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 №6

З ДИСЦИПЛІНИ “ОСНОВИ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ ”

НА ТЕМУ: “ПОТОКИ В БІБЛІОТЕЦІ MPI”

Виконав:
Студент III курсу ФІОТ
групи ІО-82
Шендріков Євгеній
Номер у списку - 25

Перевірив:
Доцент
Корочкін О. В.

м. Київ – 2020 р.

Завдання

1. Розробити програму за допомогою засобів бібліотеки MPI, яка містить ***паралельні потоки,*** кожен з яких реалізовує відповідну функцію F1, F2, F3 згідно отриманому варіанту.

2. Програма повинна складатися із пакету *Data* і основної програми — процедури (класу) *Lab6*. Пакет реалізовує ресурси, необхідні для обчислення функцій *F1, F2, F3* через підпрограми *T1, T2, T3*.

3. Кожен потік має здійснювати дії, необхідні для паралельного обчислення відповідної функції, а саме: введення відповідних даних, обчислення функції F1, F2, F3, виведення результату.

4. В тілі задачі задіяти оператор задержки **sleep** при виконанні функцій F1, F2, F3 з невеликим часом затримки.

Варіант завдання

		F1	F2	F3	
25	Шендріков Євгеній Олександрович	1.21	2.27	3.30	

1.21 D = SORT(A)+SORT(B)+SORT(C)*(MA*ME)

2.27 MF = (MG*MH)*TRANS(MK)

3.30 S = (MO*MP)*V+t*MR*(O+P)

Виконання роботи

Під час виконання лабораторної роботи за допомогою засобів бібліотеки MPI на базі мови програмування C++ було розроблено відповідну програму, що містить паралельні потоки, кожен з яких реалізовує функцію згідно варіанту. Для кожного класу було створено заголовковий файл для опису використаних у класі методів та змінних.

Для роботи з векторами та матрицями було створено спеціальний пакет *Data*, який містить функціонал для реалізації функцій F1, F2, F3.

Програмно було реалізовано «забиття» всіх елементів для T1 – як 1, для T2 – як 2, для T3 – як 3. При N > 10 можливість виводити результат в консоль не передбачена.

Висновки

1. На основі засобів бібліотеки MPI на C++ було розроблено програму, яка містить паралельні потоки, кожен з яких реалізовує відповідну функцію *F1*, *F2*, *F3*. Загалом, потоки в MPI створюються через копіювання, але на відміну від OpenMP тут копіюється вся програма. MPI дозволяє взяти послідовну програму, проаналізувати її, відокремити паралельні ділянки за допомогою функцій MPI і отримати паралельну програму. Також було розроблено пакет *Data*, який містить функціонал для роботи з матрицями та векторами.

2. За допомогою методів *MPI_Init()* та *MPI_Finalize()* було ініціалізовано та, відповідно, завершено паралельну частину програми. Для отримання ідентифікатору кожної задачі з метою подальшого встановлення кожній задачі конкретних дій щодо вводу, обчислення та виведення даних використовувався метод *MPI_Comm_rank()*. Також було використано метод *MPI_Barrier()*, який слугує певним аналогом методу *join*() в Java. Цей метод дозволяє, в точці в якій він розташований, дочекатися, коли всі задачі завершать свої дії, здійснять виклик цього метода і тільки тоді паралельна програма може виконуватись далі.

3. Проблема роботи зі спільними ресурсами для потоків вирішувалась встановленням всіх значень для T1 – як 1, для T2 – як 2, для T3 – як 3.

4. Було зроблено аналіз проблем, які виникли під час виконання, перевірено працездатність програми і програмним шляхом оброблено виключні ситуації, наприклад, при N < 0 чи при N > 10 можливість виводити результат не передбачена також виводиться помилка при введенні даних відмінних від int.

Лістинг програми

Lab6.cpp

```
1 /*---------------------------------------------------
2  |                      Labwork #6                   |
3  ---------------------------------------------------
4  |  Author  |       Jack (Yevhenii) Shendrikov       |
5  |  Group   |                IO-82                   |
6  |  Variant |                 #25                    |
7  |  Date    |             01.11.2020                 |
8  ---------------------------------------------------
9  | Function 1 |  D = SORT(A)+SORT(B)+SORT(C)*(MA*ME)  |
10  | Function 2 |       MF = (MG*MH)*TRANS(MK)          |
11  | Function 3 |      S = (MO*MP)*V+t*MR*(O+P)         |
12  ---------------------------------------------------
13 */
14
15 #include "mpi.h"
16 #include "F1.h"
17 #include "F2.h"
18 #include "F3.h"
19
20 int N;
21
```

```
22 int main(int argc, char* argv[]) {
23     //--------------------------- Input Handler --------------------------
24     // header
25     printf("-----------------------------------------------------\n"
26            "| Function 1 | D = SORT(A)+SORT(B)+SORT(C)*(MA*ME) |\n"
27            "| Function 2 |      MF = (MG*MH)*TRANS(MK)         |\n"
28            "| Function 3 |     S = (MO*MP)*V+t*MR*(O+P)        |\n"
29            "-----------------------------------------------------\n\n"
30            "!!! Note that if the value of N > 10 -> the result will not be displayed
!!!\n"
31            "!!! If you enter N <= 0 - execution will be terminated !!!\n\n"
32            "Enter N: ");
33     cin >> N;
34
35     // check for int value of N, else N = 3
36     if (cin.fail()) {
37            cout << "\n!!! You should enter data of type int, N will be taken as 3 !!!\n"
<< endl;
38            N = 3;
39     }
40
41     // check for positive value of N
42     if (N <= 0) {
43            cout << "Restart the program and enter N > 0." << endl;
44            exit(EXIT_FAILURE);
45     }
46
47     //------------------------ Main Body ---------------------------------
48     MPI_Init(&argc, &argv);
49     cout << "\nLab6 started!\n" << endl;
50
51     int rank;
52
53     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
54
55     F1 T1 = F1(N);
56     F2 T2 = F2(N);
57     F3 T3 = F3(N);
58
59     switch (rank) {
60     case 0: T1.run();
61     case 1: T2.run();
62     case 2: T3.run();
63     }
64
65     MPI_Barrier(MPI_COMM_WORLD);
66
67     cout << "\nLab6 finished!\n";
68     cin.get();
69     MPI_Finalize();
70 }
```

T1.h

```
1 #pragma once
2 #include <iostream>
3 #include <Windows.h>
4 #include "Data.h"
5
6 class F1 {
7 private:
8      int N;
9 public:
10     F1(int N);
11     DWORD run();
12 };
```

T2.h

```
#pragma once
#include <iostream>
#include <Windows.h>
#include "Data.h"

class F2 {
private:
	int N;
public:
	F2(int N);
	DWORD run();
};
```

T3.h

```
#pragma once
#include <iostream>
#include <Windows.h>
#include "Data.h"

class F3 {
private:
	int N;
public:
	F3(int N);
	DWORD run();
};
```

T1.cpp

```
#include "F1.h"

F1::F1(int N) {
	this->N = N;
}

DWORD F1::run() {
	cout << "T1 started." << endl;
	Data* data = new Data(N);

	vector<int> A, B, C;
	vector<vector<int>> MA, ME;

	// Generate Input Values
	Sleep(50);
	cout << "T1 gets a permit." << endl;

	A = data->FillVectorWithNumber(1);
	B = data->FillVectorWithNumber(1);
	C = data->FillVectorWithNumber(1);
	MA = data->FillMatrixWithNumber(1);
	ME = data->FillMatrixWithNumber(1);

	// Calculate The Result
	vector<int> result = data->Func1(A, B, C, MA, ME);
	Sleep(100);

	// Output
	if (N <= 10) {
		cout << "T1 result:\n";
		data->VectorOutput(result, 'D');
	}

	cout << "T1 releases the permit." << endl;
	cout << "T1 finished.\n" << endl;

```

```
37     delete data;
38     return 0;
39 }
```

T2.cpp

```
1 #include "F2.h"
2
3 F2::F2(int N) {
4      this->N = N;
5 }
6
7 DWORD F2::run() {
8      cout << "T2 started." << endl;
9      Data* data = new Data(N);
10
11     vector<vector<int>> MG, MH, MK;
12
13     // Generate Input Values
14     Sleep(100);
15     cout << "T2 gets a permit." << endl;
16
17     MG = data->FillMatrixWithNumber(2);
18     MH = data->FillMatrixWithNumber(2);
19     MK = data->FillMatrixWithNumber(2);
20
21     // Calculate The Result
22     vector<vector<int>> result = data->Func2(MG, MH, MK);
23     Sleep(100);
24
25     // Output
26     if (N <= 10) {
27            cout << "T2 result:\n";
28            data->MatrixOutput(result, "MF");
29     }
30
31     cout << "T2 releases the permit." << endl;
32     cout << "T2 finished.\n" << endl;
33
34     delete data;
35     return 0;
36 }
```

T3.cpp

```
1 #include "F3.h"
2
3 F3::F3(int N) {
4      this->N = N;
5 }
6
7 DWORD F3::run() {
8      cout << "T3 started." << endl;
9      Data* data = new Data(N);
10
11     int t;
12     vector<int> V, O, P;
13     vector<vector<int>> MO, MP, MR;
14
15     // Generate Input Values
16     cout << "T3 gets a permit." << endl;
17
18     t = 3;
19     V = data->FillVectorWithNumber(3);
20     O = data->FillVectorWithNumber(3);
21     P = data->FillVectorWithNumber(3);
22     MO = data->FillMatrixWithNumber(3);
23     MP = data->FillMatrixWithNumber(3);
```

```
24      MR = data->FillMatrixWithNumber(3);
25
26      // Calculate The Result
27      vector<int> result = data->Func3(t, V, O, P, MO, MP, MR);
28      Sleep(100);
29
30      // Output
31      if (N <= 10) {
32            cout << "T3 result:\n";
33            data->VectorOutput(result, 'S');
34      }
35
36      cout << "T3 releases the permit." << endl;
37      cout << "T3 finished.\n" << endl;
38
39      delete data;
40
41      return 0;
42  }
```

Data.h

```
1  #pragma once
2  #include <random>
3  #include <ctime>
4  #include <string>
5  using namespace std;
6
7  class Data {
8  private:
9       int N;
10
11 public:
12      Data(int N);
13
14      // ------------------- Fill Matrix/Vector With Specific Number -------------------
15      vector<vector<int>> FillMatrixWithNumber(int number);
16      vector<int> FillVectorWithNumber(int number);
17
18      // ---------- Data Entry Handler For Matrices, Vectors And Numbers ---------
19      vector<vector<int>> MatrixInput(string name);
20      vector<int> VectorInput(char name);
21      int NumInput(char name);
22
23      // ------------- Print Matrix, Vector And Number Into Console --------------
24      void MatrixOutput(vector<vector<int>> MA, string name);
25      void VectorOutput(vector<int> A, char name);
26      void NumOutput(int a, char name);
27
28
29      // Transposing Matrix
30      vector<vector<int>> MatrixTransp(vector<vector<int>> MA);
31
32      // Multiply 2 Matrices
33      vector<vector<int>> MatrixMult(vector<vector<int>> MA, vector<vector<int>> MB);
34
35
36      // Multiply Matrix And Vector
37      vector<int> VectorMatrixMult(vector<int> A, vector<vector<int>> MA);
38
39      // Calculates Sum Of 2 Vectors
40      vector<int> SumVectors(vector<int> A, vector<int> B);
41
42      // Multiply Integer And Matrix
43      vector<int> IntVectorMult(int a, vector<int> A);
44
45      // Sort Vector
```

```
46      vector<int> SortVector(vector<int> A);
47
48
49      // F1 -> D = SORT(A)+SORT(B)+SORT(C)*(MA*ME)
50      vector<int> Func1(vector<int> A, vector<int> B, vector<int> C, vector<vector<int>>
MA, vector<vector<int>> ME);
51
52      // F2 -> MF = (MG*MH)*TRANS(MK)
53      vector<vector<int>> Func2(vector<vector<int>> MG, vector<vector<int>> MH,
vector<vector<int>> MK);
54
55      // F3 -> S = (MO*MP)*V+t*MR*(O+P)
56      vector<int> Func3(int t, vector<int> V, vector<int> O, vector<int> P,
vector<vector<int>> MO, vector<vector<int>> MP, vector<vector<int>> MR);
57
58 };
```

Data.cpp

```
1 #include "Data.h"
2 #include <iostream>
3
4 Data::Data(int N)
5 {
6       this->N = N;
7 }
8
9
10 // ------------------- Fill Matrix/Vector With Specific Number -------------------
11 vector<vector<int>> Data::FillMatrixWithNumber(int number)
12 {
13      vector<vector<int>> MA(N, vector <int>(N));
14      for (int i = 0; i < N; i++)
15      {
16              for (int j = 0; j < N; j++)
17              {
18                      MA[i][j] = number;
19              }
20      }
21      return MA;
22 }
23
24 vector<int> Data::FillVectorWithNumber(int number)
25 {
26      std::vector<int> A(N);
27      for (int i = 0; i < N; i++)
28      {
29              A[i] = number;
30      }
31      return A;
32 }
33
34
35 // ---------- Data Entry Handler For Matrices, Vectors And Numbers ---------
36 vector<vector<int>> Data::MatrixInput(string name)
37 {
38      cout << "Enter the " << N * N << " elements of the Matrix " << name << ":" << endl;
39      vector<vector<int>> MA(N, vector <int>(N));
40
41      for (int i = 0; i < N; i++)
42      {
43              for (int j = 0; j < N; j++)
44              {
45                      cout << name << "[" << i << "][" << j << "] = ";
46                      cin >> MA[i][j];
47              }
48      }
```

```
49      return MA;
50  }
51
52  vector<int> Data::VectorInput(char name)
53  {
54      cout << "Enter the " << N << " elements of the Vector " << name << ":" << endl;
55      vector<int> A(N);
56
57      for (int i = 0; i < N; i++)
58      {
59              cout << name << "[" << i << "] = ";
60              cin >> A[i];
61      }
62      return A;
63  }
64
65  int Data::NumInput(char name)
66  {
67      int a;
68      cout << "Enter number " << name << " = ";
69      cin >> a;
70      return a;
71  }
72
73
74  // ------------- Print Matrix, Vector And Number Into Console --------------
75  void Data::MatrixOutput(vector<vector<int>> MA, string name)
76  {
77      cout << "\tMatrix " << name << ":" << endl;
78      for (int i = 0; i < N; i++)
79      {
80              cout << "\t\t";
81              for (int j = 0; j < N; j++)
82              {
83                      cout << MA[i][j] << " ";
84              }
85              cout << endl;
86      }
87  }
88
89  void Data::VectorOutput(vector<int> A, char name)
90  {
91      cout << "\tVector " << name << ": ";
92      for (int i = 0; i < N; i++)
93      {
94              cout << A[i] << " ";
95      }
96      cout << endl;
97  }
98
99  void Data::NumOutput(int a, char name)
100 {
101     cout << "\tNumber " << name << ": " << a << "\n";
102 }
103
104
105 // Transposing Matrix
106 vector<vector<int>> Data::MatrixTransp(vector<vector<int>> MA)
107 {
108     int buf;
109     for (int i = 0; i < N; i++)
110     {
111             for (int j = 0; j <= i; j++)
112             {
113                     buf = MA[i][j];
114                     MA[i][j] = MA[j][i];
115                     MA[j][i] = buf;
116             }
```

```
117    }
118    return MA;
119 }
120
121 // Multiply 2 Matrices
122 vector<vector<int>> Data::MatrixMult(vector<vector<int>> MA, vector<vector<int>> MB)
123 {
124    vector<vector<int>> MC(N, vector <int>(N));
125    for (int i = 0; i < N; i++)
126    {
127           for (int j = 0; j < N; j++)
128           {
129                  for (int k = 0; k < N; k++)
130                  {
131                         MC[i][j] += MA[i][k] * MB[k][j];
132                  }
133           }
134    }
135    return MC;
136 }
137
138
139 // Multiply Matrix And Vector
140 vector<int> Data::VectorMatrixMult(vector<int> A, vector<vector<int>> MA)
141 {
142    vector<int> B(N);
143    for (int i = 0; i < N; i++)
144    {
145           for (int j = 0; j < N; j++)
146           {
147                  B[i] += A[j] * MA[i][j];
148           }
149    }
150    return B;
151 }
152
153 // Calculates Sum Of 2 Vectors
154 vector<int> Data::SumVectors(vector<int> A, vector<int> B)
155 {
156    vector<int> C(N);
157    for (int i = 0; i < N; i++)
158    {
159           C[i] = A[i] + B[i];
160    }
161    return C;
162 }
163
164 // Multiply Integer And Matrix
165 vector<int> Data::IntVectorMult(int a, vector<int> A)
166 {
167    vector<int> B(N);
168    for (int i = 0; i < N; i++)
169    {
170           B[i] = a * A[i];
171    }
172    return B;
173 }
174
175 // Sort Vector
176 vector<int> Data::SortVector(vector<int> A)
177 {
178    sort(A.begin(), A.end());
179    return A;
180 }
181
182
183 // F1 -> D = SORT(A)+SORT(B)+SORT(C)*(MA*ME)
```

```
184 vector<int> Data::Func1(vector<int> A, vector<int> B, vector<int> C, vector<vector<int>>
MA, vector<vector<int>> ME)
185 {
186    return SumVectors(SumVectors(SortVector(A), SortVector(B)),
VectorMatrixMult(SortVector(C), MatrixMult(MA, ME)));
187 }
188
189 // F2 -> MF = (MG*MH)*TRANS(MK)
190 vector<vector<int>> Data::Func2(vector<vector<int>> MG, vector<vector<int>> MH,
vector<vector<int>> MK)
191 {
192    return MatrixMult(MatrixMult(MG, MH), MatrixTransp(MK));
193 }
194
195 // F3 -> S = (MO*MP)*V+t*MR*(O+P)
196 vector<int> Data::Func3(int t, vector<int> V, vector<int> O, vector<int> P,
vector<vector<int>> MO, vector<vector<int>> MP, vector<vector<int>> MR)
197 {
198    return SumVectors((VectorMatrixMult(V, MatrixMult(MO, MP))), IntVectorMult(t,
VectorMatrixMult(SumVectors(O, P), MR)));
199 }
200
```

Приклад роботи програми

```
-----------------------------------------------------
| Function 1 | D = SORT(A)+SORT(B)+SORT(C)*(MA*ME) |
| Function 2 |      MF = (MG*MH)*TRANS(MK)         |
| Function 3 |     S = (MO*MP)*V+t*MR*(O+P)        |
-----------------------------------------------------

!!! Note that if the value of N > 10 -> the result will not be displayed !!!
!!! If you enter N <= 0 - execution will be terminated !!!

Enter N: 3

Lab6 started!

T1 started.
T1 gets a permit.
T1 result:
        Vector D: 11 11 11
T1 releases the permit.
T1 finished.

T2 started.
T2 gets a permit.
T2 result:
        Matrix MF:
                72 72 72
                72 72 72
                72 72 72
T2 releases the permit.
T2 finished.

T3 started.
T3 gets a permit.
T3 result:
        Vector S: 405 405 405
T3 releases the permit.
T3 finished.


Lab6 finished!

D:\C++\Lab6\Debug\Lab6.exe (process 20700) exited with code 0.
Press any key to close this window . . .
```