

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

Лабораторна робота №4

Win32.

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

Київ - 2015

Лабораторна робота №4. Win32

Мета роботи: вивчення засобів бібліотеки Win32 для роботи з потоками

Мова програмування: C++, використання бібліотеки Win32

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

```
cmdopts.cpp
1  /**
2   * Parallel programming
3   * Lab 4
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 }
28
29 functions.cpp
30 1  /**
31 2   * Parallel programming
32 3   * Lab 4
33 4   *
34 5   * Functions:
35 6   * F1: C = A - B * (MA * MD)
36 7   * F2: o = Min(MK * MM)
37 8   * F3: T = (MS * MZ) * (W + X)
38 9   *
39 10  * @since 2015-10-18
40 11  * @author Olexandr Kovalchuk
41 12  * @group IP-32
42 13  */
43 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 }
lab04.cpp
1  /**
2   * Parallel programming
3   * Lab 4
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 <iostream>
16 #include "functions.h"
17 #include "cmdopts.h"
18 #include "tasks.h"
19
20 int main(int argc, char* argv[]) {
21     std::cout << "lab 04 started" << std::endl;
22     int size, threadCount;
23
24     {
25         char *sizeopt = getCmdOption(argv, argv + argc, "-s");
26         size = sizeopt ? atoi(sizeopt) : 4;
27         char *threadopt = getCmdOption(argv, argv + argc, "-t");
28         threadCount = threadopt ? atoi(threadopt) : 3;
29     }
30
31     std::vector<HANDLE> handles;
32     HANDLE thread_;
33     LPTHREAD_START_ROUTINE func;
34
35     for (int i = 0; i < threadCount; ++i) {
36         switch (i % 3) {
37             case 0:
38                 func = (LPTHREAD_START_ROUTINE) task1;
39                 break;
40             case 1:
41                 func = (LPTHREAD_START_ROUTINE) task2;
42                 break;
43             default: // case 2
44                 func = (LPTHREAD_START_ROUTINE) task3;
45                 break;
46         }
47         thread_ = CreateThread(NULL, 0, func, (LPVOID)&size, CREATE_SUSPENDED, NULL);
48         SetThreadPriority(thread_, THREAD_PRIORITY_NORMAL);
49         handles.push_back(thread_);
50     }
51
52     for (int i = 0; i < handles.size(); ++i) {

```

```

53     ResumeThread(handles[i]);
54 }
55
56 for (int i = 0; i < handles.size(); ++i) {
57     WaitForSingleObject(handles[i], INFINITE);
58     CloseHandle(handles[i]);
59 }
60
61 std::cout << "lab 04 finished" << std::endl;
62     return 0;
63 }
matrix.cpp
1  /**
2   * Parallel programming
3   * Lab 4
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  }
tasks.cpp
1  /**
2   * Parallel programming
3   * Lab 4
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 "tasks.h"
16
17  void task1(LPVOID lpSize) {
18      std::cout << "task 1 in thread " << GetCurrentThreadId() << " started" << std::endl;
19      int size = *((int *)lpSize);
20
21      Sleep(500);
22
23      vector a, b;
24      matrix ma, md;
25
26      a = generateVector(size);
27      b = generateVector(size);
28      ma = generateMatrix(size);
29      md = generateMatrix(size);
30
31      vector result = func1(a, b, ma, md);
32
33      if (size < 8) {
34          std::stringstream ss;
35          ss << "task 1 result: ";
36          ss << '[';
37          for (int i = 0; i < result.size(); ++i) {
38              ss << result[i] << " ";
39          }
40          ss << ']';
41
42          std::cout << ss.str() << std::endl;
43      }
44
45      std::cout << "task 1 in thread " << GetCurrentThreadId() << " finished" << std::endl;
46  }
47
48  void task2(LPVOID lpSize) {
49      std::cout << "task 2 in thread " << GetCurrentThreadId() << " started" << std::endl;
50      int size = *((int *)lpSize);
51
52      Sleep(500);
53
54      matrix mk, mn;
55
56      mk = generateMatrix(size);
57      mn = generateMatrix(size);
58
59      int result = func2(mk, mn);
60
61      if (size < 8) {
62          std::stringstream ss;
63          ss << "task 2 result: " << result;
64          std::cout << ss.str() << std::endl;

```

```

65     }
66
67     std::cout << "task 2 in thread " << GetCurrentThreadId() << " finished" << std::endl;
68 }
69
70 void task3(LPVOID lpSize) {
71     std::cout << "task 3 in thread " << GetCurrentThreadId() << " started" << std::endl;
72     int size = *((int *) lpSize);
73     Sleep(500);
74
75     vector w, x;
76     matrix ms, mz;
77
78     w = generateVector(size);
79     x = generateVector(size);
80     ms = generateMatrix(size);
81     mz = generateMatrix(size);
82
83     vector result = func3(ms, mz, w, x);
84
85     if (size < 8) {
86         std::stringstream ss;
87         ss << "task 3 result: ";
88         ss << '[';
89         for (int i = 0; i < result.size(); ++i) {
90             ss << result[i] << " ";
91         }
92         ss << ']';
93
94         std::cout << ss.str() << std::endl;
95     }
96
97     std::cout << "task 3 in thread " << GetCurrentThreadId() << " finished" << std::endl;
98 }

```

vector.cpp

```

1  /**
2   * Parallel programming
3   * Lab 4
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 "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[j][i] * 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 }
cmdopts.h
1  /**
2   * Parallel programming
3   * Lab 4
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_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

```

```

functions.h
1  /**
2   * Parallel programming
3   * Lab 4
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_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
matrix.h
1  /**
2   * Parallel programming
3   * Lab 4
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_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
tasks.h
1  /**
2   * Parallel programming
3   * Lab 4
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_TASKS_H
16 #define LAB_TASKS_H
17
18 #include <iostream>
19 #include "functions.h"
20 #include <windows.h>
21 #include <sstream>
22
23 void task1(LPVOID lpSize);
24 void task2(LPVOID lpSize);
25 void task3(LPVOID lpSize);
26
27 #endif
vector.h
1  /**
2  * Parallel programming
3  * Lab 4
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

```