funtras C++
Manual de Usuario

CE3102- Análisis Numérico para Ingeniería

Esteban Alvarado V.
Martín Calderón B.
Olman Castro H.
Agustín Venegaz V.

Instituto Tecnológico
de Costa Rica

funtras C++

V 1.0.1

Manual de Usuario

Esteban Alvarado Vargas
Olman Castro Hernández
Martín Calderón Blanco
Agustín Venegas Vega

Análisis Numérico para Ingeniería
CE 3102

Marzo 27, 2021

Desarrollado en el Instituto Tecnológico de Costa Rica para el curso de Análisis Numérico para Ingeniería.

Primera edición, 2021
2021, Cartago, Costa Rica.
Instituto Tecnológico de Costa Rica

Área académica de Ingeniería en Computadores
Análisis Numérico para Ingeniería, CE-3102

Contenidos

¿Qué es funtras?

Funtras es una herramienta desarrollada con fines académicos para el curso de Análisis Numérico para Ingeniería del Instituto Tecnológico de Costa Rica. Es una librería para C++ que provee un catálogo con más de 15 funciones matemáticas como seno, coseno, logaritmos, raíces, exponencial y muchas más.

Cada una de las funciones está implementada para apróximar el valor numérico de las funciones trascendentes de variable real. Las **funciones trascendentes** con aquellas funciones que no satisfacen una ecuación polinomial. Las apróximaciones se realizan mediante métodos iterativos.

```cpp
#include <funtras.hpp>

using namespace funtras;

int main(int argc, char const *argv[])
{
    double y,x;
    x = 2.71;
    y = root_t(sin_t(x)+ ln_t(x), 3) * div_t(sinh_t(sqrt_t(2)))
     + div_t(tan_t(div_t(exp_t(1))));
    return 0;
}
```

Figura 1. Ejemplo de programa con funtras.

Por eso hemos diseñado este **manual del usuario**, para que nuestros usuarios puedan sacar el máximo provecho de nuestra librería. En este manual se brindan toda la información e instrucciones necesarias para la intalación y el uso de todas las funciones que **funtras** brinda.

Con **funtras** podrá implementar el cálculo de las funciones trascendentes en sus aplicaciones de C++.

Iniciando con Funtras

Esta sección provee información detallada de la instalación de **funtras**. Incluye los detalles de los pre-requisitos y de los pasos que se deben realizar para una existosa instalación de la librería.

Pre-requisitos

Funtras es una librería para C++, por lo tanto para utilizarla debe instalar en su sistema un **compilador de C++ 11**. Nuestro equipo de desarrollo utilizó g++ para la implementación de la librería.
Puede consultar los siguientes enlaces para instalar un compilador de C++:

- **Linux:** Instalar compilador de C++ en Linux

Instalación

Primero debe obtener de https://github.com/ce-box/CE3102-FunTras el código fuente de **funtras**. Una vez que haya descargado el archivo, debe descomprimirlo en la carpeta de su preferencia.
Si se encuentra en un sistema operativo basado en UNIX puede ejecutar el archivo instalador corriendo en una terminal abierta en el directorio de los archivos:

```
$ ./install.sh
```

Nota: Es posible que deba dar permisos de ejecución al archivo con:

```
$ chmod 755 install.sh
```

Si prefiere, también es posible realizar la instalación manualmente. Lo primero que debe hacer es copiar el archivo *funtras.hpp* al directorio del sistema */usr/include* para el header esté accesible en el PATH. Puede hacerlo con el comando:

```
$ sudo cp src/funtras.hpp /usr/include
```

Luego deberá crear los directorios *build* y *lib*:

```
$ mkdir build
$ mkdir lib
```

En estos directorios se guardará el resultado de la compilación de la librería.

Finalmente deberá compilar y construir la librería para su sistema.

```
$g++ -c src/funtras.cpp -o build/funtras.o
$ar cr lib/libfuntras.a build/funtras.o
```

¿Cómo utilizar funtras?

Para incluir la librería **funtras** en su código solamente deberá realizar el include de **<funtras.hpp>**, y recomendamos que utilice el **namespace funtras** para obtener un código más legible.

```
#include <funtras.hpp>

using namespace funtras;

int main(int argc, char const *argv[])
{
    double y,x;
    x = 2.71;
    y = root_t(sin_t(x)+ ln_t(x), 3) * div_t(sinh_t(sqrt_t(2)))
     + div_t(tan_t(div_t(exp_t(1))));
    return 0;
}
```

Figura 2. Ejemplo de programa con funtras.

Compilación

Para compilar sus aplicaciones debe enlazar su archivo a la librería de funtras

```
$ g++ your_file.cpp -std=c++11 -L/path/to/lib -lfuntras -o a.out
```

Nota: En el flag -L debe colocar la dirección a la carpeta lib que contiene la librería funtras ya compilada para que el linker pueda acceder a ella y la compilación de su programa sea exitosa.

Funciones de Funtras

El propósito de esta sección es proveer al usuario los fundamentos matemáticos de las funciones implementadas en la librería y además explicar cómo utilizarlas junto con ejemplos de uso.

Funtras cuenta con 16 funciones matemáticas trascendentes y 2 funciones adicionales para calcular una apróximación de PI y otra para el cálculo del factorial de n. A continuación se detalla cada una de las funciones.

División < div_t >

Uso: Calcula el inverso de x, es decir 1/x. Se utiliza con el comando: div_t(x)
Formulación matemática: Esta función se apróxima utilizando la fórmula iterativa

$$x_{k+1} = x_k(2 - a \cdot x_k)$$

donde se necesita un valor inicial dado por:

$$x_0 = \begin{cases} \text{eps}^{15} & \text{si } 80! < a \leq 100! \\ \text{eps}^{11} & \text{si } 60! < a \leq 80! \\ \text{eps}^{8} & \text{si } 40! < a \leq 60! \\ \text{eps}^{4} & \text{si } 20! < a \leq 40! \\ \text{eps}^{2} & \text{si } \ 0! < a \leq 20! \end{cases}$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void div_test(){
    // Calcular 1/2
    cout << funtras::div_t(2) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
0.5
```

Exponencial < exp_t >

Uso: Calcula el valor numérico de e^x. Se utiliza con el comando: exp_t(x)
Formulación matemática: Esta función se apróxima utilizando el polinomio:

$$S_k(a) = \sum_{n=0}^{k} \frac{a^n}{n!}.$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void exp_test(){
    // Calcular e^2 = 7.38905
    cout << funtras::exp_t(2) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
7.38906
```

Seno < sin_t >

Uso: Calcula el valor numérico de sin(x). Se utiliza con el comando: sin_t(x)
Formulación matemática: Esta función se apróxima utilizando el polinomio:

$$S_k(a) = \sum_{n=0}^{k} (-1)^n \frac{a^{2n+1}}{(2n+1)!}.$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void sin_test(){
    // Calcular sin(pi/2) = 1
    cout << funtras::sin_t(1.5707) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
1
```

Coseno < cos_t >

Uso: Calcula el valor numérico de cos(x). Se utiliza con el comando: cos_t(x)
Formulación matemática: Esta función se apróxima utilizando el polinomio:

$$S_k(a) = \sum_{n=0}^{k} (-1)^n \frac{a^{2n}}{(2n)!}.$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void cos_test(){
    // Calcular cos(pi/2) = 0
    cout << funtras::cos_t(1.5707) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
9.63331e-05
❯ echo "Valor muy apróxido a 0 -> 0.0000963331"
```

Logaritmo natural < ln_t >

Uso: Calcula el valor numérico de ln(x). Se utiliza con el comando: ln_t(x)
Formulación matemática: Esta función se apróxima utilizando la serie:

$$S_k(a) = \frac{2(a-1)}{a+1} \sum_{n=0}^{k} \frac{1}{2n+1} \left(\frac{a-1}{a+1}\right)^{2n}$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular ln(2) = 0.693147
    cout << funtras::ln_t(2) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
0.693147
```

Seno Hiperbólico < sinh_t >

Uso: Calcula el valor numérico de sinh(x). Se utiliza con el comando: sinh_t(x)
Formulación matemática: Esta función se apróxima utilizando el polinomio:

$$S_k(a) = \sum_{n=0}^{k} \frac{a^{2n+1}}{(2n+1)!}$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular sinh(6) = 201.71
    cout << funtras::sinh_t(6) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
201.713
```

Coseno Hiperbólico < cosh_t >

Uso: Calcula el valor numérico de cosh(x). Se utiliza con el comando: cosh_t(x)
Formulación matemática: Esta función se apróxima utilizando el polinomio:

$$S_k(a) = \sum_{n=0}^{k} \frac{a^{2n}}{(2n)!}$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular cosh(15) = 1634508.68624
    cout << funtras::cosh_t(15) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
1.63451e+06
```

Tangente Hiperbólico < tanh_t >

Uso: Calcula el valor numérico de tanh(x). Se utiliza con el comando: tanh_t(x)
Formulación matemática: Esta función se apróxima con la fórmula:

$$\tanh x = \frac{\sinh x}{\cosh x}$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular tanh(15) = 1
    cout << funtras::tanh_t(15) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
1
```

Raíz enésima < root_t >

Uso: Calcula el valor numérico de (x)^1/a. Se utiliza con el comando: root_t(x,a)
Formulación matemática: Esta función se calcula al solucionar $g(x) = x^p - a.$ y a partir de allí se obtiene:

$$root\_t(x,a) = \frac{a}{2} - \frac{(\frac{a}{2})^x - a}{\frac{a}{2} \cdot x}$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular root(9,3) = 2,08008
    cout << funtras::root_t(9,3) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
2.08008
```

Raíz cuadrada < sqrt_t >

Uso: Calcula el valor numérico de sqrt(x). Se utiliza con el comando: sqrt_t(x)
Formulación matemática: Esta función se apróxima con la fórmula:

$$root\_t(x,a) = \frac{a}{2} - \frac{(\frac{a}{2})^x - a}{\frac{a}{2} \cdot x}$$

cuando el valor de a = 2.
Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular sqrt(9) = 3
    cout << funtras::sqrt_t(9) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
3
```

Arcoseno < asin_t >

Uso: Calcula el valor numérico de asin(x). Se utiliza con el comando: asin_t(x,a)
Formulación matemática: Esta función puede aproximar usando el polinomio:

$$S_k(a) = \sum_{n=0}^{k} \frac{(2n)!}{4^n(n!)^2(2n+1)} a^{2n+1}$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular asin(0.5) = 0.5235
    cout << funtras::asin_t(0.5) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
0.523599
```

Arcocoseno < acos_t >

Uso: Calcula el valor numérico de acos(x). Se utiliza con el comando: acos_t(x)
Formulación matemática: Esta función se apróxima utilizando la fórmula:

$$\text{acos }\_t(a) = \frac{\pi}{2} - \text{asin }\_t(a)$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular acos(0.5) = 1.04719
    cout << funtras::acos_t(0.5) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.ou
❯ ./run.out
1.047
```

Arcotangente < atan_t >

Uso: Calcula el valor numérico de atan(x). Se utiliza con el comando: atan_t(x)
Formulación matemática: Esta función se apróxima utilizando el polinomio:

$$S_k(a) = \sum_{n=0}^{k} (-1)^n \frac{a^{2n+1}}{2n+1}$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular atan(0.5) = 0.4636476
    cout << funtras::atan_t(0.5) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
0.463648
```

Potencia < power_t >

Uso: Calcula el valor numérico de a^x. Se utiliza con el comando: power_t(x,a)
Formulación matemática: Esta función se apróxima utilizando la fórmula:

$$power\_t(x,a) = \prod_{n=1}^{x} a$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular power(8,2) = 256
    cout << funtras::power_t(8,2) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
256
```

Logaritmo base a < log_t >

Uso: Calcula el valor numérico de log_a(x). Se utiliza con el comando: log_t(x,a)
Formulación matemática: Esta función se apróxima utilizando la fórmula:

$$\log\_t(x,a) = \frac{\ln(x)}{\ln(a)} \rightarrow log_a(x)$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    // Calcular log_5(25) = 2
    cout << funtras::log_t(25,5) << endl;
}
```

```
❯ g++ FunTras/test/doc_test.cpp -std=c++11 -L./FunTras/lib -lfuntras -o run.out
❯ ./run.out
2
```

PI <pi_t>

Uso: Apróxima el valor de pi. Se utiliza con el comando: pi_t(x)
Formulación matemática: Esta función se apróxima utilizando la serie de Leibniz:

$$\frac{\pi}{4} = \sum_{n=0}^{k} \frac{(-1)^n}{2n+1}$$

Ejemplo:

```cpp
#include <iostream>
#include <funtras.hpp>

using namespace std;
using namespace funtras;

void test(){
    cout << pi_t() << endl;
}
```

¡Acceda al código fuente escaneando este código desde su móvil!