МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»

Кафедра обчислювальної техніки

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з кредитних модулів
Основи паралельного програмування
Паралельне програмування -1.
Основи паралельного програмування

Київ – 2018/19 н.р.

Целью выполнения лабораторных работ по кредитным модулям "Основи паралельного програмування " (ОПП) и "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 потоками (процессами) в современных языках и библиотеках параллельного программирования.

СОДЕРЖАНИЕ И ОФОРМЛЕНИЕ ЛАБОРАТОРНЫХ РАБОТ

Цикл лабораторных работ по модулю включает ***шесть*** работ. Студент может выбрать соответствующий набор лабораторных работ :

Оценка	Количество работ	Лабораторные работы
A	6	1 - 6
B, C	5	1, 2, (3 или 4), 5, 6
D, E	4	1, 2, (3 или 4), (5 или 6)

Лабораторные работы связаны с изучением средств работы с потоками (процессами) в языках параллельного программирования Java, Ада, C# и библиотеках WinAPI, MPI, OpenMP.

Для выполнения лабораторных работ необходимо получить у преподавателя вариант задания для первой работы, включающий номера трех математических 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. Линейная алгебра».

Каждая лабораторная работа связана с *новым* набором функций F1-F3. Задание на каждую последующую лабораторную работу студент получает *только* после успешной сдачи предыдущей. Набор функций не меняется, если студент сдает работы строго по графику, то есть, например, при выборе шести работ он сдает каждую через одно занятие. На одном занятии можно сдать только одну работу.

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].

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 лабораторной работы студент предоставляет задание на лабораторную работу и отчет (протокол) выполненной работы. Протокол по лабораторной работе включает описание задания и **листинг** программы. Листинг формируется компилятором и имеет расширение *.lst.

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 выполнение программы на компьютере. За лабораторную работу выставляется оценка (**A,B,C,D,E**) и соответствующе количество баллов (15 за каждую работу). Суммарный баллы за лабораторные работы определяют итоговую зачетную оценку по кредитному модулю.

Лабораторная работа N 1.
ПОТОКИ В ЯЗЫКЕ АДА.ЗАДАЧИ

Цель работы: изучение средств языка Ада для работы с процессами (потоками).
Выполнение работы: Разработать программу, содержащую ***параллельные потоки*** (задачи), каждый из которых реализует функцию F1, F2, F3 из **Приложения Б** согласно полученному варианту.

Программа должна состоять 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`**). Следует изучить команды и опции компилятора ObjectAda, необходимые для компиляции и редактирования связей программы.

Пакет *Data* включает следующие ресурсы
- подпрограммы `Func1, Func2, Func3`
- необходимые ***типы*** (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 использовать ***личный (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.

При создании задач необходимо:
- указать имя задачи
- задать приоритет задачи
- задать размер стека для задачи
- выбрать и задать номер процессора (ядра) для выполнения каждой задачи.

Задачи независимы, общих данных *не имеют!*

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 использовать оператор задержки **`delay,`** поставив его перед и после выполнения функции F1, F2, F3 с небольшим временем задержки.

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 с помощью Менеджера (Диспетчера) задач ОС Windows.

Необходимые теоретические сведения: язык Ада обеспечивает возможность программирования параллельных процессов (потоков) 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 ука-

занный 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 2.
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.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 1.

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 3.
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.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 1.

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 4.
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 .

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 1.

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 5.
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.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 1.

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 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.

Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 1.

Необходимые теоретические сведения: библиотека MPI обеспечивает возможность программирования параллельных процессов с помощью задач. Для этого используются специальные функции библиотеки MPI. Задача создается путем копирования основной программы.

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

Л И Т Е Р А Т У Р А

Основная

1. Жуков І., Корочкін О. Паралельні та розподілені обчислення – Київ: «Корнійчук», 2005.- 260 с.
2. Жуков І., Корочкін О. Паралельні та розподілені обчислення.Навч.посібн. 2-ге видання - Київ: «Корнійчук», 2014. - 284 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. Breshears C. The Art of Concurrency. O'Really Media, 2009. – 280 p.
47. Burns A., Wellings A. Real-Time Systems and Programming Languages. Addison – Wesley, 2001, – 386 p.
48. Burns A., Programming in Occam2. Reading. Addison–Wesley, 1995, – 326 p.
49. Burns A., Wellings A. Concurrency in Ada. – Cambridge: Cambridge University Press, 1995., – 420 p.
50. El – Rewini H, Lewis T., Distributed and Parallel Computing.– Manning Pub. Co.1998, – 430 p.
51. Hyde P. Java Thread Programming. – Indiapolis, IN: Sams Publishing, 1999, – 324 p.
52. Hoare C.A.R. Monitors: An Operating System Structuring Concept, Communications of ACM, Vol.17, №.10, Oct.1974, pp. 549–557
53. Hoare C.A.R. Communicating Sequential Processes, Communications of ACM, vol.21, №. 8, Aug. 1978, pp. 666 – 667.
54. Hoare C.A.R. Communicating Sequential Processes,: Printice Hall, International Series in Computer Science, Englewood Cliffs NJ, 1985. – 186 p.
55. Goetz B. Java Concurrency in Practice.- Addison–Wesley Professional, 2006, – 384 p.
56. Korochkin A., Rusanova O. Scheduling Problems for Parallel and Distributed Systems– In Proceeding of the ACM Annual Conference (SIGADA'99) (The Redondo Beach, CA, USA, October 17–21, 2001) ACM Press, New York, NY, 1999, pp. 182– 190;
57. Korochkin A. Ada95 as a Foundation Language in Computer Engineering Education in Ukraine – In Proceeding of the Ada-Europe International Conference on Reliable Software Technolo gies (Ada-Europe'99), (Santander, Spain, June 7 – 11, 1999), Lecture Notes in Computer Sci- ence , – № 1622, Springer, 1999, pp. 62 – 70.

58. Korochkin A., Salah I, Korochkin D. Experimental Analyze Ada Program in Cluster System – In Proceeding of the ACM Annual Conference (SIGADA'05) (Atalanta, Georgia, USA, November 13–21, 2005) ACM Press, New York, NY, 2005, pp. 126 – 134.
59. Lea D. Concurrent Programming in Java: Design Principles and Patterns. Reading, MA: Addison – Wesley, 1999, – 344 p.
60. Mattson T., Sanders B., Massingill B. Patterns For Parallel Programming. Addison-Wesley, 2005. – 246 p.
61. Taft S., Duff R., Brukardt R., Ploedereder T. Consolidated Ada Reference Manual. Language and Standard Libraries, Springer, Berlin: 2001, – 562 p.
62. McKenney P. Is Parallel Programming Hard, And, If So, What Can You Do About It? [Електронний ресурс], pttps://www.kernel.org/pub/linux/kernel/people/paulmck/perfbook/
63. TOP500. [Електроний ресурс]. http://parallel.ru/computers/ top500.list42.html
64. Chapman B. Using OpenMP: portable shared memory parallel programming. Massachusetts Institute of Technology, London, 2008, - 350 p.
65. Intel Developer Zone [Електронний ресурс], http://software.intel.com/ru-ru/articles/more-work-sharing-with-openmp

66. Параллельные методы матричного умножения [Електронний ресурс], www.hpcc.unn.ru/file.php?id=426
67. Офіційний сайт компанії Інтел. [Електронний ресурс], www.intel.com
68. Форум MPI. [Електронний реурс], http://www.mpi-forum.org/docs/docs.html
69. 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
70. The Special Interest Group on Ada (ACM's SIGAda). [Електронний ресурс], http://www.sigada.org/
71 Офіційний сайт організації Ada-Europe. [Електронний ресурс], http://www.ada-europe.org/
72. Офіційний сайт проекту GAP. [Електронний ресурс], http://www.adacore.com/academia/universities/
73. Офіційний сайт компанії AdaCore. [Електронний ресурс], http://www.adacore.com

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 элементов 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*MC)*e
1.4 C = A + SORT(B) *(MA*ME)
1.5 C = SORT(A) *(MA*ME) + SORT(B)
1.6 MD = (B*C) *(MA*ME)
1.7 ME = (A*SORT(C)) *(MA*ME+MD)
1.8 ME = MAX(B) *(MA*MD)
1.9 MC = MIN(A) *(MA*MD)
1.10 A = B*MIN(C) *(MA*MD+MD)

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

1.21 D = SORT(A) + SORT(B) + SORT(C) *(MA*ME)
1.22 d = (B*C) + (A*B) +(C*(B*(MA*ME)))

1.23 E = A + B + C + D*(MA*MD)
1.24 E = A + C *(MA*ME) + B
1.25 e = ((A + B)*(C + D*(MA*ME)))
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*ME)
1.30 e = (A*(MA*ME) *SORT(B))

2. Функция F2

2.1 MF = MG + MH*(MK*ML)
2.2 MF = MG*(MK*ML) - MK
2.3 MF = MF*MG*k
2.4 MG = MAX(MH) *(MK*ML)
2.5 MG = SORT(MF) *MK + ML
2.6 MG = TRANS(MK) *(MH*MF)
2.7 MF = k*MG– h*MK*ML
2.8 MF = g*TRANS(MG)+ *(MK*ML)
2.9 MK = TRANS(MA)*TRANS(MB*MM) +MX
2.10 MK = MA*(MA*MZ) + TRANS(MB)]

2.11 MF = MAX(MG)*(MH*MK)
2.12 MF = TRANS(MG) + MK*ML
2.13 ML = MIN(MF)*MG + MAX(MH)* *(MK*MF)
2.14 ML = SORT(MF + MG*MH)
2.15 ML = SORT(MF*MG)
2.16 ML = SORT(TRANS(MF)*MK)
2.17 h = MAX(MF + MG*(MH*ML))
2.18 h = MIN(MG*ML)
2.19 k = MAX(MF + MG*ML)
2.20 MK = ML + MH *MG
2.21 MF = MG + (MH*MK) +ML
2.22 MF = (MG *MH)*(MK + ML)
2.23 q = MAX(MH * MK - ML)
2.24 MG = SORT(MF - MH * MK)
2.25 MF = SORT(MG + TRANS(MH*MK) - TRANS(ML))
2.26 MF = MG*(MH*ML)
2.27 MF = (MG*MH)*TRANS(MK)
2.28 MF = MIN(MH)*MK*ML
2.29 MF = (MG + MH)*(MK * ML)*(MG + ML)
2.30 f = MAX(MG*MK) - MIN(ML + MH)

3. Функция F3

3.1 O = MP*MR+ MS
3.2 O= TRANS(MP*MR)*T
3.3 O = SORT(P)*(MR*MT)
3.4 O = SORT(P)*SORT(MR*MS)
3.5 O = (SORT(MP*MR)*S)
3.6 O = MAX(MP*MR)*T
3.7 O = (P+R)*(MS*MT)
3.8 O = (MP *MR)*S + T

3.9 O = SORT(P)*(MR * MS)
3.10 O = SORT(R + S)*(MT*MP)

3.11 T= SORT(O + P)*TRANS(MR*MS)
3.12 T = MO*P + (MR*MS)*S
3.13 T = (MO*MP)*S + MR*SORT(S)
3.14 T = (O + P)*(ML * MS)
3.15 S= (O+P)*TRANS(MR * MT)
3.16 t = MAX((MO*MP)*R + MS*S)
3.17 s = MIN(A*TRANS(MB*MM) + B*SORT(C))
3.18 s = MAX(SORT(MS) + MA*MB)
3.19 S = (MO*MP)*(R + T)
3.20 S = (O + P)*SORT(MT*MR)

3.21 S = SORT(O*MO)*(MS *MT)
3.22 S = SORT(MS*MT)*O - P
3.23 s = MAX((MO*MP)(R + T))
3.24 s = MIN(MO*MP+MS)
3.25 S = (O + P + T)*(MR * MS)
3.26 s = MAX(MO*T + (MT*MS)*P + R)
3.27 S = SORT((MO*MP)*R + S)
3.28 s = MAX(MO*S) + MIN((MT*MS + MP))
3.29 S = MO*T + MP*R + (MO*MS)*T
3.30 S = (MO*MP)*T + t*MR*(O + P)

06 09 2018