

[Lazydemon341](#) / [ABC\\_HW](#)

&lt;&gt; Code

! Issues

🔗 Pull requests

▶ Actions

📁 Projects

📖 Wiki

🛡 Security

🔑 master ▾

...

[ABC\\_HW](#) / [task4](#) / README.md

Lazydemon341 Update README.md

🕒 History

👤 1 contributor

Raw

Blame



84 lines (51 sloc) 4.5 KB

# Власюк Александр, БПИ191

## Разработка многопоточных приложений с использованием OpenMP.

### Вариант 8. Условие:

Используя формулы Крамера, найти решение системы из 4х линейных уравнений при 4х неизвестных. Предусмотреть возможность деления на ноль. Входные данные: коэффициенты системы. Оптимальное количество потоков выбрать самостоятельно.

### Принцип реализации

После ввода коэффициентов системы пользователем в консоль, сразу вычислим главный детерминант матрицы системы. Если он равен нулю, то программа завершает работу, т.к. метод Крамера нельзя применить к такой системе. Иначе - задаем 4 потока в `omp`, в каждом из которых, с помощью директивы `parallel sections`, считаем вспомогательные определители (по методу Крамера). После окончания работы всех потоков, получаем решения системы, деля соответствующие вспомогательные детерминанты на главный.

Таким образом, многопоточность программы строится по модели "Управляющий и рабочие".

## Секция импорта

```
#include <omp.h>
#include <stdio.h>
#include <math.h>

using namespace std;
```

## Метод main

В первую очередь - принимаем на ввод от пользователя коэффициенты системы, проверяем ввод на корректность и выводим полученную систему на экран.

```
103 int main()
104 {
105     bool isCorrectInput = true;;
106
107     int** slae = new int* [5];
108     for (size_t i = 0; i < 5; i++)
109     {
110         // Initialize the SLAE' columns
111         slae[i] = new int[4];
112     }
113
114     for (int i = 0; i < 4; i++)
115     {
116         printf("Enter the %d line of the system:\n", (i + 1));
117         if (scanf_s("%d %d %d %d", &slae[0][i], &slae[1][i], &slae[2][i], &slae[3][i], &slae[4][i]) != 5) {
118             isCorrectInput = false;
119             break;
120         }
121     }
122
123     if (!isCorrectInput) {
124         printf("\nIncorrect input!");
125     }
126     else {
127
128         printf("\nEntered system:\n");
129         printSLAE(slae);
130     }
```

Далее - считаем основной определитель системы с помощью метода getCramerDet, и если он равен 0, завершаем программу.

```
// Count the main delta
int d = getCramerDet(slae[0], slae[1], slae[2], slae[3]);

if (d == 0) {
    printf("\nDelta0 equals zero hence the Cramer's rule can't be applied!");
}
else {
```

Иначе - находим решения системы с помощью метода getSolutions.

```
}
else {
    getSolutions(slae, d);
}
```

## Метод printSLAE

Выводит матрицу коэффициентов системы

```
void printSLAE(int** slae) {
    for (size_t i = 0; i < 4; i++)
    {
        for (size_t j = 0; j < 5; j++)
        {
            printf("%d\t", slae[j][i]);
        }
        printf("\n");
    }
}
```

## Метод getCramerDet

Составляет матрицу 4 на 4 из переданных столбцов системы и возвращает её определитель с помощью метода determinant

```
int getCramerDet(int* a1, int* a2, int* a3, int* a4) {  
    int** matrix = new int* [4];  
    matrix[0] = a1;  
    matrix[1] = a2;  
    matrix[2] = a3;  
    matrix[3] = a4;  
  
    return determinant(matrix, 4);  
}
```

## Метод determinant

Вычисляет определитель квадратной матрицы с помощью рекурсии.

```
int determinant(int** matrix, int n) {  
    int det = 0;  
    int** submatrix = new int*[n];  
    for (size_t i = 0; i < n; i++)  
    {  
        submatrix[i] = new int [n];  
    }  
  
    if (n == 2)  
        return ((matrix[0][0] * matrix[1][1]) - (matrix[1][0] * matrix[0][1]));  
    else {  
        for (int x = 0; x < n; x++) {  
            int subi = 0;  
            for (int i = 1; i < n; i++) {  
                int subj = 0;  
                for (int j = 0; j < n; j++) {  
                    if (j == x)  
                        continue;  
                    submatrix[subi][subj] = matrix[i][j];  
                    subj++;  
                }  
                subi++;  
            }  
            det = det + (pow(-1, x) * matrix[0][x] * determinant(submatrix, n - 1));  
        }  
    }  
    return det;  
}
```

## Метод getSolutions

Вычисляет вспомогательные определители системы с помощью потоков, затем вычисляет решения системы и выводит их.

Delta0 уже найден. Найдем delta1-delta4 параллельно, с помощью директивы parallel sections.

```
void getSolutions(int** slae, int d) {
    int* delta = new int[5];
    delta[0] = d;

    // Count deltas 1-4.
    omp_set_num_threads(4);
    #pragma omp parallel sections
    {
        #pragma omp section
        {
            delta[1] = getCramerDet(slae[4], slae[1], slae[2], slae[3]);
        }
        #pragma omp section
        {
            delta[2] = getCramerDet(slae[0], slae[4], slae[2], slae[3]);
        }
        #pragma omp section
        {
            delta[3] = getCramerDet(slae[0], slae[1], slae[4], slae[3]);
        }
        #pragma omp section
        {
            delta[4] = getCramerDet(slae[0], slae[1], slae[2], slae[4]);
        }
    }
    #pragma omp barrier
}
```

Подождем, пока все потоки отработают, с помощью директивы barrier. После этого найдем решения и выведем их.

```
#pragma omp barrier

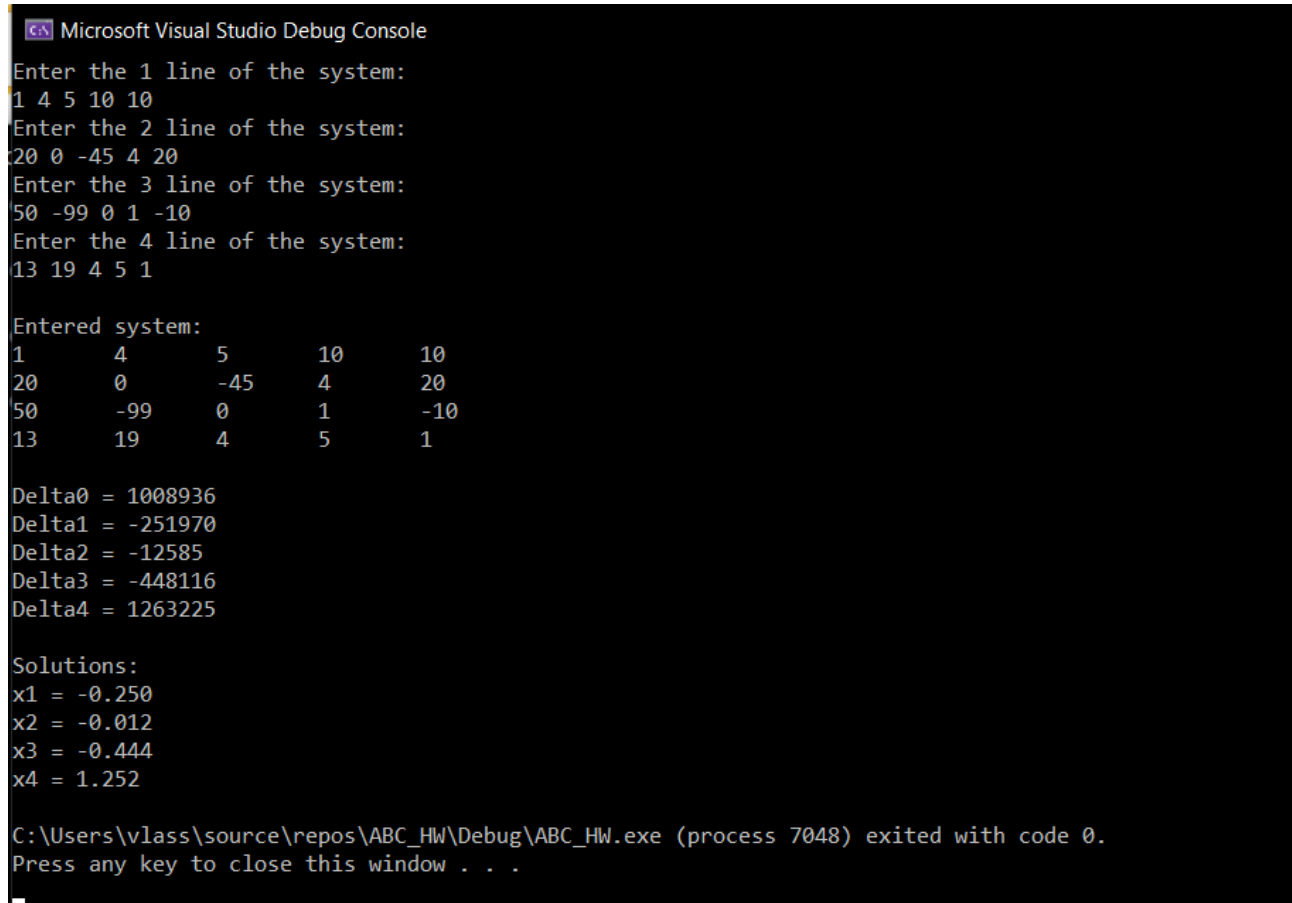
// Print the deltas.
printf("Delta0 = %d\n", delta[0]);
for (int i = 1; i < 5; i++)
{
    printf("Delta%d = %d\n", i, delta[i]);
}

// Calculate and print solutions.
printf("\nSolutions:\n");
for (int i = 1; i < 5; i++)
{
    printf("x%d = %.3f\n", i, delta[i] / (double)delta[0]);
}
return;
}
```

# Тестирование программы

## Тест 1

Введем случайную систему



```
Microsoft Visual Studio Debug Console
Enter the 1 line of the system:
1 4 5 10 10
Enter the 2 line of the system:
20 0 -45 4 20
Enter the 3 line of the system:
50 -99 0 1 -10
Enter the 4 line of the system:
13 19 4 5 1

Entered system:
1      4      5      10      10
20     0     -45     4      20
50    -99     0      1     -10
13     19     4      5       1

Delta0 = 1008936
Delta1 = -251970
Delta2 = -12585
Delta3 = -448116
Delta4 = 1263225

Solutions:
x1 = -0.250
x2 = -0.012
x3 = -0.444
x4 = 1.252

C:\Users\vlass\source\repos\ABC_HW\Debug\ABC_HW.exe (process 7048) exited with code 0.
Press any key to close this window . . .
```

Как видно, программа успешно отработала и вывела решения системы.

## Тест 2

Введем систему, главный определитель который равен 0

```
Microsoft Visual Studio Debug Console
Enter the 1 line of the system:
1 2 3 4 5
Enter the 2 line of the system:
1 2 3 4 5
Enter the 3 line of the system:
0 0 0 0 0
Enter the 4 line of the system:
0 0 0 0 0

Entered system:
1      2      3      4      5
1      2      3      4      5
0      0      0      0      0
0      0      0      0      0

Delta0 equals zero hence the Cramer's rule can't be applied!
C:\Users\vlass\source\repos\ABC_HW\Debug\ABC_HW.exe (process 6808) exited with code 0.
Press any key to close this window . . .
```

Как видно, программа сообщает о том, что данную систему невозможно решить методом Крамера, и завершает работу.

## Тест 3

Попробуем совершить некорректный ввод:

```
Microsoft Visual Studio Debug Console
Enter the 1 line of the system:
this is a system

Incorrect input!
C:\Users\vlass\source\repos\ABC_HW\Debug\ABC_HW.exe (process 13596) exited with code 0.
Press any key to close this window . . .
```

Таким образом, программа на разных входных данных работает корректно. Спасибо за внимание!