OpenMP

Параллельные потоки

OpenMP

- Open Multi-Processing
- Распараллеливание программ
  - Fortran
  - C/C++
- Многопоточные программы
- Многоядерные системы с общей памятью

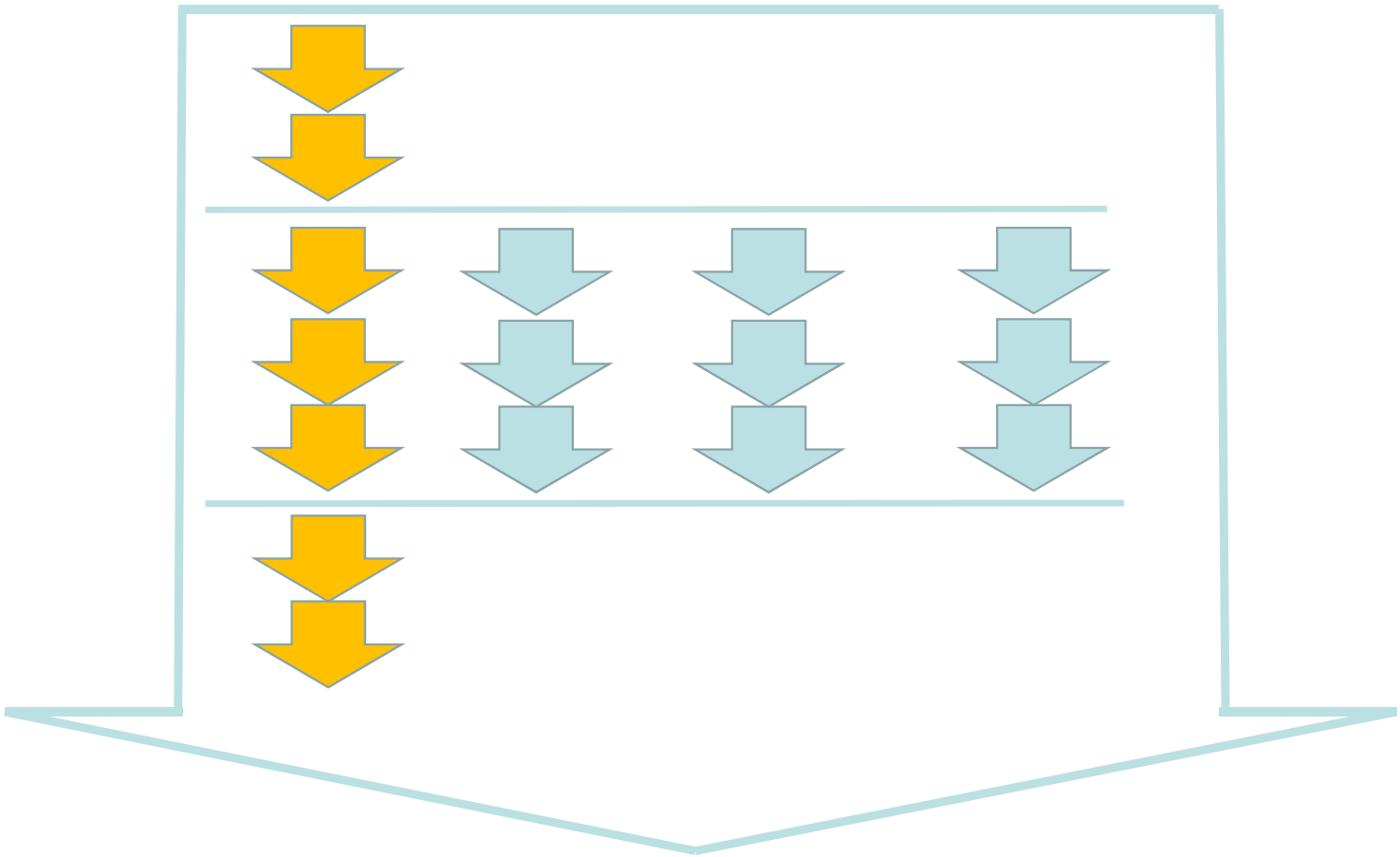
Классификация ЭВМ

- Организация параллелизма
  - SISD Single Instruction Single Data
  - SIMD Single Instruction Multiple Data
  - MISD Multiple Instruction Single Data
  - MIMD Multiple Instruction Multiple Data

Параллелизм OpenMP

- SIMD Single Instruction Multiple Data
  - ***Последовательная часть*** программы
    - Один поток
  - ***Параллельная часть*** программы
    - Несколько потоков
    - Разные наборы данных

Модель fork - join



Определения: Потоки

- thread
  - execution entity having serial flow of control
- master thread
  - thread that creates a team when a parallel region is entered
- team
  - one or more threads cooperating in the execution of a construct

Определения: Разделы программы

- parallel region
  - may be executed by multiple threads
- serial region
  - statements executed only by master thread

Определения: Доступ к переменным

- private
  - block of storage unique to thread
- shared
  - threads in team will access this single block of storage

Ресурсы

- Информационно-аналитический центр по параллельным вычислениям
  - parallel.ru
    - Учебники + примеры программ
- Национальный Открытый Университет ИНТУИТ
  - intuit.ru
    - Курсы по параллельным алгоритмам и OpenMP
- OpenMP Architecture Review Board
  - openmp.org
    - Спецификация OpenMP API
    - Tutorial Videos

Литература

- **Антонов** 2009 Параллельное программирование с использованием технологии OpenMP
  - Учебник
    - https://parallel.ru/info/parallel/openmp
  - Примеры программ
    - https://parallel.ru/tech/tech_dev/OpenMP/examples/
- **Прохоренок** 2010 Программирование на C++ в Visual Studio 2010 Express

OpenMP Specifications

http://openmp.org/wp/openmp-specifications/

- Version 4.0 (October 2013)
- Version 3.1 (September 2011)
- Version 3.0 (November 2008)
- Version 2.5 (May 2005)
- Version 2.0 (March 2002)
- Version 1.0 (October 1998)

OpenMP API 2.0 Specification

- OpenMP implementations are not required to check for
  - Dependencies
  - Conflicts
  - Deadlocks
  - Race conditions

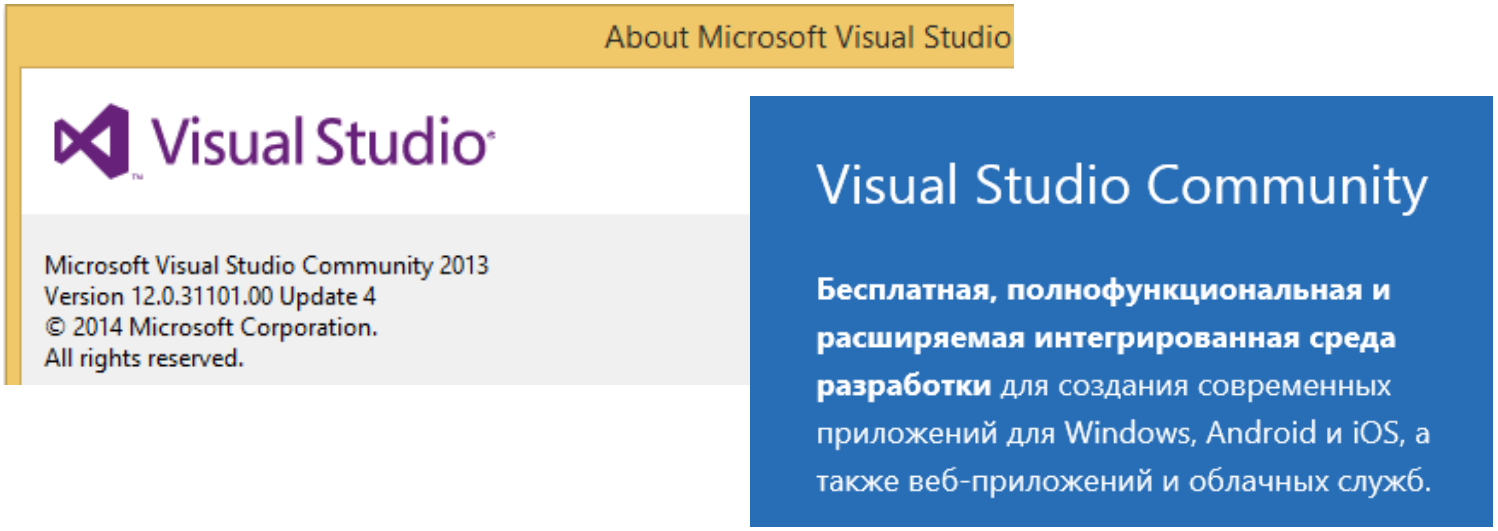
  resulting in incorrect program execution
- The user is responsible for ensuring that the application using the OpenMP C and C++ API constructs executes correctly.

Инструменты OpenMP

- Динамическое создание потоков
- Директивы **`#pragma omp`**
  - создания потоков
    - parallel
  - распределения работы между потоками
    - DO/for, section
  - определения классов переменных
    - shared, private
  - синхронизации потоков
    - critical, atomic, barrier
- Процедуры
  - omp_get_thread_num
- Переменные окружения
  - OMP_NUM_THREADS

Visual Studio

- Microsoft Visual Studio
  - Community Edition
    - https://www.visualstudio.com/



Предоставляется бесплатно для студентов, участников создания открытого исходного кода и небольших групп

Visual Studio 2015

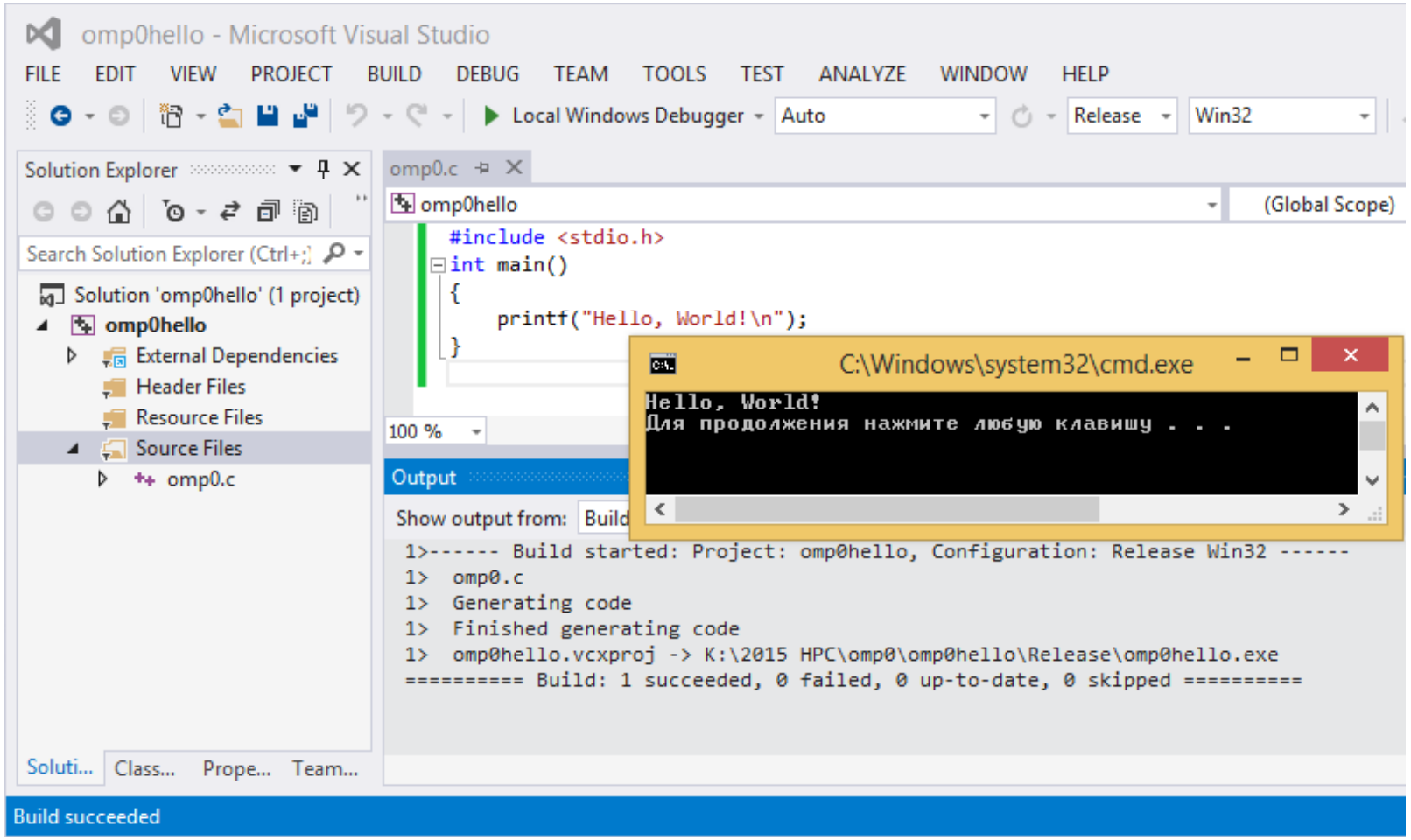
- По умолчанию компилятор C/C++ не устанавливается
  - При установке выбрать пользовательские настройки

    либо
  - Установить дополнительно
- Версия Community Edition поддерживает OpenMP

Hello, World!

```
#include <stdio.h>
void main()
{
    printf("Hello, World!\n");
}
```

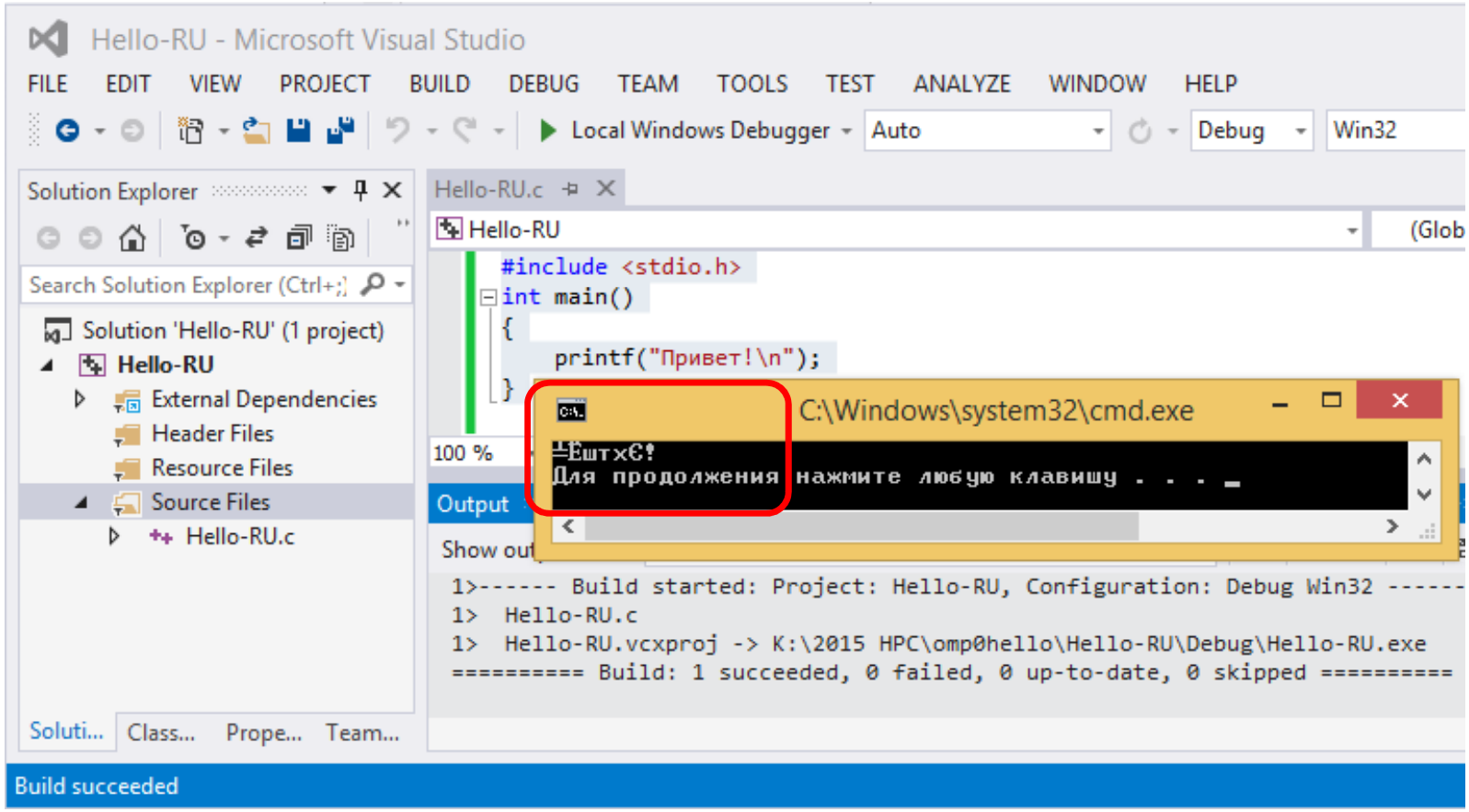
Hello, World!


omp0hello - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Release
Win32
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'omp0hello' (1 project)
omp0hello
External Dependencies
Header Files
Resource Files
Source Files
omp0.c
omp0.c
omp0hello
(Global Scope)
#include <stdio.h>
int main()
{
printf("Hello, World!\n");
}
100 %
C:\Windows\system32\cmd.exe
Hello, World!
Для продолжения нажмите любую клавишу . . .
Output
Show output from:
1>------ Build started: Project: omp0hello, Configuration: Release Win32 ------
1> omp0.c
1> Generating code
1> Finished generating code
1> omp0hello.vcxproj -> K:\2015 HPC\omp0\omp0hello\Release\omp0hello.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
Soluti...
Class...
Prope...
Team...
Build succeeded

Проблема локализации

```
#include <stdio.h>
int main()
{
    printf("Привет!\n");
}
```

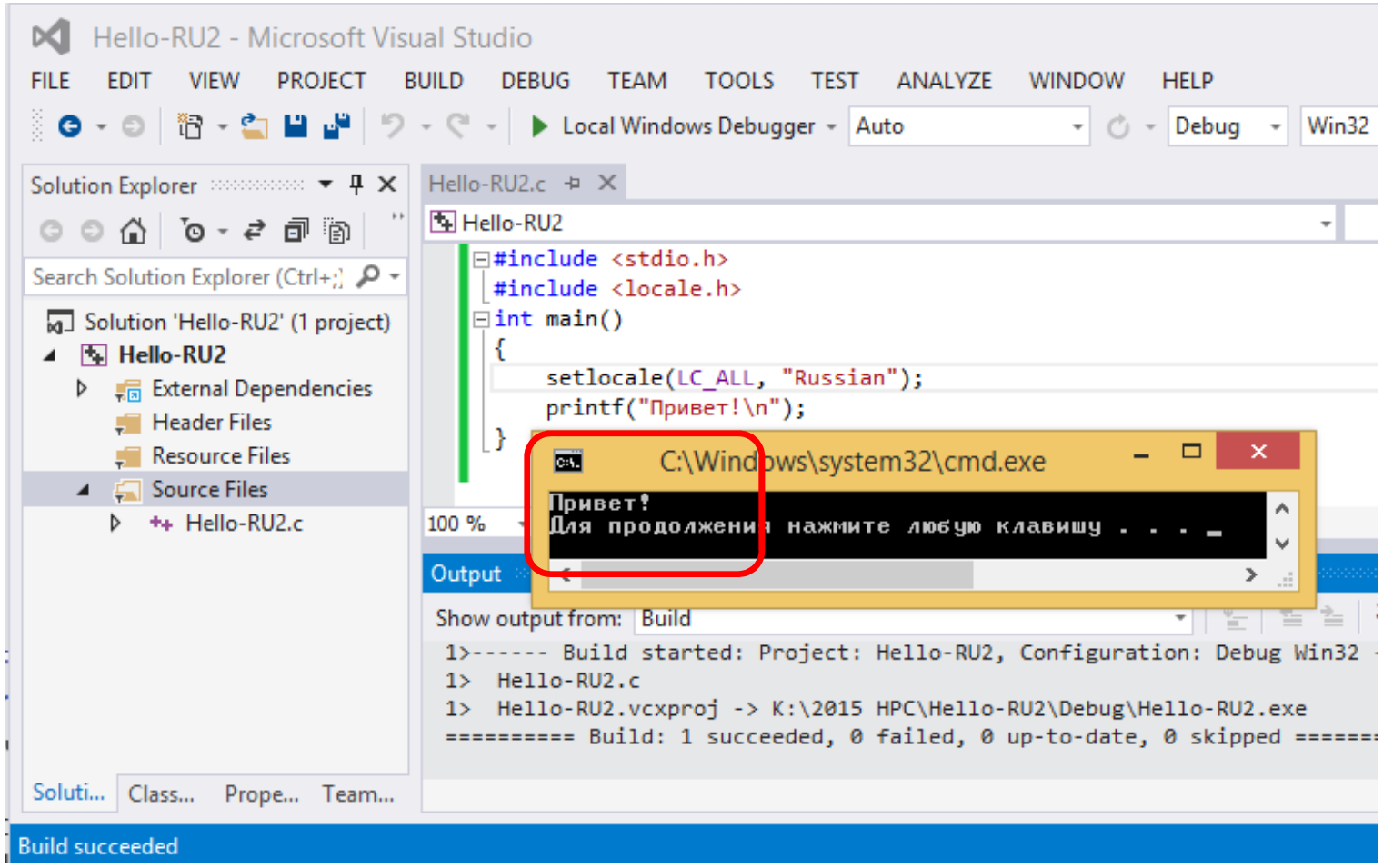
Проблема локализации



Локализация консоли

```
#include <stdio.h>
#include <locale.h>
int main()
{
setlocale(LC_ALL, "Russian");
printf("Привет!\n");
}
```

Локализация консоли



Поддержка OpenMP

```
#include <stdio.h>
int main()
{
#ifdef _OPENMP
  printf("OpenMP is supported! %d\n", _OPENMP);
#else
  printf("OpenMP is not supported!\n");
#endif
}
```

#ifdef identifier

```
#ifdef identifier
  . . .
#endif
```

- Директива препроцессора (“if defined”)
- Сохраняет группу строк, если указанный идентификатор определен директивой **#define**

Директива **#define**

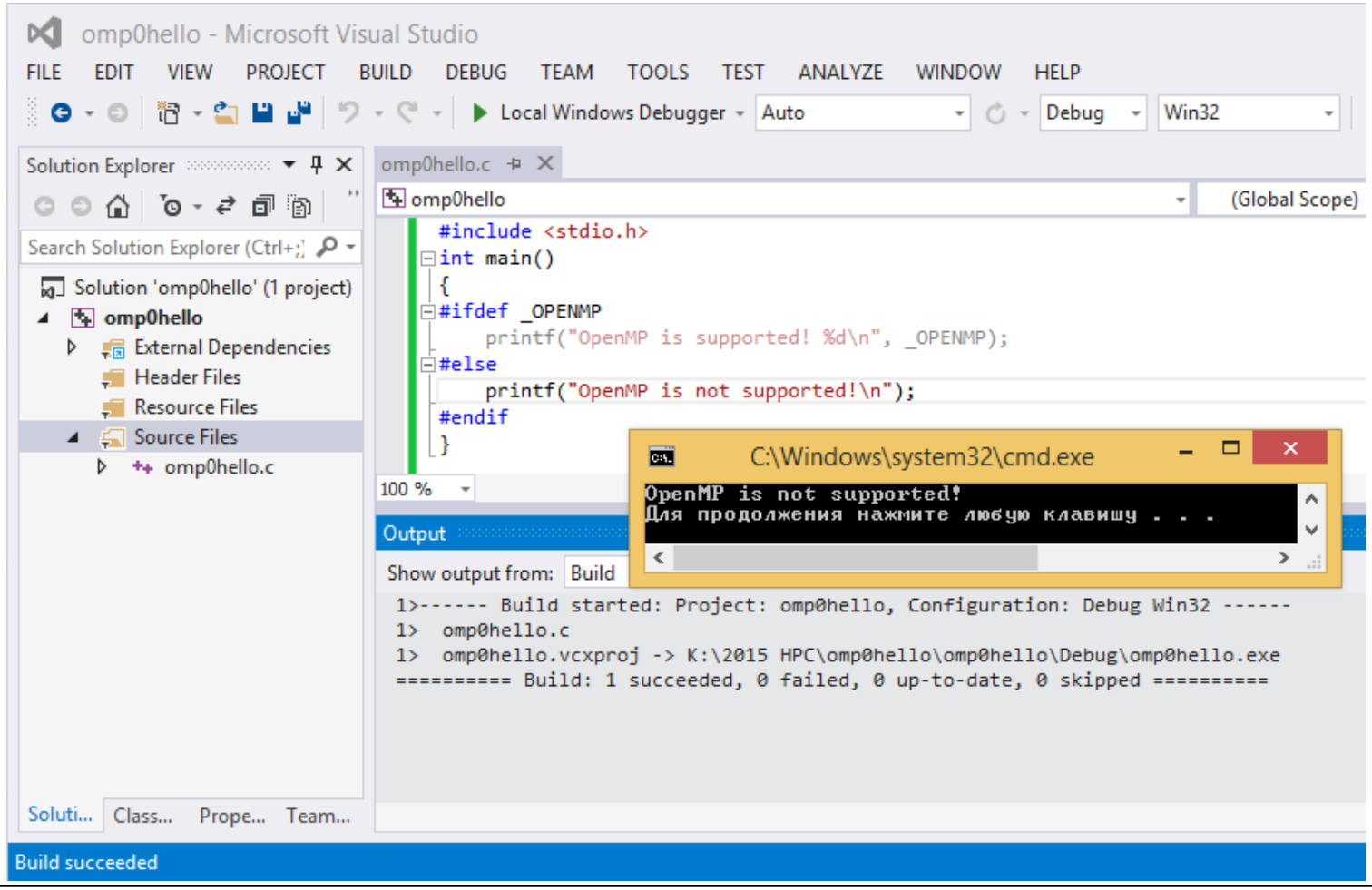
Поиск и замена строки в тексте программы

```
#define NUM 1000
x = NUM * 20;
```

Каждое **NUM** заменяется на **1000**

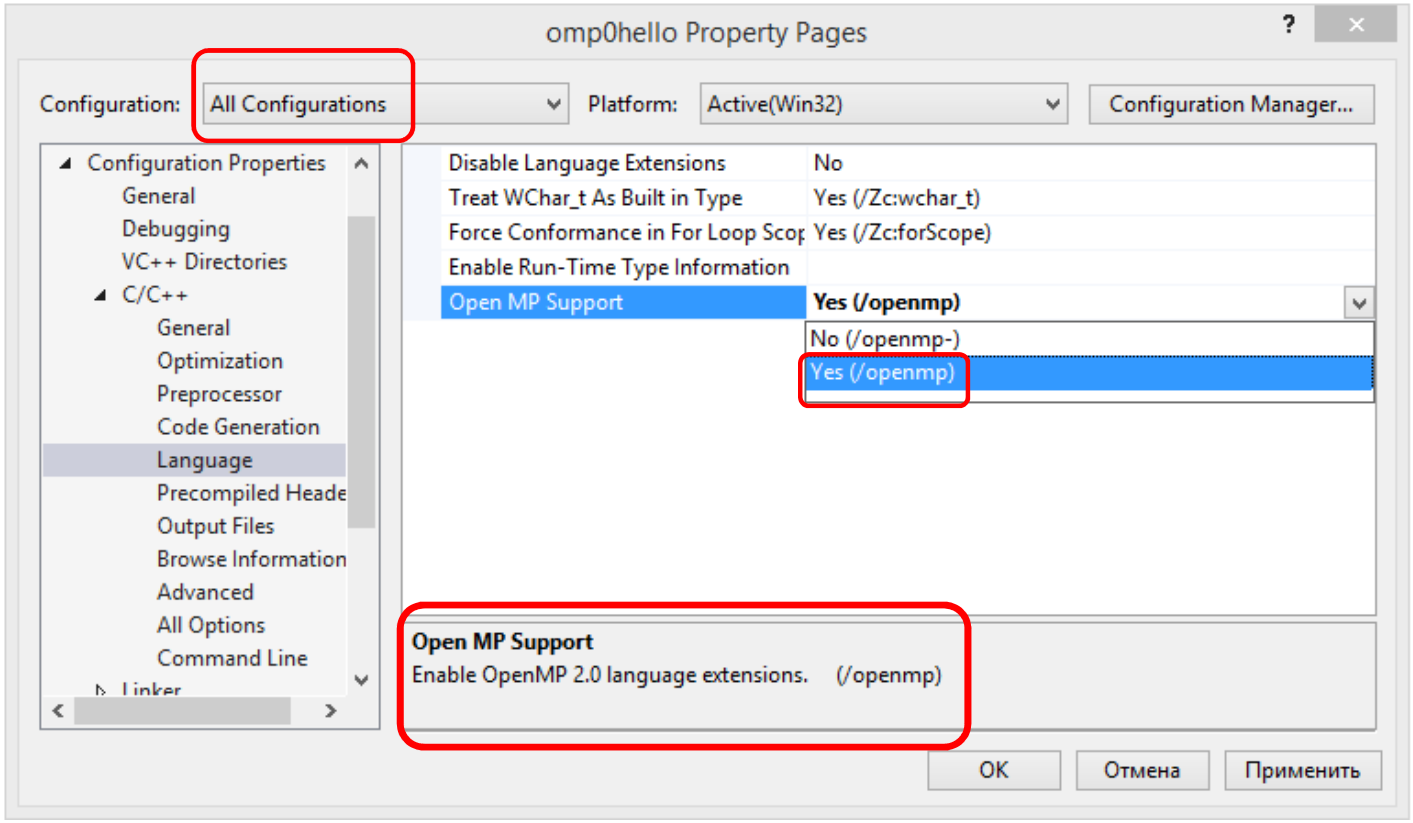
```
x = 1000 * 20;
```

Без поддержки OpenMP

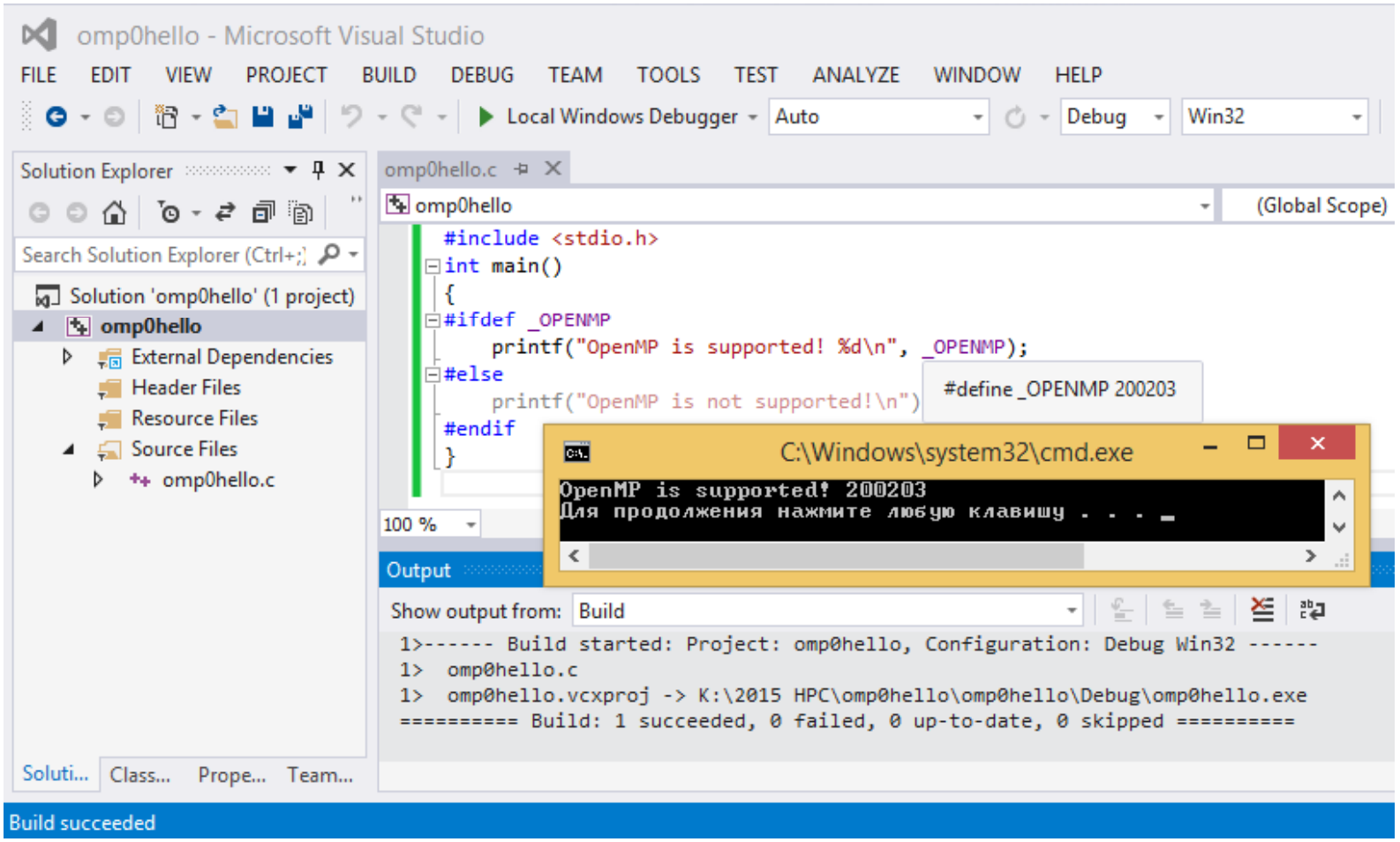


OpenMP Support

- Project → Properties → All Configurations → Configuration Properties → C/C++ → Language → OpenMP Support → Yes



Поддержка OpenMP



_OPENMP

`yyyymm` – дата принятия стандарта

`200203` = March 2002

OpenMP Specifications

OpenMP 4.0 Specifications

- **OpenMP 4.0** Complete Specifications (July 2013) (PDF)

Earlier Specifications

- **Version 2.5** - (May 2005, combined C/C++ and Fortran)
- **C/C++ version 2.0** - (March 2002)
- **C/C++ version 2.0** *with change bars reflecting changes from 1.0* - (March 2002)
- **FORTRAN version 2.0** - (November 2000)
- **FORTRAN version 2.0** *with change bars reflecting changes from 1.1* (November 2000)
- **C/C++ version 1.0** - (October 1998)
- **FORTRAN version 1.1** - (November 1999 - incorporates April 1999 Interpretations and Errata)
- **FORTRAN version 1.0** - (October 1997)

Источник: openmp.org

`#pragma omp parallel`

- Директива

  `#pragma`

- Конструкция

  `#pragma omp parallel`

  – Параллельная область программы
    - Исполняется несколькими потоками параллельно
    - После завершения параллельной области продолжает выполняться только мастер-поток

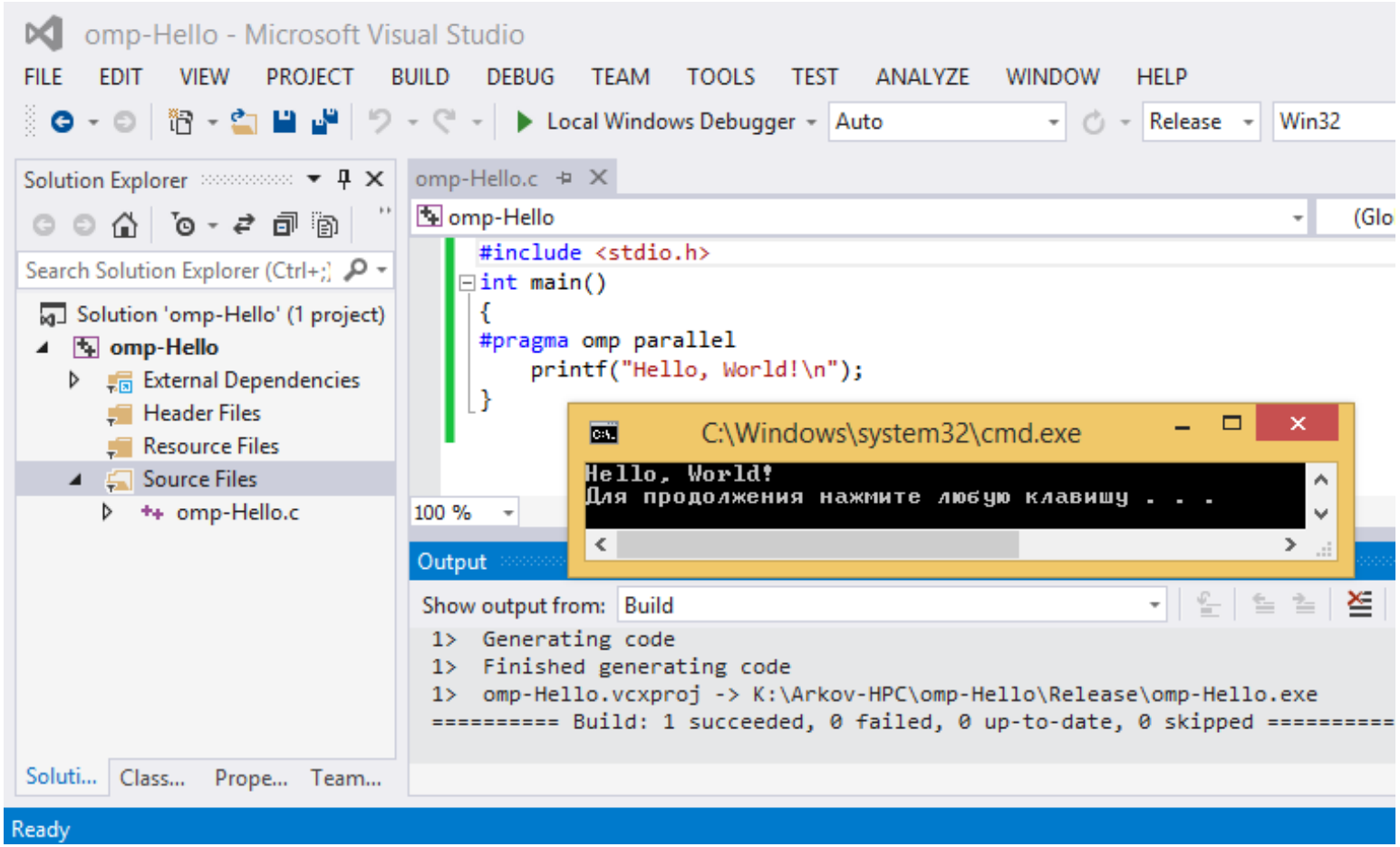
Компиляция без openMP

- Игнорирует директивы **`#pragma omp`**
- Последовательная программа
- Один вычислительный поток
- Одно ядро процессора

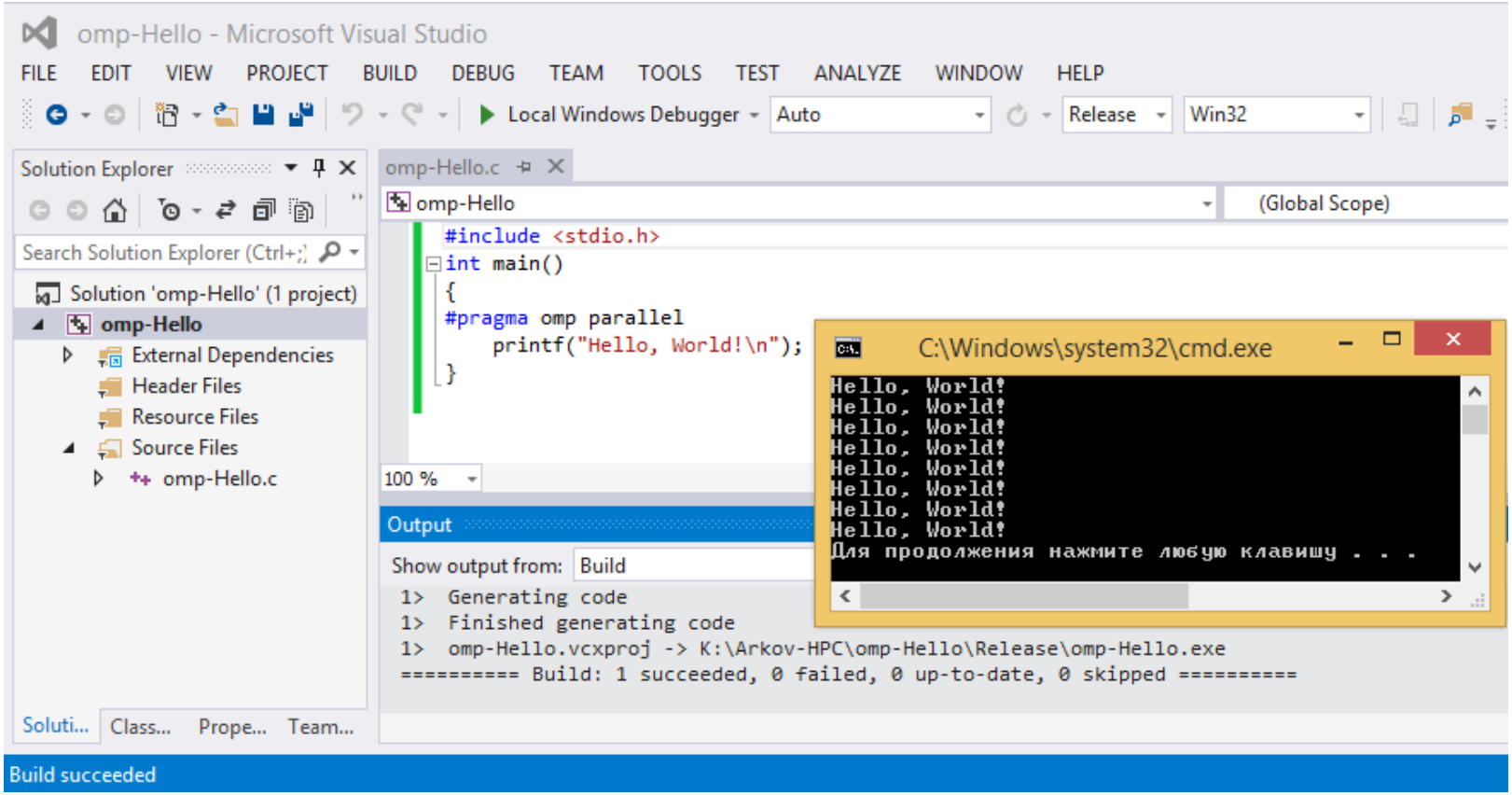
Hello, World!

```
#include <stdio.h>
int main()
{
   #pragma omp parallel
   printf("Hello, World!\n");
}
```

No (/openmp-)

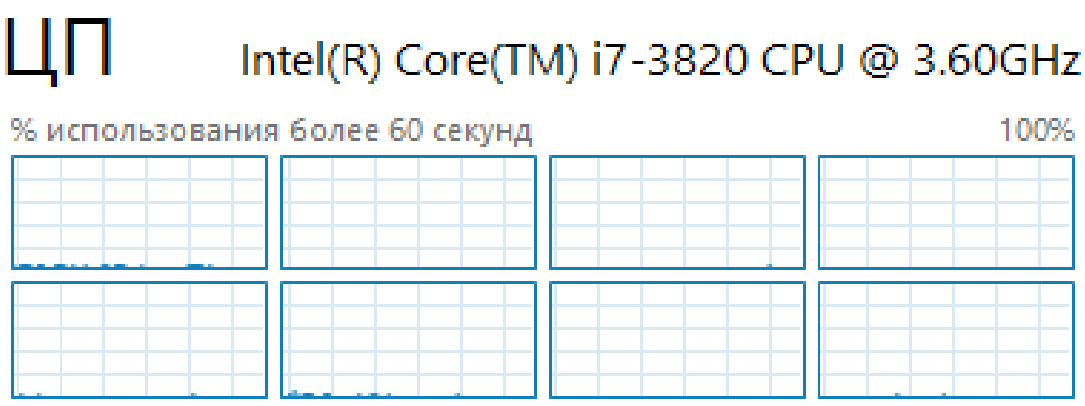

omp-Hello - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Release
Win32
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'omp-Hello' (1 project)
omp-Hello
External Dependencies
Header Files
Resource Files
Source Files
omp-Hello.c
omp-Hello.c
omp-Hello
#include <stdio.h>
int main()
{
#pragma omp parallel
printf("Hello, World!\n");
}
C:\Windows\system32\cmd.exe
Hello, World!
Для продолжения нажмите любую клавишу . . .
100 %
Output
Show output from: Build
1> Generating code
1> Finished generating code
1> omp-Hello.vcxproj -> K:\Arkov-HPC\omp-Hello\Release\omp-Hello.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
Soluti...
Class...
Prope...
Team...
Ready

Yes (/openmp)


omp-Hello - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Release
Win32
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'omp-Hello' (1 project)
omp-Hello
External Dependencies
Header Files
Resource Files
Source Files
omp-Hello.c
omp-Hello.c
omp-Hello
(Global Scope)
#include <stdio.h>
int main()
{
#pragma omp parallel
printf("Hello, World!\n");
}
100 %
C:\Windows\system32\cmd.exe
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Для продолжения нажмите любую клавишу . . .
Output
Show output from: Build
1> Generating code
1> Finished generating code
1> omp-Hello.vcxproj -> K:\Arkov-HPC\omp-Hello\Release\omp-Hello.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
Soluti... Class... Prope... Team...
Build succeeded

Распараллеливание

- Автоматическое определение числа логических ядер процессора
  - Число потоков = числу ядер процессора

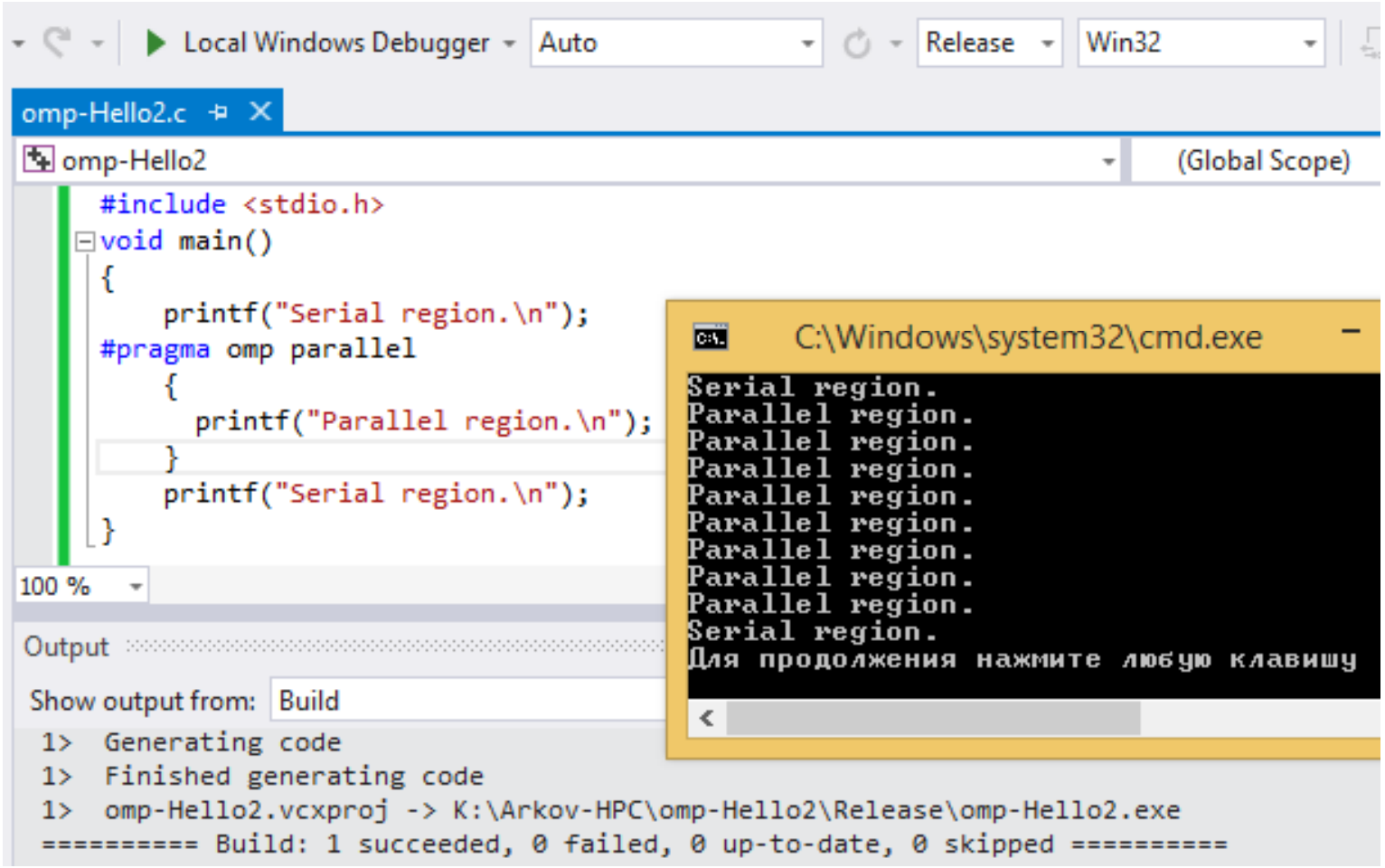



```
K:\EXE>omp-hello0
Hello, World!

K:\EXE>omp-hello1
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
```

Модель fork - join

```
#include <stdio.h>
void main()
{
printf("Serial region.\n");
#pragma omp parallel
   {
     printf("Parallel region.\n");
   }
printf("Serial region.\n");
}
```

fork - join


Local Windows Debugger
Auto
Release
Win32
omp-Hello2.c
omp-Hello2
(Global Scope)
#include <stdio.h>
void main()
{
printf("Serial region.\n");
#pragma omp parallel
{
printf("Parallel region.\n");
}
printf("Serial region.\n");
}
100 %
C:\Windows\system32\cmd.exe
Serial region.
Parallel region.
Parallel region.
Parallel region.
Parallel region.
Parallel region.
Parallel region.
Parallel region.
Parallel region.
Serial region.
Для продолжения нажмите любую клавишу
Output
Show output from: Build
1> Generating code
1> Finished generating code
1> omp-Hello2.vcxproj -> K:\Arkov-HPC\omp-Hello2\Release\omp-Hello2.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========

Число потоков

- Три способа задания пользовательского числа параллельных потоков:
  - параметр окружения
    - **`OMP_NUM_THREADS NUM`**
  - функция
    - **`omp_set_num_threads(int num);`**
  - параметр директивы
    - **`num_threads(num)`**

OMP_NUM_THREADS

- Переменная окружения
  - Environment variable
- Число потоков параллельной области

```
set
set OMP_NUM_THREADS=2
set
```

OMP_NUM_THREADS

```
K:\EXE>set
……
NUMBER_OF_PROCESSORS=8
OS=Windows_NT
……
K:\EXE>set OMP_NUM_THREADS=2
K:\EXE>set
……
NUMBER_OF_PROCESSORS=8
OMP_NUM_THREADS=2
……
K:\EXE>omp-Hello2
Serial region.
Parallel region.
Parallel region.
Serial region.
K:\EXE>set OMP_NUM_THREADS=4
```

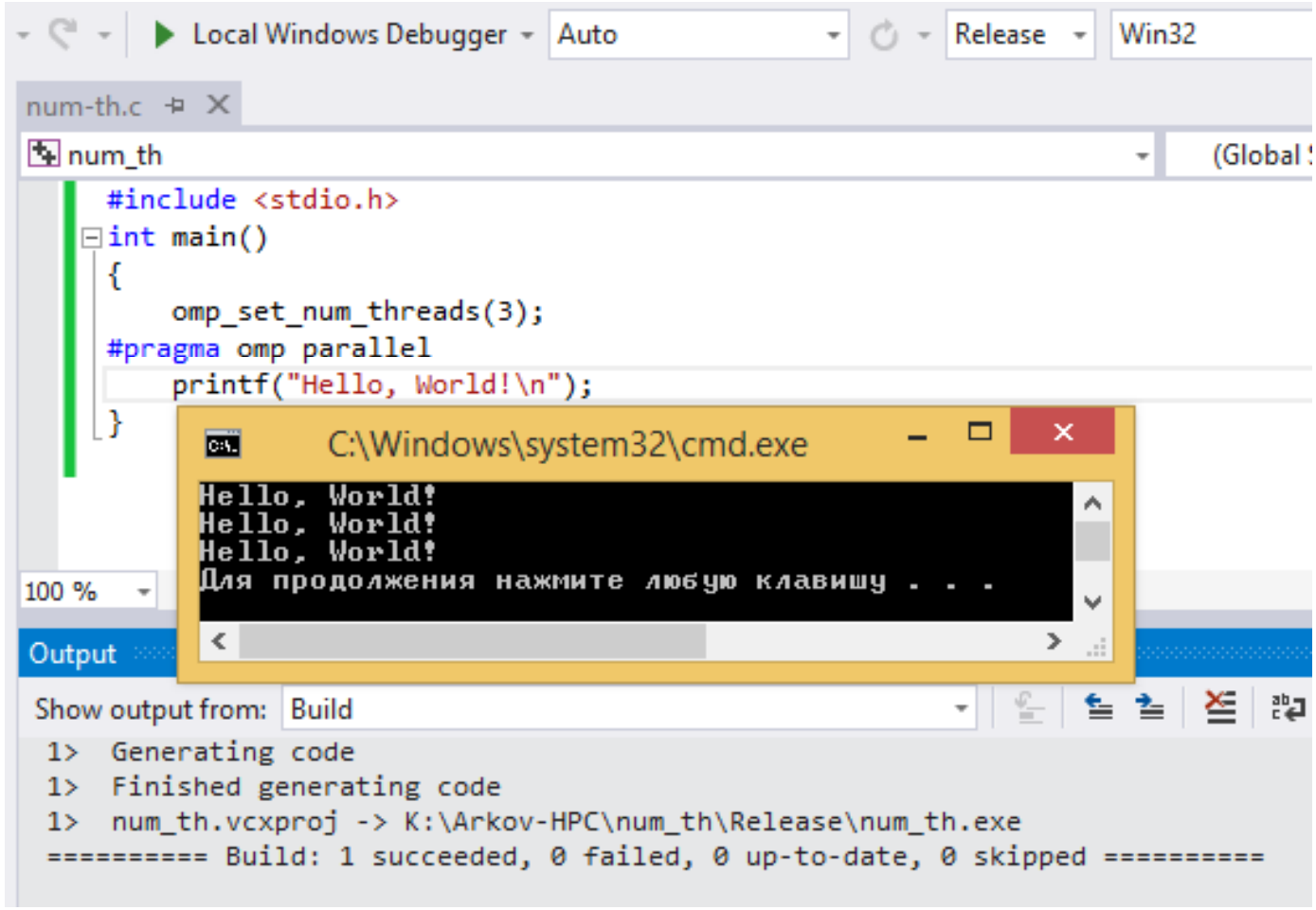
```
K:\EXE>set
……
NUMBER_OF_PROCESSORS=8
OMP_NUM_THREADS=4
……

K:\EXE>omp-Hello2
Serial region.
Parallel region.
Parallel region.
Parallel region.
Parallel region.
Serial region.

K:\EXE>
```

omp_set_num_threads

- `void omp_set_num_threads(int num);`
- Функция
  - Число потоков параллельной области

```
#include <stdio.h>
int main()
{
omp_set_num_threads(3);
   #pragma omp parallel
   printf("Hello, World!\n");
}
```

omp_set_num_threads

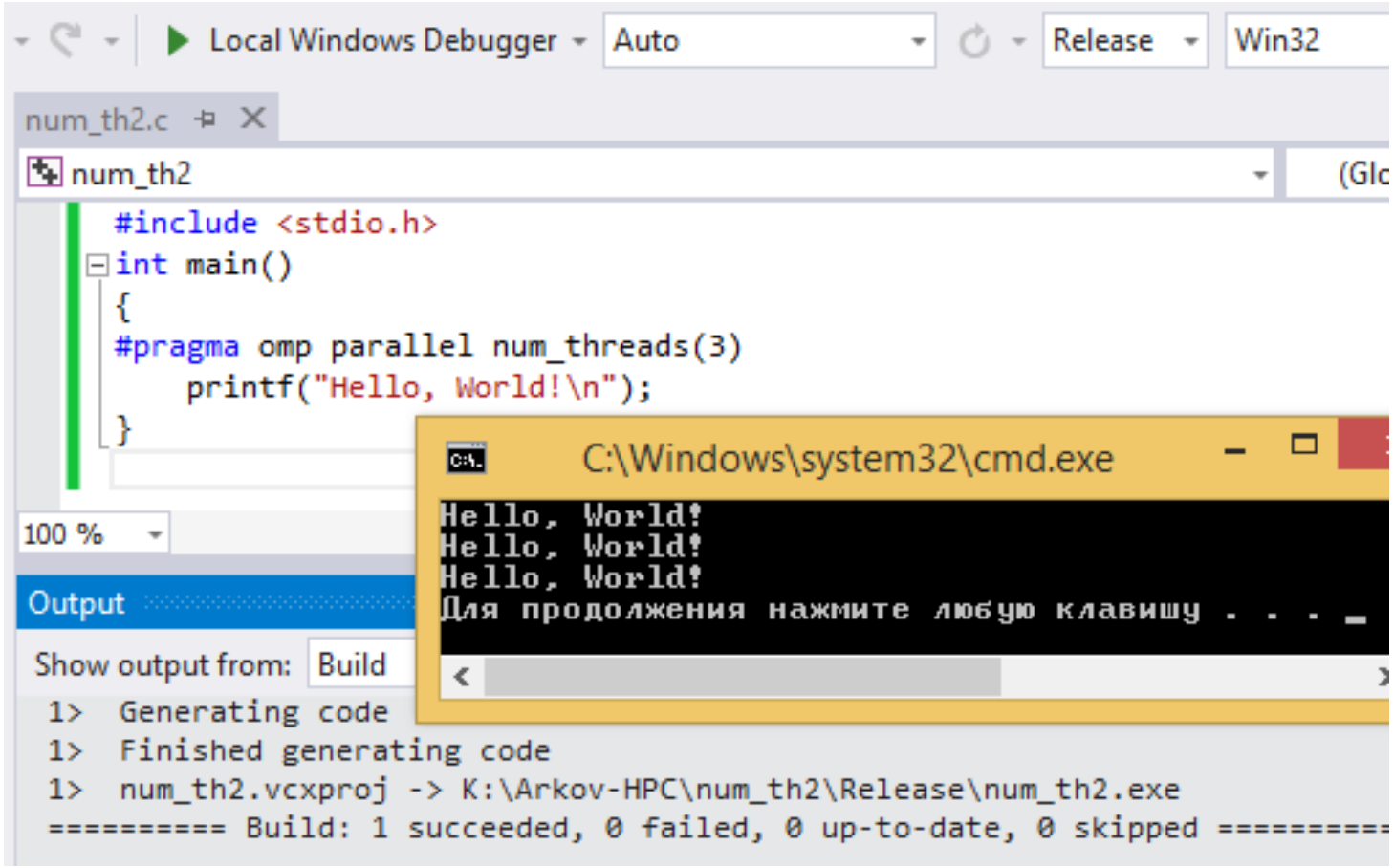


num_threads(num)

- **#pragma omp parallel num_threads**
- Опция директивы
  - Число потоков параллельной области

```
#include <stdio.h>
void main()
{
   #pragma omp parallel num_threads(3)
   printf("Hello, World!\n");
}
```

num_threads(num)

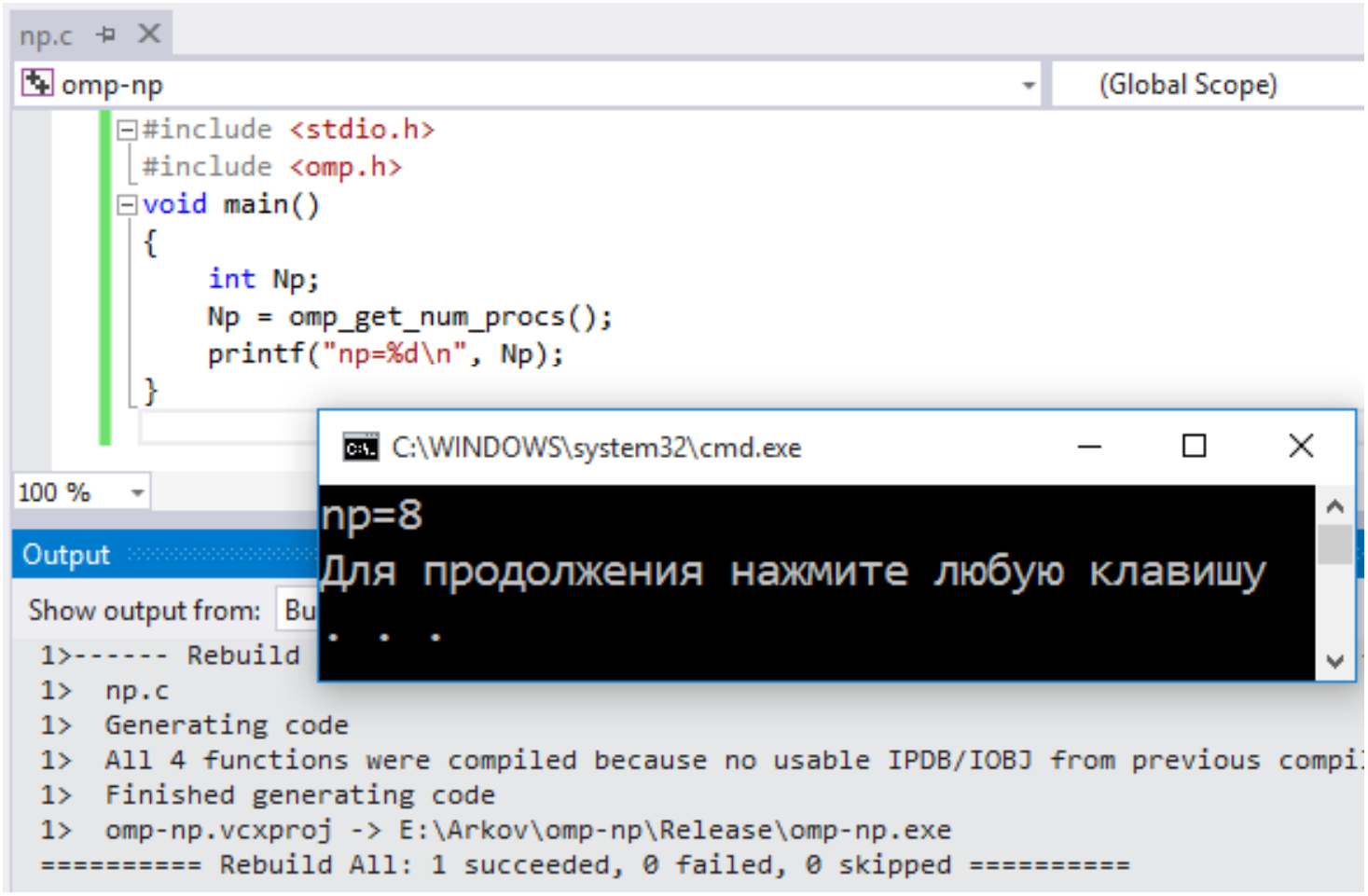


omp_get_num_procs

- **int omp_get_num_procs(void);**
  - Функция: количество доступных процессоров (логических ядер)

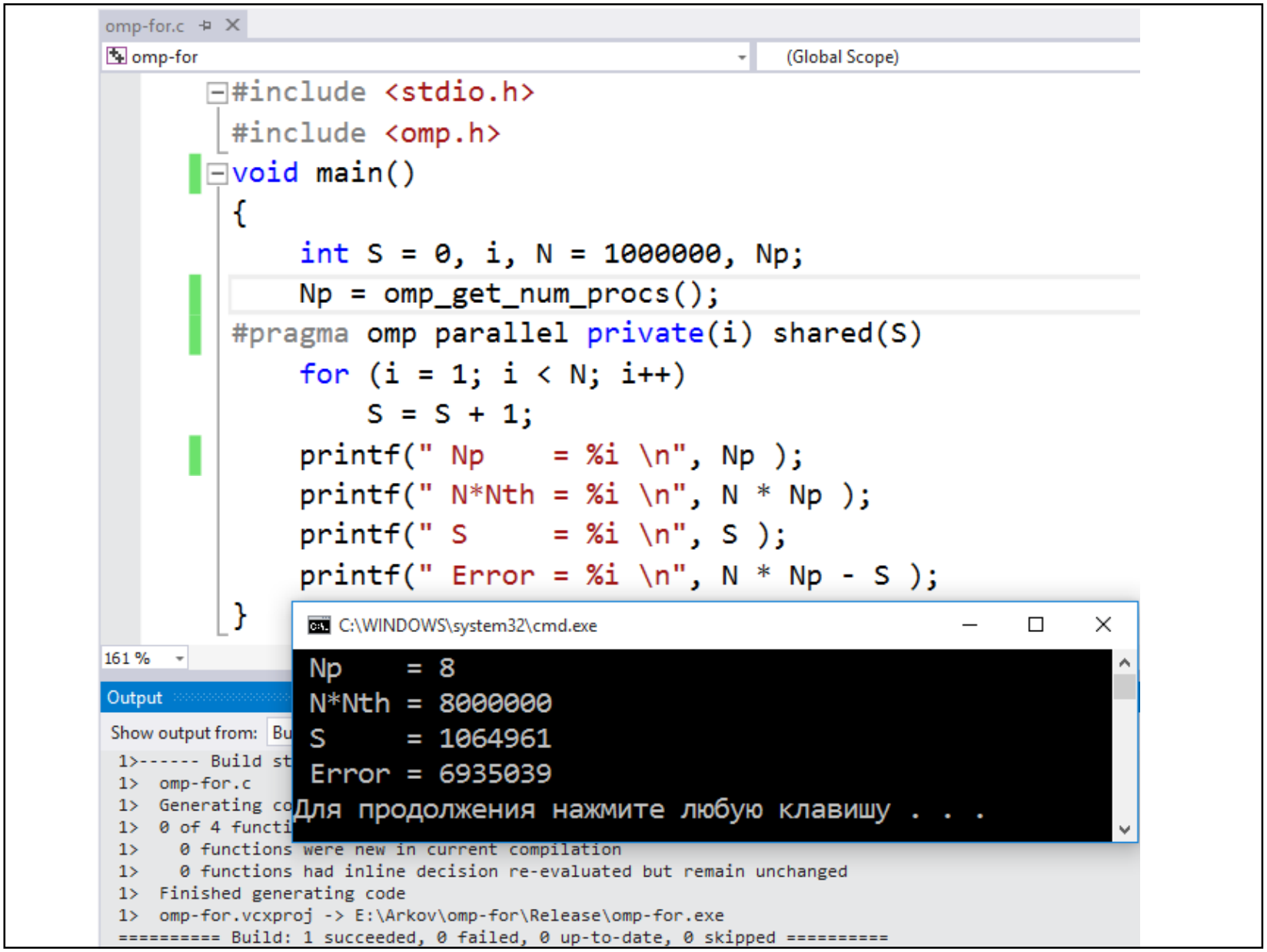
```
#include <stdio.h>
#include <omp.h>
int main()
{
  int Np;
  Np = omp_get_num_procs();
  printf("np=%d\n", Np);
}
```

omp_get_num_procs



#pragma omp parallel

```
#include <stdio.h>
#include <omp.h>
void main()
{
int S = 0, i, N = 1000000, Np;
Np = omp_get_num_procs();
#pragma omp parallel private(i) shared(S)
  for (i = 1; i < N; i++)
    S = S + 1;
printf(" Np    = %i \n", Np );
printf(" N*Nth = %i \n", N * Np );
printf(" S     = %i \n", S );
printf(" Error = %i \n", N * Np - S );
}
```


omp-for.c
omp-for
(Global Scope)
#include <stdio.h>
#include <omp.h>
void main()
{
int S = 0, i, N = 1000000, Np;
Np = omp_get_num_procs();
#pragma omp parallel private(i) shared(S)
for (i = 1; i < N; i++)
S = S + 1;
printf(" Np = %i \n", Np);
printf(" N*Nth = %i \n", N * Np);
printf(" S = %i \n", S);
printf(" Error = %i \n", N * Np - S);
}
161 %
C:\WINDOWS\system32\cmd.exe
Np = 8
N*Nth = 8000000
S = 1064961
Error = 6935039
Для продолжения нажмите любую клавишу . . .
Output
Show output from:
1>------ Build st
1> omp-for.c
1> Generating co
1> 0 of 4 functi
1> 0 functions were new in current compilation
1> 0 functions had inline decision re-evaluated but remain unchanged
1> Finished generating code
1> omp-for.vcxproj -> E:\Arkov\omp-for\Release\omp-for.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========

Типы переменных

Тип	Байт	Значения
`int`	4	–2,147,483,648 … 2,147,483,647
`unsigned int`	4	0 … 4,294,967,295
`float`	4	3.4E +/- 38 (7 цифр)
`double`	8	1.7E +/- 308 (15 цифр)

num_threads / thread_num

int omp_get_num_threads(void);

Количество потоков в группе

Number of threads

int omp_get_thread_num(void);

Номер потока в группе

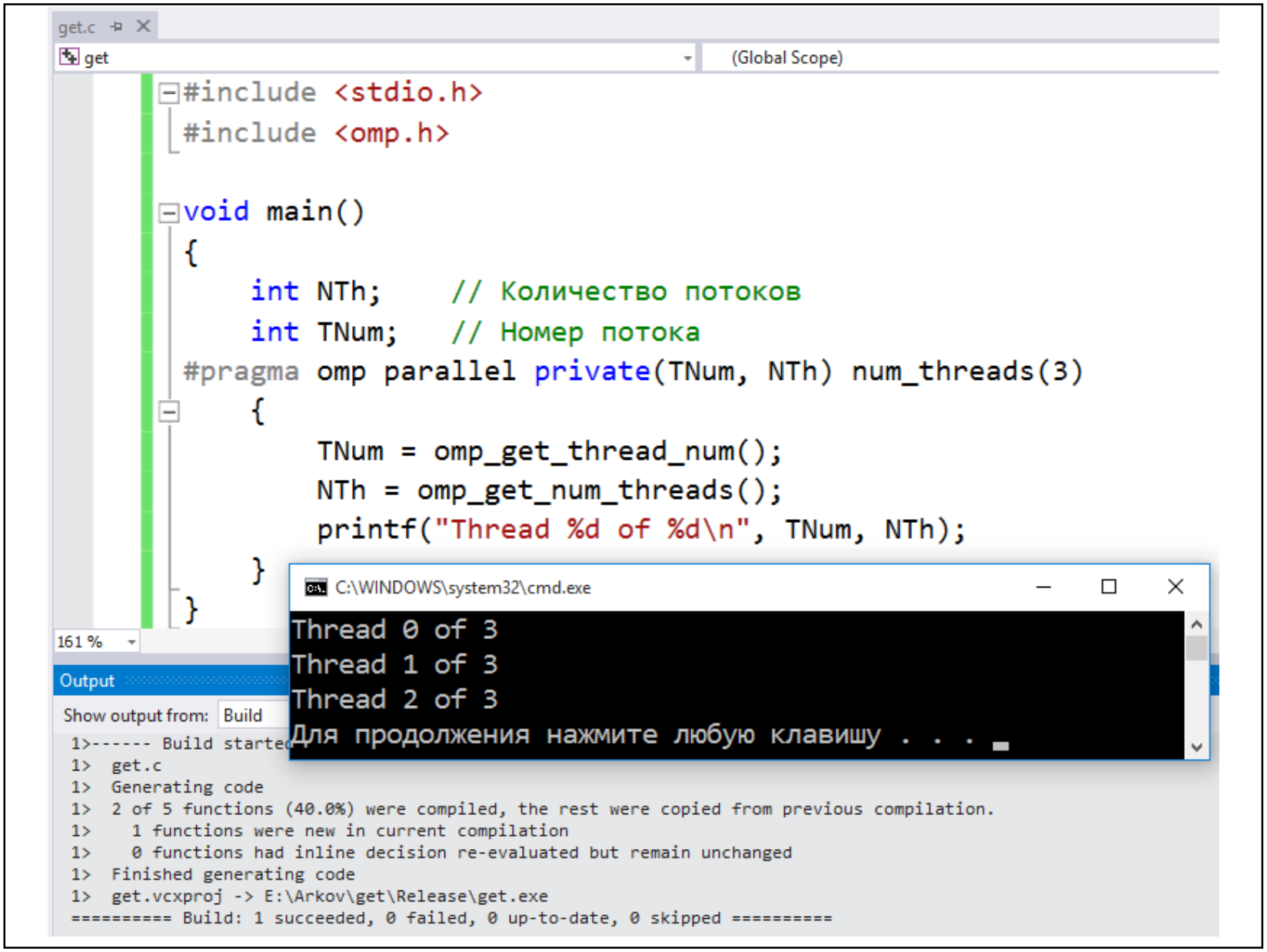
Thread number

от 0 до **omp_get_num_threads()-1**

Номер мастер-потока = 0

num_threads / thread_num

```
#include <stdio.h>
#include <omp.h>

void main()
{
int NTh;    // Количество потоков
int TNum;   // Номер потока
#pragma omp parallel private(TNum, NTh) num_threads(3)
  {
    TNum = omp_get_thread_num();
    NTh = omp_get_num_threads();
    printf("Thread %d of %d\n", TNum, NTh);
  }
}
```

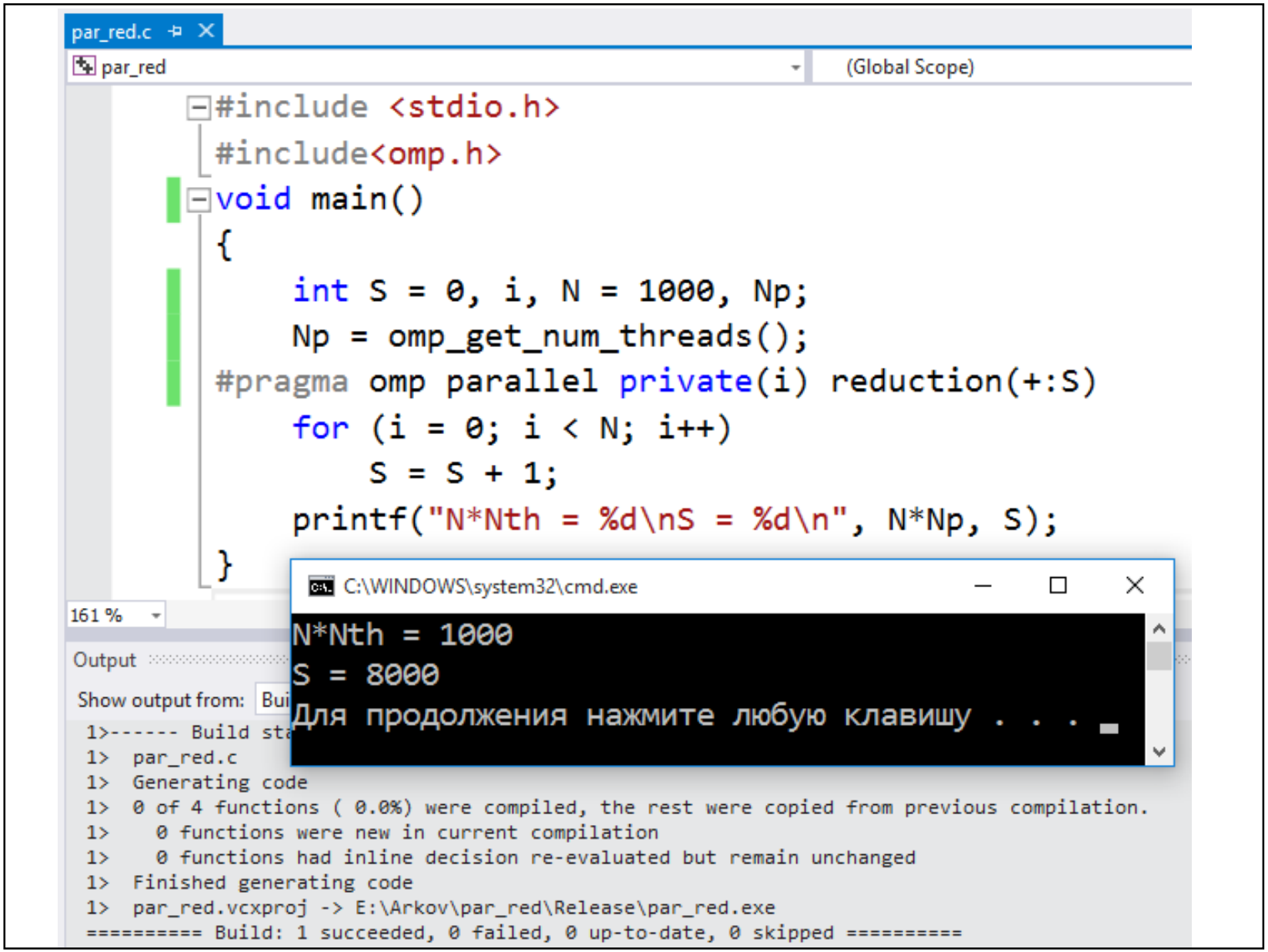

get.c
get
(Global Scope)
#include <stdio.h>
#include <omp.h>
void main()
{
int NTh; // Количество потоков
int TNum; // Номер потока
#pragma omp parallel private(TNum, NTh) num_threads(3)
{
TNum = omp_get_thread_num();
NTh = omp_get_num_threads();
printf("Thread %d of %d\n", TNum, NTh);
}
}
161 %
C:\WINDOWS\system32\cmd.exe
Thread 0 of 3
Thread 1 of 3
Thread 2 of 3
Для продолжения нажмите любую клавишу . . .
Output
Show output from: Build
1>------ Build started
1> get.c
1> Generating code
1> 2 of 5 functions (40.0%) were compiled, the rest were copied from previous compilation.
1> 1 functions were new in current compilation
1> 0 functions had inline decision re-evaluated but remain unchanged
1> Finished generating code
1> get.vcxproj -> E:\Arkov\get\Release\get.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========

Опция `reduction`

- **#pragma omp parallel reduction(***op***:** *var***)**
- **op** - оператор
- **var** – список переменных
- **#pragma omp parallel reduction(+: c)**

parallel reduction(+:S)

```
#include <stdio.h>
#include<omp.h>
void main()
{
int S = 0, i, N = 1000, Np;
Np = omp_get_num_threads();
#pragma omp parallel private(i) reduction(+:S)
   for (i = 0; i < N; i++)
        S = S + 1;
printf("N*Nth = %d\nS = %d\n", N*Np, S);
}
```


par_red.c
par_red
(Global Scope)
#include <stdio.h>
#include<omp.h>
void main()
{
int S = 0, i, N = 1000, Np;
Np = omp_get_num_threads();
#pragma omp parallel private(i) reduction(+:S)
for (i = 0; i < N; i++)
S = S + 1;
printf("N*Nth = %d\nS = %d\n", N*Np, S);
}
161 %
C:\WINDOWS\system32\cmd.exe
N*Nth = 1000
S = 8000
Для продолжения нажмите любую клавишу . . .
Output
Show output from:
1>------ Build sta
1> par_red.c
1> Generating code
1> 0 of 4 functions (0.0%) were compiled, the rest were copied from previous compilation.
1> 0 functions were new in current compilation
1> 0 functions had inline decision re-evaluated but remain unchanged
1> Finished generating code
1> par_red.vcxproj -> E:\Arkov\par_red\Release\par_red.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========

#pragma omp parallel for

- Директива **for**
  - итерации цикла выполняются параллельно
  - итерации распределяются между параллельными потоками

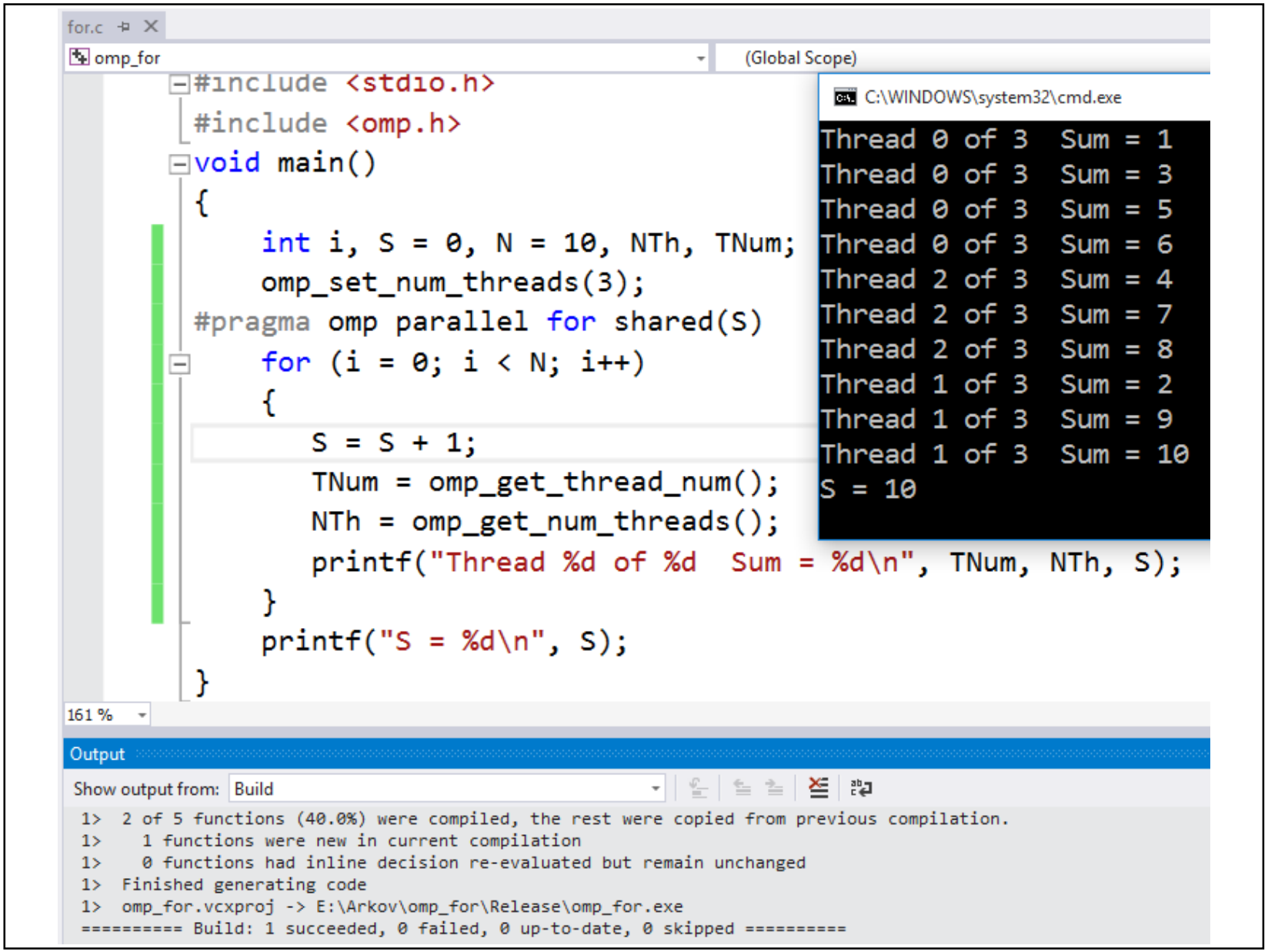
```
#pragma omp parallel for
for (i = 0; i < N; i++)
     S = S + 1;
```

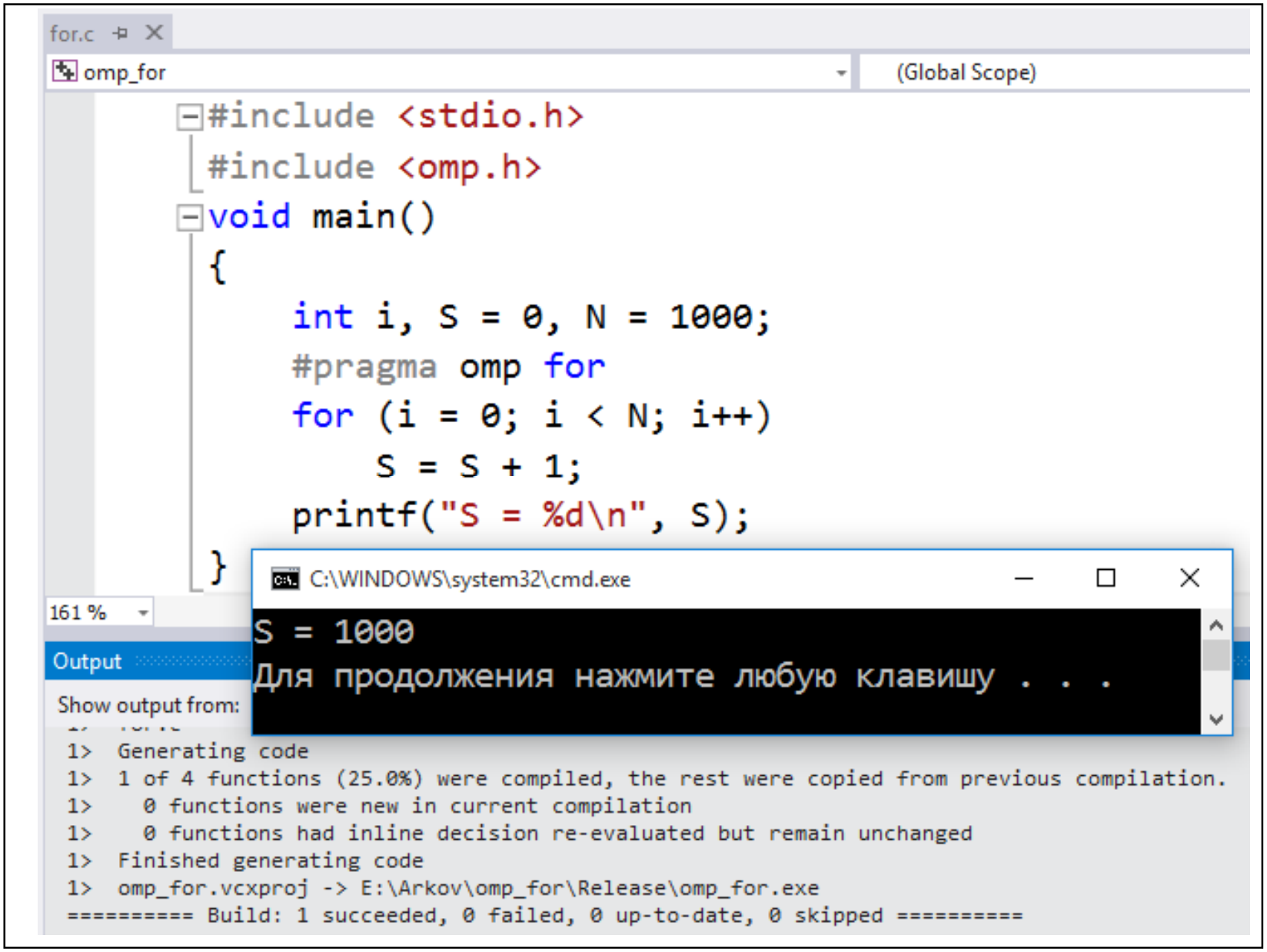
#pragma omp parallel for

```
#include <stdio.h>
#include <omp.h>
void main()
{
int i, S = 0, N = 1000;
#pragma omp parallel for
     for (i = 0; i < N; i++)
       S = S + 1;
printf("S = %d\n", S);
}
```

Автоматическое распараллеливание цикла

```
#include <stdio.h>
#include <omp.h>
void main()
{
int i, S = 0, N = 10, NTh, TNum;
omp_set_num_threads(3);
#pragma omp parallel for shared(S)
for (i = 0; i < N; i++)
  {
    S = S + 1;
    TNum = omp_get_thread_num();
    NTh = omp_get_num_threads();
    printf("Thread %d of %d  Sum = %d\n", TNum, NTh, S);
  }
printf("S = %d\n", S);
}
```

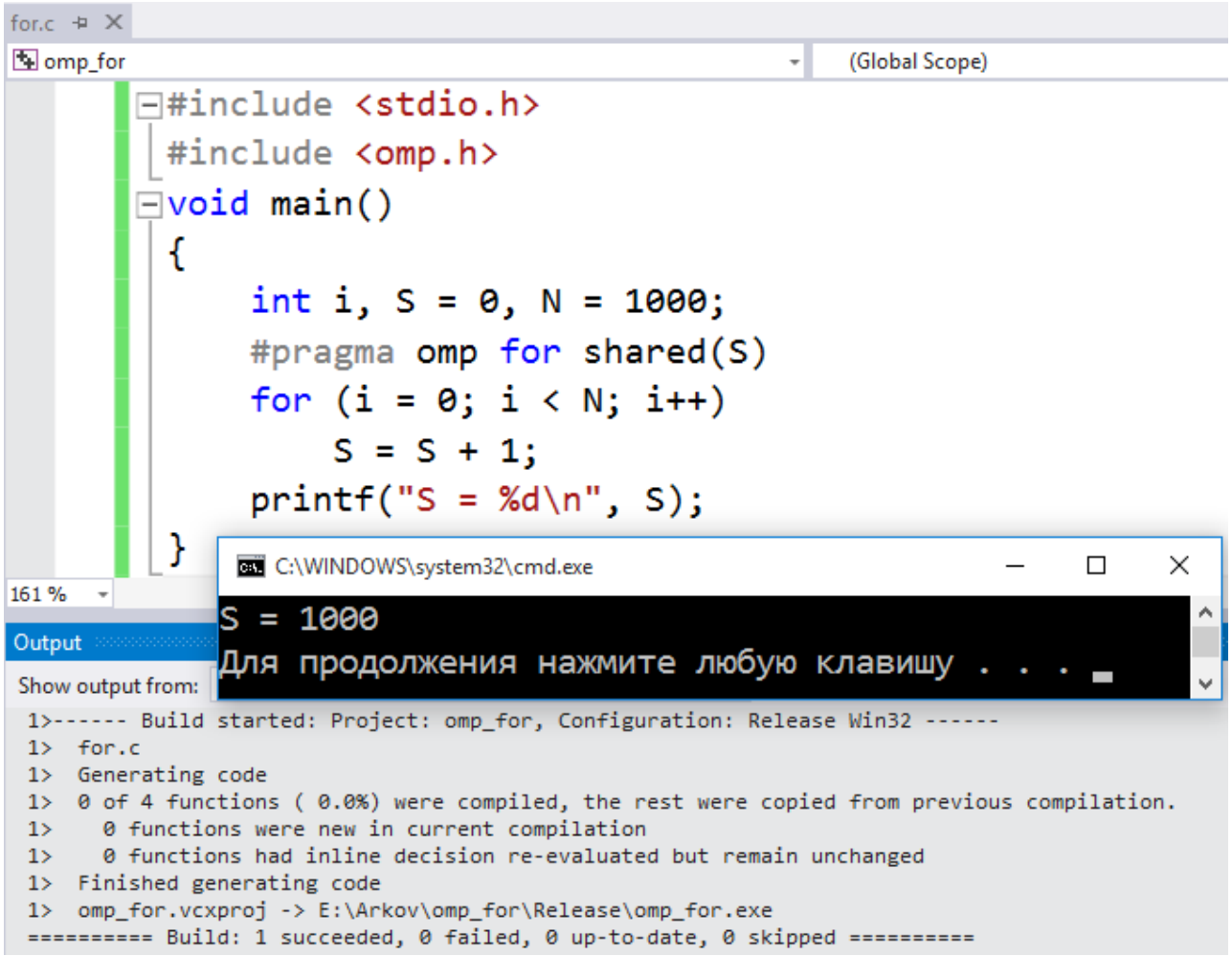

for.c
omp_for
(Global Scope)
#include <stdio.h>
#include <omp.h>
void main()
{
int i, S = 0, N = 10, NTh, TNum;
omp_set_num_threads(3);
#pragma omp parallel for shared(S)
for (i = 0; i < N; i++)
{
S = S + 1;
TNum = omp_get_thread_num();
NTh = omp_get_num_threads();
printf("Thread %d of %d Sum = %d\n", TNum, NTh, S);
}
printf("S = %d\n", S);
}
161 %
C:\WINDOWS\system32\cmd.exe
Thread 0 of 3 Sum = 1
Thread 0 of 3 Sum = 3
Thread 0 of 3 Sum = 5
Thread 0 of 3 Sum = 6
Thread 2 of 3 Sum = 4
Thread 2 of 3 Sum = 7
Thread 2 of 3 Sum = 8
Thread 1 of 3 Sum = 2
Thread 1 of 3 Sum = 9
Thread 1 of 3 Sum = 10
S = 10
Output
Show output from: Build
1> 2 of 5 functions (40.0%) were compiled, the rest were copied from previous compilation.
1> 1 functions were new in current compilation
1> 0 functions had inline decision re-evaluated but remain unchanged
1> Finished generating code
1> omp_for.vcxproj -> E:\Arkov\omp_for\Release\omp_for.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========


for.c
omp_for
(Global Scope)
#include <stdio.h>
#include <omp.h>
void main()
{
int i, S = 0, N = 1000;
#pragma omp for
for (i = 0; i < N; i++)
S = S + 1;
printf("S = %d\n", S);
}
161 %
C:\WINDOWS\system32\cmd.exe
S = 1000
Для продолжения нажмите любую клавишу . . .
Output
Show output from:
1> Generating code
1> 1 of 4 functions (25.0%) were compiled, the rest were copied from previous compilation.
1> 0 functions were new in current compilation
1> 0 functions had inline decision re-evaluated but remain unchanged
1> Finished generating code
1> omp_for.vcxproj -> E:\Arkov\omp_for\Release\omp_for.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========

#pragma omp for

```
#include <stdio.h>
#include <omp.h>
void main()
{
int i, S = 0, N = 1000;
#pragma omp for shared(S)
     for (i = 0; i < N; i++)
       S = S + 1;
printf("S = %d\n", S);
}
```

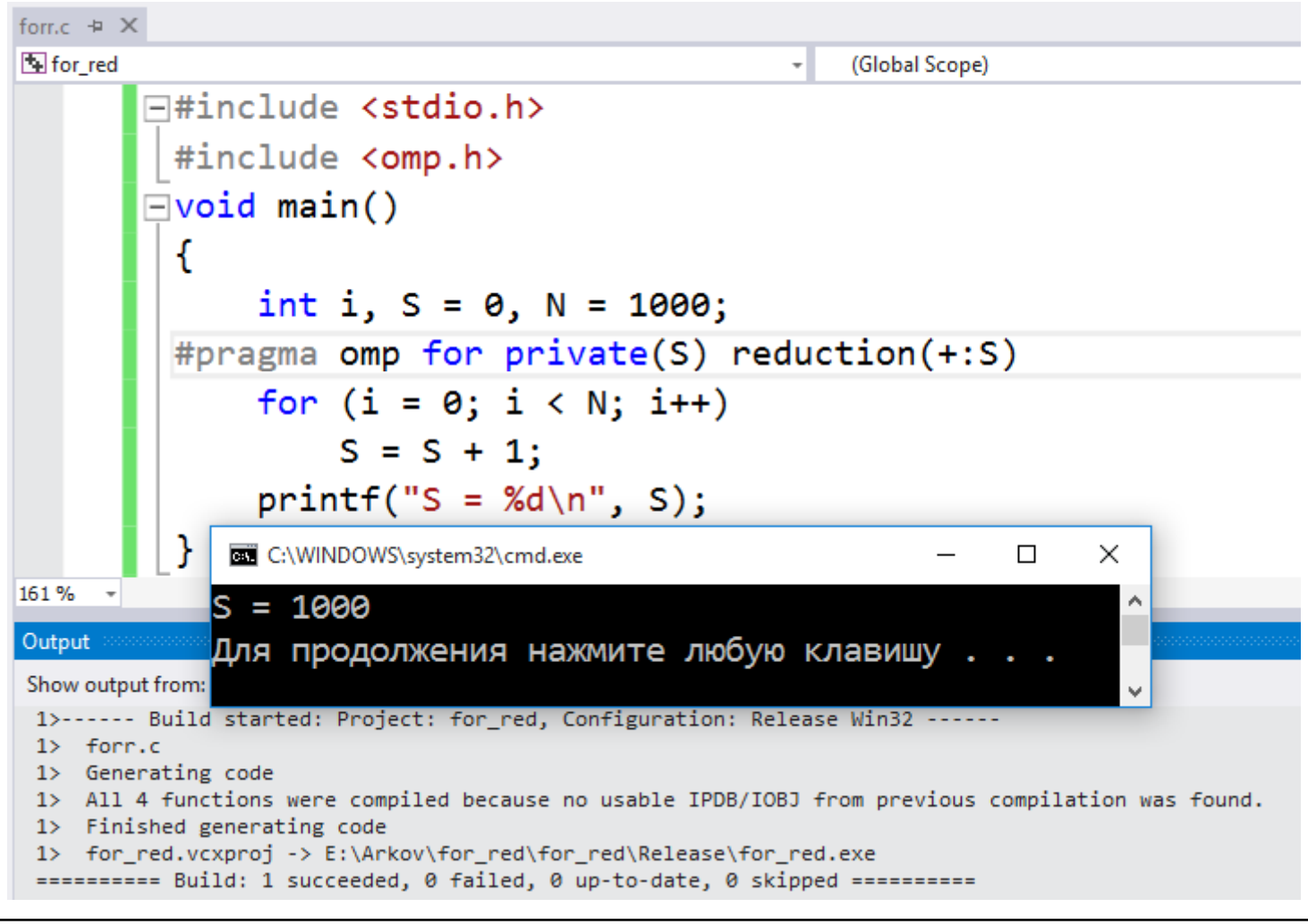
#pragma omp for



#pragma omp for reduction

```
#include <stdio.h>
#include <omp.h>
void main()
{
int i, S = 0, N = 1000;
#pragma omp for private(S) reduction(+:S)
  for (i = 0; i < N; i++)
    S = S + 1;
printf("S = %d\n", S);
}
```

#pragma omp for reduction



Самостоятельно

- Вычисление методом прямоугольников числа π
  - Определите ускорение и эффективность для различного числа потоков
  - Выберите приемлемую продолжительность расчетов
  - Используйте среднее время вычислений по 10 запускам программы
- Шаблон программы
  - Учебник Антонов OpenMP
  - Лекция MPI