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

Кафедра обчислювальної техніки

Паралельне програмування

Методичні вказівки
до виконання лабораторних робіт
з кредитного модуля
для студентів 3 курсу ОП «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»

Розробник: доцент, канд. техн. наук, доцент Корочкін О.В.
(посада, вчена ступінь та звання П.І.Б.)

Затверджено на засіданні кафедри
Протокол № 1 від «30» серпня 2021 р.

Завідувач кафедри ОТ
_____ Стіренко С.Г.
(підпис) (прізвище, ініціали)

Київ – 2021/22 н.р.

Метою виконання лабораторних робіт з кредитного модуля “Паралельне програмування” (семестр 5) є закріплення теоретичних знань та умінь, які необхідні для розробки паралельних програм, а також отримання практичних навиків по роботе з потоками (процесами) в сучасних мовах та бібліотеках паралельного програмування.

ЗМІСТ ТА ОФОРМЛЕННЯ ЛАБОРАТОРНЫХ РОБІТ

Цикл лабораторних робіт (ЛР) по модулю складає ***шість*** робіт (ЛР1-ЛР6). Студент може обрати відповідний набір лабораторних робіт :

Оцінка	Кількість робот	Лабораторні роботи
A (95-100)	5	(1 або 2), 3, 4, 5, 6
B, C (75-94)	4	(1або 2), (3 або 4), 5 , 6
D, E (60-74)	3	(1 або 2), (3 або 4), (5 або 6)

Лабораторні роботи пов’язані з вивченням засобів роботи з потоками (процесами) в мовах паралельного програмування Java, Ada, C# та бібліотеках WinAPI, MPI, OpenMP.

Для виконання робіт необхідно отримати варіант завдання для ЛР1, який включає номера трьох математичних функцій ***F1, F2, F3*** з **Додатку Б** (1.x, 2.x, 3.x). Функції пов’язані з виконанням операцій над векторами і матрицями.

Завдання на ЛР змінюється на разі несвоєчасного виконання ЛР.
Перша робота повинна бути відправлена викладачу вже на другому занятті.

Треба розробити паралельну програму за допомогою мови або бібліотеки паралельного програмування, яка забезпечує паралельне виконання трьох математичних функцій ***F1, F2, F3*** згідно варіанту ЛР.

Стандартна структура програми:

- модуль (клас) ***Data*** , що містить ресурси для створення потоків (типи, допоміжні процедури та функції та інше…);
- потоки ***T1, T2, T3***. Кожен потік здійснює дії, що необхідні для паралельного обчислення відповідної функції ***Fi***:
  - введення відповідних даних,
  - обчислення функції ***Fi,***
  - виведення результату виконання потоку Fi.

  Необхідні ресурси для побудови потоку беруться з модулю ***Data***;
- головну процедуру ***(Lab1-Lab6).***

Треба виконати налагодження паралельної програми та дослідити її виконання для малих та великих значень N (N - розмір векторів та матриць, обов’язково позначаються в програмі через змінну N).

Для невеликих розмірів N< 5 введення в потоці векторів та матриць здійсняються за допомогою клавіатуру. При цьому всі елементи векторів та матриць отримують для F1 значення 1, для F2 - значення 2, для F3 - значення 3. Дослідити проблеми введення з клавіатури і пояснити їх в протоколі ЛР1.

Для великих значень N >1000 введення даних вже передбачити через створювання в модулі Data ресурсів (методів), що дозволяють реалізувати введення шляхом:

- формування файлів з наступним зчитування даних з них
- встановлення всіх елементів даних заданому значенню (наприклад 1)
- використання генератору випадкових значень.

Під час виконання програми для N>1000 прослідити та проаналізувати процес завантаження програмою ядер багатоядерного процесора за допомогою Диспетчера задач ОС Windows. Зробити в протоколі висновки, що до завантаження процесора потоками.

УВАГА!

Наступна ЛР приймається на перевірку лише після зарахування попередньої !

Ця вимога потребує виконання і надсилання ЛР практично на кожному лабораторному занятті. Можливо зробити всі ЛР одразу, але відсилати ЛР викладачу можна лише по однієї на кожному занятті (раз на два тижня) на разі зарахування попередньої ЛР.

Більш ніж однієї роботи від студента викладач не перевіряє! Не плануйте відіслати всі ЛР на останньому занятті. Перевірятися буде лише одна робота.

Це – вимога Болонського процесу, де студент повинен працювати ВЕСЬ семестр, а не останній тиждень перед екзаменом.

Назва головні процедури (потоку) ***Lab1 – Lab6,*** пов'язана з номером ЛР, що виконується**,** назва потоку (задачі) ***T1-T3*** **-** з номером функції ***F1-F3*****.**

Функції незалежні, спільних даних *не мають!*

Лістинг програми повинен починатися з «шапки» - строк коментаріїв, де відображається наступна інформація: назва дисципліни, номер та назва ЛР, функції *F1, F2, F3,* ПІБ студента, група, дата.

Протокол виконання ЛР містить: титульний аркуш, завдання на ЛР, листінг програми з шапкою та коментарями.

Захист ЛР, якщо заняття відбувається в КПІ, здійснюється в два етапи. Спочатку студент відповідає на запитання, що пов'язані з теоретичною частиною завдання і програмою. За позитивної оцінки теоретичних знань студент показує виконання програми на комп'ютері. Кожна робота оцінюється до 8 балів.

В on-line режими студент *надсилається ЛР викладачу на вказану пошту* під час проведення ЛР в групі. Викладач через пошту інформує студента про отримання ЛР, перевіряє ЛР і надсилає студенту результати перевірки:

- *ЛР зараховано* з оцінкою А-Е; (студент може виконувати наступну роботу)
- *ЛР потребує перероблення* (є зауваження, вказуються які). Перероблена ЛР надсилається викладачеві на наступному занятті.

Сумарні оцінки за лабораторні роботи і за модульну контрольну роботу визначають підсумкову залікову оцінку з кредитного модуля.

Лабораторна робота N 1.
ПОТОКИ В МОВІ АДА. ЗАДАЧІ

Мета роботи: вивчення засобів мови Ада для роботи с потоками (процесами).

Виконання роботи: Розробити програму, яка містить ***паралельні потоки*** (задачі), кожна з яких реалізую відповідну функцію F1, F2, F3 з **Додатку Б** згідно отриманому варіанту.

Програма повинна мати пакет *Data* і основну процедуру *Lab1*. Пакет містить ресурси, необхідні для обчислення функцій F1, F2, F3 через підпрограми *Func1, Func2, Func3.*

При створенні задач необхідно:
- вказати ім'я задачі
- встановити пріоритет задачі
- задати розмір стека задачі
- обрати і задати номер процесу (ядра) для виконання кожної задачі.

В тілі задачі задіяти оператор задержки **delay** при виконанні функцій F1, F2, F3 з невеликим часом затримки.

Дослідити при виконанні програми:
- вплив пріоритетів задач на чергу запуску задач
- вплив оператора затримки **delay** на порядок виконання задач.
- завантаження центрального багатоядерного процесора паралельної комп'ютерній системи (ПКС). Зміна кількості ядер здійснюється за допомогою Менеджера (Дис петчера) задач ОС Windows.

Необхідні теоретичні відомості: мова Ада забезпечує програмування паралельних процесів (потоків) за допомогою задачних модулів **(task**). Управління виконанням задач можна здійснювати через встановлення пріоритетів задач (прагма **priority**), а також через оператор **delay,** який блокує виконання задачі на заний період часу.

Задачі мають стандартну для мови структуру, тобто містять специфікацію і тіло. Специфікація задачі дозволяє описати ім'я задачі, пріоритет, засоби взаємодії з іншими задачами та інше. Тіло задачі визначає дії задачі.

Теоретичні відомості по програмуванню задач в мові Ада можна знайти в [1, 2, 3, 16, 18, 31, 51, 62, 72, 74].

Лабораторна робота N 2.
ПОТОКИ В МОВІ JAVA

Мета роботи: вивчення засобів мови Java для роботи с потоками.

Виконання роботи: Розробити програму, яка містить ***паралельні потоки*** , що реалізують відповідну функцію F1, F2, F3 з **Додатку Б** згідно отриманому варіанту.

Вимоги що до створення потоків і завдання дослідження особливості виконання паралельної програми визначені в лабораторної роботі 1.

В потоках використати методи **sleep()** і **join().**

Необхідні теоретичні відомості: мова Java забезпечує програмування паралельних процесів (потоків) за допомогою потоків **(threads**). Використається клас **Thread** або інтерфейс **Runnable.**

Потоки визначають паралельне виконання Java програми в ПКС. Управління виконанням потоків можна здійснити за допомогою пріоритетів потоків (метод **`set_Priority()`**), метода **`sleep()`**, який викликає блокування потоку на вказаний період часу

Метод **`join()`** використається для синхронізації основного метода з потоками, яки він запускає на виконання.

Метод **`run()`** визначає дії потоку при виконанні.

Теоретичні відомості по програмуванню потоків в мові Java можна знайти в [1,2, 28, 52, 56, 60].

Лабораторна робота N 3.
ПОТОКИ В МОВІ C#

Мета роботи: вивчення засобів мови C# для роботи з потоками.

Виконання роботи: Розробити програму, яка містить ***паралельні потоки***, що реалізують відповідну функцію F1, F2, F3 з **Додатку Б** згідно отриманому варіанту.

Вимоги що до створення потоків і завдання дослідження особливості виконання паралельної програми визначені в лабораторної роботі 1.

Необхідні теоретичні відомості: мова C# забезпечує можливість програмування паралельних процесів за допомогою потоків класу *Thread* з використанням концепції потокових функцій.

Теоретичні відомості по програмуванню потоків в мові C# можна знайти в [1, 2, 3, 22, 39, 44].

Лабораторна робота N 4.
ПОТОКИ В БІБЛІОТЕЦІ WINAPI

Мета роботи: вивчення засобів бібліотеки WinAPI для роботи з потоками.

Виконання роботи: Розробити програму, яка містить ***паралельні потоки***, що реалізують відповідну функцію *F1, F2, F3* з **Додатку Б** згідно отриманому варіанту.

Вимоги що до створення потоків і завдання дослідження особливості виконання паралельної програми визначені в лабораторної роботі 1.

Необхідні теоретичні відомості: бібліотека WinAPI забезпечує можливість програмування паралельних процесів за допомогою функції *Create_Thread* з використанням концепції потокових функцій.

Теоретичні відомості по програмуванню потоків в бібліотеці WinAPI можна знайти в [1, 2, 3 40, 41, 45].

Лабораторна робота N 5.
ПОТОКИ В БІБЛІОТЕЦІ OpenMP

Мета роботи: вивчення засобів бібліотеки OpenMP для роботи з потоками.

Виконання роботи: Розробити програму, яка містить ***паралельні потоки***, що реалізують відповідну функцію *F1, F2, F3* з **Додатку Б** згідно отриманому варіанту.

Вимоги що до створення потоків і завдання дослідження особливості виконання паралельної програми визначені в лабораторної роботі 1. Для отримання оцінки А треба застосувати паралельне виконання циклів через прагму ***parallel for.***

Необхідні теоретичні відомості: бібліотека OpenMP забезпечує можливість програмування паралельних процесів за допомогою набору прагм з використанням концепції визначення паралельних ділянок і копіювання коду.

Теоретичні відомості по програмуванню потоків в бібліотеці OpenMP можна знайти в [1, 2, 3, 28, 52, 56, 60].

Лабораторна робота N 6.
ПОТОКИ В БІБЛІОТЕЦІ MPI

Мета роботи: вивчення засобів бібліотеки MPI для роботи з потоками.

Виконання роботи: Розробити програму, яка містить ***паралельні потоки***, що реалізують відповідну функцію F1, F2, F3 з **Додатку Б** згідно отриманому варіанту.

Вимоги що до створення потоків і завдання дослідження особливості виконання паралельної програми визначені в лабораторної роботі 1.

Необхідні теоретичні відомості: бібліотека MPI забезпечує можливість програмування паралельних процесів за допомогою набору процедур і типів з використанням концепції копіювання коду програми.

Теоретичні відомості по програмуванню потоків в бібліотеці MPI можна знайти в [1, 2, 3, 5, 6, 13, 17,23, 26,27, 37, 51, 69].

Л І Т Е Р А Т У Р А

Основна

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.
4. Korochkin A. Multicore Ada programming Навч. посібник [Електронний ресурс] НТУУ-КПІ 2018. - 98 с.

Додаткова

5. Антонов А.С. Параллельное программирование с использованием технологии OpenMP: Учебное пособие".-М.: Изд-во МГУ, 2009. - 77 с.
6. Богачев К.Ю. Основы параллельного программирования. – М.: БИНОМ. Лаб. знаний, 2003. – 342 с .
7. Брайант Р. Компьютерные системы: архитектура и программмирование. - BHV-СПб, 2005. - 1186 с.
8. Бройнль Т. Паралельне програмування. Початковий курс: Навч. посіб. – К.: Вища шк, 1997. – 358 с.
9. Валях Е. Последовательно-параллельные вычисления. – М.: Мир, 1985. – 456 с.

10. Воеводин В.В. Математические модели и методы в параллельных процессах. – М.: Наука, 1984. – 296 с.
11. Воеводин В., Воеводин В.Параллельные вычисления. БХВ- Петербург, 2002. 608 стр.
12. Гергель В. Высокопроизводительные вычисления для многопроцессорных многоядерных систем. . М.: Изд. МГУ.- 2010. – 544 с.
13. Гома Х. UML. Проектирование систем реального времени, параллельных и распределенных приложений. – М.: ДМК Пресс, 2002. – 704 с.
14. Гофф Макс К. Сетевые распределенные вычисления. Достижения и проблемы. – М.:Кудиц-Образ, 2005. - 320 с.
15. Дейтел Д. Введение в операционные системы. – М.: Мир,1989. – 360 с.
16. Джехани Н. Язык Ада. – М.: Мир, 1988. – 552 с.
17. Корнеев В.Д. Параллельное программирование в MPI. – Москва-Ижевск: "Институт компьютерных исследований", 2003. - 303 с.
18. Корочкин А.В. Ада95: Введение в программирование. – К.: Свит, 1999. – 260 с.
19. Линев А., Боголепов Д.Технологии параллельного программирования для процессоров новых архитектур. М.: Изд. МГУ.- 2010. – 160 с.
20. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ : Пер. с англ. – М.: Мир, 1989. – 424 с.
21. Лупин С., Посыпкин М. Технологии параллельного программирования. М.: ИД Форум, 2011. - 208 с.
22. 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.
23. Малышкин В.Э., Корнеев В.Д. Параллельное программирование мультикомпьютеров. - НГТУ, 2006. - 296 с.
24. Миллер Р.,Боксер Л. Последовательные и параллельные алгоритмы: Общий подход. – М.: Лаборатория Базовых Знаний, 2004. – 406 с.
25. Миренков Н.Н. Параллельное программирование для многомодульных вычислительных систем. – М.: Радио и связь, 1989. – 320 с.
26. Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ – Петербург, 2002. – 400 с.
27. Немнюгин С. А. Модели и средства программирования для многопроцессорных вычислительных систем. С.Петербургский ГУ, 2010. - 150 с.
28. Ноутон П., Шилдт Г. Java2: Пер. с англ. – СПб.: БХВ – Петербург, 2000. – 1072 с.
29. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. – М.: Радио и связь, 1989, – 280 с.
30. Параллельные вычисления / Под ред. Г.Родрига – М.: Наука, 1986. – 376 с.
31. Пайл Я. Ада – язык встроенных систем. – М.; Финансы и статистика, 1984. –120 с.
32. Перминов О.Н. Введение в язык программирования Ада.– М.: Радио и связь, 1991 – 228 с.
33. Программирование на параллельных вычислительных системах/ Пер. с англ./ Р.Бэбб, Дж. Мак – Гроу и др.; под ред.Бэбба II. - М.: Мир, 1991. – 376 с.
34. Русанова О.В. Программное обеспечение компьютерных систем. Особенности программ мирования и компиляции. – К.: Корнійчук, 2003. – 94 с.
35. Симкин С., Барлетт Н., Лесли А. Программирование на Java. Путеводитель – К.: НИПФ “ДиаСофт Лтд.”, 1996. – 736 с.
36. Соловьев Г.Н., Никитин В.Д. Операционные системы ЭВМ. – М.: Высш. школа., 1989. – 255 с.
37. Стіренко С. Г., Грибенко Д. В., Зіненко А. І., Михайленко А. В. Засоби паралельного програмування. - К., 2011. - 181 с.
38. Траспьютеры.Архитектура и программное обеспечение: Пер. с англ./Под ред.Г.Харпа. – М.: Радио и связь, 1993. – 304 с.
39. Троелсон Є. C# и платформа.NET. Библиотека программиста- СПб.: Питер, 2004. – 796 с.
40. Уильямс Э. Параллельное программирование на C++ в действии. ДМК, - 2012. - 672 с.
41. Хьюз К, Хьюз Т. Параллельное и распределенное программирование в C++. Пер. с англ.-

М.: Радио и связь, 1986. – 240 с.
42. Хоар Ч. Взаимодействующие последовательные процессы. - М.: Мир, 1989. – 180 с.
43. Хокни Р., Джессхоул К. Параллельные ЭВМ. Пер. с англ.- М.: Радио и связь, 1986.–240 с.
44. Шилд Г. Полный справочник по C#.: Пер. с англ. – М.: Изд. дом «Вильямс», 2007.– 752с
45.Шамим Э., Джейсон Р. Многоядерное программирование Питер, 2010. - 180 с.
46. Эндрюс Г. Основы многопоточного, паралельного и распределенного програмирования.: Пер. с англ. – М.: Изд. дом «Вильямс», 2003. – 512 с.
47. Breshears C. The Art of Concurrency. O'Really Media, 2009. – 280 p.
48. Burns A., Wellings A. Real-Time Systems and Programming Languages. Addison – Wesley, 2001, – 386 p.
49. Burns A., Programming in Occam2. Reading. Addison–Wesley, 1995, – 326 p.
50. Burns A., Wellings A. Concurrency in Ada. – Cambridge: Cambridge University Press, 1995., – 420 p.
51. El – Rewini H, Lewis T., Distributed and Parallel Computing.– Manning Pub. Co.1998, – 430 p.
52. Hyde P. Java Thread Programming. – Indiapolis, IN: Sams Publishing, 1999, – 324 p.
53. Hoare C.A.R. Monitors: An Operating System Structuring Concept, Communications of ACM, Vol.17, №.10, Oct.1974, pp. 549–557
54. Hoare C.A.R. Communicating Sequential Processes, Communications of ACM, vol.21, №. 8, Aug. 1978, pp. 666 – 667.
55. Hoare C.A.R. Communicating Sequential Processes,: Printice Hall, International Series in Computer Science, Englewood Cliffs NJ, 1985. – 186 p.
56. Goetz B. Java Concurrency in Practice.- Addison–Wesley Professional, 2006, – 384 p.
57. 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;
58. 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.
59. 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.
60. Lea D. Concurrent Programming in Java: Design Principles and Patterns. Reading, MA: Addison – Wesley, 1999, – 344 p.
61. Mattson T., Sanders B., Massingill B. Patterns For Parallel Programming. Addison-Wesley, 2005. – 246 p.
62.Taft S., Duff R., Brukardt R., Ploedereder T. Consolidated Ada Reference Manual. Language and Standard Libraries, Springer, Berlin: 2001, – 562 p.
63. 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/
64. TOP500. [Електроний ресурс]. http://parallel.ru/computers/ top500.list42.html
65. Chapman B. Using OpenMP: portable shared memory parallel programming. Massachusetts Institute of Technology, London, 2008, - 350 p.
66. Intel Developer Zone [Електронний ресурс], http://software.intel.com/ru-ru/articles/more-work-sharing-with-openmp
67. Параллельные методы матричного умножения [Електронний ресурс], www.hpcc.unn.ru/file.php?id=426
68. Офіційний сайт компанії Інтел. [Електронний ресурс], www.intel.com
69. Форум MPI. [Електронний реурс], http://www.mpi-forum.org/docs/docs.html

70. 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
71. The Special Interest Group on Ada (ACM's SIGAda). [Електронний ресурс], http://www.sigada.org/
72 Офіційний сайт організації Ada-Europe. [Електронний ресурс], http://www.ada-europe.org/
73. Офіційний сайт проекту GAP. [Електронний ресурс], http://www.adacore.com/academia/universities/
74. Офіційний сайт компанії AdaCore. [Електронний ресурс], http://www.adacore.com

ДОДАТОК А.

УМОВНІ ПОЗНАЧЕННЯ.

a	- скаляр
A	- вектор (розмірності N)
MA	- матриця (розмірності NxN)
a*B	- добуток вектора на скаляр
a*MB	- добуток матраци на скаляр
(A*B)	- скалярний добуток векторів
(MA*MB)	- добуток матриць
(B*MA)	- добуток вектора на матрицу
SORT(A)	- сортування вектора
MAX(A)	- пошук максимального елемента вектора
TRANS(MA)	- транспортування матриці
MAX(MA)	- пошук максимального елемента матриці
MIN(MA)	- пошук мінімального елемента матраці
SORT(MA)	- сортування строк матриці

ДОДАТОК Б.

ВАРІАНТИ ФУНКЦІЙ ***F1, F2, F3***

1. Функція 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. Функція 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)+ f*(MK*ML)
2.9 MK = TRANS(MG)*TRANS(MX*MM) + MX
2.10 MK = MA*(MG*MZ) + TRANS(ML)]

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. Функція *F3*

3.1 O = MP*MR+ MS
3.2 O= TRANS(MP*MR)*V
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)*V
3.7 O = (P+R)*(MS*MT)
3.8 O = S*(MP *MR) + V
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 = P*MO + (S*(MR*MS))
3.13 T = S*(MO*MP) + SORT(S)*MR
3.14 T = (O + P)*(MP * MS)
3.15 S= (O+P)*TRANS(MR * MT)
3.16 t = MAX((R*(MO*MP)) + MS*S)
3.17 s = MIN(O*TRANS(MP*MM)) + (R*SORT(S))
3.18 s = MAX(SORT(MS) + MR*MT)
3.19 S = (R + V) *(MO*MP)
3.20 S = (O + P)*SORT(MT*MR)

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