y] = tmp;
68  }
69
70  double* Gauss::solve()
71  {
72      for (int i=0;i<size;i++)
73      {
74          Position dominantPos = findDominant(i);
75          double dominant = matrix[dominantPos.m][dominantPos.n];
76          if (fabs(dominant)<pow(10,-20)) throw DivByZeroException ();
77          if (dominantPos.m==i)
78          {
79              swapColumns(i,dominantPos.n);
80          }
81          if (dominantPos.n==i)
82          {
83              swapRows(i,dominantPos.m);
84          }
85
86          for (int j=i;j<size+1;j++)
87          {
88              matrix[i][j]/=dominant;
89          }
90
91          for (int j=i+1;j<size;j++)
92          {
93              double first = matrix[j][i];
94              for (int k=i;k<size+1;k++)
95              {
96                  matrix[j][k]=matrix[j][k] - first*matrix[i][k];
97              }
98          }
99      }
100
101     double* X = new double [size];
102     for (int i=size-1;i>-1;i--)
103     {
104         double x = matrix[i][size];
105         for (int j=i+1;j<size;j++)
106         {
107             x -= matrix[i][j]*X[j];
108         }
109         if (matrix[i][i]==0)
110         {
111             throw CannotSolve();
112         }
113         X[i]=x/matrix[i][i];
114     }
115     permutate (X);
116
117     return X;
118 }
119
120 void Gauss::permutate(double *&x)
121 {
122     double* res = new double [size];
123     for (int i=0;i<size;i++)
124     {
```

```
125             res[permutation[i]]=x[i];
126         }
127         for (int i=0;i<size;i++)
128         {
129             x[i]=res[i];
130         }
131         delete res;
132     }
133
134     Gauss::~Gauss()
135     {
136         for (int i=0;i<size;i++)
137         {
138             delete matrix[i];
139         }
140         delete matrix;
141         delete permutation;
142     }
```

main.cpp

```
#include <cstdlib>
#include <iostream>
#include "gauss.h"

using namespace std;

int main(int argc, char *argv[])
{
    for (int i=1;i<argc;i++)
    {
        printf ("RUNNING ON TEST: %s\n",argv[i]);
        Gauss gauss =  Gauss(argv[i]);
        double * res = NULL;
        try
        {
            res = gauss.solve();
        }
        catch (DivByZeroException e)
        {
            printf ("ERR::DIVIZION BY ZERO\n");
            continue;
        }
        catch (CannotSolve e)
        {
            printf ("ERR::Something got wrong.Can't solve this.\n");
            continue;
        }
        for (int i=0;i<gauss.getSize();i++)
        {
            printf ("%8.3lf  ",res[i]);
        }
        printf ("\n");
        delete res;
    }
    return 0;

}
```

Часть IV

Тестовые данные

1 Тест

01.in

```
4
3  8  3  -1  4
2  3  4  1  -4
1  -3  -2  -2  3
5  -8  4  2  -8
```

stdout

```
RUNNING ON TEST: 01.in
    2.000       1.000       -3.000       1.000
```

2 Тест

03.in

```
3
0  0  0  0
0  0  0  0
0  0  0  0
```

stdout

```
RUNNING ON TEST: 03.in
ERR::DIVIZION BY ZERO
```

3 Тест

04.in

```
3
1  1  1  1
1  1  0  2
1  2  3  0
```

stdout

```
RUNNING ON TEST: 04.in
    1.000       1.000       -1.000
```

4 Тест

05.in

```
2
30  30  60
30  30  0
```

stdout

```
RUNNING ON TEST: 05.in
ERR::DIVIZION BY ZERO
```