**Практическая работа №4**

**Установка и настройка интегрированной среды распределенного программирования C++ и MPICH**

# **Цель работы**

Формирование компетенции установки, тестирования и адаптации существующих пакетов программ на примере IDE Microsoft Visual Studio Visual (Visual С++) и API MPICH.

# **Постановка задачи**

1. Провести обоснованный выбор версии (издания) MPI на основе сопоставления технических характеристик компьютера и версии используемой операционной системы Windows c требованиями, предъявляемыми MPI.

2. Скопировать (загрузить) с официального сайта производителя бесплатно распространяемую версию программного интерфейса передачи сообщений MPI, например, MPICH [2] и произвести ее установку на свой компьютер. Ссылку на официальный сайт привести в списке используемых информационных источников - в обязательном порядке.

3. Создать и настроить проект Microsoft Visual Studio для работы с библиотекой MPI.

4. Написать тестовый проект распределенного приложения для работы с библиотекой MPI, который выводит на экран монитора сообщение «Hello World!» количеством процессов «по умолчанию».

5. Произвести тестовые контрольные прогоны программы с установленным количеством процессов 8, 16 и 64.

6. Определить максимальное количество процессов, которое может быть установлено на компьютере.

7. Сделать выводы по полученным результатам и оформить отчет по выполненной работе.

8. Привести характеристики компьютера и его процессора, на котором производилась установка инструментальной среды (в том числе, в обязательном порядке: производителя, марку, тактовую частоту, количество ядер, наличие технологии hyper treading, наличие сопроцессора GPU или количество и длину векторных регистров).

# **Теоретическое введение**

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Message Passing Interface - программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. Разработан Уильямом Гроуппом, Эвином Ласком (англ.) и другими.

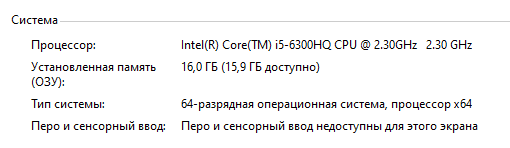
MPI является наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров. Основным средством коммуникации между процессами в MPI является передача сообщений друг другу.

Стандартизацией MPI занимается MPI Forum. В стандарте MPI описан интерфейс передачи сообщений, который должен поддерживаться как на платформе, так и в приложениях пользователя. В настоящее время существует большое количество бесплатных и коммерческих реализаций MPI. Существуют реализации для языков Фортран 77/90, Java, Си и C++.

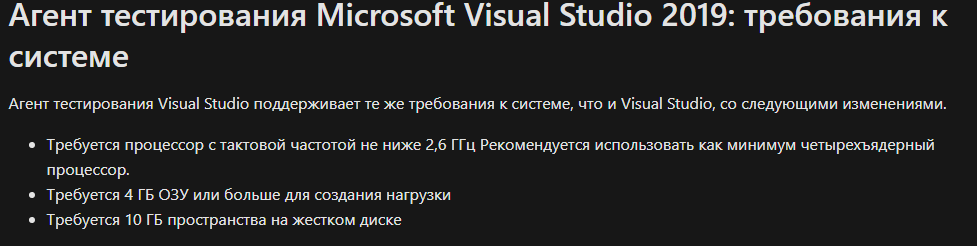
В первую очередь MPI ориентирован на системы с распределенной памятью, то есть, когда затраты на передачу данных велики, в то время как OpenMP ориентирован на системы с общей памятью (многоядерные с общим кэшем). Обе технологии могут использоваться совместно, чтобы оптимально использовать в кластере многоядерные системы.

# **Обоснование выбора**

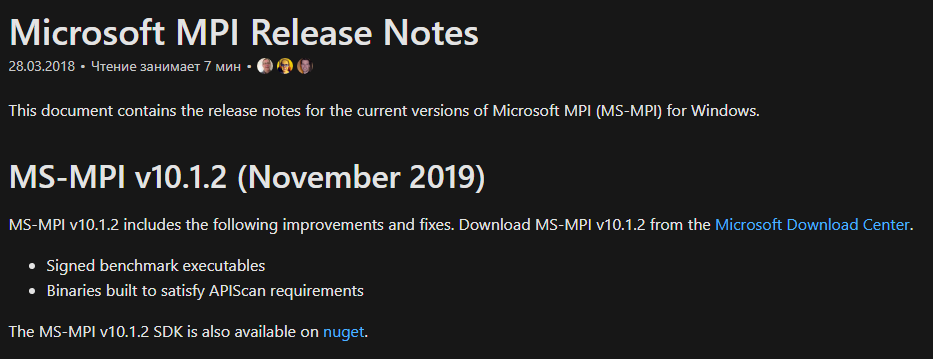
Для выполнения данной практической работы было необходимо провести сопоставление технических характеристик компьютера (рисунок 1), версии Visual Studio (рисунок 2) и рекомендованных систем требований для установки пакета MPI (рисунок 3).



*Рис. 1. Технические характеристики ПК, на котором установлена ОС.*



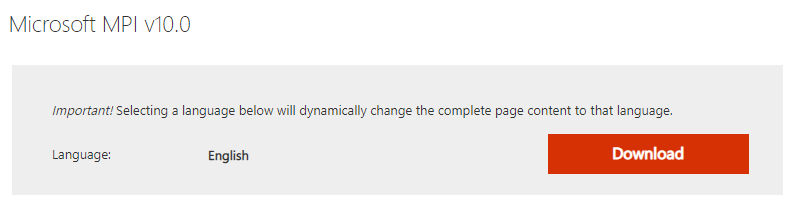
*Рис. 2. Системные требования Microsoft visual studio 2019.*



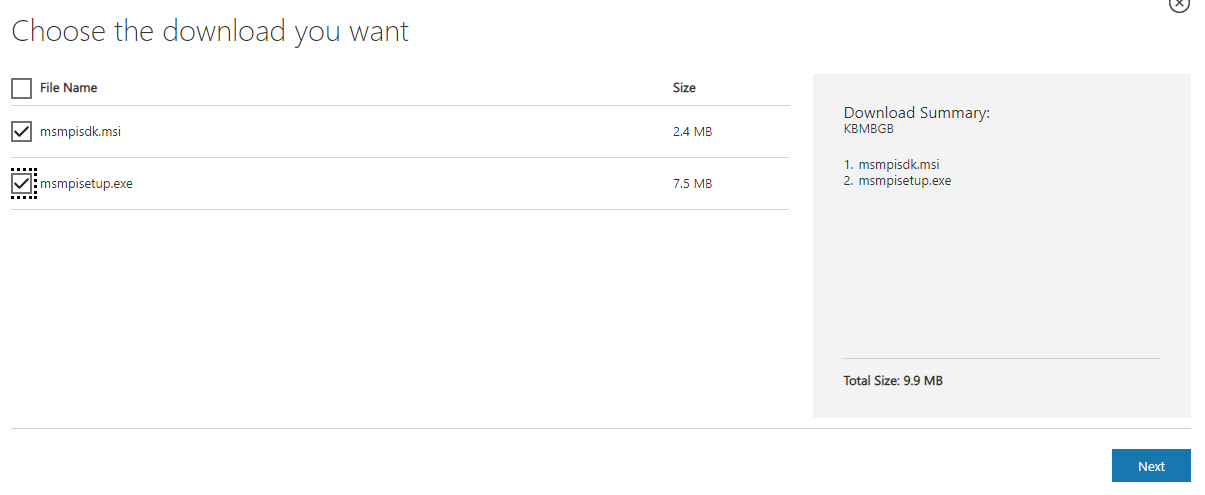
*Рис. 3. Корректная версия пакета MPI с официального сайта Microsoft.*

# **Загрузка пакета MPI**

Для того, чтобы произвести загрузку пакета, необходимо перейти по ссылке, указанной в информационных источниках, после чего нажать на кнопку Download (рисунок 4). После этого необходимо скачать оба файла с расширениями msi, exe (рисунок 5).



*Рис. 4. Официальная страница с пакетом MPI v10.0 на сайте Microsoft.*

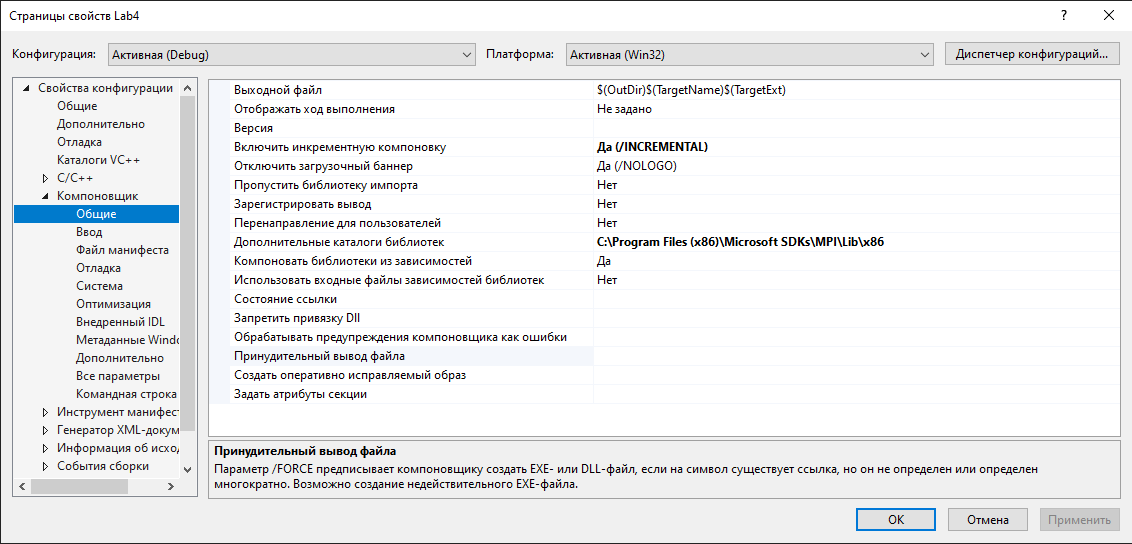


*Рис. 5. Основные файлы пакета MPI, которые необходимо скачать для последующей инсталляции.*

# **Создание и настройка проекта с использованием MPI**

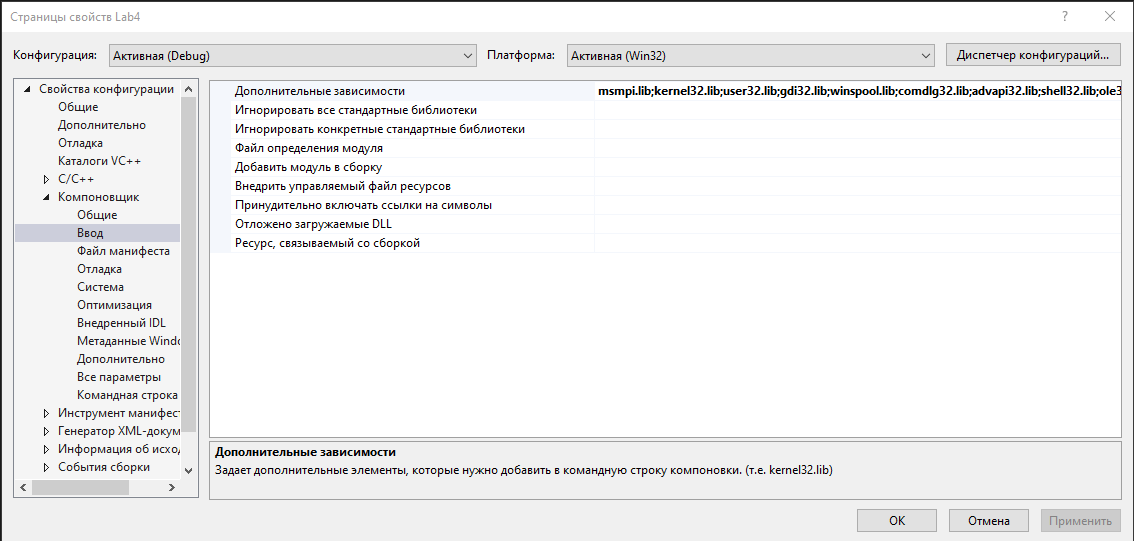
Для создания и настройки проекта в IDE MVS 2019 для работы с пакетом MPI, необходимо провести следующие действия:

1. Провести установку ранее скачанных файлов (msi и exe). Msi необходим для дальнейшего подключения самой библиотеки mpi.h, файл exe – для запуска собранной программы.
2. Перейти в IDE MVS, создать пустой проект на языке программирования C++, добавить главный файл index.cpp.
3. Далее – перейти в раздел «Отладка -> свойство отладки». Слева выбрать вкладку «Компоновщик» и вставить в параметр «Дополнительные параметры библиотек» следующий путь: C:\Program Files (x86)\Microsoft SDKs\MPI\Lib\x86 (рисунок 6).



*Рис. 6. Настройка IDE MVS (1).*

1. Затем – во вкладке «Компоновщик», перейти в раздел ввод, где в поле «Дополнительные зависимости» дописать в начало строки msmpi.lib; (рисунок 7).



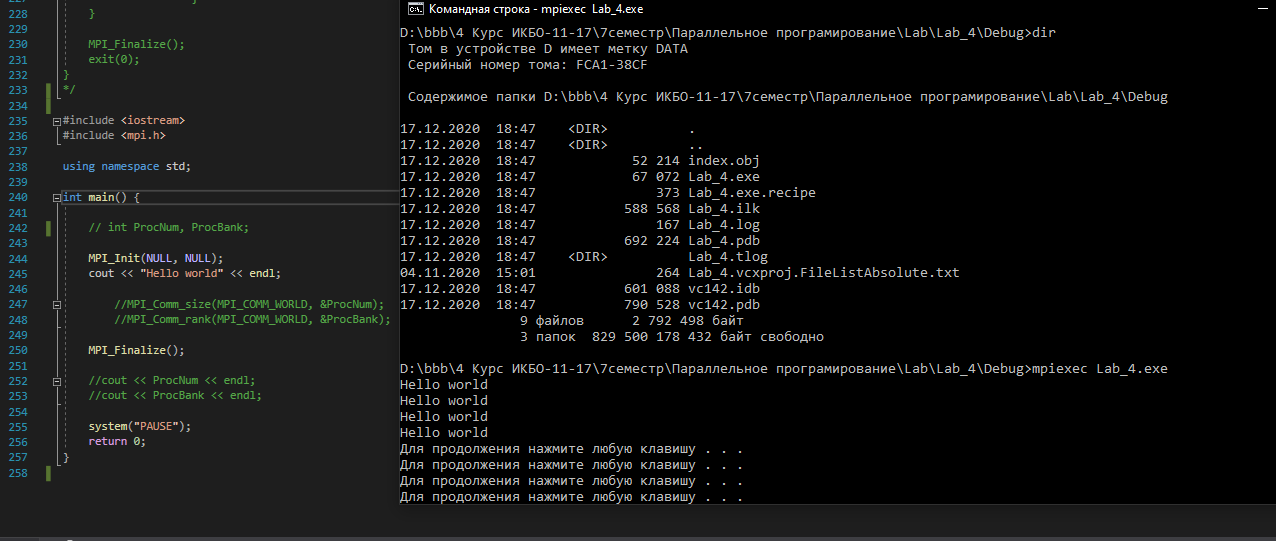
*Рис. 7. Настройка IDE MVS (2).*

1. Последним шагом необходимо написать в index.cpp в начале кода строку #include <omp.h>

# **Написание тестового проекта**

Для написания первой программы необходимо в главной функции программы объявить функции MPI\_Init (NULL, NULL), после которой уже достаточно писать код, после которого в обязательном порядке необходимо прописать MPI\_Finalize (). После этого нужно произвести сборку проекта. Затем – в командной строке перейти в папку со скомпилированным файлом exe и написать команду mpiexec <имя файла.exe>.

Напишем простейшую программу, выводящую в консоль надпись «Hello world» (рисунок 8). Количество процессов по умолчанию – 4 так как в текущем процессоре имеется 4 ядра.

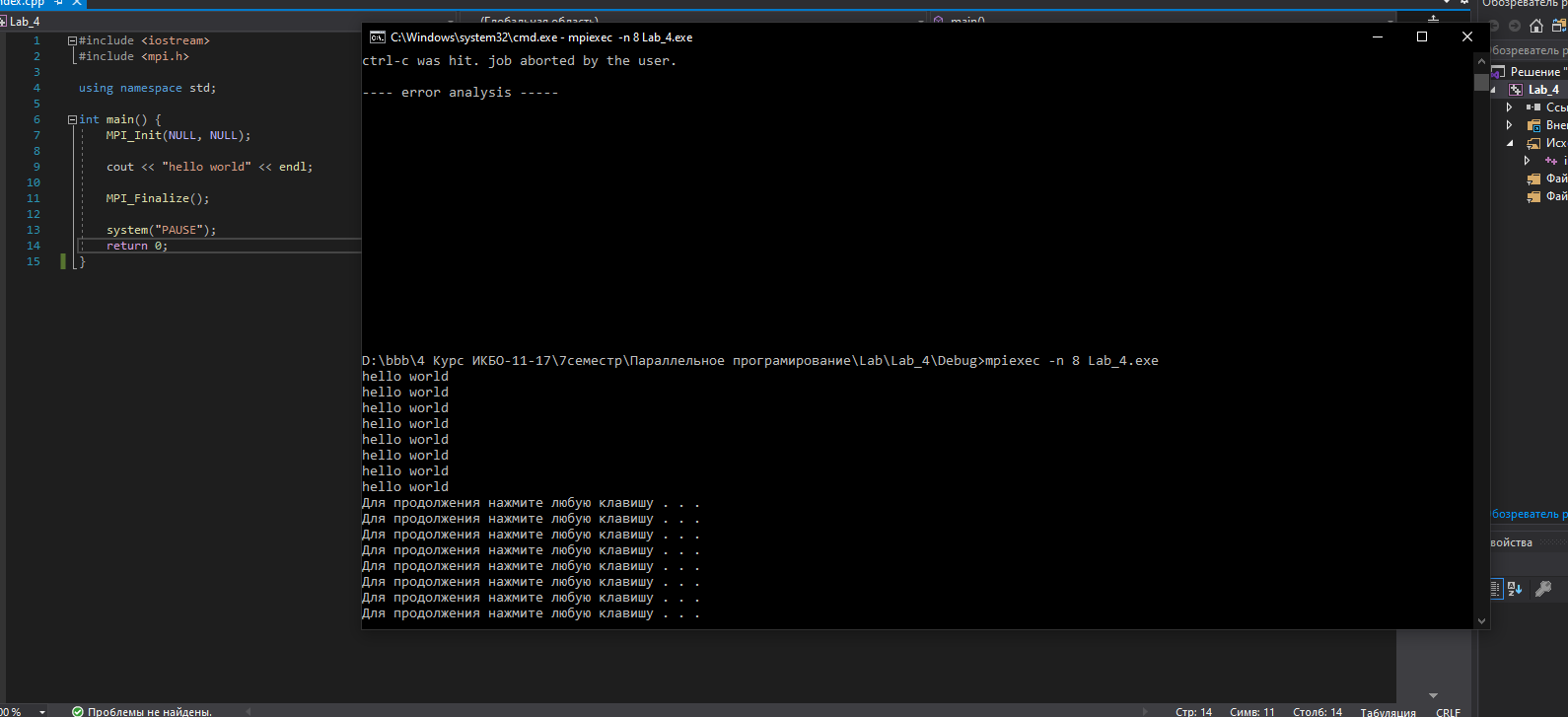


*Рис. 8. Простейшая программа с пакетом MPI, выводящая в консоль «Hello world».*

# **Контрольные прогоны программы**

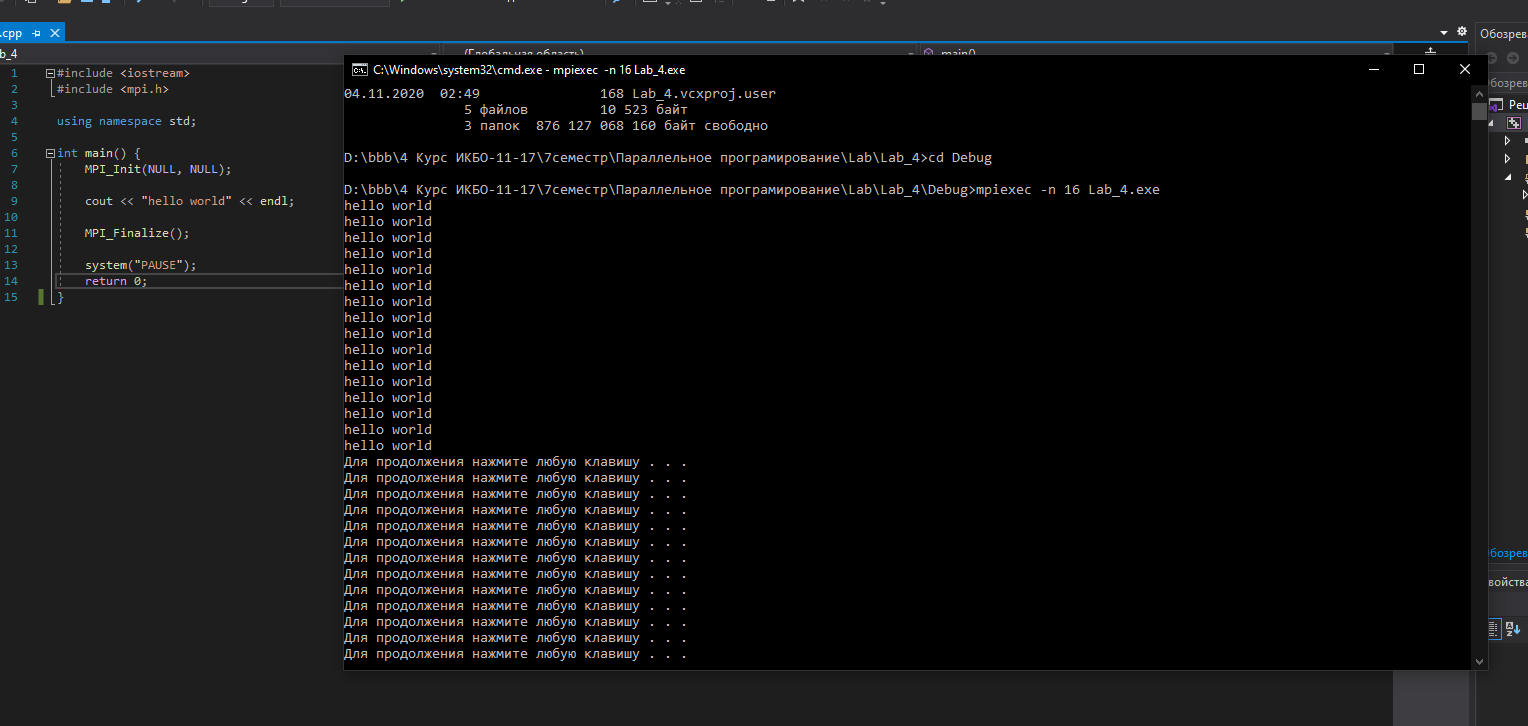
Для того, чтобы запустить программы с различным числом потоков с использованием пакета MPI, достаточно в консоли после сборки проекта прописать следующую команду: mpiexec -n <кол-во процессов> <имя файла.exe>.

Ниже продемонстрирована программа с использованием 8 процессов (рисунок 9).



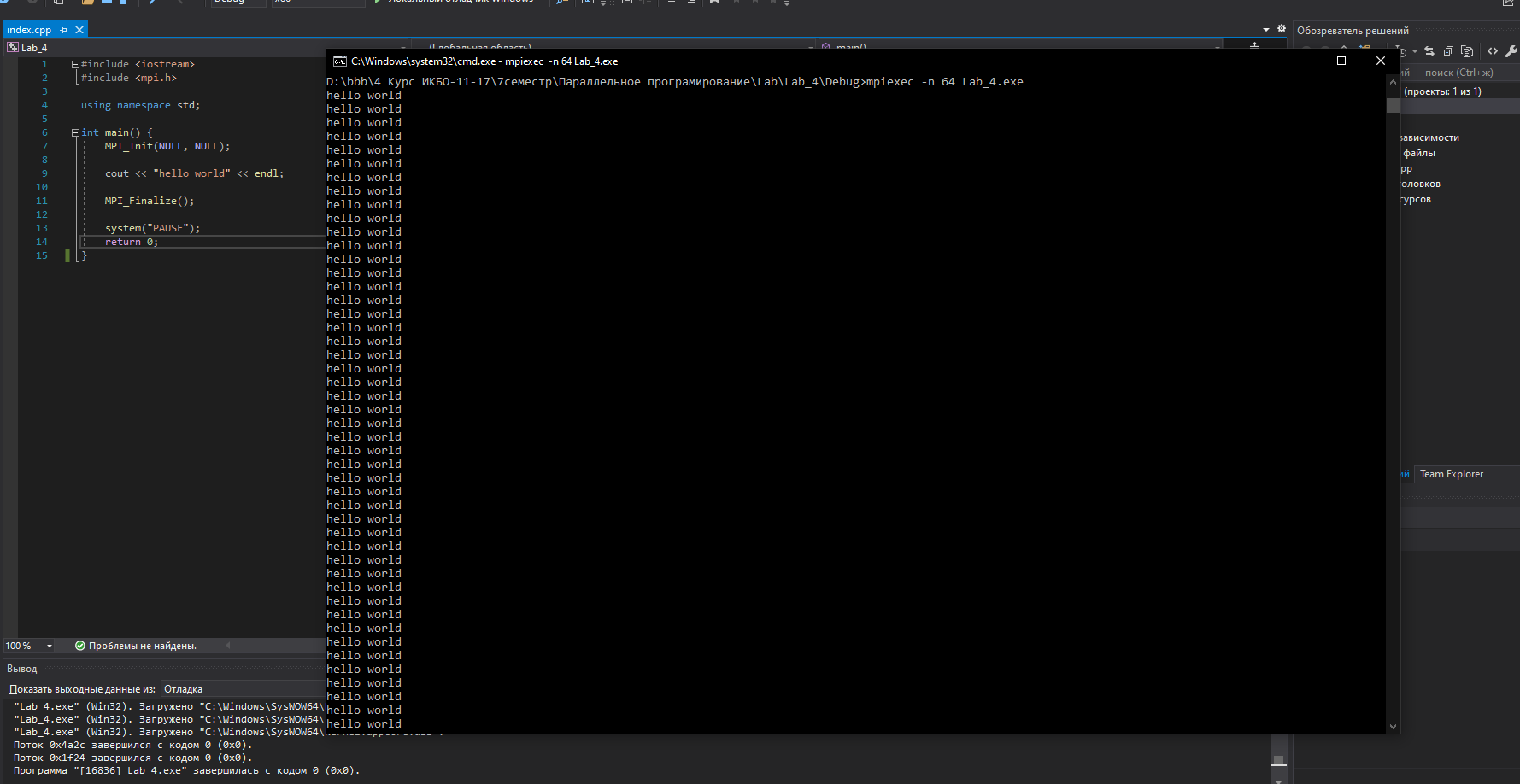
*Рис. 9. Программа, написанная с помощью библиотеки MPI, использующая 8 процессов.*

Ниже продемонстрирована программа с использованием 16 процессов (рисунок 10).



*Рис. 10. Программа, написанная с помощью библиотеки MPI, использующая 16 процессов.*

Ниже продемонстрирована программа с использованием 64 процессов (рисунок 11).



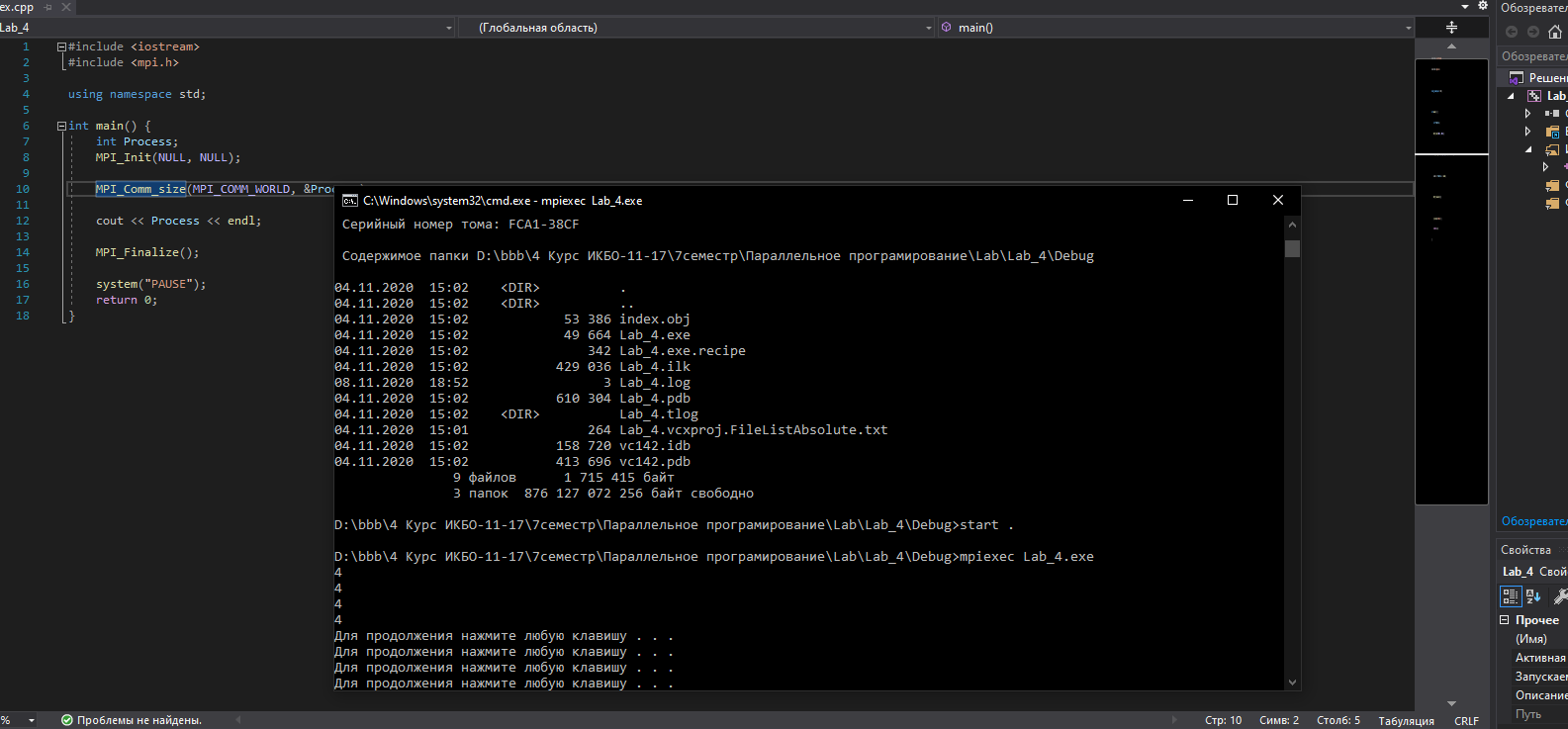
*Рис. 11. Программа, написанная с помощью библиотеки MPI, использующая 64 процесса.*

# **Максимальное количество процессов**

Для вывода максимального количества процессов служит функция int MPI\_Comm\_size(MPI\_Comm comm, int \*size).

• comm — коммуникатор, размер которого определяется,

• size — определяемое количество процессов в коммуникаторе.

*Рис. 12. Максимальное количество процессов.*

# **Выводы**

В ходе выполнения данной практической работы произвел знакомство с пакетом MPI для языка C++, написана простейшая программа при помощи данного пакета, выполнены задачи из раздела «постановка задачи».

# **Список используемых информационных источников**

1. Сыромятников В. П. Курс лекций по дисциплине «Параллельное программирование». – РТУ МИРЭА, 2020-2021 г.

2. Руководство по использованию пакета MPI [Электронный ресурс]. – Режим доступа: <http://appmath.mrsu.ru/assets/templates/appmath/pdf_docs/ParProg_MPI_OpenMP.pdf> (дата обращения 30.10.2020)

3. Официальные характеристики процессора Intel Core i5 4590 [Электронный ресурс]. – Режим доступа: <https://ark.intel.com/content/www/ru/ru/ark/products/80815/intel-core-i5-4590-processor-6m-cache-up-to-3-70-ghz.html> (дата обращения 30.10.2020)

4. Системные требования Microsoft Visual Studio [Электронный ресурс]. -  Режим доступа: <https://docs.microsoft.com/ru- ru/visualstudio/releases/2019/system-requirements> (дата обращения 30.10.2020)

5. Ссылка на загрузку официального пакета MPI для MVS 2019 [Электронный ресурс]. – Режим доступа: <https://www.microsoft.com/en-us/download/details.aspx?id=57467> (дата обращения 30.10.2020)