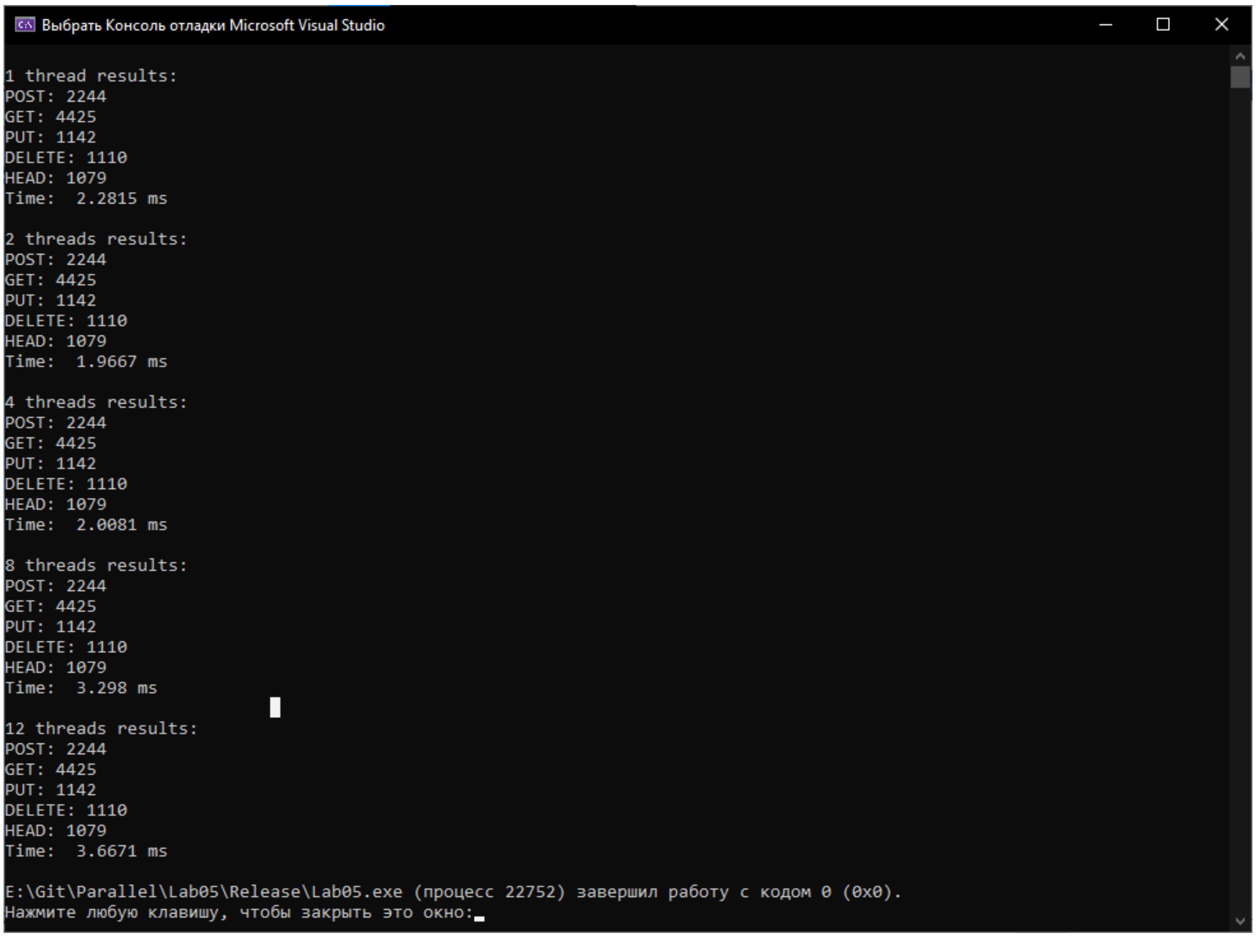

```
Выбрать Консоль отладки Microsoft Visual Studio

1 thread results:
POST: 2244
GET: 4425
PUT: 1142
DELETE: 1110
HEAD: 1079
Time:  2.2815 ms

2 threads results:
POST: 2244
GET: 4425
PUT: 1142
DELETE: 1110
HEAD: 1079
Time:  1.9667 ms

4 threads results:
POST: 2244
GET: 4425
PUT: 1142
DELETE: 1110
HEAD: 1079
Time:  2.0081 ms

8 threads results:
POST: 2244
GET: 4425
PUT: 1142
DELETE: 1110
HEAD: 1079
Time:  3.298 ms

12 threads results:
POST: 2244
GET: 4425
PUT: 1142
DELETE: 1110
HEAD: 1079
Time:  3.6671 ms

E:\Git\Parallel\Lab05\Release\Lab05.exe (процесс 22752) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:
```

```
#define _CRT_SECURE_NO_WARNINGS
#define _CRT_SECURE_NO_DEPRECATE

#include <iostream>
#include <cstdio>
#include <cstring>
#include <unordered_map>
#include <omp.h>
#include <vector>

using namespace std;
const char* pathToFile = ".log";

void readLogFile()
{
  FILE* logFile = fopen(pathToFile, "r");
  if (!logFile)
  {
    cout << "Open file error" << endl;
    return;
  }
  char line[1024];
  unordered_map<string, int> methods = { {"GET", 0}, {"POST", 0}, {"PUT", 0}, {"DELETE", 0}, {"HEAD",
0} };
  vector<string> logLines;
  while (fgets(line, sizeof(line), logFile))
  {
    logLines.push_back(line);
  }
  fclose(logFile);
  double time = omp_get_wtime();
  for (size_t i = 0; i < logLines.size(); ++i)
  {
    const char* method = strstr(logLines[i].c_str(), "GET") ? "GET"
      : strstr(logLines[i].c_str(), "POST") ? "POST"
      : strstr(logLines[i].c_str(), "PUT") ? "PUT"
      : strstr(logLines[i].c_str(), "DELETE") ? "DELETE"
      : strstr(logLines[i].c_str(), "HEAD") ? "HEAD"
      : nullptr;

    if (method)
    {
      methods[method]++;
    }
  }
  cout << endl << 1 << " thread results:" << endl;
  for (const auto& pair : methods)
  {
    cout << pair.first << ": " << pair.second << endl;
  }
```

```
    cout << "Time:  " << (omp_get_wtime() - time)*1000 << " ms" << endl;
}


void readLogFileParallel(int nThreads)
{
    FILE* logFile = fopen(pathToFile, "r");
    if (!logFile)
    {
        cout << "Open file error" << endl;
        return;
    }


    char line[1024];
    unordered_map<string, int> methods = {{"GET", 0}, {"POST", 0}, {"PUT", 0}, {"DELETE", 0}, {"HEAD",
0} };
    vector<string> logLines;
    while (fgets(line, sizeof(line), logFile))
    {
        logLines.push_back(line);
    }
    fclose(logFile);


    double time = omp_get_wtime();
#pragma omp parallel num_threads(nThreads)
    {
        unordered_map<string, int> localMethods = {{"GET", 0},
                                              {"POST", 0},
                                              {"PUT", 0},
                                              {"DELETE", 0},
                                              {"HEAD", 0} };


#pragma omp for
        for (size_t i = 0; i < logLines.size(); ++i)
        {
            const char* method = strstr(logLines[i].c_str(), "GET") ? "GET"
                : strstr(logLines[i].c_str(), "POST") ? "POST"
                : strstr(logLines[i].c_str(), "PUT") ? "PUT"
                : strstr(logLines[i].c_str(), "DELETE") ? "DELETE"
                : strstr(logLines[i].c_str(), "HEAD") ? "HEAD"
                : nullptr;

            if (method)
            {
                localMethods[method]++;
            }
        }

#pragma omp critical
        {
            for (const auto& pair : localMethods)
            {
                methods[pair.first] += pair.second;
            }
        }
    }

    cout << endl << nThreads << " threads results:" << endl;
    for (const auto& pair : methods)
    {
        cout << pair.first << ": " << pair.second << endl;
    }
    cout << "Time:  " << (omp_get_wtime() - time) * 1000 << " ms" << endl;
}

int main()
{
    readLogFile();
    readLogFileParallel(2);
```

```
  readLogFileParallel(4);

  readLogFileParallel(8);

  readLogFileParallel(12);

  return 0;

}
```