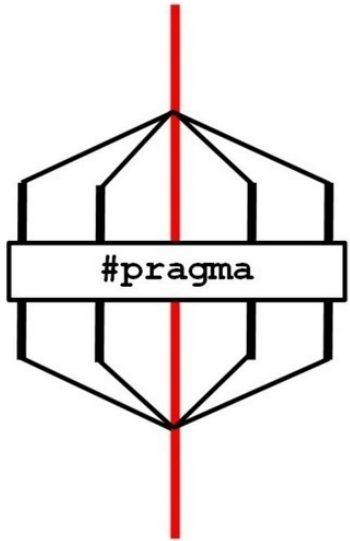
Валентин Юльевич Арьков

Организация параллельных потоков. Часть 2

Учебное пособие



Валентин Юльевич Арьков

Организация параллельных потоков. Часть 2. Учебное пособие

http://www.litres.ru/pages/biblio_book/?art=51848146
ISBN 9785449854452

Аннотация

При решении инженерных, экономических и научных задач используются высокопроизводительные вычисления – High Performance Computing или сокращённо HPC.Параллельные программы нужны для того, чтобы использовать вычислительные мощности многоядерных процессоров и графических ускорителей. В данной работе мы рассмотрим технологию автоматической организации параллельных потоков для многоядерных вычислительных машин.

Содержание

Организация параллельных потоков. Часть 2
Учебное пособие

Валентин Юльевич Арьков

ISBN 978-5-4498-5445-2 (т. 2)
ISBN 978-5-4498-5446-9
Создано в интеллектуальной издательской системе Ridero

Введение

Высокопроизводительные вычисления стали сегодня реальностью не только на уровне суперкомпьютеров и вычислительных кластеров, но и для персональных компьютеров и мобильных устройств. Речь идёт о решении инженерных, экономических и научных задач с использованием высокопроизводительных вычислений и написанием параллельных приложений (прикладных программ).

Параллельные программы нужны для того, чтобы использовать вычислительные мощности многоядерных процессоров и графических ускорителей. В данной работе мы рассмотрим технологию автоматической организации параллельных потоков для многоядерных вычислительных машин.

Нам предстоит оценить параметры эффективности распараллеливания программы на конкретной конфигурации вычислительной системы. По результатам измерения быстродействия определяются показатели ускорения и эффективности распараллеливания. Это опыт исследования и практического знакомства с технологией – не по учебнику, а в форме личногго знакомства. Каждое положение и утверждение можно проверить, «покрутить в руках» и убедиться в его правильнсти или ошибочности.

Мы рассмотрим задачу численной оценки значения опре-

делённого интеграла – по двум причинам. Во-первых, многие практические задачи сводятся к нахождению суммы или интеграла. Во-вторых, численные методы интегрирования хорошо поддаются распараллеливанию. Каждое отдельное значение подынтегрального выражения можно вычислять независимо от всех остальных значений. Поэтому численное интегрирование – подходящая задача для знакомства с высокопроизводительными вычислениями.

1. Общие сведения

1.1. Информация и литература

Параллельное программирование освещается в большом количестве учебников и пособий [1—9].

При изучении параллельного программирования полезно обращаться к библиотеке учебных материалов Лаборатории параллельных информационных технологий НИВЦ МГУ. Доступ к библиотеке осуществляется по адресу:

http://parallel.ru/info/parallel/

Учебник и учебные пособия, представленные на указанном сайте, предназначены для использования студентами вузов и доступны для бесплатного скачивания.

В данной работе мы будем опираться на некоторые примеры из учебного пособия А. С. Антонова [8]. Для первого знакомства с технологиями мы разбираем каждый пример достаточно подробно. Попутно мы обсуждаем самые общие вопросы.

Всё это нужно, чтобы студент не просто освоил стандартные, шаблонные действия с конкретным программным продуктом. В любом деле нужны специалисты с кругозором и эрудицией, с пониманием и способностью самостоятельно

развиваться. А это требует чего-то большего, чем только узкопрофессиональные знания и конкретные умения.

На сегодняшний день в интернете имеется множество онлайн курсов.

Первый пример – Национальный Открытый Университет ИНТУИТ:

https://www.intuit.ru.

Основной ресурс с отечественными массовыми открытыми онлайн-курсами (МООК) – «Открытое образование»:

https://openedu.ru/.

Международная платформа МООК «Курсера»:

https://www.coursera.org/.

Задание. Найдите на перечисленных сайтах курсы по следующим ключевым словам и перечислите их в отчёте:

– параллельные;
– parallel;
– высокопроизводительные;
– high performance computing;
– суперкомпьютеры;
– supercomputer;
– OpenMP;
– HPC;
– многоядерные;
– multicore.

1.2. Оформление отчёта

Отчёт по работе оформляем точно так же, как и в предыдущих работах [10]. Отчёт делаем в виде рабочей книги Excel. Это многостраничная книга с оглавлением.

Вначале, как и положено, должен быть титульный лист со всеми данными о работе и исполнителе.

Затем идёт оглавление со ссылками на все страницы.

Далее – задание.

Следом – шаги выполнения работы.

Текст программы вставляем как текст, а не как картинку.

На каждом листе – заголовок и пояснения о том, что заложено в данной программе. Что она должна делать и как это реализовано. Здесь же копия экрана и пояснения по поводу результатов работы.

Поскольку листов в отчёте будет много, названия листов (на вкладках) содержат только номера страниц. Подробные названия нужны в верхней части листа и в оглавлении.

Задание. Создайте файл отчёта и заполните титульный лист.

2. Технология OpenMP

В данной работе мы знакомимся с технологией автоматического распараллеливания программ **OpenMP**.

Название расшифровывается следующим образом:

Open Multi-Processing.

Распараллеливание программ поддерживается для двух языков программирования

– Fortran;

– C/C++.

2.1. Ключевые понятия

В результате использования данной технологии компилятор автоматически генерирует многопоточные программы. Такие программы дают эффект ускорения при запуске на многоядерных системах с общей памятью.

Многоядерные компьютеры – это так называемые «системы с общей памятью». Другое название – разделяемая память. Английское название: SHARED MEMORY.

Имеется в виду совместное использование оперативной памяти: любой поток имеет доступ к общим глобальным переменным процесса. Более красивая официальная формулировка звучит так: «общий доступ параллельных потоков к виртуальному адресному пространству текущего процесса».

Задание. Изучите в Википедии следующие статьи и выясните, что означают эти термины:

– «Multiprocessing» или «Многопроцессорность»;

– «Multithreading (computer architecture)» или «Многопоточность»;

– «Многоядерный процессор» или «Multi-core processor»;

– «OpenMP».

Задание. Изучите историю разработки технологии и версии спецификаций для обоих языков программирования:

https://www.openmp.org/specifications/.

OpenMP использует наиболее распространённую модель параллелизма:

Single Instruction Multiple Data (SIMD).

В этом случае параллельная часть программы состоит из нескольких одинаковых потоков, которые обрабатывают разные наборы данных.

Задание. Изучите в Википедии статьи «SIMD» и «Таксономия Флинна». Выясните, что такое SIMD.

2.2. Многопоточность

Использование OpenMP проще всего изучать на компьютере с одним многоядерным процессором.

Чтобы сделать из последовательной программы параллельную, разработчику нужно добавить всего несколько строк. Компилятор может игнорировать директивы распараллеливания. В этом случае мы получаем исходную последовательную программу. Если настроить компилятор на использование OpenMP, он автоматически сгенерирует и скомпилирует параллельную программу.

В технологии OpenMP происходит динамическое создание потоков. В процессе работы программа может создавать и уничтожать дополнительные потоки (рис. 2.1). Такая модель условно называется «fork-join». Это означаетразделение программына несколько параллельных веток, выполнение работы параллельном режиме и последующее слияние в одну последовательную ветку.

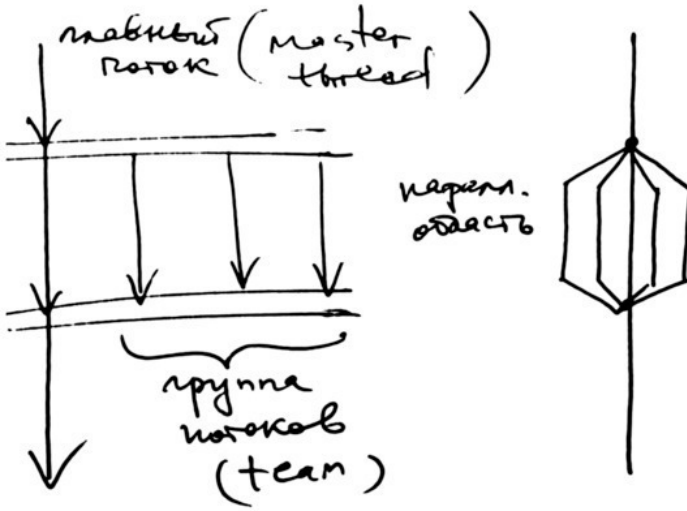


Рис. 2.1. Модель распараллеливания fork-join

Задание. Изучите в Википедии статью «Fork join».

2.3. Термины OpenMP

В технологии OpenMP используют следующие понятия и термины.

Construct – Конструкция – директива и следующий за ней структурный блок команд (часть программы, ограниченная фигурными скобками).

Directive – Директива – строка, начинающаяся как **#pragma omp** и определяющая поведение программы.

Thread – Поток – часть программы, которая может выполняться последовательно на одном ядре.

Master thread – Главный поток – поток, который существует на протяжении всего времени выполнения процесса и который создаёт группу параллельных потоков (**Team**) при входе в параллельную область.

Team – Группа параллельных потоков – один или несколько потоков, которые участвуют в выполнении какой-либо конструкции языка.

Параллельная программа состоит из параллельных и последовательных областей.

Parallel region – Параллельная область – часть программы, которая может выполняться несколькими потоками.

Serial region – Последовательная область – часть программы, которая выполняется только главным потоком.

Переменные делятся на два вида – в зависимости от способа доступа:

Private – уникальная переменная, доступная только внутри потока.

Shared – общая переменная, доступная всем параллельным потокам в рамках одной группы.

2.4. Ответственность разработчика

Лёгкость распараллеливания оборачивается повышением ответственности программиста.

Конкретная реализация OpenMP (среда разработки, компилятор и библиотека) не обязательно проверяет корректность распараллеливания и возникновение следующих нежелательных ситуаций:

– **Dependencies** – зависимости по данным между параллельными потоками;

– **Conflicts** – конфликты доступа к данным;

– **Deadlocks** – тупиковые ситуации (взаимная блокировка);

– **Race conditions** – ситуация гонки за доступ к данным.

Как видим, все эти ситуации связаны с обращением к общим данным из нескольких параллельных потоков. В последовательных программах таким проблем просто не может возникнуть.

Ответственность за корректность составления программы лежит полностью на составителе программы. Все эти положения можно найти в тексте спецификации OpenMP.

2.5. Инструменты OpenMP

Для реализации параллелизма в технологии OpenMP используют три вида инструментов:

– **Directives** – Директивы компилятора;

– **Library functions** – Готовые библиотечные функции;

– **Environment variables** – Переменные среды (параметры окружения).

2.5. Среда разработки

Всё программное обеспечение, используемое в данной работе, является бесплатным и доступно на официальных сайтах фирм-разработчиков. Программы устанавливают в операционной системе **Microsoft Windows**.

Интегрированная среда разработки **Microsoft Visual Studio Community Edition** предоставляется бесплатно для студентов, индивидуально работающих программистов и разработчиков программного обеспечения с открытым исходным кодом **Open Source.**

В данной работе мы будем использовать **Visual Studio**. Не потому, что это самый лучший компилятор. И не потому, что мы хотели бы заработать на рекламе конкретного программного продукта. Просто под руку попалось. Работает для наших задач – и на том спасибо.

Желающие могут работать в любой другой операцион-

ной системе и использовать любой другой компилятор языка программирования Си. Всё, что требуется – это поддержка технологии распараллеливания OpenMP.

Устанавливаем **Visual Studio.**

Переходим на сайт:

https://visualstudio.microsoft.com/.

Выбираем версию **Community Edition** (рис. 2.2). Она бесплатна для учебных и некоммерческих целей.

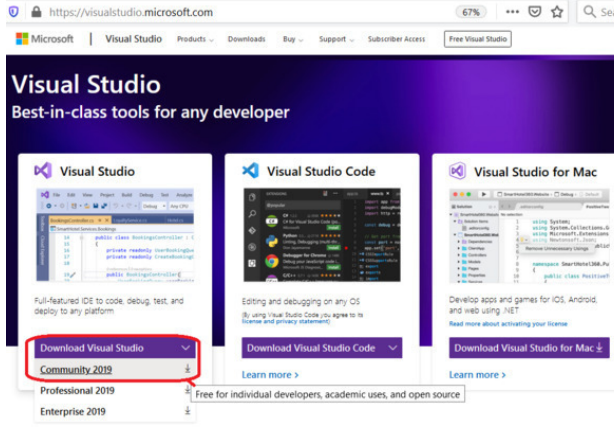


Рис. 2.2. Выбор версии среды разработки

Раньше, в далёком прошлом (несколько лет тому назад) программы были небольшие, и разработчики предлагали скачать образ диска в формате ***.ISO**. Его можно было даже на «болванку» записать. На записываемый **DVD-R** или

на многоразовый, перезаписываемый **DVD-RW**.

Сегодня выбора почти не осталось. Есть только web-установщик. Небольшая программа, которая скачает из интернета необходимые компоненты для выбранной конфигурации. Счёт может идти на десятки гигабайт, а то и поболее.

Нажимаем кнопку «Установить» и начинается скачивание установщика (рис. 2.3).

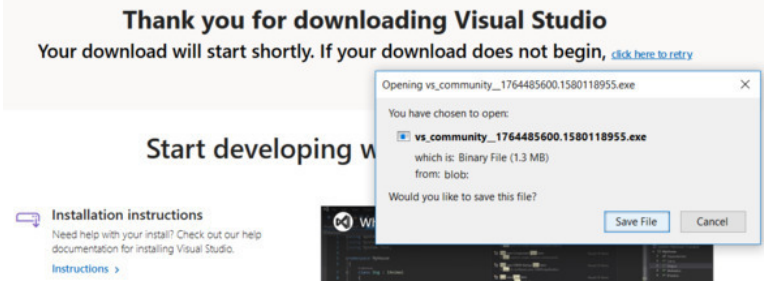


Рис. 2.3. Скачивание веб-установщика

Как видим, пока мы скачали чуть больше одного мегабайта.

Скачанную программу запускаем. Начинается процесс установки. В дополнение к стандартному набору средств разработки, который нам предлагают по умолчанию, мы выбираем компилятор языка Си. Остальные компоненты тоже можно установить, если будет такое желание ии необходимость. Веб-установщик скачает всё, что потребуется для за-

вершения установки.

Задание. Установите среду разработки и компилятор языка Си.

3. Составление параллельных программ

В этом разделе мы составим несколько программ, постепенно усложняя алгоритмы.

Каждую программу мы исследуем и рассмотрим со всех сторон.

Попутно мы уточним и проясним разные моменты и особенности.

3.1. Hello, World!

Убедимся, что среда разработки позволит нам сделать что-нибудь полезное. Мы начинаем работу с нуля. С самой простой программы «Всем привет!». Английское название: «Hello, World!»

Создаём новый проект. Пустой, незаполненный проект.

Добавляем новый элемент проекта – файл с исходным текстом программы (рис. 3.1).

Компилируем программу и запускаем её на выполнение.

Для каждой новой программы создаём новый каталог и новый файл с исходным текстом программы. Иначе потом следов не найдёшь. В руках останется последняя недоделанная программа, которая затёрла всю предыдущую работу.

E:\Arkov-OMP\Hello-1\Hello-1.c

```
1  #include <stdio.h>
2
3  void main()
4  {
5      printf("Hello World!\n");
6  }
7
```

Рис. 3.1. Программа «Всем привет»

Устанавливаем конфигурацию **Release**.

Нажимаем комбинацию клавиш:

Ctrl + F5.

Видим результаты работы программы в командном окне.

Задание. Создайте программу «Всем привет!» и запустите её на выполнение.

В данной работе мы будем иметь дело только с вариантом **Release** и не будем заниматься отладкой программы в режиме **Debug**. Чтобы больше не отвлекаться на эти красивые названия, их нужно будет разобрать и изучить.

Задание. Изучите в Википедии статьи «Отладка программы», «Debugging», «Software bug», «Стадии разработки программного обеспечения» и «Software release life cycle». Выясните значение, происхождение и перевод терминов **DEBUG** и **RELEASE**.

По умолчанию шрифт командного окна довольно мелкий.

Настроим его покрупнее.

Щелкаем по значку в левом верхнем углу окна (рис. 3.2) и выбираем свойства:

Properties.

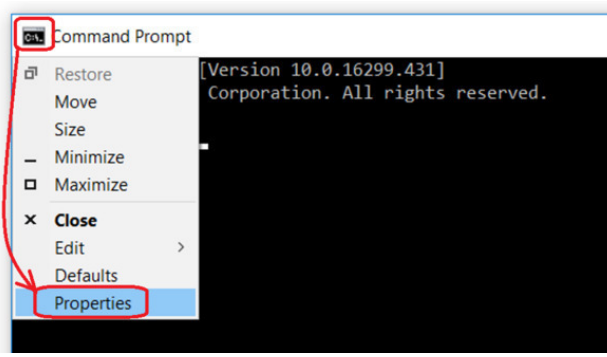


Рис. 3.2. Свойства командного окна

Здесь нас интересует раздел шрифтов:

Font.

Выбираем любой моноширинный шрифт (рис. 3.3). Это означает, что все буквы одинаковой ширины. В названии некоторых шрифтов может быть слово **mono**. «Моно» означает «один». Эти шрифты имитируют первые печатные устройства, похожие на печатные машинки. Моноширинный шрифт позволяет расположить результаты вывода на экран ровными столбцами.

Задаём подходящий размер.

Если буквы плохо различимы, делаем шрифт жирным.

В нашем примере мы выбрали такой шрифт:

– Size – 24;

– Font – Courier New;

– Bold fonts.

В нижней части окна выбора шрифта выводится пример текста на экране и указаны размеры букв в пикселах (точках).

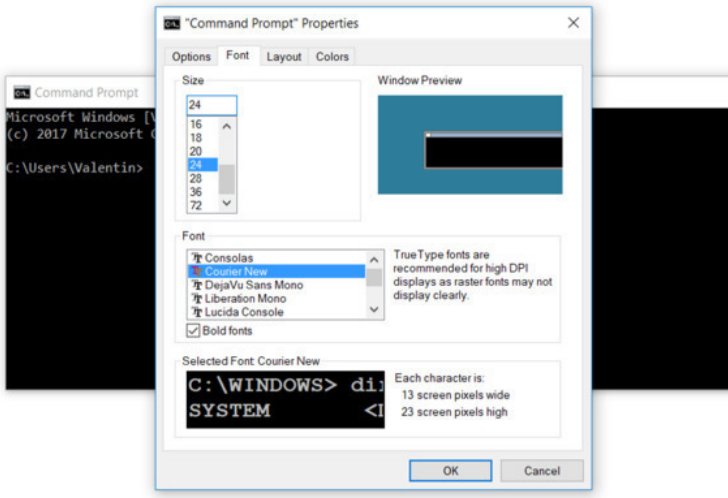


Рис. 3.3. Выбор шрифта командного окна

Задание. Настройте шрифт командного окна.

После установки шрифта ещё раз запускаем программу на выполнение.

Теперь буковки стали более читабельными (рис. 3.4).

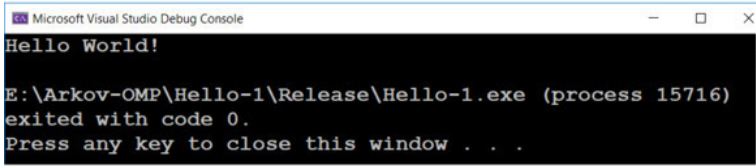


Рис. 3.4. Результаты работы программы

Задание. Убедитесь, что настроили шрифт командного окна.

3.2. Поддержка OpenMP

Проверим, как есть ли в нашем компиляторе поддержка распараллеливания.

Если в компиляторе включена поддержка OpenMP, то появится следующий идентификатор:

_OPENMP.

Для проверки задания макроса используется директива IF DEFINED:

#ifdef.

Директива предлагает компилятору проверить, определён ли указанный идентификатор.

Запускаем следующую программу (рис. 3.5).

E:\Arkov-OMP\omp-01-check\omp-01-check.c

```
 1  #include <stdio.h>
 2
 3  void main()
 4  {
 5  #ifdef _OPENMP
 6      printf("OpenMP Date: %d\n", _OPENMP);
 7  #else
 8      printf("Sorry, No OpenMP\n");
 9  #endif
10  }
```

Рис. 3.5. Проверка поддержки OpenMP

Запущенная программа сообщает нам, что указанный макрос не определён (рис. 3.6).

Это означает, что после компиляции мы получаем самую обычную последовательную программу.

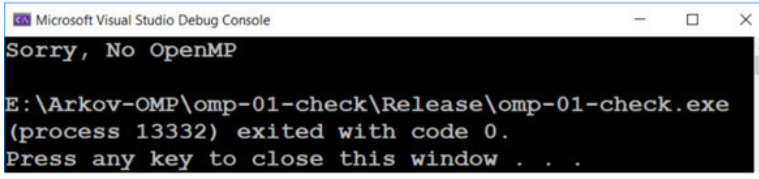


Рис. 3.6. Поддержки OpenMP пока нет

Задание. Составьте и запустите программу (рис.3.5).

Теперь включим поддержку распараллеливания. Настроим свойства текущего проекта:

Project – Properties (рис. 3.7).

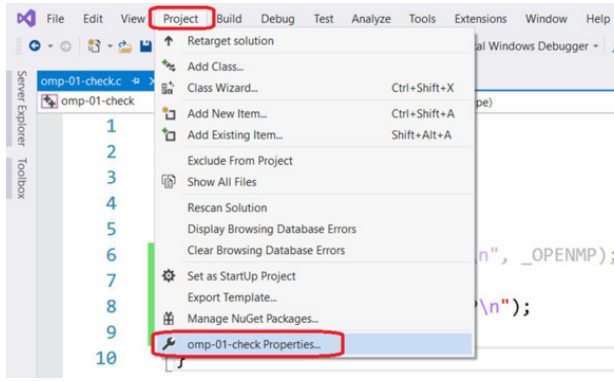


Рис. 3.7. Свойства проекта

Делаем следующие настройки (рис. 3.8):

Configuration – All Configurations;

Configuration Properties – C/C++ Language – OpenMP Support – Yes.

В нижней части окна нам сообщают, что поддерживается версия 2.0:

OpenMP Support – Enable OpenMP 2.0 language extensions.

Здесь же показана опция командной строки, которая используется при вызове компилятора:

/openmp.

Нажимаем клавиши:

Apply – OK.

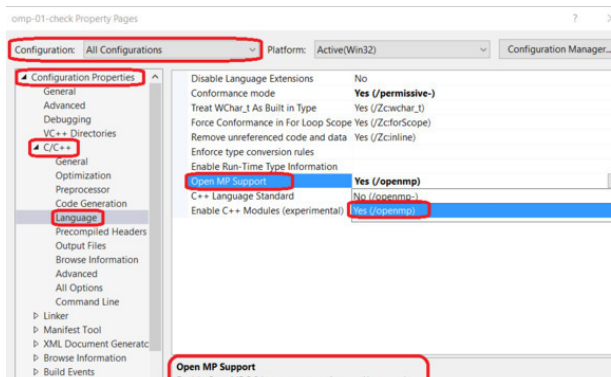


Рис. 3.8. Настройка свойств проекта

Задание. Включите поддержку OpenMP.

В окне редактирования программы сразу видно, что у нас включена поддержка OpenMP (рис. 3.9). Наводим курсор на идентификатор **_OPENMP** и видим, что он был определён.

Где-то ещё имеется следующая строчка:

#define _OPENMP 200203.

Это означает, что перед компиляцией программы все ссылки на данный идентификатор заменяются на **200203**.

Кроме того, теперь у нас подсвечивается ветка, которая фактически будет вставлена в текст программы.

```
1     #include <stdio.h>
2
3     void main()
4     {
5     #ifdef _OPENMP
6         printf("OpenMP Date: %d\n", _OPENMP);
7     #else
8         printf("Sorry, No OpenMP\n");
9     #endif
10    }
```

#define _OPENMP 200203
Expands to: 200203
Search Online

Рис. 3.9. Окно редактора с поддержкой OpenMP

Задание. Изучите изменения в окне редактора.

Запускаем программу.

На экран выводится то самое загадочное число **200203** (рис. 3.10). Возможно, у вас будет что-то другое.

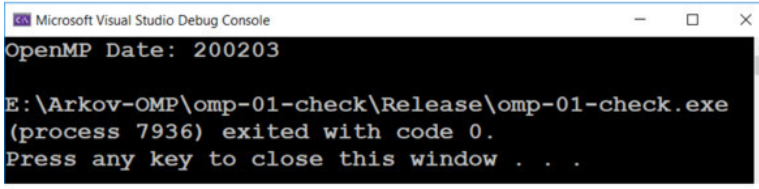


Рис. 3.10. Сообщение о поддержке OpenMP

Задание. Запустите программу на выполнение.

Полученное число сообщает нам год и месяц выпуска очередной версии спецификации.

Чтобы понять смысл сообщения, вернемся к списку версий:

https://www.openmp.org/specifications/

Находим дату: март 2002 года (рис. 3.11).

Читаем, какая версия была выпущена тогда.

Для первого знакомства с технологией и для наших простых экспериментов этого более чем достаточно. Мы не используем сложные и продвинутые инструменты.

Рис. 3.11. Версии спецификаций

Задание. Найдите версию спецификации для языка Си по дате выпуска.

3.3. Первая параллельная программа

Компилятор работает и поддерживает распараллеливание.

Составим самую первую параллельную программу.

Добавляем одну строчку (рис. 3.12):

#pragma omp parallel.

Мы объявляем часть программы, которая будет одновременно выполняться несколькими потоками.

E:\Arkov-OMP\omp-02-hello\omp-02-hello.c

```
1  #include <stdio.h>
2
3  void main()
4  {
5  #pragma omp parallel
6      printf("Hello World!\n");
7  }
```

Рис. 3.12. Параллельная программа

Запускаем программу и… ничего не изменилось (рис. 3.13).

Это всё ещё последовательная программа «Всем привет!»

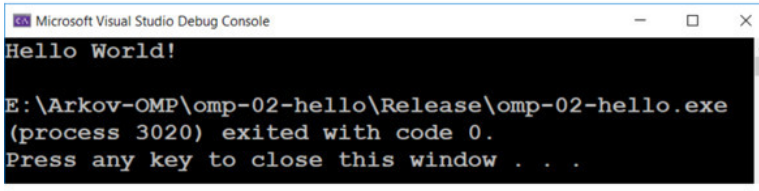


Рис. 3.13. Последовательная программа

Задание. Создайте и запустите программу (рис. 3.12).

Настроим поддержку OpenMP для текущего проекта.

Эту настройку придётся делать для каждого нашего проекта.

Запускаем программу и получаем восемь одинаковых сообщений (рис. 3.14).

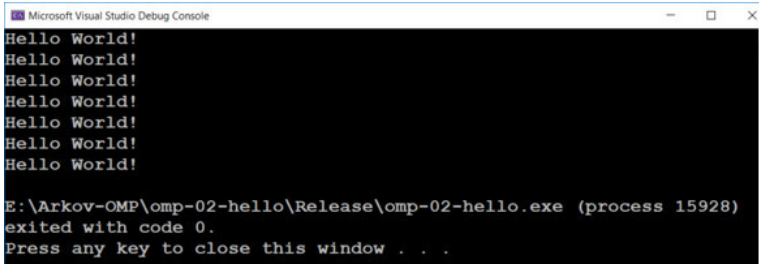


Рис. 3.14. Параллельное выполнение программы

Задание. Настройте проект и запустите параллельную

программу.

Почему именно восемь?

Запустим **Диспетчер задач – Task Manager** (рис. 3.15). Для этого одновременно нажимаем три «волшебных» клавиши:

Ctrl + Alt + Del.

Кстати, как правильно произносится английское слово CONTROL? Большинство опрошенных студентов говорят с ударением на первом слоге. Потому что в английском обычно «ударяют» на первый слог. Но это правило работает не всегда. Особенно для тех слов, которые пришли из французского.

Задание. Выясните, как правильно произносится слово CONTROL.

Итак, мы запустили Диспетчер задач:

Task Manager.

Переходим на вкладку Быстродействие:

Performance.

Перед нами появляется некоторое количество окошек с графиками – по числу логических процессоров.

Если у нас всего один график, щёлкаем по нему правой кнопкой и выбираем в контекстном меню:

Change graph to – Logical processors.

Здесь нам намекают именно на «логические процессоры»,

а не на физические процессоры и не на ядра процессора. Для операционной системы и для параллельной программы эти различия и тонкости не слишком важны. В частности, на нашем компьютере один процессор, в котором имеются четыре ядра с «гипертредингом». **HyperThreading** – это технология компании Intel для выполнения двух потоков на одном физическом ядре. Итого имеем восемь виртуальных, логических процессоров.

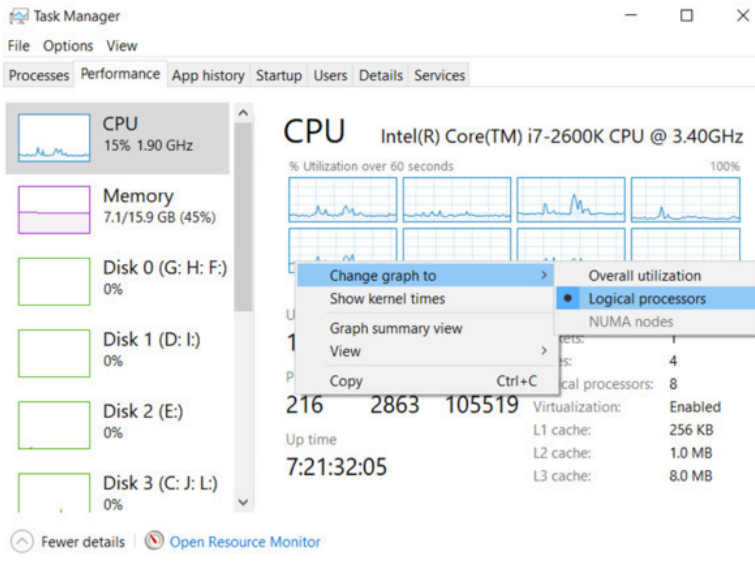


Рис. 3.15. Диспетчер задач

Задание. Выясните количество виртуальных процессо-

ров на своём компьютере.

3.4. Параллельные и последовательные области

Как мы уже говорили, здесь используется модель «fork-join».

Составим программу, в которой организуем три области:

1) первая последовательная область печатает единичку;

2) параллельная область печатает двоечку – несколько раз, по количестве виртуальных процессоров;

3) вторая последовательная область печатает троечку.

Конец ознакомительного фрагмента.