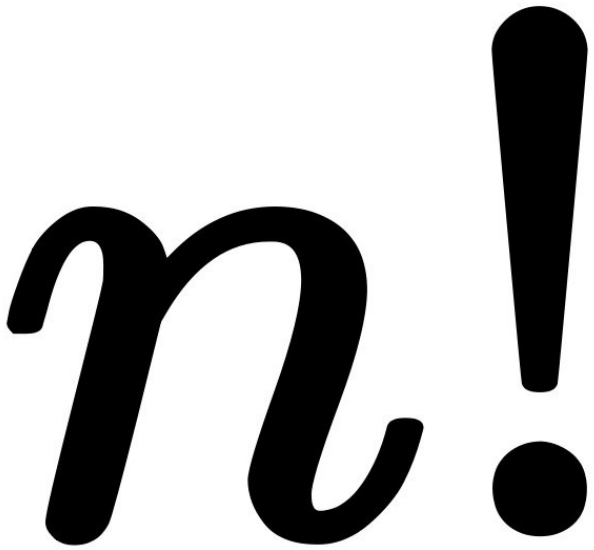
Factorial

Fernando Bobillo

Enunciado

- Calcular el factorial de un número entero



Definición de factorial

- n! denota el factorial de n
- Si n es un número natural distinto de 0, entonces

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

$$n! = \prod_{k=1}^{n} k$$

- Además, 0! = 1
- Definición recursiva:

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ (n-1)! \times n & \text{if } n > 0 \end{cases}$$

Aplicación en combinatoria

- Problema: Calcular el número de subconjuntos de k elementos a partir de un conjunto de n elementos
- Solución: número combinatorio

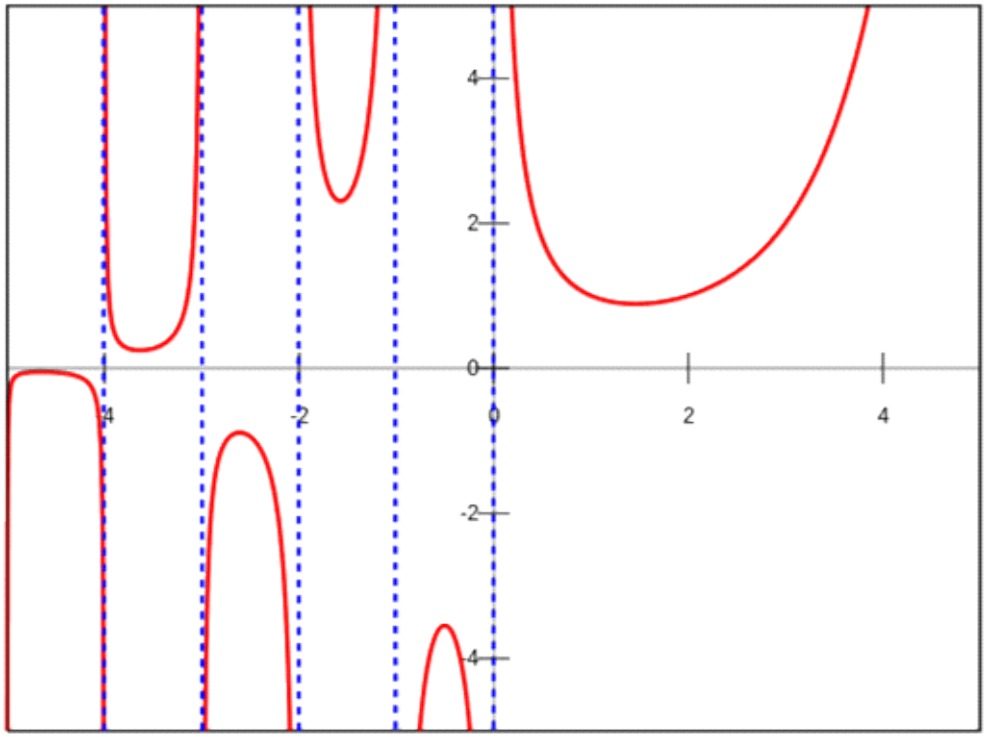
$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

- La solución requiere calcular el factorial de 3 números distintos
  - Por lo tanto, conviene implementar un método

```
public static int factorial(int n) { ... }
```

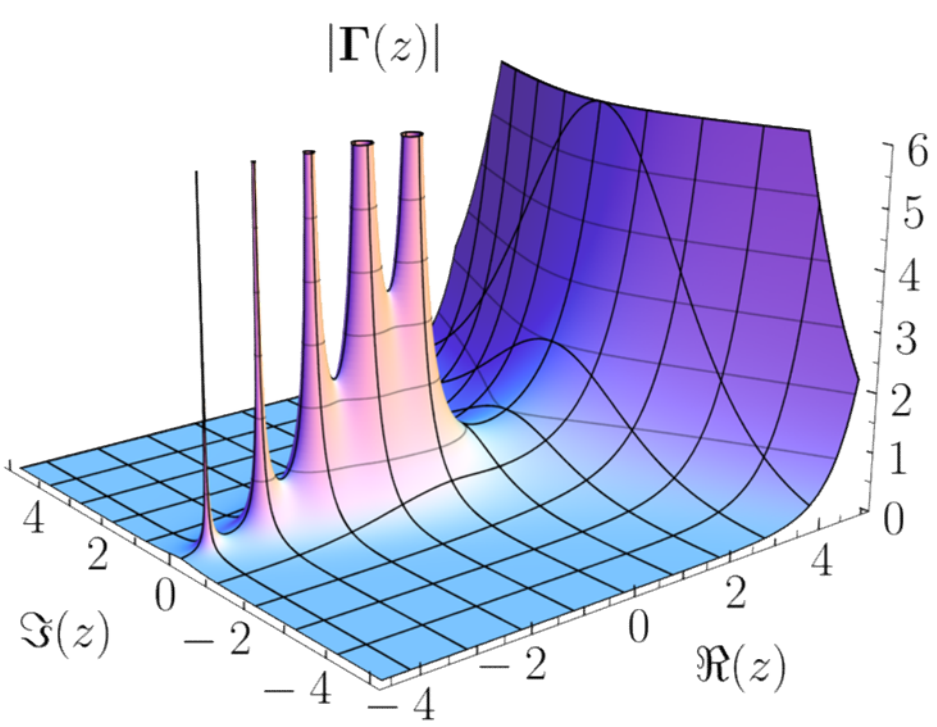
Función gamma

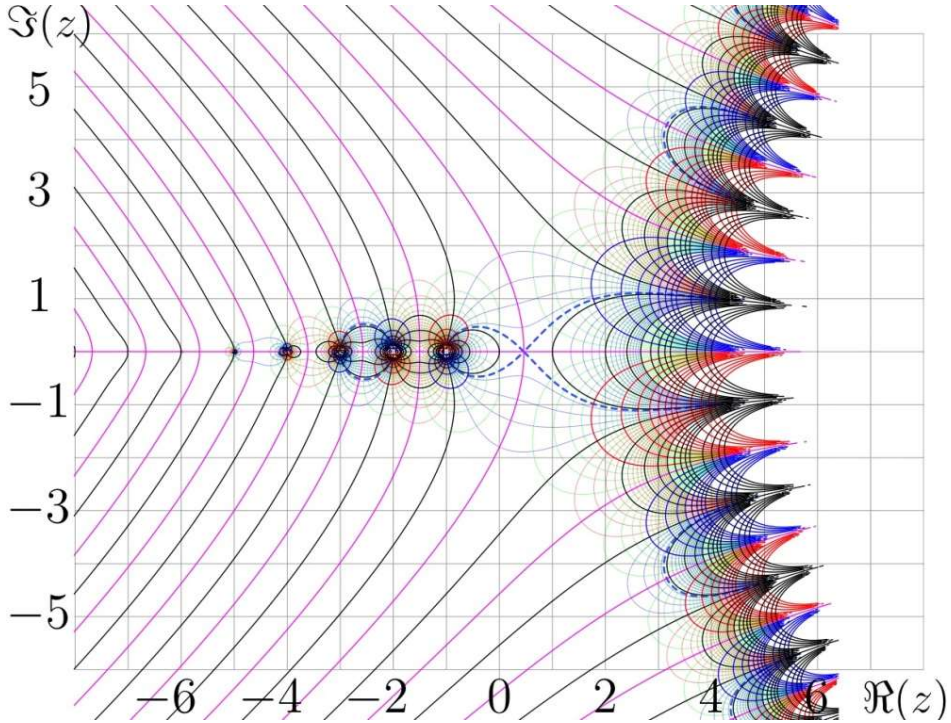
- Sea z un número complejo excepto cero y enteros negativos
- La función Γ extiende el factorial: $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt$
- La integral verifica la siguiente propiedad $\Gamma(z+1) = z\,\Gamma(z)$



Función gamma

- Sea z un número complejo excepto cero y enteros negativos
- La función Γ extiende el factorial: $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt$
- La integral verifica la siguiente propiedad $\Gamma(z+1) = z\,\Gamma(z)$





Pseudocódigo 1

```
{ Implementación sin modularizar }
programa Factorial
definición de variables
  enteros: i, fact, n;
principio
  leer(n);
  fact := 1;
  para (i:=1; i<=n; i++)
    fact := fact * i;
  escribe(fact);
fin
```

Pseudocódigo 2

```
funcion factorial(int n) : devolver entero
{ Implementación modularizada iterativa  }
principio
  entero resultado := 1;
  para (entero i := 1; i <= n; i++)
    resultado := resultado * i;
  devolver resultado;
fin
```

Pseudocódigo 3

```
funcion factorial(int n) : devolver entero
{ Implementación modularizada iterativa mejorada }
principio
  si (n >= 0)
    entero resultado := 1;
    para (entero i := 2; i <= n; i++)
      resultado := resultado * i;
    devolver resultado;
  sino
    devolver -1;
fin
```

Pseudocódigo 4

```
funcion factorial(int n) : devolver entero
{ Implementación iterativa alternativa }
principio
  si (n >= 0)
    entero resultado := 1;
    para (entero i := n; i >= 2; i--)
      resultado := resultado * i;
    devolver resultado;
  sino
    devolver -1;
fin
```

Pseudocódigo 5

```
funcion factorial(int n) : devolver entero
{ Implementación recursiva }
principio
  entero resultado;
  si (n < 0)
    resultado := -1;
  sino si (n == 0)
    resultado := 1;
  sino
    resultado := n * factorial(n-1);
  devolver resultado;
fin
```

Código en Java

¡TU TURNO!