МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
Национальный технический университет Украины
"Киевский политехнический институт"

Кафедра вычислительной техники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ
и расчетно-графической работы по курсу
"Параллельные и распределенные вычисления"
(часть 1)

Киев – 2009/10 г.

Целью выполнения лабораторных и расчетной работы по курсу "Параллельные и распределенные вычисления" (семестр 5) является практическое закрепление теоретических знаний и умений по разработке программ для параллельных вычислительных систем (ПВС), заключающихся в получении практических навыков при работе с процессами (потоками) в языках параллельного программирования Java, Ада, C# и библиотекахWin32, MPI .

СОДЕРЖАНИЕ И ОФОРМЛЕНИЕ ЛАБОРАТОРНЫХ РАБОТ

Цикл лабораторных работ по курсу составляют ***семь*** заданий. Задания охватывают изучение четырех основных видов программных модулей, используемых в языке Ада: подпрограмм, пакетов, настраиваемых модулей и задач. Еще пять заданий связаны с изучением особенностей реализации механизма потоков в в языках параллельного программирования Java, Ада, C# и библиотекахWin32, MPI .

.

Расчетно-графическая работа связана с изучением организации взаимодействия процессов при использовании посылки сообщений. При этом рассматривается механизм рандеву, реализованный в языке Ада.

Для Расчетно-Графической работы дополнительно необходимо предусмотреть:
- построение структуры взаимодействия процессов в программе **(Приложение Е);**
- описание алгоритма каждой задачи и ее взаимодействия с другими задачами
- анализ структурного графа на тупиковые ситуации.

Для выполнения лабораторных работ необходимо получить у преподавателя вариант первого задания, включающий номера функций F1, F2, F3 из **Приложения Б** (1.x, 2.x, 3.x). Функции связаны с выполнением операций над векторами и матрицами, поэтому следует восстановить соответствующие знания линейной алгебры для корректной реализации этих операций из курса «Линейная алгебра и аналитическая геометрия ».

На выполнение каждого лабораторного задания отводится **два** лабораторных занятия. После *успешной* сдачи первой работы студент получает задание на вторую работу. Если задание сдано в оговоренные сроки, то студент продолжает работу с теми же функциями, то есть каждая последующая работа будет являться продолжением и модернизацией предыдущей работы и не потребует перепрограммирования функций F1, F2, F3. Если сроки выполнения задания не соблюдаются, то на выполнение каждой новой работы выдается *новый вариант* функций F1, F2, F3. Задание на расчетно-графическую работу выдается после сдачи пятого задания по лабораторным работам. Задания на выполнение лабораторных работ и образцы их оформления приведены в **Приложениях А- Е.**

При оформление текста программы следует обратить внимание на то, что в тексте программы ключевые слова языка Ада писать строчными, а объекты, вводимые программистом (имена переменных, типов, подпрограмм) - прописными и по возможности с использованием кириллицы. С помощью строки разделителей и комментарий выделять процедуры, задачи и основные смысловые части программы. Название типов, переменных, подпрограмм, пакетов должны быть связаны их назначением (например, тип `Вектор`, переменная `Буфер`, процедура `Поиск_Максимального_Элемента` и др. [9, с. 11-29].

Отчет по лабораторной работе включает листинг программы и результаты работы программы.

Лабораторная работа N 1.
ПОДПРОГРАММЫ И ПАКЕТЫ

Цель работы: изучение структуры программы и особенностей реализации механизма подпрограмм и пакетов в языке Ада

Выполнение работы:
Часть 1. Разработать программу, содержащую *подпрограммы* `Func1, Func2, Func3` для вычисления трех функций F1, F2, F3, представленных соответствующим вариантом в **Приложении Б.** При разработке процедур разделять формальные параметры на входные (**in**) и выходные (**out**). Изучить команды и опции компилятора ObjectAda, необходимые для компиляции и редактирования связей программы.

Часть 2.Разработать и реализовать библиотечный пакет, ресурсами которого являются
- подпрограммы `Func1, Func2, Func3`,,
- необходимые ***т и п ы*** (например, `Vector и Matrix`)
- дополнительные процедуры ввода/вывода (`Vector_Input, Vector_Output, Matrix_Input, Matrix_Output`).

Ввести ***л и ч н ы й*** ***(private)*** тип и показать особенности его использования Показать использования данного пакета пользователем.

Для получения повышенной оценки реализовать пакет в виде настраиваемого пакета.

Необходимые теоретические сведения: Программа на языке Ада содержит основную программу (процедуру без параметров) и набор программных модулей. Язык включает пять видов модулей, из которых строиться программа: подпрограммы, пакеты, настраиваемые модули, задачи и защищенные модули. Модули могут быть непосредственно описаны в теле основной программы или находиться в библиотеках (библиотечные модули).

Подпрограммы – простейший вид программных модулей языка. Включают процедуры и функции. Реализация подпрограмм в языке - классическая, аналогично тому, как это сделано в Паскале. Особенностью является жесткое разделение формальных параметров процедур на входные (**in**) и выходные (**out**), позволяющее контролировать их корректное использование, как в самой подпрограмме, так и при ее вызове. Это повышает надежность программы (например, за счет запрета на изменение входных параметров, что обеспечивает их целостность при обработке в процедуре). Функции имеют параметры только вида **in** (входные).

Теоретические сведения по реализации подпрограмм в языке можно найти в [3-9].

Пакет является основным инструментом построения Ада приложения. Инкапсулирует ресурсы (типы, константы, переменные, подпрограммы и другие программные модуля) для последующего использования.

Обратить внимание на корректное формирование спецификации пакета. Спецификация есть интерфейс пакета и определяет те его ресурсы, которые (и только которые) будут доступны внешнему пользователю пакета. Поэтому тщательно продумывайте спецификацию пакета, вынося туда только те ресурсы, которые будут доступны для внешнего использования, тем самым обеспечивая защищенность остальных ресурсов пакета. Не надо путать спецификацию пакета с описательной частью всех его ресурсов. Пакет в общем случае может содержать ресурсы, необходимые для реализации других ресурсов пакета и не предназначенные для внешнего пользователя. В этом случае эти вспомогательные ресурсы не указываются в спецификации пакета и описываются непосредственно в теле пакета. Надежность пакета увеличивается с минимизацией списка его ресурсов в спецификации, когда туда *вы-*

носятся только жизненно важные для последующего использования ресурсы разрабатываемого пакета (полное сокрытие видимости ресурсов пакета).

Еще одно средство повышения надежности пакета основывается на ограничении (ужесточении) доступа к его видимым ресурсам (частичное ограничение видимости ресурсов пакета). Обеспечивается приватными (**private**) и ограниченными (**limited**) типами данных. Для объектов таких типов *автоматически вводятся и контролируются ограничения* на их использование вне пакета, заключающиеся в том, что изначально *запрещаются все* операции, кроме нескольких фиксированных элементарных операций. Все другие необходимые операции над объектами приватных (ограниченных) типов должны быть реализованы разработчиком с помощью введения дополнительных подпрограмм в спецификации пакета.

Пакеты, использующие приватные типы, имеют приватную часть спецификации, в которой раскрывается действительная реализация приватного типа, которая предназначена только для компилятора, видима в теле пакета и используется разработчиком пакета при реализации видимых ресурсов пакета.

Теоретические сведения по реализации пакетов и настраиваемых модулей можно найти в [3-9].

Лабораторная работа N 2.
ИЕРАРХИЧЕСКИЕ БИБЛИОТЕКИ. РАЗДЕЛЬНАЯ КОМПИЛЯЦИЯ. ИСКЛЮЧЕНИЯ

Цель работы: изучение механизма пакетов объектно-ориентированного программирования на примере построения иерархических библиотек, изучение средств и методов построения сложных программ, изучение средств и методов обеспечения их надежности выполнения программы.

Выполнение работы Разделить пакет из лабораторной работы № 2 на четыре пакета, поместив в первый пакет реализацию функций `Func1` и типа `Vector`, во второй – реализацию функции `Func2` и тип `Matrix`, в третий – реализацию функции `Func3`, а в последний - реализацию дополнительных процедур ввода-вывода `Vector_Input, Vector_Output, Matrix_Input, Matrix_Output`). Установить между пакетами отношения иерархии, выбрав первый пакет в качестве ***родительского***, а остальные пакеты реализовав через ***дочерние*** модули (***приватные и публичные***).

Организовать ***раздельную*** компиляцию пакета и одной из подпрограмм из лаб.работы N 2.(спецификации и тела).

Рассмотреть три варианта раздельной компиляции:

- пакет является библиотечным модулем (отдельный файл);
- спецификация и тело пакета размещены в отдельных файлах;
- спецификация пакета вложена в пользовательскую программу, тело находится в отдельном файле.

Реализовать обработку ***исключительных*** ситуаций, связанных с вводов/выводом (ввод неправильного символа) и счетом (деление на 0 или переполнение) в лабораторной работе № 4.

Необходимые теоретические сведения: Ада является языком объектно-ориентированного программирования. Инкапсуляция обеспечивается механизмом пакетов. Полиморфизм – перекрывающимися подпрограммами. Наследование - механизмом иерархических библиотек и

тэговых типов. Новый пакет B может быть реализован путем расширения уже существующего пакета A с помощью нотации A.B. При этом дочерний пакет A.B наследует ресурсы пакета A и может быть расширен путем добавления новых ресурсов. Правила наследования и видимости между родительскими и дочерними пакетами регулируются с помощью приватных разделов и приватных дочерних модулей.

Теоретические сведения по построению иерархических библиотек в языке можно найти в [3-9].

Ада обеспечивает разнообразные средства построения больших программных продуктов, основанные на модульных и объектно-ориентированных технологиях.

Раздельная компиляция позволяет формировать программу в виде набора отдельных модулей, что важно при разработке больших и сверхбольших программных систем. При этом возможна раздельная компиляция каждого модуля, когда одновременно проверяется и правильность согласованности между модулями. За счет универсальной структуры всех модулей языка, включающей спецификацию и тело, язык обеспечивает три вида организации раздельной компиляции:
- раздельная компиляция основной программы и пакета
- раздельная компиляция основной программы, спецификации пакета и тела пакета
- раздельная компиляция основной программы и тела пакета (спецификации пакета вложена в основную программу.

Теоретические сведения по организации раздельной компиляции и реализации механизма исключений в языке можно найти в [3-9].

Ада обеспечивает поддержку надежного выполнения программ, основанную на использовании механизма исключительных ситуаций.

Надежность – основная отличительная особенность языка Ада, которая обеспечивается всеми его составляющими на всех этапах разработки и выполнения программы. Механизм исключений (исключительных ситуаций) реализует средства, обеспечивающие надежность выполнения Ада приложения. Механизм исключений в языке реализован с помощью стандартных и пользовательских исключений (**`exception`**), а также обработчиков исключений. Если при выполнении программы происходит исключительная ситуация (например, переполнение), то выполнение программы прерывается (а не завершается) и управление передается обработчику данного исключения, который выполняет действия по устранению данной ошибки и продолжению выполнения прерванной программы.

Теоретические сведения по организации реализации механизма исключений в языке можно найти в [3-9].

Лабораторная работа N 3.
ПРОЦЕССЫ В ЯЗЫКЕ АДА

Цель работы: изучение средств языка Ада для работы с процессами.
Выполнение работы: Разработать программу, содержащую ***п а р а л л е л ь н ы е*** задачи, каждая из которых реализует функцию F1, F2, F3 из лабораторной работы номер 1 или 2. Задачи независимы, общих данных *не имеют!*

Исследовать влияние приоритетов задач и оператора задержки **`delay`** на очередность запуска и выполнения задач.

Необходимые теоретические сведения: язык Ада обеспечивает возможность программирования параллельных процессов с помощью задачных модулей **(task**). Задачи обеспечивают параллельное выполнение частей одной программы в параллельных вычислительных системах или конкурирующее - в последовательных. Управление выполнением задач можно осуществлять с помощью установления приоритетов задач (прагма **priority**), а также оператора **delay,** который вызывает приостановку (блокирование) задачи на указанный отрезок времени. При этом задача блокируется и управление передается другой задаче, готовой к выполнению.

Задачи как программный модуль имеют стандартную для модулей языка структуру, то есть состоят из спецификации и тела. Спецификация задачи позволяет описать имя задачи, ее приоритет, средства взаимодействия с другими задачами (входы задачи) и др. Тело задачи определяет ее действия при выполнении. Для описания задач также используется задачный тип, важной составляющей которого является дискриминант, позволяющей параметризацию типа и соответственно создание разных задач на основе одного шаблона.

Теоретические сведения по программированию задач и управлению задачами в языке Ада можно найти в [1. с. 27-29, 2. с.15-19].

Лабораторная работа N 4.
ПОТОКИ В ЯЗЫКЕ JAVA

Цель работы: изучение средств языка Java для работы с потоками (процессами).
Выполнение работы: Разработать программу, содержащую ***п а р а л л е л ь н ы е*** потоки, каждый из которых реализует функцию F1, F2, F3 из лабораторной работы номер 2. Исследовать влияние приоритетов потоков и методов **sleep()** и **join()** на очередность запуска и выполнения задач.
Необходимые теоретические сведения: язык Java обеспечивает возможность программирования параллельных процессов с помощью потоков **(threads**). Для этого используется класс **Thread** или интерфейс **Runnable.** Потоки обеспечивают параллельное выполнение частей одной Java программы в параллельных вычислительных системах или конкурирующее - в последовательных системах. Управление выполнением задач можно осуществлять с помощью установления приоритетов потоков (метод **set_Priority()**), а также метода **sleep(),** который вызывает приостановку (блокирование) потока на указанный отрезок времени. При этом поток блокируется и управление передается другому потоку, готовому к выполнению.

Рассмотреть использование метода **join()** для синхронизации основного метода с потоками, которые он запускает на выполнение.

Создаваемый поток должен переопределять метод **run(),** который определяет действия данного потока при выполнении. При создании экземпляра касса – потока следует использовать соответствующий конструктор, с помощью которого задать внутренне имя потока, его приоритет, особенности поведения и др.

Теоретические сведения по программированию потоков и управлению ими в языке Java можно найти в [1. с. 22-26, 2. с.8-14].

Лабораторная работа N 5.
ПОТОКИ В ЯЗЫКЕ C#

Цель работы: изучение средств языка C# для работы с потоками (процессами).

Выполнение работы: Разработать программу, содержащую ***п а р а л л е л ь н ы е*** потоки, каждый из которых реализует функцию F1, F2, F3 из лабораторной работы номер 2. Исследовать влияние приоритетов потоков на очередность запуска и выполнения задач.
Необходимые теоретические сведения: язык C# обеспечивает возможность программирования параллельных процессов с помощью потоков и операций Sleep, Priority, Join, Start класса Thread.

Теоретические сведения по программированию потоков и управлению ими в языке C# можно найти в [2. с.33-38].

Лабораторная работа N 6.
ПОТОКИ В БИБЛИОТЕКЕ WIN32

Цель работы: изучение средств библиотеки Win32 для работы с потоками (процессами).
Выполнение работы: Разработать программу, содержащую ***п а р а л л е л ь н ы е*** потоки, каждый из которых реализует функцию F1, F2, F3 из лабораторной работы номер 2. Исследовать влияние приоритетов потоков и на очередность запуска и выполнения задач.
Необходимые теоретические сведения: библиотека Win32 обеспечивает возможность программирования параллельных процессов с помощью потоков (`threads`) и потоковых функций.

Теоретические сведения по программированию потоков и управлению ими в языке Win32 можно найти в [1. с. 35-36, 2. с.22-27].

Лабораторная работа N 7.
ПОТОКИ В БИБЛИОТЕКЕ MPI

Цель работы: изучение средств библиотеки MPI для работы с задачами (процессами).
Выполнение работы: Разработать программу, содержащую ***п а р а л л е л ь н ы е*** потоки, каждый из которых реализует функцию F1, F2, F3 из лабораторной работы номер 1 или 2.

Необходимые теоретические сведения: библиотека MPI обеспечивает возможность программирования параллельных процессов с помощью задач (`tasks`). Для этого используются функции библиотеки MPI.

Теоретические сведения по программированию потоков и управлению ими в языке MPI можно найти в [1. с. 30-34, 2. с.20-21].

Расчетно-графическая работа

Цель работы: изучение средств языка для организации взаимодействия процессов, основанного на модели посылки сообщений.

Выполнение работы: Разработать программу, содержащую набор *параллельных* задач (T1 – T6) для реализации функций F1, F2, F3 из лабораторной работы номер 1, а также задач для ввода и вывода данных и результатов для этих функций. Организовать запуск указанных задач, а также передачу необходимых данных и результатов между соответствующими задачами с помощью *механизма рандеву*.

Варианты расчета соответствующих функций в задачах, а также задач для ввода и вывода данных приведены в **Приложении Д.** Для каждой задачи разработать и представить

алгоритм ее выполнения, включающий шаги связанные с вводом-выводом данных, счетом соответствующей функции, а также приемом или передачей информации другой задаче.

Например:

Алгоритм задачи Т3

1. Принять от Т1 вектора А и В
2. Принять от Т4 матрицу МС
3. Выполнить счет функции X = F3(A, B, MC)
4. Передать вектор X в задачу Т2

Пример структурной схемы взаимодействия задач при использовании посылки сообщений (механизма рандеву) приведен в **Приложении Е.**

Необходимые теоретические сведения: взаимодействие задач, основанное на передаче сообщений в языке Ада реализовано с помощью механизма рандеву. Рандеву основывается на использовании операторов входа и вызова входа. Задача, имеющая описанный в спецификации вход, может быть вызвана из другой задачи для осуществления приема или передачи данных. Описание входа определяет протокол взаимодействия задач, так как он задает объем и направление передаваемой информации. При формировании операторов входа следует обращать внимание на минимизацию объемов передаваемой информации, так как это в итоге влияет на время выполнения программы.

Рандеву объединяет передачу данных с синхронизацией, так как передача данных осуществляется только в том случае, если взаимодействующие задачи достигли определенных точек (точек синхронизации), которыми являются оператор вызова входа в вызываемой задаче и оператор принятия вызова входа (**`accept`**) в вызываемой задаче. Если одна из взаимодействующих задач достигла своей точки синхронизации (например, оператор **`accept`**) раньше другой, то она блокируется до тех пор, пока вторая задача не выйдет на свою точку синхронизации (оператор вызова данного входа). Возможность блокирования задач во время рандеву может привести к тупиковой ситуации (**`deadlock`**), когда обе задачи не могут выйти на точки синхронизации, блокируются и выполнение программы становится невозможным. Борьба с тупиками является одной из важных проблем при программировании параллельных процессов.

Запуск задачи осуществляется автоматически, если задача описана в основной программе. Принудительное выполнение задач может быть осуществлено либо путем вызова подпрограммы, в которой эта задача описана, либо использованием задачного типа, ссылочного типа и генератора **`new.`**

Теоретические сведения по реализации механизма рандеву можно найти в [1, с. 112-116, 72 с.130-133]. Графическая нотация для отображения взаимодействия задач представлена в [3, с. 54-74].

Образцы титульного листа и лист задания на расчетно-графическую работу приведены в **Приложениях В и Г.**

Л И Т Е Р А Т У Р А

1. Жуков І.А., Корочкін О.В. Паралельні та розподілені обчислення.- К.:Корнійчук, 2005. - 224 с.
2. Жуков И.А., Корочкин А.В. Параллельные и распределенные вычисления. Лабораторный практикум. – К.: Корнейчук, 2008.- 226 с.
3. Бар Р. Язык Ада в проектировании систем. - М.; Мир, 1988.- 320с.
4. Пайл Я. Ада - язык встроенных систем. -М.; Финансы и статистика, 1984. - 120 с.
5. Джехани Н. Язык Ада. - М.; Мир, 1988.- 552 с.
6. Василеску Ю. Прикладное программирование на языке Ада.- М.:Мир,1990. - 332 с.
7. Вегнер П. Программирование на языке Ада. - М.; Мир, 1983.- 78 с.
8. Органик Э. Организация системы ИНТЕЛ - 432. - М.; Мир, 1987.-224 с.
9. Корочкин А.В. Ада 95: Введение в программирование. - Киев; Свит, 1998.- 260 с.
10. Бройнль Т. Паралельне програмування. Початковий курс: Навч. Посібник - Київ.: Вища школа, 1997. – 358 с.
11. Соловьев Г.Н., Никитин В.Д. Операционные системы ЭВМ.- М.: Высшая школа, 1989. - 255 с.
12. Дейтел Д. Введение в операционные системы.- М.:: Мир, 1989, - 360 с.
13. Элементы параллельного программирвания / Вальковский В.А., Котов В.Е., Марчук А.Г.,/ Под ред. Котова В.Е. – М.: Радио и связь, 1983.- 240 с.
14. Воеводин В.В. Математические модели и методы в параллельных процессах. – М.: Наука, 1984. – 296 с.
15. Миренков Н.Н. Параллельное программирование для многомодульных вычислительных систем. – М.: Радио и связь, 1989. -320 с.
16. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. – М.: Радио и связь, 1989, с. 280
17. Параллельные вычисления/ Под ред.Г. Родрига: Пер. с англ./ Под ред.
18. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ : Пер. с англ. – М.: Мир,1989. - 424 с.
19. Корочкин А.В. Ада 95: Введение в программирование. - Киев; Свит, 1998.- 260 с.
20. Гома Х. UML. Проектирование систем реального времени, параллельных и распределенных приложений. – М.: ДМК Пресс, 2002. – 704 с.
21. Богачев К.Ю. Основы параллельного программирования. – М.: БИНОМ. Лаб знаний, 2003. – 342 с .
22. Корочкин А., Мустафа Акрам Параллельные вычисления: Ада и Java. – Вісн. НТУУ “КПІ”, Інформатика, управління та обчислювальна техніка, 1999, К.: – № 32, С. 13–17.
23. Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ – Петербург, 2002. – 400 с.
24. Ноутон П., Шилдт Г. Java2: Пер. с англ. – СПб.: БХВ – Петер бург, 2000. – 1072 с.
25. Симкин С., Барлетт Н., Лесли А. Программирование на Java. Путеводитель – К.: НИПФ “ДиаСофт Лтд.”, 1996. – 736 с.
26. Эндрюс Г. Основы многопоточного, параллельного и распределенного программирования.: Пер. с англ. – М.: Изд. Дом «Вильямс», 2003. – 512 с.

ЛОЖЕНИЕ А. Условные обозначения.

a - скалярная величина
A - вектор размерности N
MA - матрица размерности NxN

a*B - произведение вектора на скаляр
a*MB - произведение матрицы на скаляр
(A*B) - скалярное произведение векторов A и B

(MA*MB) - произведение матриц MA и MB
(MA*B) - произведение матрицы на вектор
SORT(A) - сортировка вектора A по возрастанию
MIN(A) - поиск минимального элемента вектора
MAX(A) - поиск максимального элемента вектора

TRANS(MA) - транспонирование матрицы MA
MAX(MA) - поиск максимального элемента матрицы
MIN(MA) - поиск минимального элемента матрицы
SORT(MA) - сортировка строк матрицы по убыванию

ПРИЛОЖЕНИЕ Б. ВАРИАНТЫ ФУНКЦИЙ F1, F2, F3 ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

1. **Операции с векторами** (Функция F1)

1.1 A = SORT(B)
1.2 C = A + B
1.3 C = A – B*x
1.4 C = A + SORT(B)
1.5 C = SORT(A) - SORT(B)
1.6 a = (B*C)
1.7 b = (A*SORT(C))
1.8 a = MAX(B)
1.9 b = MIN(A+C)
1.10 A = B*MIN(C)

1.11 c = MAX(A)*(A*B)
1.12 A = B + C – D*e
1.13 C = A - B + D
1.14 D = SORT(A + B) - C
1.15 d = MAX(A + B + C)
1.16 d = ((A + B)* C)
1.17 d = (A * (B + C))
1.18 d = (A*B) + (C*B)
1.19 d = MAX(B - C) + MIN(A + B)
1.20 D = MIN(A + B) * (B – C + X)

1.21 D = SORT(A) + SORT(B) - SORT(C)
1.22 d = (B*C) - (A*B) +(C*B)
1.23 E = A + B + C – D*e
1.24 E = A + C – B*e + D
1.25 e = ((A +B)*(C + D))
1.26 e = ((A + SORT(B)*(C + SORT(C))
1.27 e = (A*B) + (C*D)
1.28 E = MAX(A)*(A - B + C)
1.29 E = A*(B*C) + D*e
1.30 e = ((A + SORT(B))*(C - SORT(D))

2. Операции с матрицами (Функция F2)

2.1 MA = MB + MC*MT
2.2 MA = MC*MP - MB
2.3 MC = MA*MB*e
2.4 x = MAX(MB*MM)
2.5 MX = SORT(MA*MM)
2.6 MC = TRANS(MB*MM)
2.7 MB = a*MA + c*MZ*MK
2.8 MB = b*TRANS(MA*MM)
2.9 MC = TRANS(MA*MT)*TRANS(MB)
2.10 MC = MA*MH - TRANS(MB)]

2.11 MD = MAX(MA*MM)*MB
2.12 MA = TRANS(MB) + MC*MM
2.13 MZ = MIN(MA)*MB + MAX(MB)*MA
2.14 MC = SORT(MA + MB*MO)
2.15 MR = SORT(MA*MB)
2.16 MC = SORT(TRANS(MA)*MB)
2.17 a = MAX(MA – MB*MT)
2.18 c = MIN(MA*MB)
2.19 v = MAX(MA + MB*MC)
2.20 MD = MA + MB – MC*MM
2.21 MD = MA + (MB*MC)
2.22 MT = (MA - MB)*(MA + MC)
2.23 q = MAX(MA *MM+ MB - MC)
2.24 MG = SORT(MA – MB*MT - MC)
2.25 MU = SORT(MA *TRANS(MB) - TRANS(MC))
2.26 MD = (MA*MB)*MC
2.27 MD = MA*(MB*TRANS(MC))
2.28 MD = MIN(MA)*(MB*MC)
2.29 MD = (MA + MB)*(MA *MC)*(MB + MA)
2.30 d = MAX(MA*MB) - MIN(MB + MC)

3. Операции с векторами и матрицами (Функция F3)

3.1 A = MB*C
3.2 B = TRANS(MC)*A
3.3 C = SORT(A)*MB
3.4 B = SORT(A)*SORT(MB)
3.5 A = (SORT(MB)*C)
3.6 A = MAX(MB)*C

3.7 D = (A+B)*MC
3.8 D = (MA -MB)*C
3.9 D = SORT(A)*(MA + MB)
3.10 D = SORT(A + B)*MC

3.11 D = SORT(A - M)*TRANS(MC)
3.12 E = MC*A + MB*B
3.13 E = MA*B + MB*SORT(A)
3.14 D - (A + B)*(MA + MB)
3.15 S = (B - C)*TRANS(MA + MB)
3.16 d = MAX(MA*A + MB*C)
3.17 d = MIN(A*TRANS(MB) - B*SORT(C))
3.18 p = MAX(SORT(MS) + MA*MB)
3.19 T = (MA*MB)*(A + B)
3.20 R = SORT(B + C)*SORT(MA*MB)

3.21 W = SORT(B*MD)*(MA -MB)
3.22 S = SORT(MA*T) - F
3.23 e = MAX(MA*(B - C))
3.24 k = MIN(MA*MB*E)
3.25 E = (A + B - C)*(MA - MB)
3.26 s = MAX(MA*B + MB*C - R)
3.27 E = SORT(MA*MA*C - D + W)
3.28 e = MAX(MA*B) - MIN(MA*(B + C))
3.29 F = MA*D + MB*C + MX*E
3.30 K = MA*MV*C - a*MB*(A - B)

ПРИЛОЖЕНИЕ В. Титульный лист для оформления расчетно-графической работи

М І Н І С Т Е Р С Т В О О С В І Т И І Н А У К И У К Р А Ї Н И
Національний технічний університет України
"Київський політехнічний інститут"
Кафедра обчислювальної техніки

РОЗРАХУНКОВО-ГРАФИЧНА РОБОТА

з дисципліни
"Паралельні та розподілені обчислення"

Керівник роботи
Доц. Корочкин А.В.

Виконавець роботи
ст. Іванов А.А.

Допущена к захисту
_____»__» 2009 р.

група №

Захищена
_____»__» 2009 р.

Київ - 2009 р.

ПРИЛОЖЕНИЕ Г. Лист задания на расчетно-графическую работу

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАІНИ
Національний технічний університет України
"Київський політехнічний інститут"
Кафедра обчислювальної техніки

Факультет_____

Кафедра_____

ЗАВДАННЯ НА РОЗРАХУНКОВО-ГРАФИЧНУ РОБОТУ

з дисципліни _____

Група_____ студент_____ Термін отримання_____

Термін виконання_____Керивник роботи_____

НАИМЕНУВАННЯ ЗАВДАННЯ

_____
_____
_____
_____
Дани_____
_____
_____
_____

ПЕРЕЛИК ГРАФИЧНОІ ДОКУМЕНТАЦІІ

_____
_____
_____

РЕКОМЕНДОВАНА ЛИТЕРАТУРА

_____
_____
_____

ПОДПИС КЕРИВНИКА РОБОТИ_____

ПРИЛОЖЕНИЕ Д. Варианты заданий на выполнение РГР. Включают описание действий, которые будут выполнять параллельные задачи Т1 –Т6. Например, в варианте 3 :

- задача **Т1** вычисляет функцию F3. При этом она должна *принять* исходные данные от задач Т4 и Т5, а результаты счета *передать* в Т6.
- задача **Т2** вычисляет функцию F2 . При этом она должна *принять* необходимые исходные данные от задач Т4 и Т5, а результаты счета *передать* в Т6.
- задача **Т3** вычисляет функцию F1. При этом она должна *принять* необходимые исходные данные от задач Т4 и Т5, а результаты счета *передать* в Т6.
- задача **Т4** вводит вектора и *передает* их в задачи Т1 и Т3.
- задача **Т5** вводит матрицы и *передает* их в задачи Т3 и Т2 .
- задача **Т6** *принимает* от задач Т1, Т2, Т3 результаты счета функций F3, F2, F1 и выводит их.

Вариант РГР	**Задача Т1**	**Задача Т2**	**Задача Т3**	**Задача Т4**	**Задача Т5**	**Задача Т6**
1	Ввод матриц	F3 F1 Вывод скаляров	Вывод векторов	Вывод матриц	Ввод векторов F2	-
2	F1	F2	F3	Ввод данных	Вывод результатов	-
3	F3	F2	F1	Ввод векторов	Ввод матриц	Вывод результатов
4	-	F2	F1	Ввод и вывод векторов	Ввод и вывод матриц	F3
5	F1	F3	F2	Ввод матриц и вывод векторов	Ввод векторов и вывод матриц	-
6	F2	F1	F3 Вывод результатов	Ввод векторов	Ввод матриц	Запуск задач Т1-Т3
7	F2 Вывод скаляров	F1 Вывод векторов	F3	Ввод векторов	Ввод матриц	Вывод матриц
8	F1 Вывод векторов	F2 Ввод матриц	F3 Ввод векторов	Вывод матриц	-	Вывод скаляров
9	F1	F2 Вывод матриц	F3	Ввод матриц	Ввод векторов	Вывод векторов
10	F2 Ввод данных	F3	F1 Вывод результатов	-	-	-
11	F1	F3	F2	Вывод матриц	Ввод данных	Вывод векторов
12	Ввод данных	F1	F2	F3	Вывод результатов	
13	Вывод матриц	Ввод данных. Вывод векторов	F1	F2	F3	Вывод скаляров
14	Ввод данных. Вывод матриц	F1	F2	F3 Вывод векто-	-	-

				ров		
15	Ввод и вывод матриц	Вывод векторов	F1 Ввод векторов	F2	F3	-
16	F1 F2	Ввод данных и вывод результатов	F3	-	-	-
17	Вывод результатов	F2	F3 F1	Ввод данных	-	-
18	F1	F2 Ввод данных	F3	Ввод векторов	Ввод матриц	-
19	F1 Ввод данных и вывод результатов	F2	F3	-	-	Запуск задачи T3
20	F1 F3	F2	Запуск задач T1 и T2	-	Ввод данных и вывод результатов	-
21	F1 F3	F2 Вывод векторов	-	Ввод векторов	Ввод матриц	Вывод матриц
22	F3	F1 Ввод данных	F2	-	Вывод результатов	-
23	F3	F2 Вывод результатов	F1	-	-	Ввод данных
24	F2	Ввод матриц	F1 F3 Вывод матриц	Ввод скаляров	Ввод и вывод векторов	-
25	Ввод данных	F1 F2 F3	-	Вывод результатов	-	Запуск всех задач
26	Ввод векторов	Ввод матриц	Вывод матриц	F1 F2 F3	Вывод векторов	-
27	-	Вывод матриц	Вывод векторов	F1 Ввод векторов	F2 Ввод матриц	F3
28	F1	F2 Ввод матриц	F3	Вывод векторов	Ввод векторов	Вывод матриц
29	F3 Вывод матриц	F2	F2	Ввод векторов	Вывод векторов	Ввод матриц
30	Запуск задач T2-T6	F3	Ввод векторов	F2	Ввод скаляров	F1 Ввод матриц
31	Ввод данных	F1 F3	-	Вывод результатов	F2	-
32	F1	F2	F3	Ввод данных и вывод результатов	-	-
33	Вывод скаляров	Ввод данных. Запуск задач	F3 Вывод векто-	Вывод матриц	-	F2

		T6 и T3	ров			F1
34	Ввод матриц. Запуск задач T5 и T2	F3 F1 Вывод скаляров	Вывод векторов	Вывод матриц	F2 Ввод векторов	-
35	F1	Вывод матриц Вывод векторов	F2	Вывод и вывод скаляров	Ввод данных. Запуск всех задач	F3
36	F2 Вывод скаляров	F1 Ввод матриц	F3 Ввод векторов	Вывод векторов	Запуск всех задач	Вывод матриц
37	F2 Запуск задач T3 и T5	Ввод данных. Вывод векторов	F1	-	F3	Вывод матриц
38	F2	F1	Ввод данных. Вывод матриц	F3 Вывод векторов	-	Запуск задач T2-T4
39	-	-	F1 Вывод матриц	Запуск всех задач	F3 Вывод векторов и скаляров	Ввод данных
40	F2 F1	Ввод данных	F3 Вывод векторов	Вывод матриц и скаляров	-	-
41	Ввод матриц. Ввод скаляров	F3 F1 Вывод скаляров	Вывод векторов	Вывод матриц	F2 Ввод векторов	Запуск всех задач
42	Ввод данных. Запуск всех задач	-	F3	Вывод результатов	F1	F2
43	Запуск всех задач	Вывод скаляров	F2 Вывод матриц	F3 F1	Ввод данных	Вывод векторов
44	F3	Вывод результатов	F1	Ввод векторов	F2	Ввод матриц. Запуск задач T1,T3, T5
45	Ввод векторов	Запуск задач T4 и T5	Ввод матриц	F2	F3	F1 Вывод результатов
46	F1 Вывод матриц и скаляров	Запуск задачи T4	-	F2 F3	Ввод данных	Вывод векторов
47	-	Ввод матриц F3	Вывод результатов	F1	F2 Вывод скаляров	Ввод векторов. Запуск задач T4 и T5
48	Запуск задач T2 и T6	F3	Вывод скаляров	-	Ввод данных и вывод результатов	F1 F2
49	F1 F3	F2 Вывод векторов	Запуск всех задач	Ввод векторов	Ввод матриц. Вывод скаляров	Вывод матриц
50	F3	F2	F1	Вывод векто-	Вывод мат-	Ввод матриц

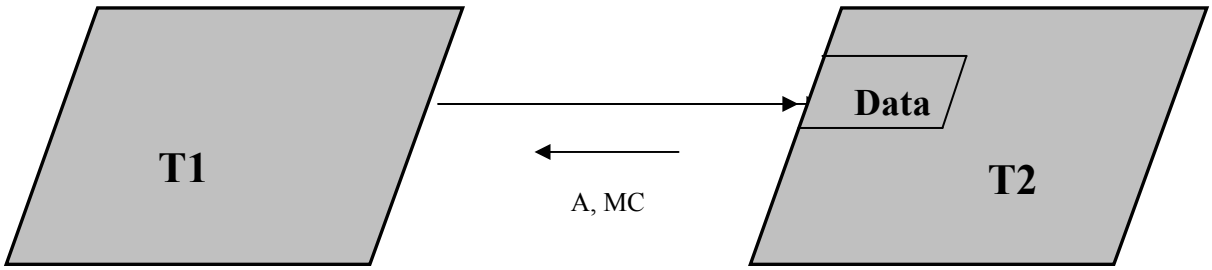
			Ввод векторов	ров	рид	
51	Запуск задач Т2 и Т3	F3	F2	Ввод данных и вывод результатов	F1	-
52	Вывод результатов	-	F1	Ввод данных и запуск остальных задач	F2	F3
53	F3	F1 Ввод данных	F2 Вывод результатов	-	Запуск задач Т2 и Т3	-
54	F2	F1 Вывод результатов	-	F3 Запуск задач Т6 и Т3	-	Ввод данных
55	F3	-	F1 F2 Вывод матриц	Ввод матриц. Ввод скаляров	Ввод и вывод векторов	Вывод скаляров
56	Ввод данных	F2 F3	-	Вывод результатов	F1	Запуск задач Т2 и Т5
57	Запуск задач Т3 и Т6	Ввод матриц	Вывод матриц	F1 F2 F3	Вывод векторов	Ввод векторов
58	Запуск всех задач	Вывод векторов	F1 Ввод векторов	F2	F3	Ввод и вывод матриц
59	-	Ввод данных и вывод результатов	F2	Запуск задач Т3 и Т5	F1 F3	-
60	Вывод результатов	-	F3 F2	Ввод данных. Запуск всех задач	-	F1
61	Вывод матриц	Ввод матриц	Ввод векторов	F2	F3 Вывод векторов	F1
62	F3	Вывод результатов	F1	Ввод векторов	F2	Ввод матриц
63	-	F1	Ввод матриц	F2	Ввод векторов	F3 Вывод результатов
64	F2 Ввод матриц	-	Вывод результатов	F1	F3	Ввод векторов
65	-	F3	Ввод данных и вывод результатов	F2	F1	-
66	-	F1	F3	Вывод матриц	F2 Вывод векторов	Ввод данных
67	F2	Вывод скаляров	Вывод векторов	F3 Вывод матриц	F1 Вывод скаляров	-
68	Вывод матриц	F2	Ввод матриц и вывод скаляров	Ввод векторов	F3	F1
69	Ввод векторов	-	F3 Вывод матриц	F1 Вывод векто-	F2	Вывод скаляров

				ров	Ввод матриц	
70	F3 Вывод векторов	Вывод скаляров	F2 Ввод данных	-	F1 Вывод матриц	-
71	-	Вывод матриц	F1 Вывод векторов	F3	-	F2 Ввод данных
72	-	F2 F3 Вывод векторов	-	F1 Ввод данных	Вывод матриц	Вывод скаляров
73	Ввод матриц	F1 Вывод векторов	Ввод векторов	Вывод матриц	F2 F3	-
74	F3 Вывод матриц	-	-	F2 F1	Ввод данных	Вывод векторов
75	Вывод векторов	Вывод скаляров	F1 F3 Вывод матриц	Ввод данных	F3	-
76	-	F1	Ввод данных	F2	F3 Вывод результатов	-
77	Запуск всех задач	F2 F3	F1 Ввод векторов	Вывод векторов	Вывод матриц	Ввод матриц
78	F2	Вывод результатов	F1	Ввод векторов	F3	Ввод матриц
79	F2 Ввод матриц	F3	F1	Ввод векторов Запуск всех задач	Вывод векторов	Вывод матриц
80	Запуск задач T2, T4, T6	Ввод скаляров F3	Ввод векторов	F2	-	Ввод матриц F1
81	-	F1 F3	Ввод данных	Вывод результатов	F2	Запуск задач T2 и T5
82	F1	F3	F2	Ввод данных и вывод результатов	Запуск задач T1-T3	-
83	-	F2	F3 Вывод результатов	Запуск всех задач	Ввод данных	F1
84	F2	F3	F1	Вывод матриц	Ввод данных и запуск всех задач	Вывод векторов
85	Ввод данных	F1	F2	F3	Вывод результатов	Запуск всех задач
86	Вывод матриц	Ввод данных. Вывод векторов	F1	F2	Запуск всех задач	F3 Вывод скаляров
87	F1	F3	F2	Ввод данных и вывод результатов	-	Запуск всех задач
88	F3	Вывод результатов	-	Ввод векторов	F2 F1	Ввод матриц
89	F1	F2	Запуск задач	Запуск всех	F3 Вывод векто-	Ввод данных

	Вывод матриц		T1 и T2	задач	ров и скаля-ров	
90	Запуск задач T5 и T4	Ввод данных	F2 Вывод векто-ров	Вывод мат-риц и скаля-ров	F3 F1	-
91	Ввод матриц	F3 F1 Вывод скаля-ров	Вывод векто-ров	Вывод мат-риц	F2 Ввод векто-ров. Ввод скаля-ров	Запуск задач T1 и T5
92	F1 Вывод векто-ров	Вывод скаля-ров	F2 Ввод данных	-	F3 Вывод мат-риц	Запуск задач T1-T3
93	-	Вывод матриц	F1 Вывод векто-ров	F3	-	Ввод данных F2
94	Ввод скаляров	F2 F3 Вывод векто-ров	-	Ввод данных Запуск задач T2 и T5. F1	Вывод мат-риц	Вывод скаля-ров
95	Вывод матриц	F1	Ввод векторов	Ввод матриц	F2 F3 Вывод векто-ров	-
96	F2 Вывод матриц	Запуск за-дач T4-T6	-	F3 F1	Ввод данных	Вывод векто-ров
97	Вывод векто-ров	Вывод скаля-ров	F1 F3 Вывод матриц	Ввод данных	F3	-
98	Запуск задач T2-T5	Вывод ре-зультатов	F1	Ввод векто-ров	F3 F2	Ввод матриц
99	Ввод матриц	F3 Вывод векто-ров	Запуск всех задач	Вывод матриц	F2 F1	Ввод век-торов
100	F3 Ввод данных	F2	-	-	F1 Вывод ре-зультатов	Ввод скаля-ров
101	F1	F3	F2 Вывод скаля-ров	Вывод мат-риц	Ввод данных	Вывод векто-ров
102	Ввод данных	F2	F1	-	Вывод ре-зультатов	F3
103	-	Ввод данных. Вывод матриц	F1	F2 Вывод скаля-ров	F3	Вывод скаля-ров и вывод век-торов
104	Запуск всех задач	F2	Ввод данных и вывод ре-зультатов	F3	F1	-
105	-	F1	F3	Вывод мат-риц	F2 Вывод векто-ров	Ввод данных
106	F3	-	Вывод векто-ров	F2 Вывод мат-риц	F1 Вывод скаля-ров	Вывод скаля-ров
107	Вывод матриц F2	-	Ввод матриц и вывод ска-ляров	Ввод векто-ров	F3	F1

108	Ввод векторов	-	F2 Вывод матриц	F1 Вывод векто-ров	F3 Ввод матриц	Вывод скаля-ров
109	F3 Вывод векто-ров	Вывод скаля-ров	F2 Ввод данных	-	F1 Вывод мат-риц	Запуск всех задач

ПРИЛОЖЕНИЕ Е. Структурная схема взаимодействия задач T1 и T2. Используя вход Data задача T2 передаёт в задачу T1 вектор A и матрицу MC.



02.09.2009