

Gabriela Lessen Cassanova

26 08 2025

Scribe

Algoritmo (pseudocódigo)

Seleccionar entre 5 funciones con la cual precis usar la serie de Taylor para aproximar cualquier función que el usuario de hasta un nivel de error menor al error máximo permitido.

Gabriela Lessen Cassanova

26-08-2025

Algoritmo (pseudocódigo)

Funciones:

$$f_1(x) = e^x$$

$$f_2(x) = \text{sen}(x)$$

$$f_3(x) = \text{cos}(x)$$

$$f_4(x) = \ln(1+x)$$

$$f_5(x) = 1/(1-x)$$

Variables

enteras opcion, n=0

double x, epsilon, suma=0, termino=1, fact=1

Inicio

escribir ("Elige una función: ")

escribir ("1. e^x ")

escribir ("2. $\text{sen}(x)$ ")

escribir ("3. $\text{cos}(x)$ ")

escribir ("4. $\ln(1+x)$ ")

escribir ("5. $1/(1-x)$ ")

leer (opcion)

escribir ("Ingrese el valor de x: ")

leer (x)

escribir ("Ingrese epsilon (error máximo): ")

leer (epsilon)


```
Para (init, i = 1; i <= n; i++)
```

```
fact * = i
```

```
Seun (opcion)
```

```
Caso 1: //  $e^x$ 
```

```
mientras (termino > epsilon)
```

```
termino = pow(x, n) / fact(n+1)
```

```
suma += termino
```

```
n++
```

```
escribir ("Aproximación  $e^x$ : ", suma)
```

```
escribir ("Valor de n: ", n)
```

```
Finaliza el caso
```

```
Caso 2: //  $\sin x$ 
```

```
mientras (termino > epsilon)
```

```
termino = pow(-1, n) * pow(x, 2*n+1) / fact(2*n+1)
```

```
suma += termino
```

```
n++
```

```
escribir ("Aproximación  $\sin(x)$ : ", suma)
```

```
escribir ("Valor de n: ", n)
```

```
Finaliza el caso
```

```
Caso 3: //  $\cos(x)$ 
```

```
mientras (termino > epsilon)
```

```
termino = pow(-1, n) * pow(x, 2*n) / fact(2*n+1)
```

```
suma += termino
```

```
n++
```

```
escribir ("Aproximación  $\cos(x)$ : ", suma)
```

```
escribir ("Valor de n: ", n)
```

```
Finaliza el caso
```


Caso 4: $\ln(1+x)$, $|x| < 1$

Si $(x \leq -1)$

escribir ("Error: $\ln(1+x)$ no se puede calcular")

regresar a 1

$n = 1$

mientras (termino $>$ epsilon)

termino = $\text{pow}(-1, n+1) * \text{pow}(x, n) / \text{fact}$

suma += termino

$n++$

escribir ("Aproximación $\ln(1+x)$: ", suma)

escribir ("Valor de n: ", $n-1$)

Finalizar caso

Caso 5: $1/(1-x)$, $|x| < 1$

Si $(x \geq 1)$

escribir ("Error: no se puede calcular para $|x| < 1$ ")

regresar a 1

mientras (termino $>$ epsilon)

termino = $\text{pow}(x, n)$

suma += termino

$n++$

}

escribir ("Aproximación $1/(1-x)$: ", suma)

escribir ("Valor de n: ", n)

Finalizar caso

Cuando no

escribir ("No se puede calcular esta derivada")

regresar a 1

Fin

Diagrama de flujo

Inicio

Declarar variables enteras Option, n = 0

Declarar variables dobles x, epsilon, suma = 0,
termino = 0, fact =

Imprimir Elige una funcion:

1. e^x
2. $\text{Sen}(x)$
3. $\text{Cos}(x)$
4. $\ln(1+x)$
5. $1/(1-x)$

Imprimir Ingresar el valor de x.

Imprimir Ingrese epsilon (error maximo permitido)

```
for (int i = 1; i <= n; i++) {
    fact *= i;
}
```

Switch (opcion)

NO

Si

case 1:

while (termino > epsilon) {

termino = pow(x, n) / fact(n+1);

suma += termino;

n++;

Imprimir Aproximacion e^x

Imprimir Valor de n

break;

default {

Imprimir no se puede
calcular la derivada

Fin

Case 2:

```
while (termino > epsilon) {  
    termino = pow(-1, n) * pow(x, 2 + n) / fact(2 + n);  
    suma += termino;  
    n++;  
}
```

Imprimir Aproximacion $\sin(x)$

Imprimir Valor de n

break;

Case 3:

```
while (termino > epsilon) {  
    termino = pow(-1, n) * pow(x, 2 + n) / fact(2 + n + 1);  
    suma += termino;  
    n++;  
}
```

Imprimir Aproximacion $\cos(x)$

Imprimir Valor de n

break;

Case 4:

if ($x \leq -1$) {

```
    Imprimir ("Error:  $\ln(1+x)$  no se puede calcular para  $x \leq -1$ ");  
    return -1;  
}
```

n = 1;

while (termino > epsilon) {

```
    termino = pow(-1, n + 1) * pow(x / n) / fact;  
    suma += termino;
```

n++;

Imprimir Aproximacion $\ln(1+x)$

Imprimir valor de n

break;

Código: Si

if ($x \geq 1$) {Imprimir error no se puede calcular para $|x| \geq 1$

return 0;

while (termino $>$ epsilon) {

termino = pow(x, n);

suma += termino;

n++;

Imprimir Aproximación $x(1-x)^n$

Imprimir Valor de n

break;

Fin