

Національний технічний університет України  
«Київський політехнічний інститут»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки

Лабораторна робота №7

МРІ.

Виконав:  
студент групи ІП-32  
Ковальчук О. М.  
Перевірив:  
Корочкін О. В.

Київ - 2015

## Лабораторна робота №7. MPI

**Мета роботи:** вивчення засобів MPI для роботи з потоками

**Мова програмування:** C++, використання MPI

**Завдання:** Розробити програму, що містить паралельні потоки, кожен з яких реалізує функції F1, F2, F3 з лабораторної роботи №1. Вимоги до потоків такі ж, як в лабораторній роботі № 2.

**Функції:**

F1:  $C = A - B * (MA * MD)$

F2:  $o = \text{Min}(MK * MM)$

F3:  $T = (MS * MZ) * (W + X)$

### Лістинг програми

```
src/lab07.cpp
1  /**
2   * Parallel programming
3   * Lab 7
4   *
5   * Functions:
6   * F1: C = A - B * (MA * MD)
7   * F2: o = Min(MK * MM)
8   * F3: T = (MS * MZ) * (W + X)
9   *
10  * @since 2015-11-11
11  * @author Olexandr Kovalchuk
12  * @group IP-32
13  */
14
15 #include <iostream>
16 #include <mpi.h>
17 #include "functions.h"
18 #include "cmdopts.h"
19 #include "tasks.h"
20
21 int main(int argc, char* argv[]) {
22     int size, threads, tid, ctid;
23     int group_size = 1;
24
25     int ranks[] = {1};
26
27     MPI::Group group;
28     MPI::Intercomm communicator;
29
30     MPI::Init(argc, argv);
31
32     group = MPI::COMM_WORLD.Get_group().Excl(group_size, ranks);
33     communicator = MPI::COMM_WORLD.Create(group);
34
35     threads = MPI::COMM_WORLD.Get_size();
36     tid = MPI::COMM_WORLD.Get_rank();
37     ctid = (communicator == MPI::COMM_NULL) ? -1 : communicator.Get_rank();
38
39     {
40         char *sizeopt = getCmdOption(argv, argv + argc, "-s");
41         size = sizeopt ? atoi(sizeopt) : 4;
42     }
43
44     switch(tid % 3) {
45     case 0:
46         task1(size, tid, ctid);
```

```

47         break;
48         case 1:
49             task2(size, tid, ctid);
50             break;
51         default: case 2:
52             task3(size, tid, ctid);
53             break;
54     }
55
56     MPI::Finalize();
57
58     return 0;
59 }
src/vector.h
1  /**
2   * Parallel programming
3   * Lab 7
4   *
5   * Functions:
6   * F1:  $C = A - B * (MA * MD)$ 
7   * F2:  $o = \text{Min}(MK * MM)$ 
8   * F3:  $T = (MS * MZ) * (W + X)$ 
9   *
10  * @since 2015-10-18
11  * @author Olexandr Kovalchuk
12  * @group IP-32
13  */
14
15  #ifndef LAB_VECTOR_H
16  #define LAB_VECTOR_H
17
18  #include <vector>
19  #include "matrix.h"
20  #include <assert.h>
21  typedef std::vector<int> vector;
22
23  vector generateVector(int size, int filler = 1);
24
25  vector operator*(matrix left, vector right);
26  vector operator*(vector left, matrix right);
27  vector operator+(vector left, vector right);
28  vector operator-(vector left, vector right);
29
30  #endif // LAB_VECTOR_H
31
src/functions.cpp
1  /**
2   * Parallel programming
3   * Lab 7
4   *
5   * Functions:
6   * F1:  $C = A - B * (MA * MD)$ 
7   * F2:  $o = \text{Min}(MK * MM)$ 
8   * F3:  $T = (MS * MZ) * (W + X)$ 
9   *
10  * @since 2015-10-18
11  * @author Olexandr Kovalchuk
12  * @group IP-32
13  */
14
15  #include "functions.h"
16
17  vector func1(vector a, vector b, matrix ma, matrix md) {
18      return (a - b * (ma * md));
19  }
20

```

```

21 int func2(matrix mk, matrix mn) {
22     return (min(mk*mn));
23 }
24
25 vector func3(matrix ms, matrix mz, vector w, vector x) {
26     return ((ms * mz) * (w + x));
27 }
src/vector.cpp
1  /**
2   * Parallel programming
3   * Lab 7
4   *
5   * Functions:
6   * F1: C = A - B * (MA * MD)
7   * F2: o = Min(MK * MM)
8   * F3: T = (MS * MZ) * (W + X)
9   *
10  * @since 2015-10-18
11  * @author Olexandr Kovalchuk
12  * @group IP-32
13  */
14
15 #include "vector.h"
16
17 vector generateVector(int size, int filler) {
18     vector result = vector(size, filler);
19     return result;
20 }
21
22 vector operator*(matrix left, vector right) {
23     assert(left.size() > 0);
24     assert(right.size() > 0);
25     assert(right.size() == left[0].size());
26
27     vector result = generateVector(left.size(), 0);
28     for (int i = 0; i < left.size(); ++i) {
29         for (int j = 0; j < right.size(); ++j) {
30             result[i] += left[i][j] * right[j];
31         }
32     }
33     return result;
34 }
35
36 vector operator*(vector left, matrix right) {
37     assert(left.size() > 0);
38     assert(right.size() > 0);
39     assert(left.size() == right[0].size());
40
41     vector result = generateVector(left.size(), 0);
42     for (int i = 0; i < right[0].size(); ++i) {
43         for (int j = 0; j < right.size(); ++j) {
44             result[i] += right[i][j] * left[j];
45         }
46     }
47     return result;
48 }
49
50 vector operator+(vector left, vector right) {
51     assert(left.size() > 0);
52     assert(left.size() == right.size());
53
54     vector result = vector(left);
55     for (int i = 0; i < result.size(); ++i) {
56         result[i] += right[i];
57     }
58 }

```

```

59     return result;
60 }
61
62 vector operator-(vector left, vector right) {
63     assert(left.size() > 0);
64     assert(left.size() == right.size());
65
66     vector result = vector(left);
67     for (int i = 0; i < result.size(); ++i) {
68         result[i] -= right[i];
69     }
70
71     return result;
72 }

```

src/cmdopts.cpp

```

1  /**
2   * Parallel programming
3   * Lab 7
4   *
5   * Functions:
6   * F1: C = A - B * (MA * MD)
7   * F2: o = Min(MK * MM)
8   * F3: T = (MS * MZ) * (W + X)
9   *
10  * @since 2015-10-18
11  * @author Olexandr Kovalchuk
12  * @group IP-32
13  */
14
15 #include "cmdopts.h"
16
17 char *getCmdOption(char **begin, char **end, const std::string &option) {
18     char **itr = std::find(begin, end, option);
19     if (itr != end && ++itr != end) {
20         return *itr;
21     }
22     return 0;
23 }
24
25 bool cmdOptionExists(char **begin, char **end, const std::string &option) {
26     return std::find(begin, end, option) != end;
27 }

```

src/functions.h

```

1  /**
2   * Parallel programming
3   * Lab 7
4   *
5   * Functions:
6   * F1: C = A - B * (MA * MD)
7   * F2: o = Min(MK * MM)
8   * F3: T = (MS * MZ) * (W + X)
9   *
10  * @since 2015-10-18
11  * @author Olexandr Kovalchuk
12  * @group IP-32
13  */
14
15 #ifndef LAB_FUNCTIONS_H
16 #define LAB_FUNCTIONS_H
17
18 #include "matrix.h"
19 #include "vector.h"
20
21 vector func1(vector a, vector b, matrix ma, matrix md);
22 int func2(matrix mk, matrix mn);
23 vector func3(matrix ms, matrix mz, vector w, vector x);

```

```

24
25 #endif // LAB_FUNCTIONS_H
26
src/tasks.cpp
1  /**
2  * Parallel programming
3  * Lab 7
4  *
5  * Functions:
6  * F1:  $C = A - B * (MA * MD)$ 
7  * F2:  $o = \text{Min}(MK * MM)$ 
8  * F3:  $T = (MS * MZ) * (W + X)$ 
9  *
10 * @since 2015-11-11
11 * @author Olexandr Kovalchuk
12 * @group IP-32
13 */
14
15 #include "tasks.h"
16
17 void task1(int size, int tid, int ctid) {
18     {
19         std::stringstream ss;
20         ss << "task 1 with tid " << tid << " and ctid " << ctid << " started" << std::endl;
21         std::cout << ss.str();
22     }
23
24     vector a, b;
25     matrix ma, md;
26
27     a = generateVector(size);
28     b = generateVector(size);
29     ma = generateMatrix(size);
30     md = generateMatrix(size);
31
32     vector result = func1(a, b, ma, md);
33
34     if (size < 8) {
35         std::stringstream ss;
36         ss << "task 1 result: ";
37         ss << '[';
38         for (int i = 0; i < result.size(); ++i) {
39             ss << result[i] << " ";
40         }
41         ss << ']';
42         ss << std::endl;
43         std::cout << ss.str();
44     }
45
46     {
47         std::stringstream ss;
48         ss << "task 1 with tid " << tid << " and ctid " << ctid << " finished" << std::endl;
49         std::cout << ss.str();
50     }
51 }
52
53 void task2(int size, int tid, int ctid) {
54     {
55         std::stringstream ss;
56         ss << "task 2 with tid " << tid << " and ctid " << ctid << " started" << std::endl;
57         std::cout << ss.str();
58     }
59
60     matrix mk, mn;
61
62     mk = generateMatrix(size);

```

```

63     mn = generateMatrix(size);
64
65     int result = func2(mk, mn);
66
67     if (size < 8) {
68         std::stringstream ss;
69         ss << "task 2 result: " << result;
70         ss << std::endl;
71         std::cout << ss.str();
72     }
73
74     {
75         std::stringstream ss;
76         ss << "task 2 with tid " << tid << " and ctid " << ctid << " finished" << std::endl;
77         std::cout << ss.str();
78     }
79 }
80
81 void task3(int size, int tid, int ctid) {
82     {
83         std::stringstream ss;
84         ss << "task 3 with tid " << tid << " and ctid " << ctid << " started" << std::endl;
85         std::cout << ss.str();
86     }
87
88     vector w, x;
89     matrix ms, mz;
90
91     w = generateVector(size);
92     x = generateVector(size);
93     ms = generateMatrix(size);
94     mz = generateMatrix(size);
95
96     vector result = func3(ms, mz, w, x);
97
98     if (size < 8) {
99         std::stringstream ss;
100         ss << "task 3 result: ";
101         ss << '[';
102         for (int i = 0; i < result.size(); ++i) {
103             ss << result[i] << " ";
104         }
105         ss << ']';
106         ss << std::endl;
107         std::cout << ss.str();
108     }
109
110     {
111         std::stringstream ss;
112         ss << "task 3 with tid " << tid << " and ctid " << ctid << " finished" << std::endl;
113         std::cout << ss.str();
114     }
115 }

```

src/matrix.cpp

```

1  /**
2   * Parallel programming
3   * Lab 7
4   *
5   * Functions:
6   * F1: C = A - B * (MA * MD)
7   * F2: o = Min(MK * MM)
8   * F3: T = (MS * MZ) * (W + X)
9   *
10  * @since 2015-10-18
11  * @author Olexandr Kovalchuk
12  * @group IP-32

```

```

13  */
14
15  #include "matrix.h"
16
17  matrix generateMatrix(int size, int filler) {
18      return generateMatrix(size, size, filler);
19  }
20
21  matrix generateMatrix(int rows, int columns, int filler) {
22      matrix result = matrix();
23      for (int r = 0; r < rows; r++) {
24          result.push_back(std::vector<int>(columns, filler));
25      }
26      return result;
27  }
28
29  matrix operator*(matrix left, matrix right) {
30      assert(left.size() > 0);
31      assert(right.size() > 0);
32      assert(left[0].size() == right.size());
33
34      matrix result = generateMatrix(left.size(), right[0].size(), 0);
35      for (int i = 0; i < left.size(); ++i) {
36          for (int j = 0; j < right[0].size(); ++j) {
37              for (int k = 0; k < left[0].size(); ++k) {
38                  result[i][j] += left[i][k] * right[k][j];
39              }
40          }
41      }
42      return result;
43  }
44
45  int min(matrix mtrx) {
46      int result = mtrx[0][0];
47      for (int r = 0; r < mtrx.size(); ++r) {
48          for (int c = 0; c < mtrx[r].size(); ++c) {
49              if (mtrx[r][c] < result) {
50                  result = mtrx[r][c];
51              }
52          }
53      }
54      return result;
55  }

```

src/matrix.h

```

1  /**
2   * Parallel programming
3   * Lab 7
4   *
5   * Functions:
6   * F1: C = A - B * (MA * MD)
7   * F2: o = Min(MK * MM)
8   * F3: T = (MS * MZ) * (W + X)
9   *
10  * @since 2015-10-18
11  * @author Olexandr Kovalchuk
12  * @group IP-32
13  */
14
15  #ifndef LAB_MATRIX_H
16  #define LAB_MATRIX_H
17
18  #include <vector>
19  #include <assert.h>
20
21  typedef std::vector<std::vector<int>> matrix;
22

```



```

23 matrix generateMatrix(int size, int filler = 1);
24 matrix generateMatrix(int rows, int columns, int filler = 1);
25
26 matrix operator*(matrix left, matrix right);
27 int min(matrix mtrx);
28
29 #endif // LAB_MATRIX_H
src/cmdopts.h
1 /**
2  * Parallel programming
3  * Lab 7
4  *
5  * Functions:
6  * F1: C = A - B * (MA * MD)
7  * F2: o = Min(MK * MM)
8  * F3: T = (MS * MZ) * (W + X)
9  *
10 * @since 2015-10-18
11 * @author Olexandr Kovalchuk
12 * @group IP-32
13 */
14
15 #ifndef LAB_CMD_OPTS
16 #define LAB_CMD_OPTS
17
18 #include <algorithm>
19 #include <string>
20
21 char *getCmdOption(char **begin, char **end, const std::string &option);
22 bool cmdOptionExists(char **begin, char **end, const std::string &option);
23
24 #endif
src/tasks.h
1 /**
2  * Parallel programming
3  * Lab 7
4  *
5  * Functions:
6  * F1: C = A - B * (MA * MD)
7  * F2: o = Min(MK * MM)
8  * F3: T = (MS * MZ) * (W + X)
9  *
10 * @since 2015-11-11
11 * @author Olexandr Kovalchuk
12 * @group IP-32
13 */
14
15 #ifndef LAB_TASKS_H
16 #define LAB_TASKS_H
17
18 #include <iostream>
19 #include "functions.h"
20 #include <sstream>
21
22 void task1(int size, int tid, int ctid);
23 void task2(int size, int tid, int ctid);
24 void task3(int size, int tid, int ctid);
25
26 #endif

```