2. Технология программирования MPI

Задание 10. Создание проекта в среде MS Visual Studio с поддержкой MPI

Создайте проект в среде Visual Studio 2010 с поддержкой MPI.

Задание 11. Программа «I am!»

Напишите программу, в которой каждый процесс выводит на экран свой номер и общее количество процессов в приложении в формате:
`I am <Номер процесса> process from <Количество процессов> processes!`

Входные данные: нет.

Выходные данные: строки в формате «`I am <Номер процесса> process from <Количество процессов> processes!`».

Пример входных и выходных данных

Входные данные	Выходные данные
3	I am 0 process from 3 processes! I am 1 process from 3 processes! I am 2 process from 3 processes!

Задание 12. Программа «На первый-второй рассчитайся!»

Напишите программу, в которой каждый процесс с четным номером выводит на экран строку «`I am <Номер процесса>: FIRST!`», а каждый процесс с нечетным номером – «`I am <Номер процесса>: SECOND!`». Процесс с номером 0 должен вывести на экран общее количество процессов в приложении в формате «`<Количество процессов> processes.`».

Входные данные: нет.

Выходные данные: строки в формате «`I am <Номер процесса>: FIRST!`» или «`I am <Номер процесса>: SECOND!`» или «`<Количество процессов> processes.`».

Пример входных и выходных данных

Входные данные	Выходные данные
4	4 processes. I am 1 process: FIRST! I am 2 process: SECOND! I am 3 process: FIRST!

Указания к заданию 10. Создание проекта в среде MS Visual Studio с поддержкой MPI

1. Создайте проект mpi в Microsoft Visual Studio 2010 с минимальным кодом:

```
int main() {
  return 0;
}
```

2. ***Для включения поддержки MPI*** установите дополнительные параметры компиляции проекта:

   - В главном меню выберите *Project-> Имя_проекта Properties*
   - В открывшемся окне выберите *Configuration Properties / C/C++ / General*. Установите значение параметра *Additional Include Directories* в значение «**C:\Program Files\MPICH2\include**». См. Рис.1.

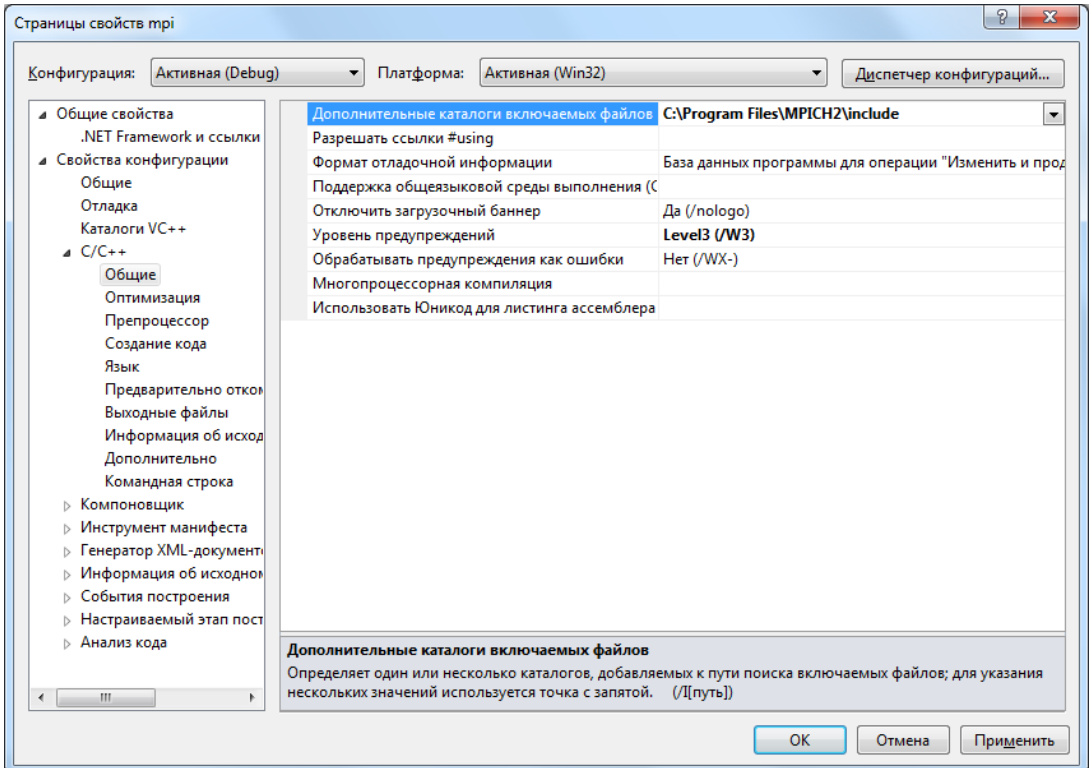

Рис. 1

- Раскройте *Configuration Properties / Linker / General*. Установите значение параметра *Additional Library Directories* в значение «**C:\Program Files\MPICH2\lib**». См. Рис.2.

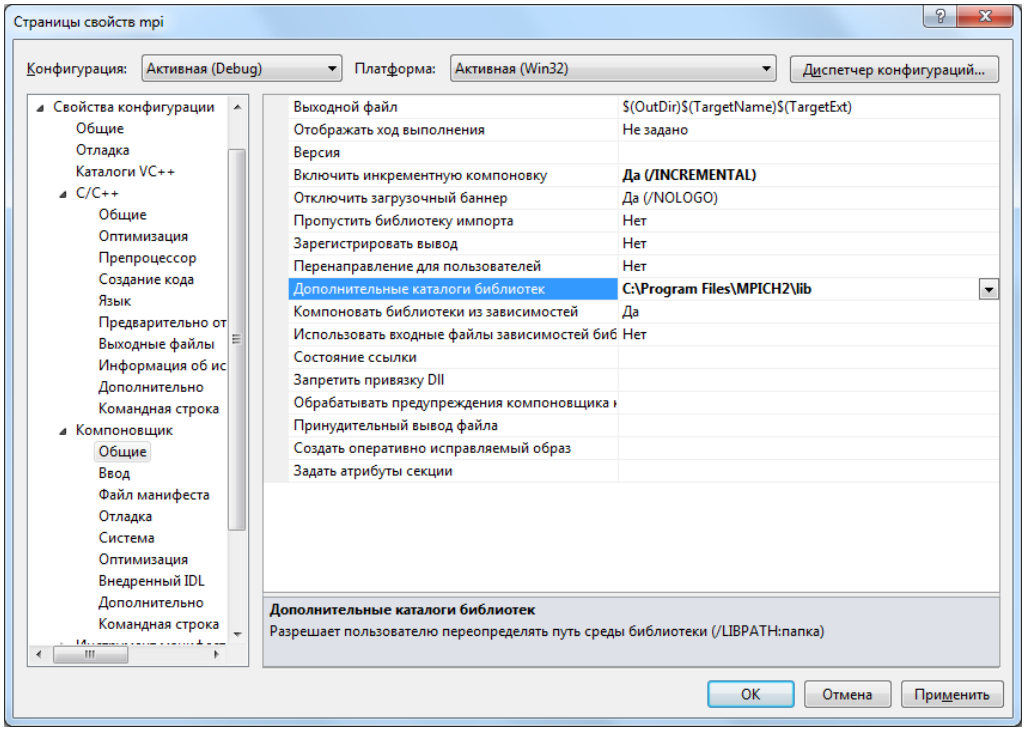

Рис. 2

- Раскройте вкладку *Configuration properties / Linker / Input*. Добаьте к значению параметра *Additional Dependencies*: «**mpi.lib**». См. Рис.3.

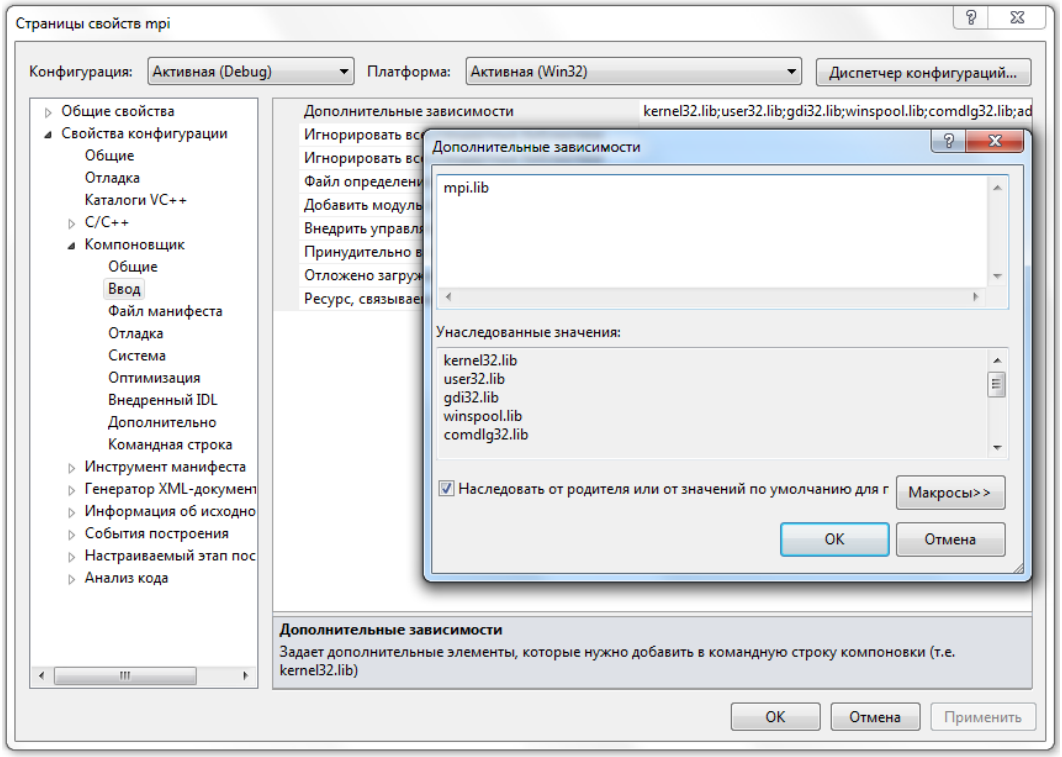

Рис. 3

- ***Для компиляции*** приложения нажмите F7.
- ***Для запуска*** приложения создайте файл run.bat в директории, в которой находится ваше скомпилированное приложение, следующего содержания:

```
"C:\Program Files\MPICH2\bin\mpiexec.exe" -np 2 -noprompt
mpi.exe
PAUSE
```

где
`-np` – параметр задающий количество процессов в приложении,
`-noprompt` – параметр для отмены запроса регистрации,
`mpi.exe` – имя вашего приложения.

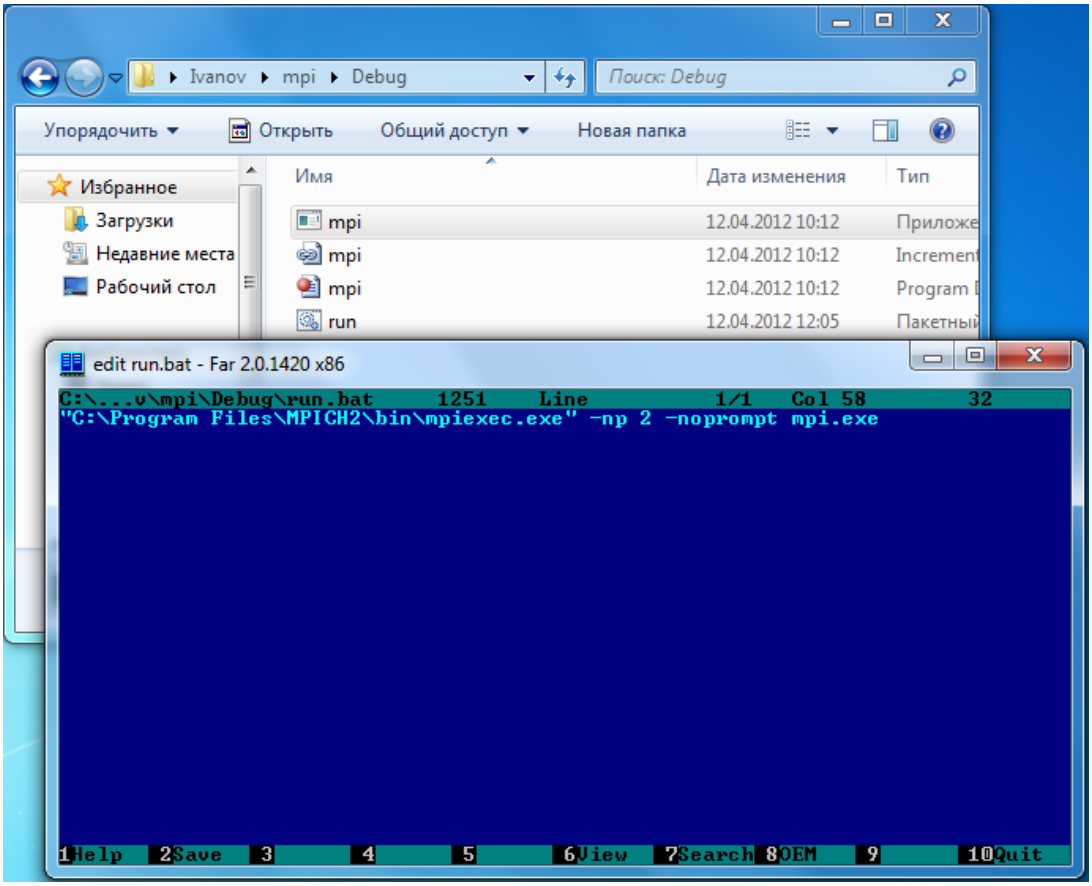

Рис. 4

- Пройдите авторизацию для запуска mpich. Для это перейдите в каталог C:\Program Files\MPICH2\bin\, запустите wmpiregister.exe. В открывшемся окне введите в поле Account ваш логин для входа на компьютеры в учебном классе:

  class\ваш_логин

  и в поле password ваш пароль. См. Рис. 5. Нажмите кнопку Register, убедитесь, что вывелось сообщение «Password encrypted into the Registry.» Нажмите кнопку OK.

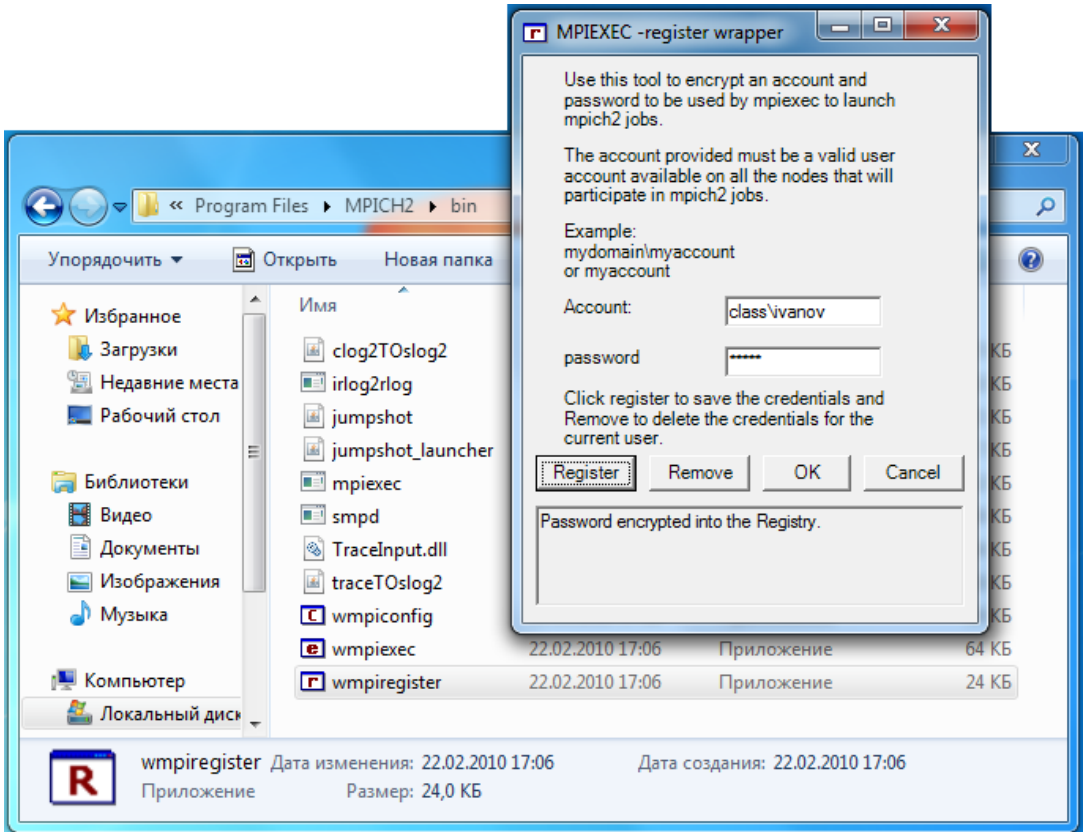

Рис. 5

- Теперь можно запускать созданный вами ранее bat-файл.

Указания к заданию 11. Программа «I am!»

1. Создайте проект mpi_i_am в Microsoft Visual Studio 2010 с поддержкой MPI (см. указания к заданию 10).
2. Подключите заголовочный файл mpi.h с функциями MPI. Строка подключения заголовочного файла:

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

3. Инициализируйте библиотеку MPI. Для этого в функции main вызовите MPI-функцию ***MPI_Init***:

   ```
   MPI_Init(&argc, &argv);
   ```

   Реальная инициализация для каждого приложения выполняется не более одного раза, а если MPI уже был инициализирован, то никакие действия не выполняются и происходит немедленный возврат из подпрограммы. **Все остальные MPI-функции могут быть вызваны только после вызова MPI_Init!**

В качестве параметров MPI_Init требует параметры командной строки, которые ваша программа получает через параметры функции main argc и argv. Добавьте параметры argc и argv в функцию main:

```
int main(int argc, char *argv[]){
  //...
}
```

4. Определите номер процесса в приложении с помощью функции ***MPI_Comm_rank***:

```
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

где

первый параметр указывает коммуникатор, в котором определяется номер текущего процесса. `MPI_COMM_WORLD` – коммуникатор, объединяющий все процессы в MPI-приложении. Создается по умолчанию.

`rank` – целочисленная переменная, в которой функция возвращает номер текущего процесса.

5. Определите количество процессов в приложении с помощью функции ***MPI_Comm_size***:

```
int size;
MPI_Comm_size(MPI_COMM_WORLD, &size);
```

где

первый параметр указывает коммуникатор, в котором определяется количество процессов.

`size` – целочисленная переменная, в которой функция возвращает количество процессов.

6. Вызовите команду вывода строки «`I am <Номер процесса> process from <Количество процессов> processes!`»:

```
printf("I am %d process from %d processes!\n", rank, size);
```

7. В конце программы вызовите функцию MPI_Finalize:

```
MPI_Finalize();
```

MPI_Finalize – завершение параллельной части приложения. Все последующие обращения к любым MPI-процедурам, в том числе к MPI_Init, запрещены. К моменту вызова MPI_Finalize процессом все действия, требующие его участия в обмене сообщениями, должны быть завершены.

8. Скомпилируйте и запустите ваше приложение. Убедитесь, что на экран выводится верный результат.

Указания к заданию 12. Программа «На первый-второй рассчитайся!»

1. Создайте проект mpi_fist_second в Microsoft Visual Studio 2010 с поддерж-кой MPI (см. указания к заданию 10).
2. Подключите заголовочный файл mpi.h.
3. Инициализируйте библиотеку MPI с помощью функции MPI_Init.
4. Определите номер процесса в приложении с помощью функции MPI_Comm_rank.
5. С помощью оператора switch либо if определите три случая:
    - номер процесса равен 0, тогда в качестве операторов напишите определение количества процессов в приложении с помощью функции MPI_Comm_size и вывод на экран строки «`<Количество процессов> processes.`»
    - номер процесса является *четным* числом, т.е. оно не 0 и делится на два без остатка (`rank%2==0`) – выведите строку «`I am <Номер процесса>: SECOND!`».
    - номер процесса является *нечетным* числом, т.е. оно не делится на два без остатка (`rank%2!=0`) – выведите строку «`I am <Номер процесса>: FIST!`».
6. Завершите MPI-приложение функцией MPI_Finalize.
7. Скомпилируйте и запустите ваше приложение. Убедитесь, что на экран выводится верный результат.