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

Кафедра вычислительной техники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ
по кредитному модулю
"Параллельные и распределенные вычисления -1"
(часть 1)

Киев – 2012/13г.

Целью выполнения лабораторных работ по модулю "Параллельные и распределенные вычисления-1" (семестр 5) является закрепление теоретических знаний и умений по разработке программ для параллельных вычислительных систем (ПВС), а также получение практических навыков по работе с процессами (потоками) в современных языках и библиотеках параллельного программирования.

СОДЕРЖАНИЕ И ОФОРМЛЕНИЕ ЛАБОРАТОРНЫХ РАБОТ

Цикл лабораторных работ по модулю составляют ***семь*** заданий. Задания охватывают изучение средств работы с потоками (процессами) в языках параллельного программирования Java, Ада, C# и библиотекахWin32, MPI, OpenMP.

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 «Высшая математика».

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 требует перепрограммирования функций F1, F2, F3. Если сроки выполнения задания не соблюдаются, то на выполнение каждой новой работы выдается *новый вариант* функций F1, F2, F3.

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. [11, с. 11-29].

Отчет по лабораторной работе включает задание и листинг программы.

Защита лабораторной работы выполняется в два этапа. Сначала студент отвечает на вопросы, связанные с теоретической частью задания и текстом программы. При положительной оценке теоретических знаний он показывает преподавателю работу программы на компьютере. За лабораторную работу выставляется оценка. Оценки за лабораторные работы и оценки за модульные контрольные работы определяют итоговую оценку (зачет) за семестр.

Лабораторная работа N 1.
ПОДПРОГРАММЫ И ПАКЕТЫ

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

Выполнение работы: 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 пара-

метры на входные (**in**) и выходные (**out**). Изучить команды и опции компилятора ObjectAda, необходимые для компиляции и редактирования связей программы.

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.

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. с. 367-376, 4 с.128-160, 7, с.60-70 ,].

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 *выносятся 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 построению иерархических библиотек в языке можно найти в [3. с. 377-387, 4, с.178-203, 7, с.71-81, 161-168 ,].

Лабораторная работа N 2.
ПРОЦЕССЫ В ЯЗЫКЕ АДА.ЗАДАЧИ

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. 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 ими в языке Ада можно найти в [3. с. 398-413, 2. с.76-86, с.167- 179, 7, с. 82-103 , 17. с. 27-30, 18. с. 15-19].

Лабораторная работа N 3.
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 [17. с. 22-26, 18. с.8-14].

Лабораторная работа N 4.
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.

Теоретические сведения по программированию потоков и управлению ими в языке C# можно найти в [18. с. 33-38].

Лабораторная работа N 5.
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)`**.

Теоретические сведения по программированию потоков и управлению ими в языке Win32 можно найти в [17. с. 35-36, 18. с.22-28].

Лабораторная работа N 6.
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 1или 2.
Необходимые теоретические сведения: библиотека MPI обеспечивает возможность программирования параллельных процессов с помощью задач **`(task)`**. Для этого используются специальные функции библиотеки MPI.

Теоретические сведения по программированию потоков и управлению ими в языке MPI можно найти в [17. с. 30-34, 18. с.20-21, 25. с. 166-181].

Лабораторная работа N 7.
ПОТОКИ В БИБЛИОТЕКЕ OpenMP

Цель работы: изучение средств библиотеки OpenMP 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 1или 2.
Необходимые теоретические сведения: библиотека OpenMP обеспечивает возможность программирования параллельных процессов с помощью специальных директив.

Теоретические сведения по программированию потоков и управлению ими в языке OpenMP можно найти в [18. с. 28-32].

Л И Т Е Р А Т У Р А

1. Бар Р. Язык Ада в проектировании систем. - М.; Мир, 1988.- 320 с.
2. Пайл Я. Ада - язык встроенных систем. -М.; Финансы и статистика, 1984. - 120 с.
3. Джехани Н. Язык Ада. - М.; Мир, 1988.- 552 с.
4. Василеску Ю. Прикладное программирование на языке Ада.- М.:Мир,1990. - 332 с.
5. Вегнер П. Программирование на языке Ада. - М.; Мир, 1983.- 78 с.
6. Органик Э. Организация системы ИНТЕЛ - 432. - М.; Мир, 1987.-224 с.
7. Корочкин А.В. Ада 95: Введение в программирование. - Киев; Свит, 1998.- 260 с.
8. Бройнль Т. Паралельне програмування. Початковий курс: Навч. Посібник - Київ.: Вища школа, 1997. – 358 с.
9. Соловьев Г.Н., Никитин В.Д. Операционные системы ЭВМ.- М.: Высшая школа, 1989. - 255 с.
10. Дейтел Д. Введение в операционные системы.- М.:: Мир, 1989, - 360 с.
11. Элементы параллельного программирвания / Вальковский В.А., Котов В.Е., Марчук А.Г.,/ Под ред. Котова В.Е. – М.: Радио и связь, 1983.- 240 с.
12. Воеводин В.В. Математические модели и методы в параллельных процессах. – М.: Наука, 1984. – 296 с.
13. Миренков Н.Н. Параллельное программирование для многомодульных вычислительных систем. – М.: Радио и связь, 1989. -320 с.
14. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. – М.: Радио и связь, 1989, с. 280
15. Параллельные вычисления/ Под ред.Г. Родрига: Пер. с англ./ Под ред.
16. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ : Пер. с англ. – М.: Мир,1989. - 424 с.
17. Жуков І., Корочкін О. Паралельні та розподілені обчислення - Киев; Світ, 2004.- 260 с.
18. Жуков И., Корочкин А. Паралельные и распределенные вычисления. Лабораторный практикум. Киев; Світ, 2008.- 240
19. Гома Х. UML. Проектирование систем реального времени, параллельных и распределенных приложений. – М.: ДМК Пресс, 2002. – 704 с.
20. Богачев К.Ю. Основы параллельного программирования. – М.: БИНОМ. Лаб знаний, 2003. – 342 с .
21. Корочкин А., Мустафа Акрам Параллельные вычисления: Ада и Java. – Вісн. НТУУ “КПІ”, Інформатика, управління та обчис лювальна техніка, 1999, К.: – № 32, С. 13–17.
22. Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ – Петербург, 2002. – 400 с.
23. Ноутон П., Шилдт Г. Java2: Пер. с англ. – СПб.: БХВ – Петер бург, 2000. – 1072 с.
24. Симкин С., Барлетт Н., Лесли А. Программирование на Java. Путеводитель – К.: НИПФ “ДиаСофт Лтд.”, 1996. – 736 с.
25. Эндрюс Г. Основы многопоточного, параллельного и распределенного программирования.: Пер. с англ. – М.: Изд. Дом «Вильямс», 2003. – 512 с.

ПРИЛОЖЕНИЕ 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. Функция F1

1.1 A = SORT(B) (MB*MC)
1.2 C = A + B*(MO*ME)
1.3 C = A – B*(MA*MZ)
1.4 C = A + SORT(B) *(MA*MZ)
1.5 C = SORT(A) *(MA*MZ) + SORT(B)
1.6 MX = (B*C) *(MX*MZ)
1.7 MR = (A*SORT(C)) *(MA*MM+MN)
1.8 MM = MAX(B) *(MA*MV)
1.9 MC = MIN(A) *(MA*MP)
1.10 A = B*MIN(C) *(MA*MK+MQ)

1.11 c = MAX(MA*MM)*(A*B)
1.12 A = B + C + D*(MA*MZ)
1.13 C = A*(MA*MZ) + B + D
1.14 D = (SORT(A + B) + C) *(MA*MZ)
1.15 d = MAX(A + B + C) *(MA*MK)
1.16 d = ((A + B)* C) *(MA*MZ)
1.17 d = (A * (B + C)) *(MA*MK)
1.18 d = (A*B) + (C*B) *(MA*MZ)
1.19 d = MAX(B + C) + MIN(A + B*(MA*MK))
1.20 D = MIN(A + B) * (B + C) *(MA*MZ)

1.21 D = SORT(A) + SORT(B) + SORT(C) *(MA*MZ)

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

2. Функция F2

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

2.11 MD = MAX(MA)*(MA*MZ)
2.12 MA = TRANS(MB) + MA*MZ
2.13 MZ = MIN(MA)*MB + MAX(MB)* *(MA*MZ)
2.14 MC = SORT(MA + MB*MM)
2.15 MR = SORT(MA*MB)
2.16 MC = SORT(TRANS(MA)*MB)
2.17 a = MAX(MA + MB*(MA*MZ))
2.18 c = MIN(MA*MB)
2.19 v = MAX(MA + MB*MC)
2.20 MD = MA + MB *MC
2.21 MD = MA + (MB*MC) +MT
2.22 MT = (MA *MB)*(MA + MC)
2.23 q = MAX(MA * MB - MC)
2.24 MG = SORT(MA - MB * MC)
2.25 MU = SORT(MA + TRANS(MB*MC) - 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. Функция F3

3.1 A = MB*MC+ MM
3.2 B = TRANS(MC*MM)*A
3.3 C = SORT(A)*(MB*MM)
3.4 B = SORT(A)*SORT(MB*MM)
3.5 A = (SORT(MB*MM)*C)
3.6 A = MAX(MB*MM)*C
3.7 D = (A+B)*(MC*MX)

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

3.11 D = SORT(A + M)*TRANS(MC*MM)
3.12 E = MC*A + (MB*MK)*B
3.13 E = (MA*MM)*B + MB*SORT(A)
3.14 D = (A + B)*(MA * MB)
3.15 S = (B+C)*TRANS(MA * MB)
3.16 d = MAX((MA*MM)*A + MB*C)
3.17 d = MIN(A*TRANS(MB*MM) + 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*MT)*V - F
3.23 e = MAX((MA*MV)(B + C))
3.24 k = MIN(MA*MB+MM)
3.25 E = (A + B + C)*(MA * MB)
3.26 s = MAX(MA*B + (MB*MM)*C + R)
3.27 E = SORT((MA*MA)*C + W)
3.28 e = MAX(MA*B) + MIN((MA*MM + MC))
3.29 F = MA*D + MB*C + (MX*ME)*P
3.30 K = (MA*MV)*C + a*MB*(A + B)