1. Скачать и установить Microsoft MPI (оба файла, msmpisetup.exe и msmpisdk.exe).
   https://www.microsoft.com/en-us/download/details.aspx?id=57467

2. Проверить, что все компоненты (а именно: …\Microsoft MPI\Bin, …\Microsoft SDKs\MPI\Include, …\Microsoft SDKs\MPI\Lib\x86 и/или x64. Для этого запустить командную строку от имени администратора и ввести set MSMPI. (Benchmarks может отсутствовать):

```
C:\WINDOWS\system32>set MSMPI
MSMPI_BENCHMARKS=C:\Program Files\Microsoft MPI\Benchmarks\
MSMPI_BIN=C:\Program Files\Microsoft MPI\Bin\
MSMPI_INC=C:\Program Files (x86)\Microsoft SDKs\MPI\Include\
MSMPI_LIB32=C:\Program Files (x86)\Microsoft SDKs\MPI\Lib\x86\
MSMPI_LIB64=C:\Program Files (x86)\Microsoft SDKs\MPI\Lib\x64\
```

3. Создать пустой проект C++ в Visual Studio. Добавить в файлы исходного кода файл с расширением .c или .cpp в зависимости от выбранного языка, C или C++.

4. Открыть свойства проекта. В разделе Configuration Properties -> C/C++ -> General найти пункт Additional Include Directories и добавить в него путь к …\Microsoft SDKs\MPI\Include, а также …\Include\x86 или …\Include\x64 в зависимости от платформы.

5. В разделе Configuration Properties -> Linker -> All Options найти пункт Additional Library Directories и добавить в него путь …\Microsoft SDKs\MPI\Lib\x64 или x86 в зависимости от платформы.

6. Там же найти пункт Additional Dependencies и добавить в него имя библиотеки msmpi.lib.

7. Исходный код параллельного Hello, world на MPI:

```
#include <stdio.h>
#include "mpi.h"

using namespace std;

int main(int *argc, char **argv)
{
    int numtasks, rank;

    MPI_Init(argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);

    printf("Hello MPI from process = %d, total number of processes: %d\n", rank, numtasks);

    MPI_Finalize();

}
```

   Функция MPI_Init инициализирует MPI и сразу создает процессы (все выполняемые после неё команды распараллеливаются на заданное число процессов). MPI_Comm_rank возвращает номер текущего процесса, MPI_Comm_size – общее чсле процессов. MPI_Finalize завершает работу MPI.

8. Сборка проекта происходит как обычно для Visual Studio, но при его запуске его же средствами программа будет запускаться лишь на одном процессе. Для запуска на большем количестве существует два варианта:
   - Запустить командную строку, перейти в путь, где расположен исполняемый файл проекта, созданный при сборке, и запустить его командой:
     mpiexec –n 4 HelloMPI.exe
     где параметр -n задаёт число процессов.

- В Visual Studio перейти в Tools -> External Tools. Добавить новый инструмент и заполнить пустые поля следующим образом:
  - ✓ Title: MPI
  - ✓ Command: C:\Program Files\Microsoft MPI\Bin\mpiexec.exe
  - ✓ Arguments: -n 4 $(TargetName).exe
  - ✓ Initial Directory: $(BinDir)

  После этого можно запускать исполняемый файл из главного меню командой Tools -> MPI. Для изменения количества процессов необходимо отредактировать External Tools, поменяв аргумент -n.

9. Отладку средствами Visual Studio производить можно лишь на одном процессе, отладка многопроцессорных и многопоточных программ сложна. Рекомендуется использовать отладочную печать, указывая в каждом сообщении номер процесса, вызвавшего эту печать. Для гарантированной печати сообщения в случае ошибки приложения после печати можно вызывать fflush(stdout).

10. Для выполнения варианта программы, в котором на группы строк разрезается лишь матрица A, понадобится ещё одна дополнительная функция, помимо описанных в HelloMPI:

    MPI_Barrier(MPI_COMM_WORLD);

    Эта функция используется для синхронизации процессов и гарантирует, что ни один процесс не пойдёт дальше неё, пока все её не исполнят.
    Для второго варианта программы необходимы функции для передачи сообщений, которые будут описаны на лекциях.