Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА №4

З ДИСЦИПЛІНИ “ ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ”

НА ТЕМУ: “OpenMP. Бар’єри, критичні секції”

Виконав:
Студент III курсу ФІОТ
групи ІО-82
Шендріков Євгеній
Номер у списку - 24

Перевірив:
Доцент
Корочкін О. В.

м. Київ – 2021 р.

Технічне завдання

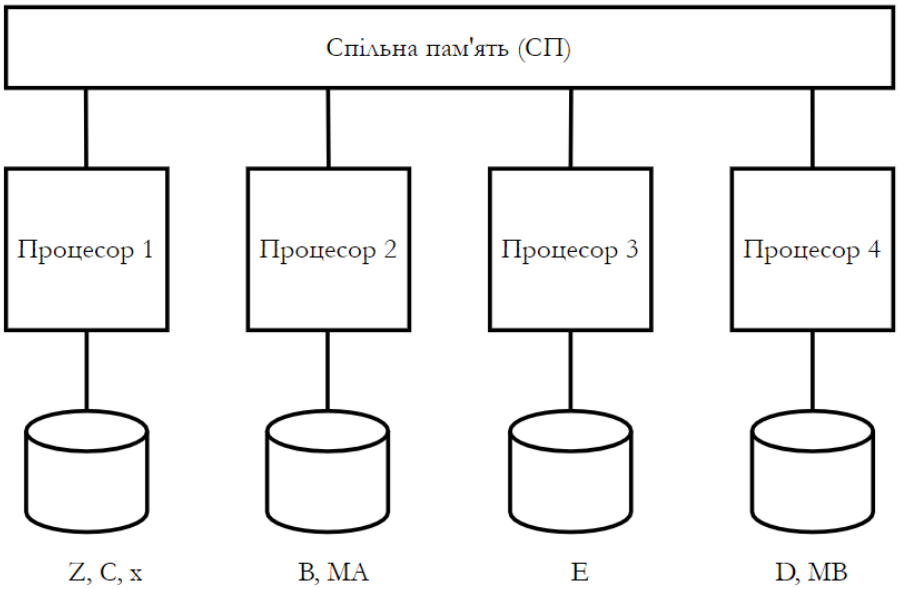
1. Розробити паралельний алгоритм рішення математичної задачі $Z = (B * C) * D + E * (MA * MB) * x$ з використанням бібліотеки OpenMp на мові C++;
2. Виявити спільні ресурси;
3. Описати алгоритм кожного потоку (T1 – Tp) з визначенням критичних ділянок (КД) та бар'єрів;
4. Розробити структурну схему взаємодії задач, де застосувати всі вказані засоби взаємодії процесів;
5. Розробити програму (обов'язкові "шапка", коментарі);
6. Виконати налагодження програми;
7. Отримати правильні результати обчислень;
8. За допомогою Диспетчеру задач Windows проконтролювати завантаження ядер процесору.

Задача: $Z = (B * C) * D + E * (MA * MB) * x$;

Мова програмування: C++;

Засоби організації взаємодії: бар'єри, замки, критичні секції OpenMP;

Структурна схема ПКС



Виконання роботи

Етап 1. Побудова паралельного алгоритму

1) $b_i = B_H * C_H$, $i = \overline{1,P}$
2) $b = b + b_i$, $i = \overline{1,P}$
3) $Z_H = b * D_H + E * (MA * MB_H) * x$

Спільний ресурс: b, x, E, MA

Етап 2. Розроблення алгоритмів роботи кожного процесу

Задача T1		КД та Бар'єри
1	**Введення** C, x	
2	Бар'єр. Синхронізація вводу	Бар′єр
3	**Обчислення 1:** $b_1 := B_H * C_H$	
4	**Обчислення 2:** $b := b + b_1$	КД
5	Бар'єр. Синхронізація обчислення 2	Бар′єр
6	**Копіювання** $b_1 := b$	КД
7	**Копіювання** $x_1 := x$	КД
8	**Копіювання** $E_1 := E$	КД
9	**Копіювання** $MA_1 := MA$	КД
10	**Обчислення 3**: $Z_H = b_1 * D_H + E_1 * (MA_1 * MB_H) * x_1$	
11	Бар'єр. Синхронізація обчислення 3	Бар′єр
12	**Виведення** Z	

Задача T2		КД та Бар'єри
1	**Введення** B, MA	
2	Бар'єр. Синхронізація вводу	Бар′єр
3	**Обчислення 1:** $b_2 := B_H * C_H$	
4	**Обчислення 2:** $b := b + b_2$	КД
5	Бар'єр. Синхронізація обчислення 2	Бар′єр
6	**Копіювання** $b_2 := b$	КД
7	**Копіювання** $x_2 := x$	КД
8	**Копіювання** $E_2 := E$	КД
9	**Копіювання** $MA_2 := MA$	КД
10	**Обчислення 3**: $Z_H = b_2 * D_H + E_2 * (MA_2 * MB_H) * x_2$	

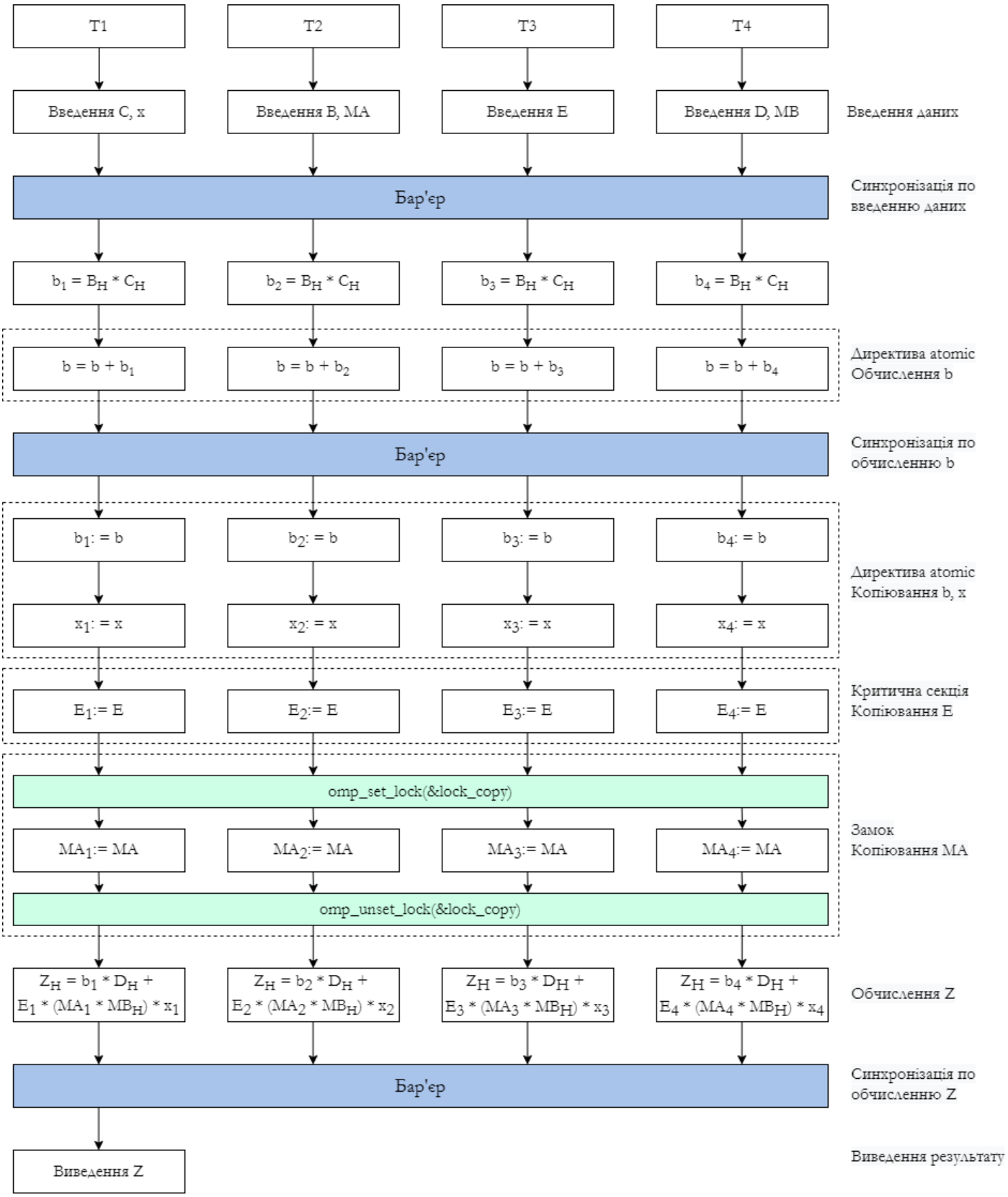
Задача T3		КД та Бар'єри
1	**Введення** E	
2	Бар'єр. Синхронізація вводу	Бар′єр
3	**Обчислення 1:** $b_3 := B_H * C_H$	
4	**Обчислення 2:** $b := b + b_3$	КД
5	Бар'єр. Синхронізація обчислення 2	Бар′єр
6	**Копіювання** $b_3 := b$	КД
7	**Копіювання** $x_3 := x$	КД
8	**Копіювання** $E_3 := E$	КД

9	**Копіювання** MA_3:= MA	КД
10	**Обчислення 3**: $Z_H = b_3 * D_H + E_3 * (MA_3 * MB_H) * x_3$	
Задача Т4		КД та Бар'єри
1	**Введення** D, MB	
2	Бар'єр. Синхронізація вводу	Бар′єр
3	**Обчислення 1:** b_4: = $B_H * C_H$	
4	**Обчислення 2:** b: = b + b_4	КД
5	Бар'єр. Синхронізація обчислення 2	Бар′єр
6	**Копіювання** b_4: = b	КД
7	**Копіювання** x_4: = x	КД
8	**Копіювання** E_4:= E	КД
9	**Копіювання** MA_4:= MA	КД
10	**Обчислення 3**: $Z_H = b_4 * D_H + E_4 * (MA_4 * MB_H) * x_4$	

Етап 3. Розроблення структурної схеми взаємодії задач

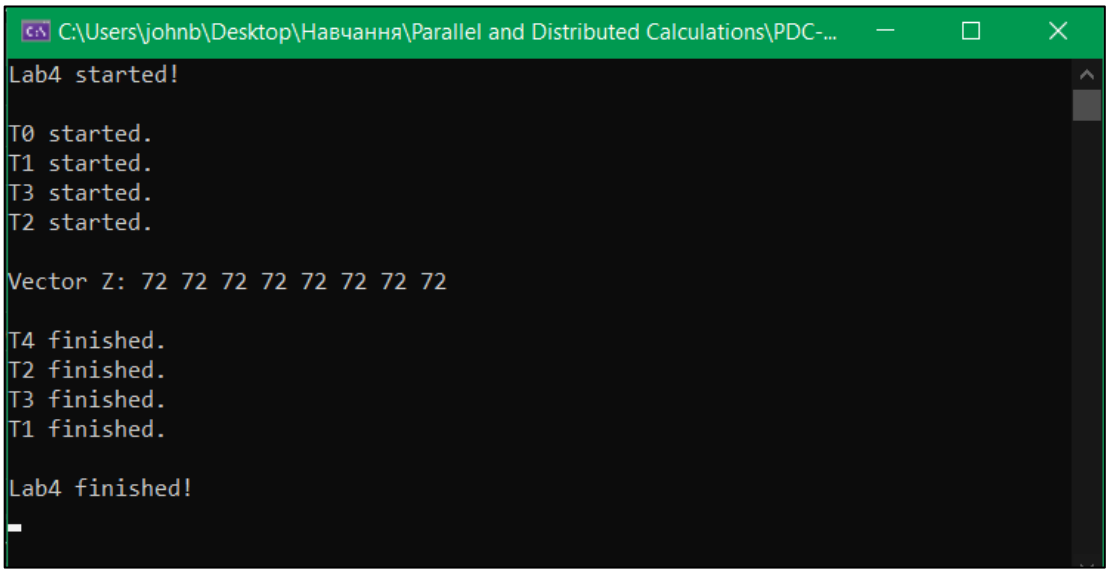
На структурній схемі взаємодії задач зображено такі засоби організації взаємодії потоків, як:

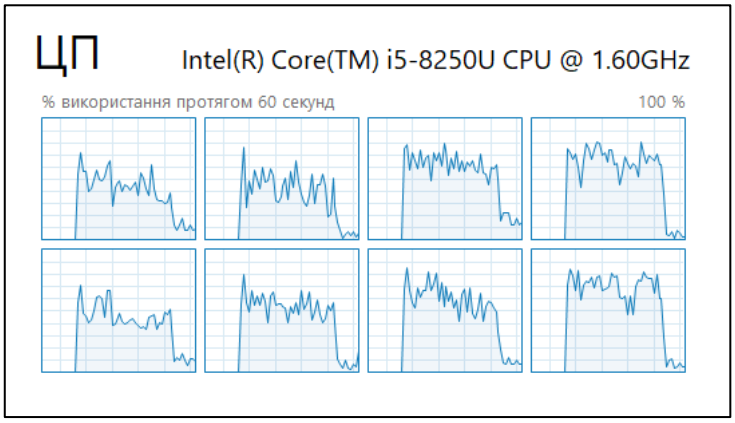
- **бар'єри** для синхронізації потоків;
- директива **atomic** для обчислення b;
- директива **atomic** для забезпечення послідовного доступу до спільних ресурсів b та x;
- директива **critical** для керування доступом до спільного ресурсу E;
- **замок** *lock_copy* для керування доступом до спільного ресурсу MA.


T1
T2
T3
T4
Введення C, x
Введення B, MA
Введення E
Введення D, MB
Введення даних
Бар'єр
Синхронізація по введенню даних
$b_1 = B_H * C_H$
$b_2 = B_H * C_H$
$b_3 = B_H * C_H$
$b_4 = B_H * C_H$
$b = b + b_1$
$b = b + b_2$
$b = b + b_3$
$b = b + b_4$
Директива atomic
Обчислення b
Бар'єр
Синхронізація по обчисленню b
$b_1 := b$
$b_2 := b$
$b_3 := b$
$b_4 := b$
$x_1 := x$
$x_2 := x$
$x_3 := x$
$x_4 := x$
Директива atomic
Копіювання b, x
$E_1 := E$
$E_2 := E$
$E_3 := E$
$E_4 := E$
Критична секція
Копіювання E
omp_set_lock(&lock_copy)
$MA_1 := MA$
$MA_2 := MA$
$MA_3 := MA$
$MA_4 := MA$
Замок
Копіювання MA
omp_unset_lock(&lock_copy)
$Z_H = b_1 * D_H + E_1 * (MA_1 * MB_H) * x_1$
$Z_H = b_2 * D_H + E_2 * (MA_2 * MB_H) * x_2$
$Z_H = b_3 * D_H + E_3 * (MA_3 * MB_H) * x_3$
$Z_H = b_4 * D_H + E_4 * (MA_4 * MB_H) * x_4$
Обчислення Z
Бар'єр
Синхронізація по обчисленню Z
Виведення Z
Виведення результату

Етап 4. Розробка програми

Результат роботи





Висновки

1. На основі бар'єрів, замків та критичних секцій бібліотеки OpenMP на мові C++ було розроблено програму та паралельний алгоритм для рішення математичної задачі заданої за варіантом.

2. Було описано алгоритм кожного потоку (T1 – T4) з визначенням критичних ділянок (КД) та бар'єрів;

3. Розроблено структурну схему взаємодії задач, де було застосовано вказані в завданні засоби взаємодії процесів. Засобом організації взаємодії слугували бар'єри, директиви atomic, замки та критичні секції.

4. Було написано програму згідно з завданням та перевірено її працездатність, а також проконтрольовано завантаження ядер процесору за допомогою Диспетчеру задач. Програма забезпечує 80% завантаженості.

Лістинг коду

Lab4.cpp

```
/*-------------------------------------------------------
 |                      Labwork #4                      |
 |          OpenMP. Barriers, critical sections         |
 -------------------------------------------------------
 |  Author  |       Jack (Yevhenii) Shendrikov          |
 |  Group   |                IO-82                      |
 |  Variant |                 #25                       |
 |  Date    |             23.03.2021                    |
 -------------------------------------------------------
  | Function |     Z = (B*C)*D + E*(MA*MB)*x            |
  ---------------------------------------------------------
  */

#include "omp.h"
#include "Data.h"
#include <iostream>
#include <windows.h>

using namespace std;

// Розмір стеку
#pragma comment(linker, "/stack:160000000")

const int P = 4;
int N = 8;

int main()
{
  int b = 0;
  int x = 0;

  vector Z = new int[N];
  vector B = new int[N];
  vector C = new int[N];
  vector D = new int[N];
  vector E = new int[N];

  matrix MA;
  matrix MB;

  const int H = N / P;

  void cs();
  omp_lock_t lock_copy;
  omp_init_lock(&lock_copy);

  cout << "Lab4 started!\n\n";

  omp_set_num_threads(P);
```

```
50.
51.   #pragma omp parallel
52.   {
53.         int tid = omp_get_thread_num();
54.         cout << "T" << tid << " started.\n";
55.
56.         //---------------------------- Ввід Даних ------------------------
--
57.         switch (tid)
58.         {
59.               case 0:
60.                     C = inVector(1);
61.                     x = 1;
62.                     break;
63.               case 1:
64.                     B = inVector(1);
65.                     MA = inMatrix(1);
66.                     break;
67.               case 2:
68.                     E = inVector(1);
69.                     break;
70.               case 3:
71.                     D = inVector(1);
72.                     MB = inMatrix(1);
73.                     break;
74.         }
75.
76.         //----------------- Синхронізація по введенню даних --------------
--
77.         #pragma omp barrier
78.               int bi = 0;
79.               for (int i = tid * H; i < (tid + 1) * H; i++)
80.                     bi += B[i] * C[i];
81.
82.               // Обчислення b
83.               #pragma omp_atomic
84.               {
85.                     b += bi;
86.               }
87.
88.         //------------------ Синхронізація по обчисленню b ----------------
--
89.         #pragma omp barrier
90.               int xi = 0;
91.               vector Ei = new int[N];
92.               matrix MAi = new vector[N];
93.
94.               // Директива atomic - копіювання b
95.               #pragma omp_atomic
96.               {
97.                     bi = b;
98.               }
```

```
99.
100.              // Директива atomic - копіювання x
101.              #pragma omp_atomic
102.              {
103.                    xi = x;
104.              }
105.
106.              // Критична секція - копіювання E
107.              #pragma omp critical(cs)
108.              {
109.                    Ei = copyVector(E);
110.              }
111.
112.              // Замок - копіювання MA
113.              omp_set_lock(&lock_copy);
114.                    MAi = copyMatrix(MA);
115.              omp_unset_lock(&lock_copy);
116.
117.
118.              // Обчислення Z
119.              int buf;
120.              for (int i = tid * H; i < (tid + 1) * H; i++) {
121.                    buf = 0;
122.                    Z[i] = 0;
123.
124.                    Z[i] += bi * D[i];
125.
126.                    for (int j = 0; j < N; j++) {
127.                          buf = 0;
128.                          for (int k = 0; k < N; k++)
129.                                buf += MAi[i][k] * MB[k][j];
130.                          Z[i] += Ei[j] * buf * xi;
131.                    }
132.              }
133.
134.        //------------------ Синхронізація по обчисленню Z ----------------
--
135.        #pragma omp barrier
136.              if (tid == 0 && N < 15) {
137.                    outVector(Z, 'Z');
138.              }
139.              Sleep(100);
140.              cout << "T" << tid + 1 << " finished." << endl;
141.  }
142.
143.  cout << "\nLab4 finished!\n";
144.  getchar();
145.  return 0;
146. }
```

Data.cpp

```
1. #include <windows.h>
2. #include <iostream>
3. #include "Data.h"
4.
5. using namespace std;
6.
7. // ------------------- Fill Matrix/Vector With Specific Number -------------
------
8. vector inVector(int value) {
9.    vector result = new int[N];
10.   for (int i = 0; i < N; i++)
11.         result[i] = value;
12.
13.   return result;
14. }
15.
16. matrix inMatrix(int value) {
17.   matrix result = new vector[N];
18.   for (int i = 0; i < N; i++)
19.         result[i] = new int[N];
20.
21.   for (int i = 0; i < N; i++)
22.         for (int j = 0; j < N; j++)
23.               result[i][j] = value;
24.
25.   return result;
26. }
27.
28.
29. // ------------- Print Vector Into Console --------------
30. void outVector(vector vec, char name) {
31.   cout << "\nVector " << name << ": ";
32.   for (int i = 0; i < N; i++)
33.   {
34.         cout << vec[i] << " ";
35.   }
36.   cout << "\n" << endl;
37. }
38.
39.
40. // ------------- Copy Vector, Matrix --------------
41. vector copyVector(vector vec) {
42.   vector result = new int[N];
43.   for (int i = 0; i < N; i++)
44.   {
45.         result[i] = vec[i];
46.   }
47.   return result;
48. }
49.
50. matrix copyMatrix(matrix matr) {
```

```
51.   matrix result = new vector[N];
52.   for (int i = 0; i < N; i++)
53.         result[i] = new int[N];
54.
55.   for (int i = 0; i < N; i++)
56.         for (int j = 0; j < N; j++)
57.               result[i][j] = matr[i][j];
58.
59.   return result;
60. }
```

Data.h

```
1. typedef int* vector;
2. typedef int** matrix;
3.
4. extern int N;
5.
6. vector inVector(int);
7. matrix inMatrix(int);
8.
9. void outVector(vector, char);
10.
11. vector copyVector(vector);
12. matrix copyMatrix(matrix);
```