

**Софийски Университет „Св. Климент Охридски”**

**Факултет по математика и информатика**

**Курсов проект**

**По Системи за паралелна обработка**

**Задача:** *Пресмятане на числото π*

**Изготвил: Анна Георгиева Ангелова, фак.№: 80306, 3 курс, Компютърни науки**

**Ръководител: ас. Христо Христов**

**Дата: 14.06.2011г. Проверил: ………………….. (подпис)**

**1.*Условие на курсовата задача.***

**Зад. 6 (Пресмятане на π):** Числото (стойността на) π може да бъде изчислена по различни начини. Разглеждаме следния метод за приближено пресмятане на π:

1) Имаме окръжност, вписана в квадрат;

2) Генерираме по произволен начин точки в квадрата;

3) Определяме броя на точките, които се намират в окръжността;

4) Нека **k** е число равно на броя на точките в окръжността, разделен на броя на точките в квадрата;

5) Тогава **π ~ 4 \* k**;

***Бележка***: Колкото повече точки генерираме толкова по-добра ще бъде точността с която пресмятаме π. Визуално, можем да представим метода със следната картинка:



***Изискванията към програмата са следните***:

1. Размерността на квадрата се задава в точки (пиксели) от подходящо избран команден параметър – например **“-s 10240”**. Точките в квадрата, генерираме произволно с помощта на **Math.random()** или класа **java.util.Random;**
2. Друг команден параметър задава максималния брой нишки (задачи) на които разделяме работата по пресмятането на π – например **“–t 1”** или **“–tasks 3”;**
3. Програмата извежда подходящи съобщения на различните етапи от работата си, както и времето отделено за изчисление и резултата от изчислението (стойността на π);
4. Да се осигури възможност за „quiet“ режим на работа на програмата, при който се извежда само времето отделено за изчисление на π (и самото число) , отново чрез подходящо избран друг команден параметър – например **“-q”;**

***ЗАБЕЛЕЖКА***: В условието на задачата се говори за разделянето на работата на две или повече нишки. Работата върху съответната задача на една нишка ще служи за еталон, по който да измерваме евентуално ускорение. Тоест в кода реализиращ решенията на задачите трябва да се предвиди и тази възможност – задачата да бъде решавана от единствена нишка; Пускайки програмата да работи върху задачата с помощта на единствена нишка, ще считаме че използваме серийното решение на задачата;

***2.Описание на курсовия проект***

Проекта е реализирана на Java с помощта на многонишково програмиране, то спомага за разпределянето на работата на няколко нишки, чрез което се постига по-добро ускорение и ефективност на програмата.

За изчислението има реализирани 3 класа: class Point, class MyThread и class Pi:

1)Класа Point е за създаване на точка с координати xCoordinate и yCoordinate, чрез него можем да извличаме координатите на точката и да ги променяме.

2)Класа MyThread е за създаване на нишките (процесите) и работата им в програмата. Той наследява класа Thread, който е част от пакета java.lang в Java. В този класа имаме метод run(), в който е написан програмния код на всяка нишка, т.е. какво да прави нишката, когато я стартираме. Кода се състои в това че всяка нишка създава определен брой точки с random координати в квадрата и проверява колко от тях се намират в окръжността, вписана в квадрата. За генерирането на random координати в квадрата се използва java.util.random. Чрез (new random().nextDouble())\*(страната на квадрата), сме сигурни, че нишката няма да е генерирала точки извън квадрата. Метода inCircle(long radius, Point point), на който му подаваме радиуса на окръжността и точката, която е създала нишката, проверява дали точката е в окръжността, ако е то нишката увеличава променливата, която ги брои в противен случай продължава работата си.

3)Класа Pi е за пресмятането на π. В метода main() е целия алгоритъм. Аргументите на програмата трябва да са (без значение на подредбата): 1) -q – ако имаме този аргумент то програмата ни е в quiet режим, т.е. ще изведе на стандартния изход само времето за изчисление на π и приближението на π. 2) -s [размер на страната на квадрата > 0] – чрез този аргумент въвеждаме колко пиксела да е страната на квадрата . 3) -t [брой нишки > 0] – чрез този аргумент въвеждаме боя на нишките (процесите), които искаме да се създадат. Първо проверяваме дали има правилно въведени аргументи, т.е. дали аргументите са 4 или 5 и дали са въведени положителни числа за страната на квадрата и броя нишки, ако някое от условията не е правилно, то се извежда съобщение на стандартния изход „ERROR: Arguments is not correct и присвояваме стойностите на съответните полета. Ако сме въвели правилно аргументити, то създаваме толкова на брой нишки, колкото сме подали като аргумент. Нишка се стартира чрез извикването на метода start(). Той от своя страна вика run() метода от класа MYThread, като цялата функционалност на нишката e в този метод. За приключването на работата на нишка се извиква метода join(). След приключване на работата на всички нишки се изчислява броя на всички точки в окръжността, като се сумират стойностите на променливите за броя точки в окръжността за всяка нишка. Накрая се изчислява приблизителната стойност на π, по формулата: 4\*(брой точки в окръжността / брой точки в квадрата). Ако не сме в quite режим на стандартния изход се изписва коя нишка е стартирала и с колко точки за обработка, коя нишка е приключила и с колко преброени точки, колко са точките в квадрата, колко е времето за изчисление на π и приблизителната стойност на π.

***3.Диаграми на ускорението и ефективността.***

Тестването на проекта е на 8 ядрен мултипроцесор с цел да се оценят ускорението S и ефективността Е на описания алгоритъм. Формулите за изчисляването им са:

S(n) =T1/Tn и E(n) = S(n)/n, където n е броя нишки, а Tn е времето необходимо за завършване на работата на алгоритъм с n на брой нишки.