**Проект**

**по Системи за Паралелна Обработка**

**Изчисление на π с произволно висока точност**

Ръководител:

Ас. Христо Христов

Изготвили:

Светослав Кръстев, 80803, КН, група 2

Ивайло Георгиев, 80880, КН, група 2

Проверил: ........................

(ас. Христо Христов)

**Постановка на задачата**

Числото Pi може да бъде изчислено по различни начини. Използвайки сходящи редове, можем да сметнем стойността на Pi с произволно висока точност.  В този проект ще използваме формулата:

  Това е един от бързо сходящите към Pi редове, открит от индийския математик Srinivasa Ramanujan през 1910­-1914 година. Със всяка негова сума той ни дава до 8 нови цифри от числото Pi.

Задачата на проекта е да се напише програма за изчисление на чилсото Pi изполвайки цитирания ред, която използва паралелни процеси (нишки) и осигурява пресмятането на Pi със зададена от потребителя точност. Изискванията към програмата са следните:

* Команден параметър задава точността на пресмятанията. Точността се изразява в брой членове на реда. Командният параметър задаващ точността има ви – “-p 1000”;
* Друг команден параметър задава максималния брой нишки (задачи) на които разделяме работата по пресмятането на Pi – например **“–t 1”** или **“—threads 3”;**
* Програмата извежда подходящи съобщения на различните етапи от работата си, както и времето отделено за изчисление на стойността на Pi;
* Записва резултата от работа си (стойността на Pi) във изходен файл, зададен с подходящ параметър, например **“­o result.txt”.** Ако този параметър е изпуснат, се избира име по подразбиране фиксирано отнапред във Вашата програма;
* Да се осигури възможност за **„quiet“** режим на работа на програмата, при който се извежда само времето отделено за изчисление на Pi, отново чрез подходящо избран друг команден параметър – например **“-­q”;**

**Алгоритъм за пресмятане на Pi**

Стандартният начин за смятане на реда на Ramanujan би бил всяка една сума и да намерим техният сбор. Тъй като обаче може да се наложи да работим с много големи числа и n!, за голямо n става много сложна операция. Затова понеже n! = 1\*2\*…\*(n-1)\*n можем да опростим сметкитe за например (n+1)! Ако сме сметнали n!. Така може да опростим частично сметките на една сума като вземем извършени сметки вече за предхождаща сума.

Частичната сума при n-тата сума, която може да бъде преизползвана е . Получаването на n+1 съответно става по следната формула:

(1)

*\* С g(k) сме означили тази част от всяка сума, която не може да използва вече сметната предхождаща я сума*.

Паралелната обработка на сумата ще я разделим между нишките по равен брой частни суми. Ако имаме t нишки, съответно на една нишка ще и се паднат по брой суми да сметне. Всяка нишка би получила индекс, на който би отговаряла ако се смятат една след друга, така че резултатите да са подредени правилно.

Проблем би се появил за първата сума, която се пада на всяка нишка да сметне, понеже не можем да вземем вече пресметната предишна сума, тък като тя е се смята от друга нишка и тази сума би била последна за нея. Тоест тази нишка трябва да приключи своята работа, за да можем да вземем от нея исканата частична сума. Ако се наложи да изчакваме да приключи тази нишка то би се изгубил паралелизма. Така всяка нишка ще чака предишната да приключи преди да започне своята работа.

Ако означим формула (1) по следния начин: f(n+1) = f(n) \* p(n) , то забелязваме че за сбора от всички следващи суми бихме могли да извадим пред скоби f(n). Тогава за сумите от n+1 натам получаваме:

(2)

След като всяка нишка свърши своята работа, трябва да вземем полученати суми от всички нишка и да ги съберем. Първо обаче трябва всяка сума получена (без тази от нишката с индекс 1) да бъде умножена с частичната сума от нишката с предишен индекс. Това идва от формула (2).

**Анализ на метричните показатели**

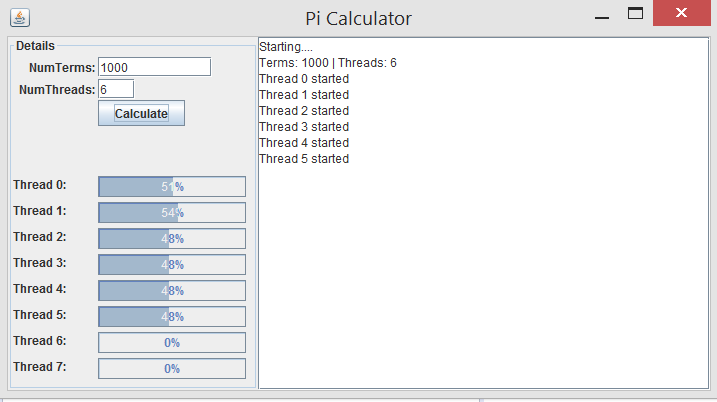
Тестовете са направени на четириядрения Intel Core i7-3612QM. На първата графика са показани броя нишки и съответното изпълнение на програмата в милисекунди.На втората и третата графика са показани ускорението Sp и ефективността Ep намирани съответно по формулите Sp = T1/Tp и Ep = Sp/p, където T1 е времето за изпълнението на серийната програма, а Tp е времето за изпъление на паралелната програма, използваща p нишки. Програмата е пусната с 1000 терма.

В лявата част е времето за изпълнение в милисекунди, а отдолу броя нишки.

В лявата част е ускоренито, а отдолу броя нишки.

Ефективносттта е показана отляво, а броя нишки отдолу.

**UI Реализация**



С помощта на javax.swing е направена UI реализация. Горе вляво в текстовите полета се въвеждат броя термове и броя нишки като при натискане на бутона Calculate отдолу се показва прогреса по всяка една нишка в Progressbar-а, а вдясно стои лога и при приключване на изчисленията се връща времето за изчисление в милисекунди.