МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра обчислювальної техніки

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з програмного модулю
"Паралельне програмування -1.
Основи паралельного програмування"

Київ – 2015/16 н.р.

Целью выполнения лабораторных работ по дисциплине "**Паралельне програмування-1.Основи паралельного програмування"** (ПП-1) (семестр 5) является закрепление теоретических знаний и умений, необходимых для разработки программ для параллельных компьютерных систем (ПКС), 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*** обязательных заданий. Восьмое задание выполняют студенты, претендующие на высокий итоговый балл (А). 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 задания на первую работу, включающий номера трех математических функций F1, F2, F3 из **Приложения Б** (1.х, 2.х, 3.х). Функции связаны с выполнением операций над векторами и матрицами произвольной размерности, поэтому следует восстановить соответствующие знания из курса «Линейная алгебра» для корректной реализации этих операций.

На выполнение каждого задания отводится **два** учебных 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 (F1-F3), то есть каждая последующая лабораторная работа будет являться продолжением и модернизацией предыдущей и не требует перепрограммирования функций F1, F2, F3. Если сроки выполнения задания не соблюдаются, то на выполнение следующей работы выдается *новый вариант* функций F1, F2, F3. При этом оценка за лабораторную работу снижается.

При оформление текста программы следует обратить внимание на то, что в тексте программы ключевые слова 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].

Название основной программы лабораторной работы должно соответствовать номеру работы **Lab1, Lab2, … Lab8.**

Листинг программы обязательно должен начинаться с «шапки» - строк комментариев, где отображается следующая информация: название дисциплины, номер и название лабораторной работы, функции F1,F2, F3, фамилия студента и группа, дата.

При защите каждой лабораторной работы студент предоставляет задание на лабораторную работу. При отсутствии задания работа не рассматривается.

Протокол по лабораторной работе включает :

- титульный лист
- задание на работу
- **листинг** программы. Листинг формируется компилятором и имеет расширение *.lst. Листинг в формате .doc не рассматривается!

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 студент показывает преподавателю работу программы на компьютере для данных, которые задает преподаватель. Программа должна работать для векторов и матриц **ЛЮБОЙ РАЗМЕРНОСТИ.**

За лабораторную работу выставляется оценка (A,B,C,D,E). Оценки за лабораторные работы и оценки за модульные контрольные работы определяют итоговую оценку по модулю ПП-1 (зачет).

Лабораторная работа N 1.
АДА.ПОДПРОГРАММЫ И ПАКЕТЫ

Цель работы: изучение структуры программы и особенностей реализации механизма подпрограмм и пакетов в языке Ада, который является базовым в дисциплине «ПП»

Выполнение работы: разработать программу на языке Ада, реализующую вычисления трех функций F1, F2, F3, представленных соответствующим вариантом в **Приложении Б.** Программа состоит из пакета ***Data*** и основной программы – процедуры ***Lab1.*** Пакет реализует ресурсы, необходимые для вычисления функций F1, F2, F3 через подпрограммы *Func1, Func2, Func3.*

При разработке процедур следует разделять формальные параметры на входные (**in**) и выходные (**out**). Необходимо изучить команды и опции компилятора ObjectAda, необходимые для компиляции и редактирования связей программы.

Разработать и реализовать библиотечный пакет *Data* , 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.

Для получения высокого балла по лабораторной работе 1 следует показать умение использовать ***л и ч н и й*** ***(private)*** тип при проектировании пакета ***Data*** и показать особенности его использования. Еще более высокий балл обеспечивает разработка и использование пакета ***Data*** как настраиваемого (**generic)** 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). Все модули имеют стандартную структуру и состоят из *спецификации* (интерфейсной части) и *тела* (части реализации). Спецификация и тело модуля могут находиться в разных файлах и компилироваться отдельно: сначала спецификация, а затем - тело.

*Подпрограммы (**procedure, function**)* – 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 [15,17, 30, 31 47].

Пакеты (***package***) является основным инструментом построения Ада приложения. Это аналог класса в ООП. Инкапсулируют ресурсы (типы, константы, переменные, подпрограммы и другие программные модуля) для последующего использования. Пакет может быть настраиваемым **(generic),** иметь параметры настройки и перед использованием следует создать его экземпляр, указав соответствующие значения параметров в спецификации настройки.

Следует обратить внимание на корректное формирование спецификации пакета!

Спецификация пакета - интерфейс пакета и определяет только те его ресурсы, которые *будут доступны* внешнему пользователю пакета. Поэтому **тщательно продумывайте** спецификацию пакета, вынося туда *не все* ресурсы пакета, а только те ресурсы, которые бу-

дут необходимы и 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 внутренние (вспомогательные) 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 (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 для объектов данного типа *запрещаются все* операции, кроме нескольких фиксированных элементарных операций (**присваивания, сравнения**). 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 [15, 17].

Лабораторная работа N 2.
ПРОЦЕССЫ В ЯЗЫКЕ АДА.ЗАДАЧИ

Цель работы: изучение средств языка Ада для работы с задачами.

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. Обязательно использовать пакет **`Data`** из лабораторной работы 1 !

Задачи независимы, общих данных *не имеют!*

При создании задач необходимо:

- указать имя задачи *T1,T2, T3*
- задать приоритет каждой задачи
- задать размер стека для задачи
- выбрать и задать номер процессора (ядра) для выполнения каждой задачи.

Первый и последний операторы тела задачи выводят на экран информацию о старте и завершении соответствующей задачи (“Task T1 started”, “ Task T2 finished”).

В теле задачи показать использование оператора задержки **`delay,`** поставив его перед выполнением функции F1, F2, F3.

Исследовать при выполнении программы:

- влияние приоритетов задач на очередность запуска задач (при использовании одного или двух ядер).
- влияние оператора задержки **`delay`** на порядок выполнения задач.
- загрузку параллельной компьютерной системы (ПКС) (ядер процессора) при изменении их количества. Изменение количества ядер задается и контролируется с помощью Менеджера (Диспетчера или Task Manager) задач ОС Windows.

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 [1, 2, 15, 17].

Лабораторная работа N 3.
ПОТОКИ В ЯЗЫКЕ JAVA

Цель работы: изучение средств языка Java для работы с потоками.

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.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 2.

В потоках использовать методы **sleep()** и **join().**

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 [1,2, 34, 51, 55, 59].

Лабораторная работа N 4.
ПОТОКИ В ЯЗЫКЕ C#

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

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.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 2.

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 [21, 38, 43].

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 1.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 2.

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 [1, 2].

Лабораторная работа N 6.
ПОТОКИ В БИБЛИОТЕКЕ MPI

Цель работы: изучение средств библиотеки MPI для работы с задачами.

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.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 2.

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

Теоретические сведения по программированию потоков и управлению ими в языке MPI можно найти в [1, 2, 5,16, 25, 36, 68].

Лабораторная работа N 7.
ПОТОКИ В БИБЛИОТЕКЕ OpenMP

Цель работы: изучение средств библиотеки OpenMP для работы с задачами.

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.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 2.

Необходимые теоретические сведения: библиотека OpenMP обеспечивает возможность программирования параллельных процессов с помощью специальных директив.

Теоретические сведения по программированию потоков и управлению ими в библиотеке OpenMP можно найти в [1, 2, 4, 64].

Лабораторная работа N 8. (дополнительно)
ПОТОКИ В ЯЗЫКЕ Python

Цель работы: изучение средств языка Python 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.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 2.

Необходимые теоретические сведения: язык Python обеспечивает возможность программирования параллельных процессов. Теоретические сведения по программированию потоков и управлению ими в языке Python можно найти на сайте
http://rus-linux.net/MyLDP/BOOKS/python.pdf

Л И Т Е Р А Т У Р А

Основная

1. Жуков І., Корочкін О. Паралельні та розподілені обчислення.Навч.посібн. 2-ге видання - Київ: «Корнійчук», 2014. - 284 с.
2. Жуков І., Корочкін О. Паралельні та розподілені обчислення – Київ: «Корнійчук», 2005.- 260 с.
3. Жуков И., Корочкин А. Паралельные и распределенные вычисления. Лабораторный практикум. Киев: «Корнійчук», 2008.- 240 с.

Дополнительная

4. Антонов А.С. Параллельное программирование с использованием технологии OpenMP: Учебное пособие".-М.: Изд-во МГУ, 2009. - 77 с.
5. Богачев К.Ю. Основы параллельного программирования. – М.: БИНОМ. Лаб. знаний, 2003. – 342 с .
6. Брайант Р. Компьютерные системы: архитектура и программмирование. - BHV-СПб, 2005. - 1186 с.
7. Бройнль Т. Паралельне програмування. Початковий курс: Навч. посіб. – К.: Вища шк, 1997. – 358 с.
8. Валях Е. Последовательно-параллельные вычисления. – М.: Мир, 1985. – 456 с.
9. Воеводин В.В. Математические модели и методы в параллельных процессах. – М.: Наука, 1984. – 296 с.
10. Воеводин В., Воеводин В.Параллельные вычисления. БХВ- Петербург, 2002. 608 стр.
11. Гергель В. Высокопроизводительные вычисления для многопроцессорных многоядерных систем. . М.: Изд. МГУ.- 2010. – 544 с.
12. Гома Х. UML. Проектирование систем реального времени, параллельных и распределенных приложений. – М.: ДМК Пресс, 2002. – 704 с.
13. Гофф Макс К. Сетевые распределенные вычисления. Достижения и проблемы. – М.:Кудиц-Образ, 2005. - 320 с.
14. Дейтел Д. Введение в операционные системы. – М.: Мир,1989. – 360 с.
15. Джехани Н. Язык Ада. – М.: Мир, 1988. – 552 с.
16. Корнеев В.Д. Параллельное программирование в MPI. – Москва-Ижевск: "Институт компьютерных исследований", 2003. - 303 с.
17. Корочкин А.В. Ада95: Введение в программирование. – К.: Свит, 1999. – 260 с.
18. Линев А., Боголепов Д.Технологии параллельного программирования для процессоров новых архитектур. М.: Изд. МГУ.- 2010. – 160 с.
19. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ : Пер. с англ. – М.: Мир, 1989. – 424 с.
20. Лупин С., Посыпкин М. Технологии параллельного программирования. М.: ИД Форум, 2011. - 208 с.
21. Мак-Дональд М., Шпушта М.Microsoft ASP.NET 2.0 с примерами на C# 2005 для профессионалов.: Пер. с англ. – М.: Изд. дом «Вильямс», 2006. - 1408 с.
22. Малышкин В.Э., Корнеев В.Д. Параллельное программирование мультикомпьютеров. -

НГТУ, 2006. - 296 с.
23. Миллер Р.,Боксер Л. Последовательные и параллельные алгоритмы: Общий подход. – М.: Лаборатория Базовых Знаний, 2004. – 406 с.
24. Миренков Н.Н. Параллельное программирование для многомодульных вычислительных систем. – М.: Радио и связь, 1989. – 320 с.
25. Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ – Петербург, 2002. – 400 с.
26. Немнюгин С. А. Модели и средства программирования для многопроцессорных вычислительных систем. С.Петербургский ГУ, 2010. - 150 с.
27. Ноутон П., Шилдт Г. Java2: Пер. с англ. – СПб.: БХВ – Петербург, 2000. – 1072 с.
28. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. – М.: Радио и связь, 1989, – 280 с.
29. Параллельные вычисления / Под ред. Г.Родрига – М.: Наука, 1986. – 376 с.
30. Пайл Я. Ада – язык встроенных систем. – М.; Финансы и статистика, 1984. –120 с.
31. Перминов О.Н. Введение в язык программирования Ада.– М.: Радио и связь, 1991 – 228 с.
32. Программирование на параллельных вычислительных системах/ Пер. с англ./ Р.Бэбб, Дж. Мак – Гроу и др.; под ред.Бэбба II. - М.: Мир, 1991. – 376 с.
33. Русанова О.В. Программное обеспечение компьютерных систем. Особенности программ мирования и компиляции. – К.: Корнійчук, 2003. – 94 с.
34. Симкин С., Барлетт Н., Лесли А. Программирование на Java. Путеводитель – К.: НИПФ "ДиаСофт Лтд.", 1996. – 736 с.
35. Соловьев Г.Н., Никитин В.Д. Операционные системы ЭВМ. – М.: Высш. школа., 1989. – 255 с.
36. Стіренко С. Г., Грибенко Д. В., Зіненко А. І., Михайленко А. В. Засоби паралельного програмування. - К., 2011. - 181 с.
37. Траспьютеры.Архитектура и программное обеспечение: Пер. с англ./Под ред.Г.Харпа. – М.: Радио и связь, 1993. – 304 с.
38. Троелсон Є. C# и платформа.NET. Библиотека программиста- СПб.: Питер, 2004. – 796 с.
39. Уильямс Э. Параллельное программирование на C++ в действии. ДМК, - 2012. - 672 с.
40. Хьюз К, Хьюз Т. Параллельное и распределенное программирование в C++. Пер. с англ.- М.: Радио и связь, 1986. – 240 с.
41. Хоар Ч. Взаимодействующие последовательные процессы. - М.: Мир, 1989. – 180 с.
42. Хокни Р., Джессхоул К. Параллельные ЭВМ. Пер. с англ.- М.: Радио и связь, 1986.–240 с.
43. Шилд Г. Полный справочник по C#.: Пер. с англ. – М.: Изд. дом «Вильямс», 2007.– 752 с.
44.Шамим Э., Джейсон Р. Многоядерное программирование Питер, 2010. - 180 с.
45. Эндрюс Г. Основы многопоточного, паралельного и распределенного програмирования.: Пер. с англ. – М.: Изд. дом «Вильямс», 2003. – 512 с.

46. Intel Developer Zone [Електронний ресурс], http://software.intel.com/ru-ru/articles/more-work-sharing-with-openmp
47. Параллельные методы матричного умножения [Електронний ресурс], www.hpcc.unn.ru/file.php?id=426
48. Офіційний сайт компанії Інтел. [Електронний ресурс], www.intel.com
49. Форум MPI. [Електронний реурс], http://www.mpi-forum.org/docs/docs.html
50. Who's Using Ada? Real-World Projects Powered by the Ada Programming Language. [Електронний ресурс], http://www.seas.gwu.edu/~mfeldman/ada-project-summary.html
51. The Special Interest Group on Ada (ACM's SIGAda). [Електронний ресурс], http://www.sigada.org/
52 Офіційний сайт організації Ada-Europe. [Електронний ресурс], http://www.ada-europe.org/
53. Офіційний сайт проекту GAP. [Електронний ресурс],

http://www.adacore.com/academia/universities/
54. Офіційний сайт компанії AdaCore. [Електронний ресурс], http://www.adacore.com

ПРИЛОЖЕНИЕ А.

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ.

a - скалярная величина
A - вектор размерности N (размерность произвольная, задается при запуске программы)
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 каждой строки 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*(MD*ME)
1.3 C = A – B*(MA*MD)
1.4 C = A + SORT(B) *(MA*MD)
1.5 C = SORT(A) *(MA*MD) + SORT(B)
1.6 MA= (B*C) *(MD*ME)
1.7 MB = (A*SORT(C)) *(MD*ME+MC)
1.8 MC = MAX(B) *(MA*MD)
1.9 MC = MIN(A) *(MD*MB)
1.10 A = B*MIN(C) *(MF*MD+ME)

1.11 c = MAX(MA*MB)*(A*B)
1.12 A = B + C + D*(MA*MC)
1.13 C = A*(MA*MD) + B + D
1.14 D = (SORT(A + B) + C) *(MA*MD)
1.15 d = MAX(A + B + C) *(MA*MD)
1.16 d = ((A + B)* (C *(MA*MD)))
1.17 d = (A * ((B + C)*(MA*MD))

1.18 d = (A*B) + (C*(B*(MA*MD))
1.19 d = MAX(B + C) + MIN(A + B*(MA*MK))
1.20 D = MIN(A + B) * (B + C) *(MA*MD)

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

2. Функция F2

2.1 MF = MK + ML*(MN*MM)
2.2 MA = MC*(MX*MZ) - MB
2.3 MM = MK*MG*f
2.4 MN = MAX(ML) *(MK*MO)
2.5 MN = SORT(MK) *MF + ML
2.6 MF= TRANS(MK) *(ML*MN)
2.7 MO = n*MK – m*ML*MN
2.8 MM = f*TRANS(MO)+ *(MF*MK)
2.9 ML = TRANS(MF)*TRANS(MO*MM) +MK
2.10 MO = MF*(MK*ML) + TRANS(MM)]

2.11 MO = MAX(MK)*(ML*MM)
2.12 MO = TRANS(MK) + ML*MM
2.13 MO = MIN(MK)*ML + MAX(MM)* (MK*MN)
2.14 MF = SORT(MK + ML*MM)
2.15 MF = SORT(MK*MN)
2.16 MF = SORT(TRANS(MK)*MN)
2.17 o = MAX(MK + ML*(MM*MN))
2.18 o = MIN(MK*MM)
2.19 o = MAX(MF + MK*ML)
2.20 MM = ML + MK *MO
2.21 MM = ML + (MK*MO) +MN
2.22 MM = (MF *MK)*(ML + MO)
2.23 f = MAX(MK * ML - MO)
2.24 MM = SORT(MF - MK * MO)
2.25 MM = SORT(MF + TRANS(MK*MO) - TRANS(MN))
2.26 MM = MF*(MO*MN)
2.27 MK = (MF*MO)*TRANS(MN)
2.28 MK = MIN(MM)*MN*MO
2.29 MK = (MO + MM)*(MF * MN)*(ML + MO)
2.30 f = MAX(MO*MF) - MIN(MM + MN)

3. Функция F3

3.1 X = MT*MR+ MS
3.2 X = TRANS(MQ*MX)*T

3.3 X = SORT(T)*(MX*MQ)
3.4 Z = SORT(R)*SORT(MW*MV)
3.5 Z = (SORT(MR*MY)*R)
3.6 Z = MAX(MT*MR)*R
3.7 Q = (X+Y)*(MX*MY)
3.8 Q = (MW *MV)*Y + Z
3.9 Q = SORT(R)*(MT * MZ)
3.10 Q = SORT(X + Z)*(MT*MS)

3.11 R = SORT(S + T)*TRANS(MS*MR)
3.12 R = MQ*T + (MY*MX)*S
3.13 R = (MX*MT)*S + MS*SORT(V)
3.14 R = (X + S)*(MT * MR)
3.15 S = (B+C)*TRANS(MA * MB)
3.16 s = MAX((MS*MT)*Q + MW*V)
3.17 s = MIN(X*TRANS(MX*MY) + Z*SORT(Z))
3.18 s = MAX(SORT(MS) + MX*MT)
3.19 T = (MS*MZ)*(W + X)
3.20 R = (X + T)*SORT(MV*MY)

3.21 W = SORT(R*MT)*(MX *MS)
3.22 S = SORT(MW*MZ)*V - F
3.23 q = MAX((MT*MS)(Z + X))
3.24 q = MIN(MX*MQ+MT)
3.25 V = (X + Y + Z)*(MZ * MS)
3.26 v = MAX(MT*R + (MS*MY)*T + R)
3.27 V = SORT((MX*MX)*S + W)
3.28 V = MAX(MR*S) + MIN((MT*MW + MV))
3.29 W = MT*S + MX*Z + (MX*MT)*W
3.30 W = (MX*MS)*T + x*MQ*(X + S)

27 08 2015