

Herramientas Computacionales para Matemática Aplicada

Curso 2020



Introducción a la
programación en C++

A diferencia de ‘C’, ‘C++’ es un lenguaje “orientado a objetos”

La POO se fue convirtiendo en el estilo de programación dominante a mediados de los años 1980, en gran parte debido a la influencia de C++, una extensión del lenguaje de programación C. Su dominación fue consolidada gracias al auge de las interfaces gráficas de usuario, para las cuales la POO está particularmente bien adaptada. En este caso, se habla también de programación dirigida por eventos.

Entre los lenguajes orientados a objetos se destacan los siguientes:

- ABAP⁸
- ABL⁹
- ActionScript
- ActionScript 3
- C Sharp (C#)
- Clarion
- Clipper¹⁰
- D
- Object Pascal (Embarcadero Delphi)
- Gambas
- GObject
- Genie
- R
- Pauscal (en español)
- Perl^{13 14}
- Perl 6
- PHP¹⁵
- PowerScript
- Processing.¹⁶
- Python
- Ruby
- Self
- Smalltalk¹⁷
- Swift
- Magik

Paradigmas fundamentales de la POO (en pocas palabras ...)

- **Encapsulación** : término por el que se conoce al ocultamiento de los detalles internos de la implementación de una clase (en general, permite que nos ocupemos en que métodos y datos tiene una dada clase, más que en los detalles de implementación).
- **Herencia** : es la capacidad de generar una clase derivada a partir de una clase base, heredando y ampliando sus características. Una clase puede tener más de una clase base, lo que se conoce como “herencia múltiple”.
- **Polimorfismo** : en la POO, un objeto puede responder de forma diferente ante llamadas similares, con sensibilidad al contexto con el que se lo llama. Esto se logra usando, bien funciones virtuales, bien la sobrecarga de operadores o bien la sobrecarga de funciones.

Paradigmas fundamentales de la POO (en pocas palabras ...)

- **Funciones virtuales** : una función de una clase derivada que es redefinida en la clase derivada. Por lo cual, pueden realizar operaciones distintas, aunque se llamen igual.
- **Sobrecarga de operadores**: pueden redefinirse los operadores para que actúen de manera controlada sobre los objetos de una clase.
- **Sobrecarga de funciones** : una misma función puede redefinirse varias veces para que actúe de manera distinta según el tipo de variable de los parámetros de entrada. El programa, en tiempo de ejecución, detecta los tipos de que se trata y llama a la función correspondiente.

Primeras consideraciones al pasar de C a C++:

- C++ es un superconjunto de C, por lo que los programadores pueden utilizar un compilador C++ para compilar sus programas existentes en C, y de ahí modificar de forma gradual estos programas a C++.
- C requiere que un comentario quede delimitado por `/*` y `*/`. C++ permite que un comentario empiece con `//` y que termine al final de la línea.
- C++ permite que las declaraciones aparezcan en cualquier parte en que un enunciado ejecutable pueda aparecer. Las declaraciones variables pueden también aparecer en la sección de inicialización de una estructura `for`.
- C++ proporciona una alternativa a las funciones `printf` y `scanf` para manejar la entrada/salida de los tipos de datos estándar y de las cadenas. El flujo de salida `cout`, y el operador `<<` (operador de inserción de flujo, que se lee “colocar en”) le permite la salida de los datos. El flujo de entradas `cin` y el operador `>>` (el operador de extracción de flujo, que se lee “obtener de”) permite que se introduzcan datos.

Hola, Mundo! en C++

```
#include <iostream>

int main()
{
    std::cout << "Hola, Mundo!" << std::endl; //esta es la única línea de código que hace algo "visible" ;-)

    return(0);
}
```

Compilador gnu de C++



```
$ g++ Hola.cpp -o Hola -lm
$ ./Hola
Hola, Mundo!
$
```

Palabras reservadas en C++:

asm	do	inline	short	typeid
auto	double	int	signed	typename
bool	dynamic_cast	long	sizeof	union
break	else	mutable	static	unsigned
case	enum	namespace	static_cast	using
catch	explicit	new	struct	virtual
char	extern	operator	switch	void
class	false	private	template	volatile
const	float	protected	this	wchar_t
const_cast	for	public	throw	while
continue	friend	register	true	
default	goto	reinterpret_cast	try	
delete	if	return	typedef	

El modificador const :

The `const` keyword is used to create a “read only” object. As an object of this type is constant, it cannot be modified at a later stage and must be initialized during its definition.

EXAMPLE: `const double pi = 3.1415947;`

Thus the value of `pi` cannot be modified by the program. Even a statement such as the following will merely result in an error message:

```
pi = pi + 2.0; // invalid
```

Archivos de cabecera de la biblioteca estándar de C++ :

Header files of the C++ standard library

algorithm	ios	map	stack
bitset	iosfwd	memory	stdexcept
complex	iostream	new	streambuf
dequeue	istream	numeric	string
exception	iterator	ostream	typeinfo
fstream	limits	queue	utility
functional	list	set	valarray
iomanip	locale	sstream	vector

Sobrecarga de funciones y argumentos por defecto :

C++ allows you to overload functions, that is, different functions can have the same name.

Example: `int max(int x, int y);
double max(double x, double y);`

In our example two different function share the same name, `max`. The function `max()` was overloaded for `int` and `double` types. The compiler uses a function's signature to differentiate between overloaded functions.

Example: `void moveTo(int x = 0, int y = 0);`

Parameter names can be omitted, as usual.

Example: `void moveTo(int = 0, int = 0);`

The function `moveTo()` can then be called with or without one or two arguments.

Example: `moveTo(); moveTo(24); moveTo(24, 50);`

The first two calls are equivalent to `moveTo(0, 0);` or `moveTo(24, 0);`.

It is also possible to define default arguments for only some of the parameters. The following general rules apply:

- the default arguments are defined in the function prototype. They can also be supplied when the function is defined, if the definition occurs in the same source file and before the function is called
- if you define a default argument for a parameter, all following parameters must have default arguments

Escritura en archivos:

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <string>

using namespace std;

int main(int numarg,char **cargs)
{
ofstream Outfile;
string Encabezado("El número importante es: ");

Outfile.open ("Datos.dat",ios_base::out);
Outfile.precision(9);

Outfile << Encabezado << M_PI << endl;

cout << "Listo, mensaje guardado en el archivo " << "Datos.dat" << "!" << endl;
return(0);
}
```

```
$ ./a.out
Listo, mensaje guardado en el archivo Datos.dat!
$ more Datos.dat
El número importante es: 3.14159265
```

Asignación dinámica de memoria:

```
#include <iostream>
#include <new>
#include <cmath>

using namespace std;

int main(int argc, char const *argv[])
{
double *array;
int size;

cout << "Que tamaño queres darle al array ? (introducir un entero)" << endl;
cin >> size; //leemos el entero ingresado

array = new double[size];

for (int i = 0; i < size; ++i)
{
    array[i] = pow(-1,i)*i*i;
    cout << i << " " << array[i] << endl;
}

delete[] array; //liberamos la memoria

return(0);
}
```

```
$ ./a.out
Que tamaño queres darle al array ? (introducir un entero)
13
0  0
1  -1
2  4
3  -9
4  16
5  -25
6  36
7  -49
8  64
9  -81
10  100
11  -121
12  144
```

Plantillas (templates) de funciones:

```
// function template
#include <iostream>
using namespace std;

template <class T>
T sum (T a, T b)
{
    T result;
    result = a + b;
    return result;
}

int main () {
    int i=5, j=6, k;
    double f=2.0, g=0.5, h;
    k=sum<int>(i,j);
    h=sum<double>(f,g);
    cout << k << '\n';
    cout << h << '\n';
    return 0;
}
```

11
2.5

```
// Using template functions
#include <iostream.h>

template <class T>
void printArray(T *array, const int count)
{
    for (int i = 0; i < count; i++)
        cout << array[i] << " ";
    cout << '\n';
}

main()
{
    const int aCount = 5, bCount = 7, cCount = 6;
    int a[aCount] = {1, 2, 3, 4, 5};
    float b[bCount] = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7};
    char c[cCount] = "HELLO"; // 6th position for null

    cout << "Array a contains:\n";
    printArray(a, aCount); // integer version

    cout << "Array b contains:\n";
    printArray(b, bCount); // float version

    cout << "Array c contains:\n";
    printArray(c, cCount); // character version

    return 0;
}
```

```
Array a contains:
1 2 3 4 5
Array b contains:
1.1 2.2 3.3 4.4 5.5 6.6 7.7
Array c contains:
H E L L O
```

Ejemplo de uso de la clase std::string :

```
#include <iostream>
#include <string>

int main(int argc, char const *argv[])
{
    std::string my_string1("Esto ");
    std::string my_string2("es una ");

    std::cout << my_string1 + my_string2 + "prueba" << std::endl;

    std::string my_string3("Estado");
    std::string my_string4("Orquestado");
    std::string my_string5;

    std::cout << my_string3 << " tiene " << my_string3.length() << " caracteres" << std::endl;
    std::cout << my_string4 << " tiene " << my_string4.length() << " caracteres" << std::endl;

    if(my_string3 < my_string4){
        std::cout << my_string3 << " es lexicográficamente menor que " << my_string4 << std::endl;
    }
    else if(my_string3 > my_string4){
        std::cout << my_string3 << " es lexicográficamente mayor que " << my_string4 << std::endl;
    }
    else{
        std::cout << my_string3 << " es lexicográficamente igual que " << my_string4 << std::endl;
    }
}
```

Ejemplo de uso de la clase std::string (cont.):

```
my_string5 = my_string3;

if(my_string3 < my_string5){
    std::cout << my_string3 << " es lexicográficamente menor que " << my_string5 << std::endl;
}
else if(my_string3 > my_string5){
    std::cout << my_string3 << " es lexicográficamente mayor que " << my_string5 << std::endl;
}
else{
    std::cout << my_string3 << " es lexicográficamente igual que " << my_string5 << std::endl;
}

return 0;
}
```

```
$ g++ EjStrings.cpp -lm
$ ./a.out
Esto es una prueba
Estado tiene 6 caracteres
Orquestado tiene 10 caracteres
Estado es lexicográficamente menor que Orquestado
Estado es lexicográficamente igual que Estado
```

Ejemplo : creando una clase para números complejos*

* obviamente, solo con fines didácticos. La biblioteca estándar ya cuenta con tal clase, debidamente implementada y optimizada...

Archivo de cabecera complejos.h

```
namespace flib
{
    class Complejo
    {
        public:
            Complejo(); //constructor
            void Setear(double, double);
            void ImprimirComplejo(void);
            void ImprimirParteRe(void);
            void ImprimirParteIm(void);
            double Norma(void);
            Complejo Conjugado();

            Complejo operator+(const Complejo& b)
            {
                Complejo ret;

                ret.Re = this->Re + b.Re;
                ret.Im = this->Im + b.Im;

                return (ret);
            }

            Complejo operator-(const Complejo& b)
            {
                Complejo ret;

                ret.Re = this->Re - b.Re;
                ret.Im = this->Im - b.Im;
            }
    };
}
```

Espacio de nombres

“métodos” o “funciones miembro de la clase”

Sobrecarga de los operadores aritméticos para operar entre complejos de modo natural

Ejemplo : creando una clase para números complejos

Archivo de
cabecera
complejos.h
(continuación)

```
Complejo ret;

ret.Re = this->Re - b.Re;
ret.Im = this->Im - b.Im;

return (ret);
}

Complejo operator*(const Complejo& b)
{
    Complejo ret;

    ret.Re = this->Re*b.Re-this->Im*b.Im;
    ret.Im = this->Im*b.Re+this->Re*b.Im;

    return (ret);
}

Complejo operator/(const Complejo& b)
{
    Complejo ret;

    ret.Re = (this->Re*b.Re+this->Im*b.Im)/(b.Re*b.Re+b.Im*b.Im);
    ret.Im = (this->Im*b.Re-this->Re*b.Im)/(b.Re*b.Re+b.Im*b.Im);

    return (ret);
}

private:
double Re,Im;
};

}//end namespace flib
```

Datos privados de la clase. No pueden ser leídos ni modificados directamente desde fuera de la interfaz que provee la clase (encapsulación)

Ejemplo : creando una clase para números complejos

Archivo fuente
complejos.cpp,
con la
implementación
de los métodos
de la clase

```
using namespace std;
using namespace flib;

/*constructor*/
Complejo::Complejo()
{
Re=0.0;
Im=0.0;
}

void Complejo::Setear(double x, double y)
{
Re=x;
Im=y;
}

void Complejo::ImprimirComplejo()
{
std::cout << "(" << Re << "," << Im << ")" << std::endl;
}

void Complejo::ImprimirParteRe()
{
std::cout << Re << std::endl;
}
```

Ejemplo : creando una clase para números complejos

Archivo fuente
complejos.cpp,
con la
implementación
de los métodos
de la clase
(continuación)

```
void Complejo::ImprimirParteIm()
{
    std::cout << Im << std::endl;
}

double Complejo::Norma()
{
    return(sqrt(Re*Re+Im*Im));
}

Complejo Complejo::Conjugado()
{
    Complejo ret;

    ret.Re = this->Re;

    ret.Im = -this->Im;
    return(ret);
}
```

Usando la clase creada

```
#include <iostream>
#include <complejos.h>

using namespace std;
using namespace flib;

int main(int numarg,char **cargs)
{
int NumTotLineas;

Complejo x,v,w,z;

v.Setear(1,5);
w.Setear(1,-1);

z=v+w;
x=v/w;

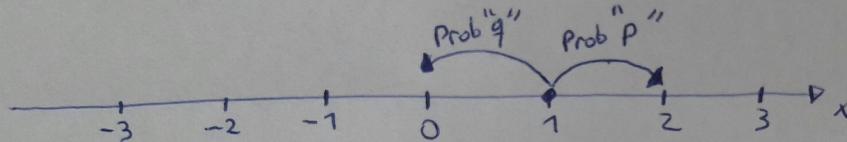
z.ImprimirComplejo();
cout << endl;
x.ImprimirComplejo();
cout << endl;
z.ImprimirParteIm();
cout << endl;
z.ImprimirParteRe();
cout << endl;
cout << z.Norma() << endl;
cout << endl;
x=v.Conjugado();
x.ImprimirComplejo();
cout << endl;

return(0);
}
```

```
$ ./a.out
(2,4)
(-2,3)
4
2
4.47214
(1,-5)
```

Caso de estudio: Random Walk (1D)

Caminata aleatoria 1D



Al cabo de N pasos (N puede considerarse una variable temporal discreta), la probabilidad de que hayan ocurrido r pasos a la derecha ($y N-r$ a la izquierda) está dada por la distribución binomial (" N pruebas de Bernoulli")

$$P_r(N) = \binom{N}{r} P^r q^{N-r}$$

Caso de estudio: Random Walk (1D)

Si la caminata es "insegada" ($P = q = \frac{1}{2}$)
es fácil mostrar que la variable aleatoria
posición al tiempo "N" $X(N)$ tiene valor medio

$$\boxed{\langle X \rangle(N) = 0}$$

Por su parte, si consideramos el valor medio
de la distancia del caminante al punto de
partida (asumir que es el origen) puede
demostrarse que, para $N \gg 1$, se tiene

$$\boxed{\langle d \rangle(N) \sim \sqrt{\frac{2N}{\pi}}}$$

{Ver artículo en }
mathworld.wolfram.com

Implementación en C++: random_walk1D.cpp

```
#include <iostream>
#include <vector>

#define NumSteps 10000
#define NumRealizaciones 100

int main()
{
int PosCaminante;

std::vector<double> SumaPosiciones(NumSteps); //para promediar entre realizaciones
std::vector<double> SumaDistancias(NumSteps); //para promediar entre realizaciones
//todos los elementos de SumaPosiciones y SumaDistancias son inicializados en 0

for(int k=0; k<NumRealizaciones; k++){//nueva realización del RW

    PosCaminante=0; //inicializamos la posición del caminante en cada realización,
                    //para que salga desde el origen

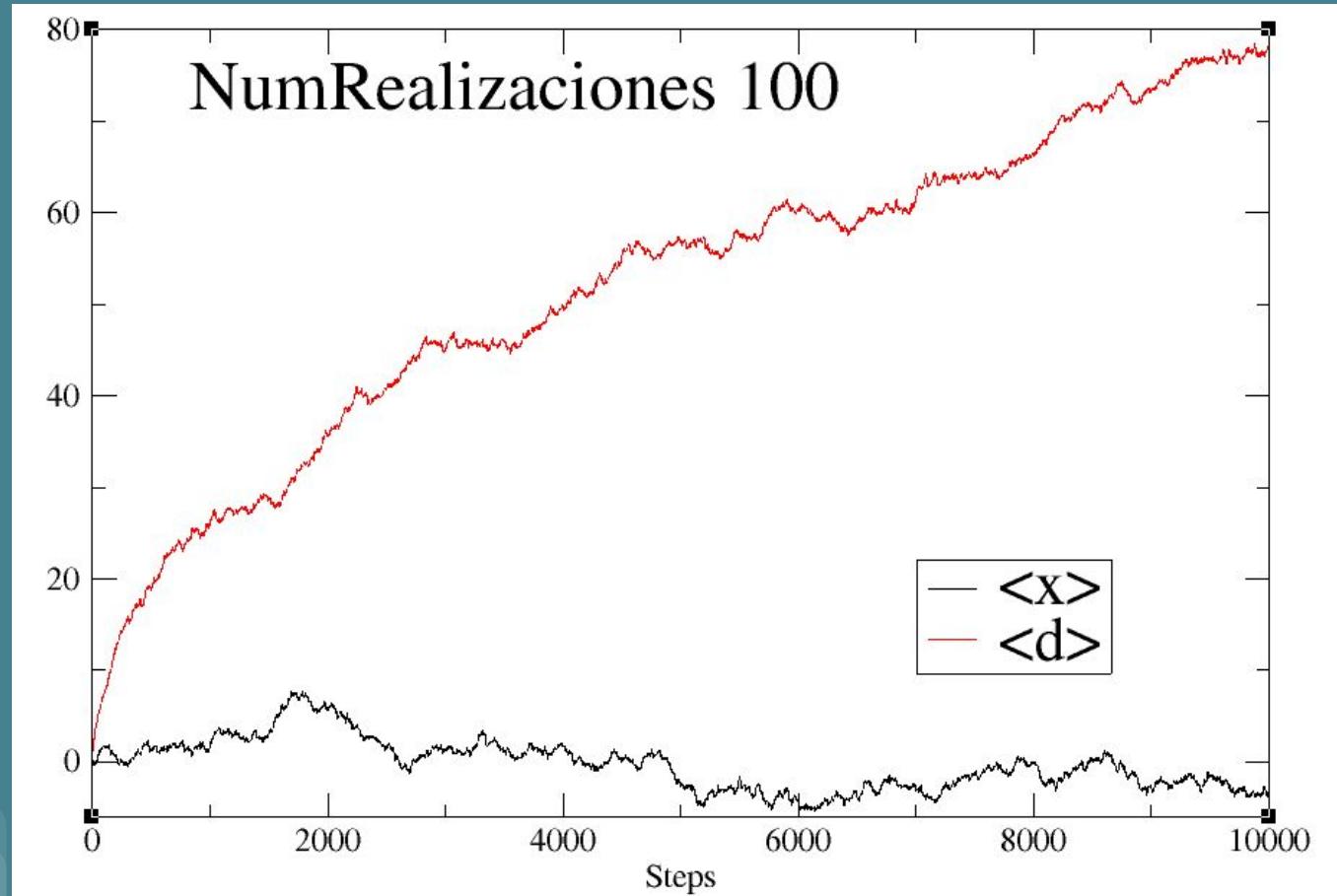
    for (int i = 0; i < NumSteps; ++i){ //realizamos la caminata aleatoria del caminante

        SumaPosiciones[i] += PosCaminante;
        SumaDistancias[i] += abs(PosCaminante);
```

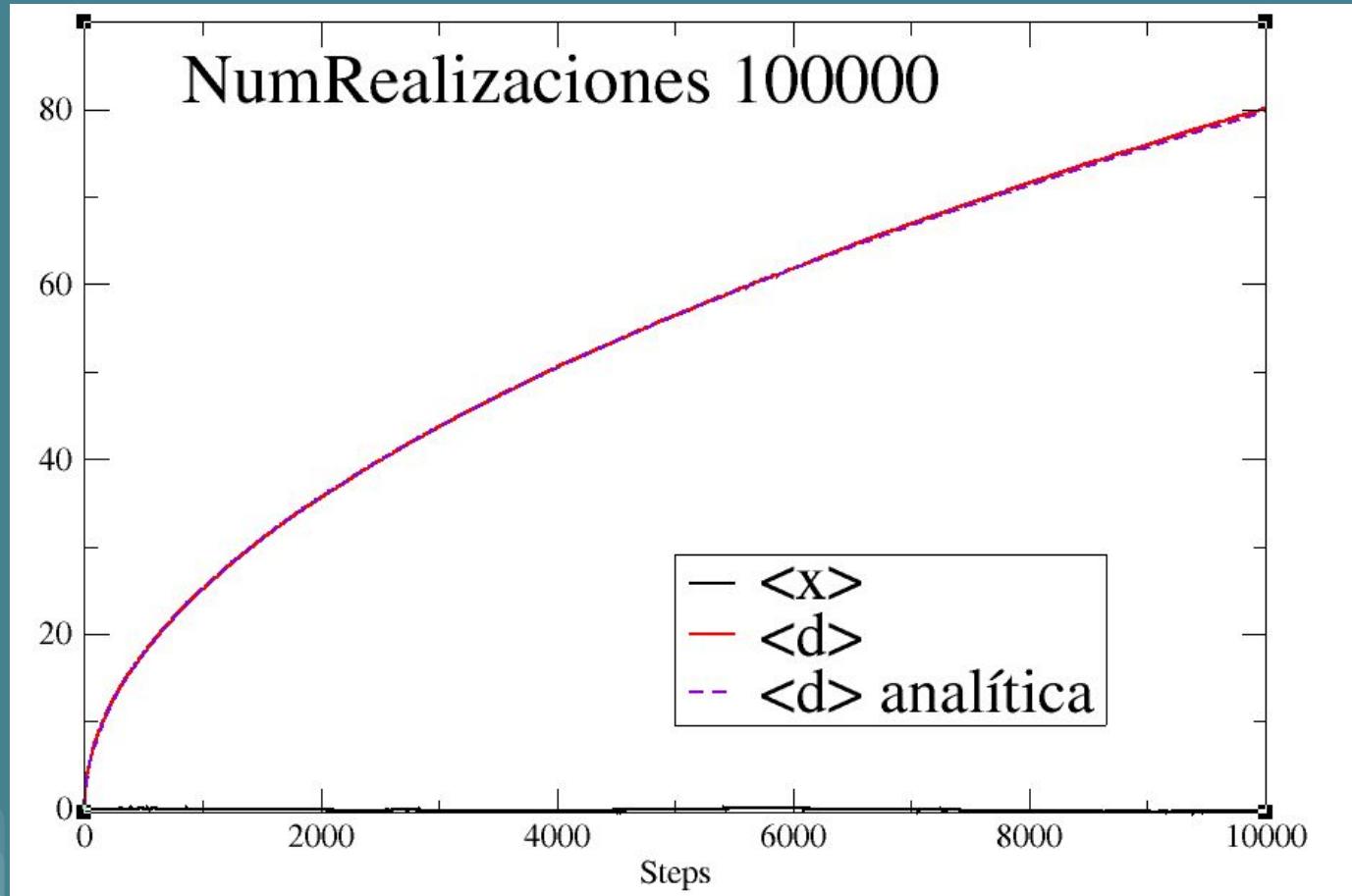
Implementación en C++: random_walk1D.cpp (cont.)

```
//usamos el valor por defecto de la semilla.  
//usamos rand()%2 para obtener 0 o 1 con probabilidad 0.5. Esta no es la mejor manera  
//de hacer esto (mejor usar generadores de números aleatorios optimizados por ejemplo  
//los provistos en la biblioteca gsl) pero sirve para este ejemplo sencillo  
    if(rand()%2 == 1){ //paso a la derecha  
        PosCaminante++;  
    }  
    else{ //paso a la izquierda  
        PosCaminante--;  
    }  
  
    }  
  
}  
  
//imprimimos promedios  
for (int i = 0; i < NumSteps; ++i){  
    std::cout << i << " " << SumaPosiciones[i]/NumRealizaciones << " " <<  
        SumaDistancias[i]/NumRealizaciones << std::endl;  
}  
return(0);  
}
```

Resultados numéricos:



Resultados numéricos:



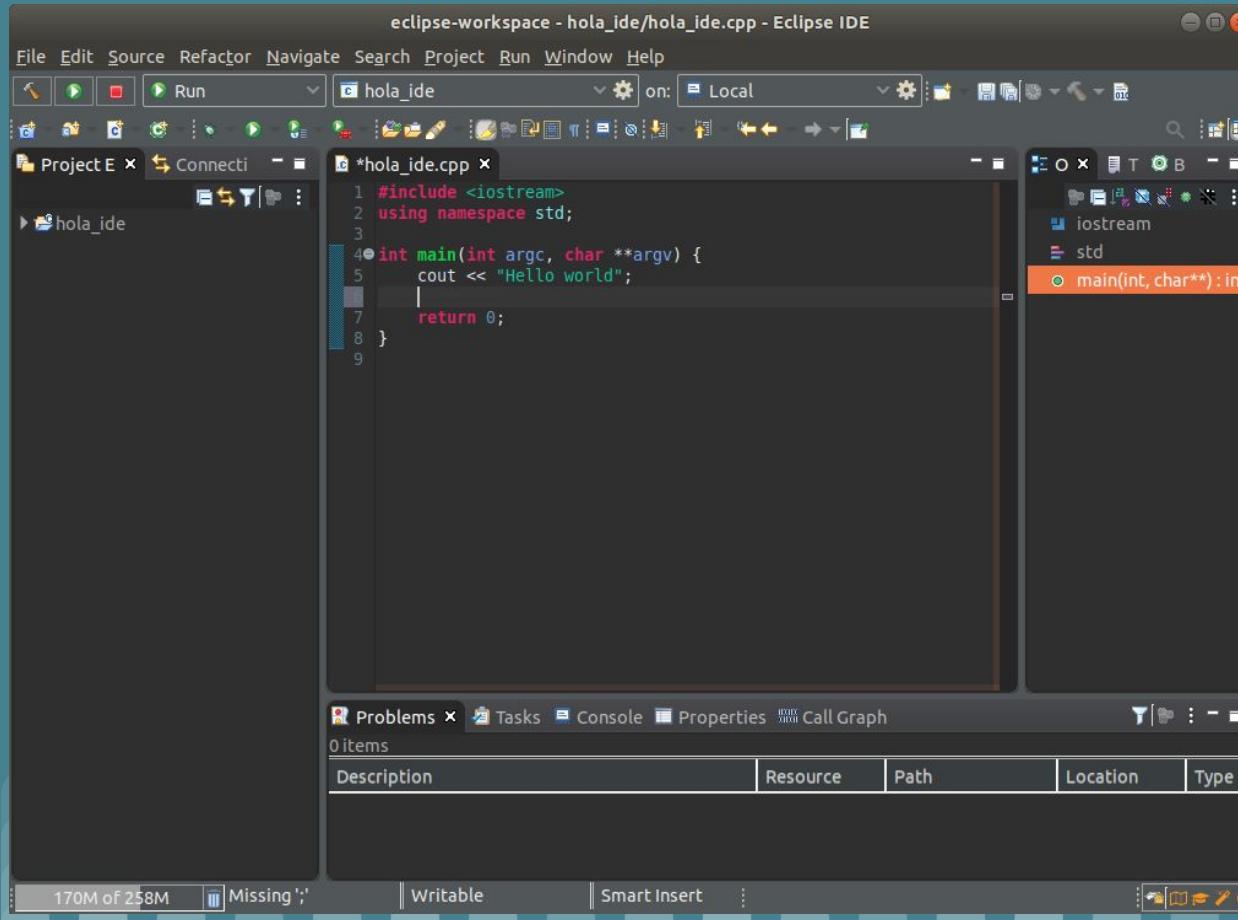
Programando en entornos integrados (IDEs):

The screenshot shows the 'Welcome to the Eclipse IDE for C/C++ Developers' screen. At the top left is the Eclipse logo. To its right is the title 'Welcome to the Eclipse IDE for C/C++ Developers'. In the top right corner is a yellow play button icon with the word 'Workbench' below it. The main area contains several items with icons and descriptions:

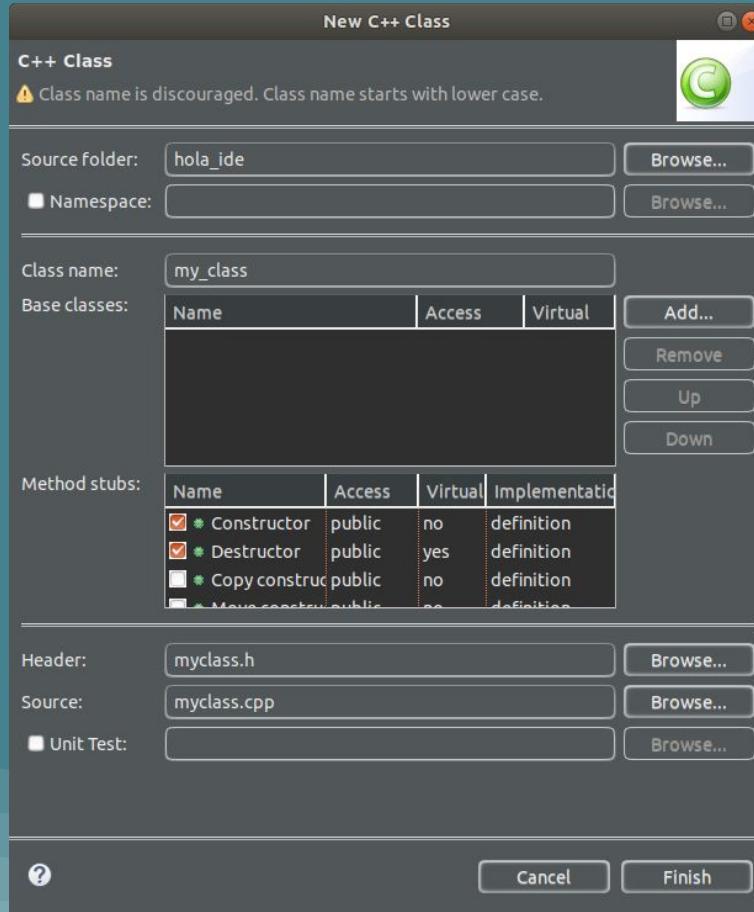
- Tutorial: Import an existing project** (with a list icon): A guided walk-through how to import an existing project.
- Overview** (with a book icon): Get an overview of the features.
- Review IDE configuration settings** (with a gear icon): Review the IDE's most fiercely contested preferences.
- Tutorials** (with a graduation cap icon): Go through tutorials.
- Create a new C/C++ project** (with a plus icon): Create a new Eclipse project for C/C++ source code.
- Samples** (with a paintbrush icon): Try out the samples.
- Import a project with a working Makefile** (with a plus icon): Open the New item wizard.
- What's New** (with a star icon): Find out what is new.
- Checkout projects from Git** (with a diamond icon): Checkout Eclipse projects hosted in a Git repository.

At the bottom left, there is a download icon next to the text 'Import existing projects' and a description: 'Import existing Eclipse projects from the filesystem or archive'. At the bottom right, there is a checked checkbox labeled 'Always show Welcome at start up'.

Programando en entornos integrados (IDEs):



Algunas funcionalidades de Eclipse:



```
1/*  
2 * miclase.h  
3 *  
4 * Created on: Mar 11, 2020  
5 * Author: fede  
6 */  
7  
8#ifndef MICLASE_H_  
9#define MICLASE_H_  
10  
11class mi_clase {  
12public:  
13    mi_clase();  
14    virtual ~mi_clase();  
15};  
16  
17#endif /* MICLASE_H */  
18
```

Compilación de proyectos complejos:

Crear un archivo Makefile “a mano” para un proyecto complejo puede ser una tarea ardua. Por eso, existen varias aplicaciones que tienen la capacidad de crear Makefiles de manera automatizada

```
KqkWallFunctionFvPatchFields.C --> ../derivedFvPatchFields/wallFunctions/kqkWallFunctions/kqkWallFunction/kqkWallFunctionFvPatchField.H
kqRWallFunctionFvPatchFields.C --> ../derivedFvPatchFields/wallFunctions/kqWallFunctions/kqRWallFunction/kqRWallFunctionFvPatchFields.C
kqRWallFunctionFvPatchFields.H --> ../derivedFvPatchFields/wallFunctions/kqWallFunctions/kqRWallFunction/kqRWallFunctionFvPatchFields.H
laminarModel.C --> ../laminar/laminarModel/laminarModel.C
laminarModel.H --> ../laminar/laminarModel/laminarModel.H
laplaceFilter.C --> ./LES/LESfilters/laplaceFilter/laplaceFilter.C
laplaceFilter.H --> ./LES/LESfilters/laplaceFilter/laplaceFilter.H
LauderSharmaKE.C --> ../RAS/LauderSharmaKE/LauderSharmaKE.C
LauderSharmaKE.H --> ../RAS/LauderSharmaKE/LauderSharmaKE.H
LESDelta.C --> ../LES/LESdeltas/LESdelta/LESdelta.C
LESDelta.H --> ../LES/LESdeltas/LESdelta/LESdelta.H
LESddyViscosity.C --> ../LES/LESddyViscosity/LESddyViscosity.C
LESddyViscosity.H --> ../LES/LESddyViscosity/LESddyViscosity.H
LESfilter.C --> ./LES/LESfilters/LESfilter/LESfilter.C
LESfilter.H --> ./LES/LESfilters/LESfilter/LESfilter.H
LESmodel.C --> ./LES/LESmodel/LESmodel.C
LESmodel.H --> ./LES/LESmodel/LESmodel.H
linearViscousStress.C --> ./linearViscousStress/linearViscousStress.C
linearViscousStress.H --> ./linearViscousStress/linearViscousStress.H
LRR.C --> ./RAS/LRR/LRR.C
LRR.H --> ./RAS/LRR/LRR.H
makeTurbulenceModel.H --> ./makeTurbulenceModel.H
maxDeltaxyz.C --> ../LES/LESdeltas/maxDeltaxyz/maxDeltaxyz.C
maxDeltaxyz.H --> ../LES/LESdeltas/maxDeltaxyz/maxDeltaxyz.H
Maxwell... --> ./laminar/Maxwell/Maxwell.C
Maxwell... --> ./laminar/Maxwell/Maxwell.H
nonlinearEddyViscosity.C --> ./nonlinearEddyViscosity/nonlinearEddyViscosity.C
nonlinearEddyViscosity.H --> ./nonlinearEddyViscosity/nonlinearEddyViscosity.H
nutkRoughWallFunctionFvPatchScalarField.C --> ./derivedFvPatchFields/wallFunctions/nutkRoughWallFunction/nutkRoughWallFunctionFvPatchScalarField.C
nutkRoughWallFunctionFvPatchScalarField.H --> ./derivedFvPatchFields/wallFunctions/nutkRoughWallFunction/nutkRoughWallFunctionFvPatchScalarField.H
nutkWallFunctionFvPatchScalarField.C --> ./derivedFvPatchFields/wallFunctions/nutkWallFunction/nutkWallFunctionFvPatchScalarField.C
nutkWallFunctionFvPatchScalarField.H --> ./derivedFvPatchFields/wallFunctions/nutkWallFunction/nutkWallFunctionFvPatchScalarField.H
nutLowReWallFunctionFvPatchScalarField.C --> ./derivedFvPatchFields/wallFunctions/nutLowReWallFunction/nutLowReWallFunctionFvPatchScalarField.C
nutLowReWallFunctionFvPatchScalarField.H --> ./derivedFvPatchFields/wallFunctions/nutLowReWallFunction/nutLowReWallFunctionFvPatchScalarField.H
nutURoughWallFunctionFvPatchScalarField.C --> ./derivedFvPatchFields/wallFunctions/nutURoughWallFunction/nutURoughWallFunctionFvPatchScalarField.C
nutURoughWallFunctionFvPatchScalarField.H --> ./derivedFvPatchFields/wallFunctions/nutURoughWallFunction/nutURoughWallFunctionFvPatchScalarField.H
nutUSpaldingWallFunctionFvPatchScalarField.C --> ./derivedFvPatchFields/wallFunctions/nutUSpaldingWallFunction/nutUSpaldingWallFunctionFvPatchScalarField.C
nutUSpaldingWallFunctionFvPatchScalarField.H --> ./derivedFvPatchFields/wallFunctions/nutUSpaldingWallFunction/nutUSpaldingWallFunctionFvPatchScalarField.H
nutUTabulatedWallFunctionFvPatchScalarField.C --> ./derivedFvPatchFields/wallFunctions/nutUTabulatedWallFunction/nutUTabulatedWallFunctionFvPatchScalarField.C
nutUTabulatedWallFunctionFvPatchScalarField.H --> ./derivedFvPatchFields/wallFunctions/nutUTabulatedWallFunction/nutUTabulatedWallFunctionFvPatchScalarField.H
nutUWallFunctionFvPatchScalarField.C --> ./derivedFvPatchFields/wallFunctions/nutUWallFunction/nutUWallFunctionFvPatchScalarField.C
nutUWallFunctionFvPatchScalarField.H --> ./derivedFvPatchFields/wallFunctions/nutUWallFunction/nutUWallFunctionFvPatchScalarField.H
nutWAllFunctionFvPatchScalarField.C --> ./derivedFvPatchFields/wallFunctions/nutWAllFunction/nutWAllFunctionFvPatchScalarField.C
nutWAllFunctionFvPatchScalarField.H --> ./derivedFvPatchFields/wallFunctions/nutWAllFunction/nutWAllFunctionFvPatchScalarField.H
omegaWallFunctionFvPatchScalarField.C --> ./derivedFvPatchFields/wallFunctions/omegaWallFunctions/omegaWallFunction/omegaWallFunctionFvPatchScalarField.C
omegaWallFunctionFvPatchScalarField.H --> ./derivedFvPatchFields/wallFunctions/omegaWallFunctions/omegaWallFunction/omegaWallFunctionFvPatchScalarField.H
porousBafflePressureFvPatchField.C --> ./derivedFvPatchFields/porousBafflePressure/porousBafflePressureFvPatchField.C
porousBafflePressureFvPatchFieldHd.C --> ./derivedFvPatchFields/porousBafflePressure/porousBafflePressureFvPatchFieldHd.C
porousBafflePressureFvPatchFieldHd.H --> ./derivedFvPatchFields/porousBafflePressure/porousBafflePressureFvPatchFieldHd.H
PrandtlDelta.C --> ../LES/LESdeltas/PrandtlDelta/PrandtlDelta.C
PrandtlDelta.H --> ../LES/LESdeltas/PrandtlDelta/PrandtlDelta.H
RASModel.C --> ./RAS/RASModel/RASModel.C
RASModel.H --> ./RAS/RASModel/RASModel.H
realizableKE.C --> ./RAS/realizableKE/realizableKE.C
realizableKE.H --> ./RAS/realizableKE/realizableKE.H
ReynoldsStress.C --> ./ReynoldsStress/ReynoldsStress.C
ReynoldsStress.H --> ./ReynoldsStress/ReynoldsStress.H
```

Compilación de proyectos complejos:

Build script generation [edit]

These *generator* tools do not build directly, but rather generate files to be used by a *native* build tool (as the ones listed in the previous two sections).

- [Bakefile](#)
- [BuildAMation](#), a multi-platform tool, using a declarative syntax in C# scripts, that builds C/C++ code in a terminal using multiple threads, or generates project files for Microsoft Visual Studio, Xcode or MakeFiles.
- [CMake](#) generates files for various build tools, such as [make](#), [ninja](#), Apple's [Xcode](#), and [Microsoft Visual Studio](#).^[2] CMake is also directly used by some [IDE](#) as [Qt Creator](#)^[3], [KDevelop](#) and [GNOME Builder](#)^[4].
- [GNU Build System](#) (aka autotools), a collection of tools for portable builds. These in particular include [Autoconf](#) and [Automake](#), cross-platform tools that together generate appropriate localized makefiles.
- [GYP](#) (Generate Your Projects) - Created for [Chromium](#); it is another tool that generates files for the native build environment
- [imake](#)
- [Meson](#), a build system optimized for performance and usability is based on [ninja](#) on Linux, [Visual Studio](#) on Windows and [Xcode](#) on macOS. Meson is also directly used by [GNOME Builder](#).^[4]
- [OpenMake Software Meister](#)
- [Premake](#), a Lua-based tool for making makefiles, Visual Studio files, Xcode projects, and more
- [qmake](#)
- [xmake](#)

CMake



Este artículo o sección necesita [referencias](#) que aparezcan en una publicación acreditada.

Este aviso fue puesto el 28 de julio de 2015.

CMake es una herramienta multiplataforma de generación o automatización de código. El nombre es una abreviatura para "cross platform make" (make multiplataforma); más allá del uso de "make" en el nombre, CMake es una suite separada y de más alto nivel que el sistema make común de Unix, siendo similar a las [autotools](#).

CMake es un [software libre y de código abierto](#) multiplataforma para gestionar la automatización de la construcción del software utilizando un método independiente del compilador. Soporta jerarquías de directorios y aplicaciones que dependen de múltiples bibliotecas. Se utiliza en conjunción con entornos de construcción nativos como [Make](#), [Qt Creator](#), [Ninja](#), [Xcode](#) de Apple, y [Microsoft Visual Studio](#). Tiene dependencias mínimas, requiriendo sólo un compilador C++ en su propio sistema de construcción.

Índice [ocultar]

- 1 Características
- 2 Proceso de construcción
- 3 Internos
- 4 Historia
- 5 Documentación y tutoriales
- 6 Principales funcionalidades
- 7 CTest, CPack, CDash



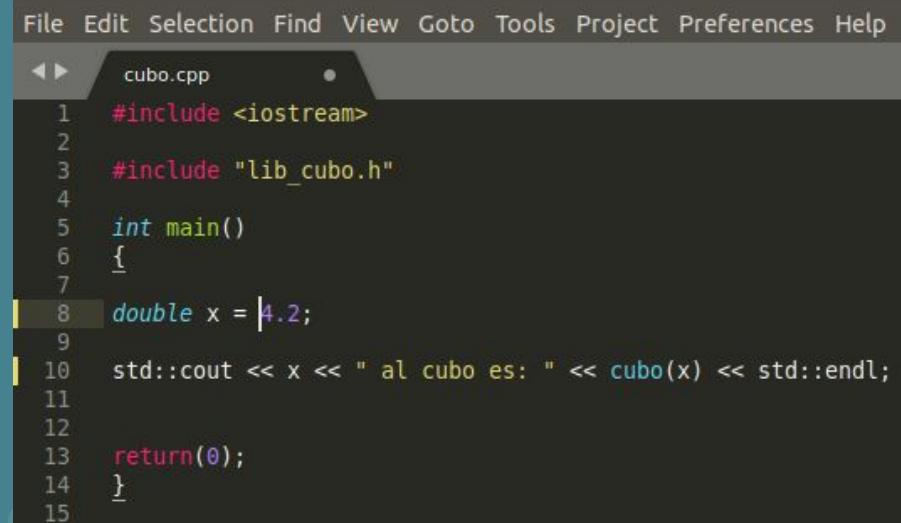
Tipo de

build automation

Ejemplo simple de uso de Cmake:

```
fede@fedeLaptopLenovo:~/tmp/cubo_cmake$ tree
.
├── build
├── CMakeLists.txt
├── cubo.cpp
└── my_lib
    ├── CMakeLists.txt
    ├── lib_cubo.cpp
    └── lib_cubo.h

2 directories, 5 files
fede@fedeLaptopLenovo:~/tmp/cubo_cmake$
```



The screenshot shows a code editor window with the title "cubo.cpp". The file contains the following code:

```
File Edit Selection Find View Goto Tools Project Preferences Help
cubo.cpp
1 #include <iostream>
2
3 #include "lib_cubo.h"
4
5 int main()
6 {
7
8     double x = 4.2;
9
10    std::cout << x << " al cubo es: " << cubo(x) << std::endl;
11
12
13    return(0);
14
15 }
```

Compilación de proyectos complejos:

A screenshot of a code editor showing two files: `cubo.cpp` and `lib_cubo.cpp`. The `cubo.cpp` file contains the following code:

```
1 | double cubo(double x)
2 | {
3 |     return(x*x*x);
4 |
5 | }
```

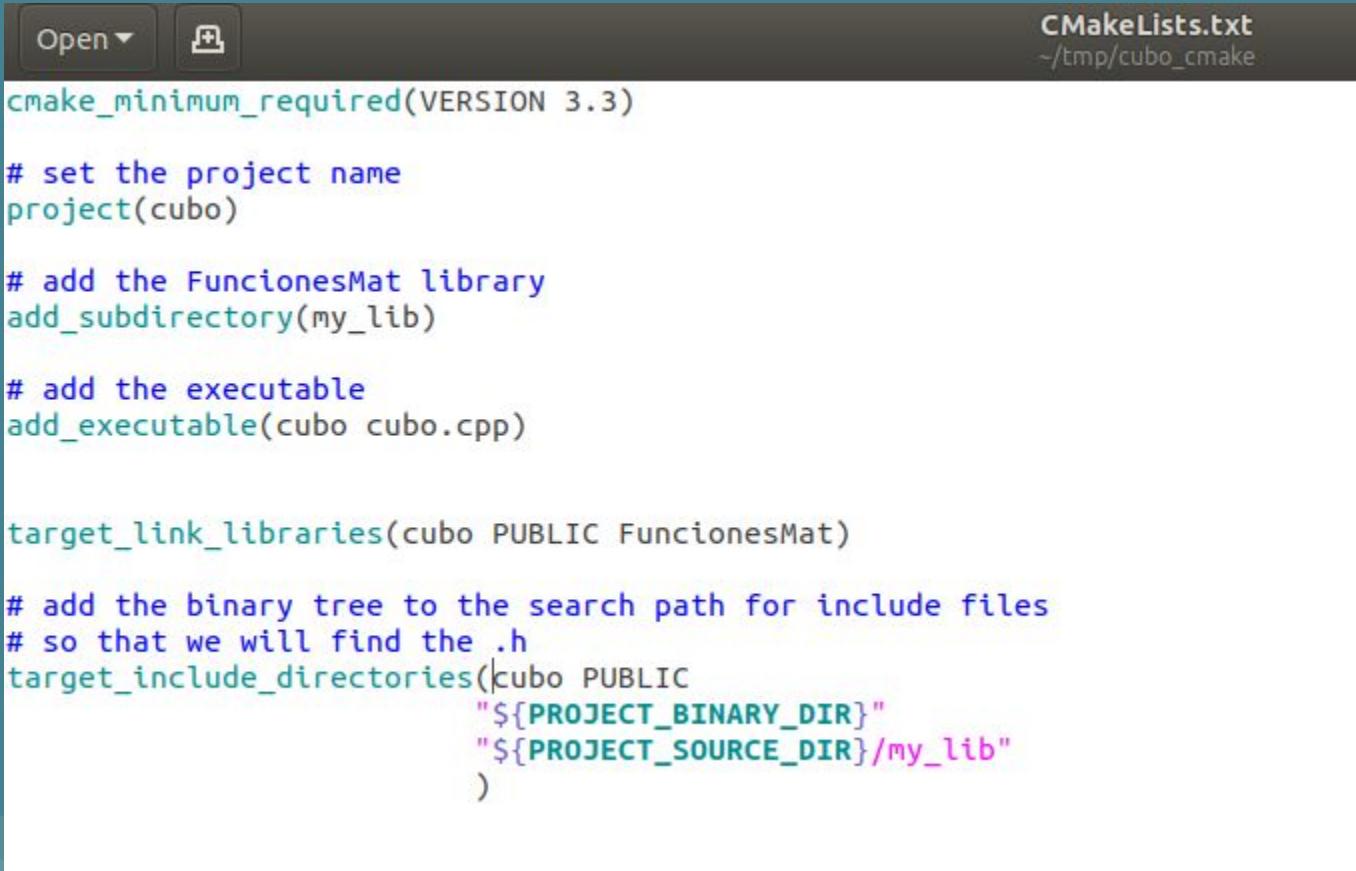


A screenshot of a code editor showing the header file `lib_cubo.h`. It contains the following code:

```
1 | double cubo(double x);
```

Archivo CMakeLists.txt
auxiliar, dentro de my_lib

Archivo CMakeLists.txt para este proyecto:



The image shows a screenshot of a code editor window with a dark theme. The title bar reads "CMakeLists.txt" and " ~/tmp/cubo_cmake". The editor contains the following CMake configuration script:

```
cmake_minimum_required(VERSION 3.3)

# set the project name
project(cubo)

# add the FuncionesMat library
add_subdirectory(my_lib)

# add the executable
add_executable(cubo cubo.cpp)

target_link_libraries(cubo PUBLIC FuncionesMat)

# add the binary tree to the search path for include files
# so that we will find the .h
target_include_directories(cubo PUBLIC
    "${PROJECT_BINARY_DIR}"
    "${PROJECT_SOURCE_DIR}/my_lib"
)
```

Generando el proyecto: dentro de build, tipear **ccmake** . .

File Edit View Search Terminal Help

Page 1 of 3

CMAKE_AR	/usr/bin/ar
CMAKE_BUILD_TYPE	ON
CMAKE_COLOR_MAKEFILE	/usr/bin/c++
CMAKE_CXX_COMPILER	-g
CMAKE_CXX_FLAGS	-Os -DNDEBUG
CMAKE_CXX_FLAGS_DEBUG	-O3 -DNDEBUG
CMAKE_CXX_FLAGS_MINSIZEREL	-O2 -g -DNDEBUG
CMAKE_CXX_FLAGS_RELEASE	/usr/bin/cc
CMAKE_CXX_FLAGS_RELWITHDEBINFO	-g
CMAKE_C_COMPILER	-Os -DNDEBUG
CMAKE_C_FLAGS	-O3 -DNDEBUG
CMAKE_C_FLAGS_DEBUG	-O2 -g -DNDEBUG
CMAKE_C_FLAGS_MINSIZEREL	CMAKE_C_FLAGS_RELEASE
CMAKE_C_FLAGS_RELWITHDEBINFO	CMAKE_EXE_LINKER_FLAGS
CMAKE_EXE_LINKER_FLAGS_DEBUG	CMAKE_AR: Path to a program.

Press [enter] to edit option

Press [c] to configure

Press [g] to generate and exit

Press [h] for help

Press [q] to quit without generating

Press [t] to toggle advanced mode (Currently On)

CMake Version 3.3.0

Compilación de proyectos complejos:

Una vez que generamos el Makefile (tipeando ‘c’ ~> ‘c’ ~> ‘g’ en la gui de cmake), compilamos todo mediante make

```
fede@fedeLaptopLenovo:~/tmp/cubo_cmake/build$ make
Scanning dependencies of target FuncionesMat
[ 25%] Building CXX object my_lib/CMakeFiles/FuncionesMat.dir/lib_cubo.cpp.o
[ 50%] Linking CXX static library libFuncionesMat.a
[ 50%] Built target FuncionesMat
Scanning dependencies of target cubo
[ 75%] Building CXX object CMakeFiles/cubo.dir/cubo.cpp.o
[100%] Linking CXX executable cubo
[100%] Built target cubo
fede@fedeLaptopLenovo:~/tmp/cubo_cmake/build$ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  cubo  Makefile  my_lib
fede@fedeLaptopLenovo:~/tmp/cubo_cmake/build$ ./cubo
4.2 al cubo es: 74.088
fede@fedeLaptopLenovo:~/tmp/cubo_cmake/build$
```

La biblioteca gsl:

GNU Scientific Library

GNU Scientific Library (GSL) es una [biblioteca](#) escrita en [C](#), destinada a cálculos numéricos en [matemáticas y ciencia](#), distribuida bajo la [licencia GNU GPL](#).

Incorpora, entre otras, rutinas para el manejo de [números complejos](#), [funciones elementales](#) y [funciones especiales](#), combinatoria, álgebra lineal, [integración](#) y [derivación numéricas](#), transformada rápida de Fourier, transformada wavelet discreta, generación de números aleatorios y estadística.

- Funciones matemáticas básicas
- Números complejos
- Polinomios
- Funciones especiales
- Vectores y matrices
- Permutaciones
- Combinaciones
- Multiset
- Ordenación

- Álgebra lineal
- Eigenvectores
- Transformada rápida de Fourier
- Integración numérica
- Generación aleatoria de números
- Secuencias Quasi-aleatorias
- Distribuciones de números aleatorios
- Estadísticas
- Histogramas
- N-tuplas
- Integración de Monte Carlo

- Ecuaciones diferenciales ordinarias
- Interpolación
- Derivación numérica
- Aproximaciones de Chebyshev
- **Aceleración de Series**
- Transformada discreta de **Hankel**
- Búsqueda de raíces en una y varias dimensiones
- Minimización en una y varias dimensiones
- Medida de mínimos cuadrados
- **Medida de mínimos cuadrados no lineales**
- Constantes Físicas

GNU Scientific Library

gnu.org/software/gsl/



Tipo de programa	biblioteca de software
	Paquete GNU
Desarrollador	Proyecto GNU
Lanzamiento	1996
Última versión estable	2.0
	31 de octubre de 2015
Género	Biblioteca numérica
Programado en	C
Sistema operativo	Multiplataforma
Licencia	GNU GPL
En español	X No

Porque usar bibliotecas numéricas optimizadas?

- Para no “reinventar” la rueda.
- Para no perder tiempo en implementar características que ya están programadas y dedicar nuestras energías en aquellas que no lo están.
- Para promover la reutilización del código.
- Porque, en general, tales bibliotecas en general van a tener una performance y confiabilidad muy superiores a lo que podamos obtener nosotros, fruto de años de evolución y mucho trabajo puesto en la optimización de algoritmos numéricos.
- etc...

Aún en los cálculos más sencillos hay lugar para la optimización ...

We assume that you know enough *never* to evaluate a polynomial this way:

```
p=c[0]+c[1]*x+c[2]*x*x+c[3]*x*x*x+c[4]*x*x*x*x;
```

or (even worse!),

```
p=c[0]+c[1]*x+c[2]*pow(x,2.0)+c[3]*pow(x,3.0)+c[4]*pow(x,4.0);
```

Come the (computer) revolution, all persons found guilty of such criminal behavior will be summarily executed, and their programs won't be! It is a matter of taste, however, whether to write

```
p=c[0]+x*(c[1]+x*(c[2]+x*(c[3]+x*c[4])));
```

or

```
p=((c[4]*x+c[3])*x+c[2])*x+c[1])*x+c[0];
```

If the number of coefficients $c[0..n]$ is large, one writes

```
p=c[n];
for(j=n-1;j>=0;j--) p=p*x+c[j];
```

En pág. 173 del libro
“Numerical Recipes
in C, The Art of
Scientific
Computing”, W. H.
Press, S. A.
Teukolsky, W. T.
Vetterling and B. P.
Flannery.

Manual online de gsl:

The screenshot shows a web browser displaying the official documentation for the GNU Scientific Library (GSL). The header features a blue navigation bar with the GSL logo and version 2.6. Below the header is a search bar labeled "Search docs". The main content area has a white background with a large title "GNU Scientific Library" in bold black font. To the left is a sidebar with a dark grey background containing a list of topics: Introduction, Using the Library, Error Handling, Mathematical Functions, Complex Numbers, Polynomials, Special Functions, Vectors and Matrices, Permutations, Combinations, Multisets, Sorting, and BLAS Support. The main content area also contains a bulleted list of topics under "GNU Scientific Library", including "Introduction", "Using the Library", and several sub-topics for each.

Docs » GNU Scientific Library

GNU Scientific Library

- [Introduction](#)
 - [Routines available in GSL](#)
 - [GSL is Free Software](#)
 - [Obtaining GSL](#)
 - [No Warranty](#)
 - [Reporting Bugs](#)
 - [Further Information](#)
 - [Conventions used in this manual](#)
- [Using the Library](#)
 - [An Example Program](#)
 - [Compiling and Linking](#)
 - [Shared Libraries](#)
 - [ANSI C Compliance](#)

Como comenzar? : Buscar ejemplos en el manual online de gsl

[One Dimensional Root-Finding](#)

[One Dimensional Minimization](#)

[Multidimensional Root-Finding](#)

[Multidimensional Minimization](#)

[Linear Least-Squares Fitting](#)

[Nonlinear Least-Squares Fitting](#)

[Basis Splines](#)

[Sparse Matrices](#)

[Sparse BLAS Support](#)

[Sparse Linear Algebra](#)

[Physical Constants](#)

[IEEE floating-point arithmetic](#)

[Debugging Numerical Programs](#)

[Contributors to GSL](#)

[Autoconf Macros](#)

[GSL CBLAS Library](#)

[GNU General Public License](#)

[GNU Free Documentation License](#)

Example programs for vectors

This program shows how to allocate, initialize and read from a vector using the functions

`gsl_vector_alloc()`, `gsl_vector_set()` and `gsl_vector_get()`.

```
#include <stdio.h>
#include <gsl/gsl_vector.h>

int
main (void)
{
    int i;
    gsl_vector * v = gsl_vector_alloc (3);

    for (i = 0; i < 3; i++)
    {
        gsl_vector_set (v, i, 1.23 + i);
    }

    for (i = 0; i < 100; i++) /* OUT OF RANGE ERROR */
    {
        printf ("v_%d = %g\n", i, gsl_vector_get (v, i));
    }

    gsl_vector_free (v);
    return 0;
}
```

Ejemplo de uso de la biblioteca desde C++:

```
Open ▾  EjFuncionesEspecialesConGSL.cpp  
~/tmp
```

```
#include <iostream>
#include <math.h>
#include <gsl/gsl_sf_bessel.h>
#include <gsl/gsl_sf_gamma.h>

/*Para compilar programas que usen la libreria gsl, poner, ademas de los headers
necesarios, los flags siguientes para el linkeador:
-lgsl -lgslcblas

cblas provee "C Basic Linear Algebra Subroutines"
*/

int main (void)
{
    std::cout << "J_3 (5.5) = " << gsl_sf_bessel_Jn(3, 5.5) << std::endl;
    std::cout << "Gamma (6) = 5! = " << gsl_sf_gamma(6) << std::endl;
    return 0;
}
```

```
fede@fedeLaptopLenovo:~/tmp$ g++ EjFuncionesEspecialesConGSL.cpp -lm -lgsl -lgslcblas
fede@fedeLaptopLenovo:~/tmp$ ./a.out
J_3 (5.5) = 0.256118
Gamma (6) = 5! = 120
```

Dónde encontrar más información sobre Funciones Especiales?

Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables

Edited by
Milton Abramowitz and Irene A. Stegun

National Bureau of Standards
Applied Mathematics Series • 55

Issued June 1964
Tenth Printing, December 1972, with corrections

Dónde encontrar más información sobre Funciones Especiales?

4. Elementary Transcendental Functions	65
Logarithmic, Exponential, Circular and Hyperbolic Functions	
RUTH ZUCKER	
5. Exponential Integral and Related Functions	227
WALTER GAUTSCHI and WILLIAM F. CAHILL	
6. Gamma Function and Related Functions.	253
PHILIP J. DAVIS	
7. Error Function and Fresnel Integrals	295
WALTER GAUTSCHI	
8. Legendre Functions	331
IRENE A. STEGUN	
9. Bessel Functions of Integer Order	355
F. W. J. OLVER	
10. Bessel Functions of Fractional Order.	435
H. A. ANTOSIEWICZ	
11. Integrals of Bessel Functions	479
YUDELL L. LUKE	
12. Struve Functions and Related Functions	495
MILTON ABRAMOWITZ	
13. Confluent Hypergeometric Functions	503
LUCY JOAN SLATER	
14. Coulomb Wave Functions	537
MILTON ABRAMOWITZ	
15. Hypergeometric Functions	555
FRITZ OBERHETTINGER	
16. Jacobian Elliptic Functions and Theta Functions	567
L. M. MILNE-THOMSON	
17. Elliptic Integrals	587

Dónde encontrar más información sobre Funciones Especiales?

9. Bessel Functions of Integer Order

Mathematical Properties

Notation

The tables in this chapter are for Bessel functions of integer order; the text treats general orders. The conventions used are:

$$z=x+iy; \quad x, y \text{ real.}$$

n is a positive integer or zero.

ν, μ are unrestricted except where otherwise indicated; ν is supposed real in the sections devoted to Kelvin functions 9.9, 9.10, and 9.11.

The notation used for the Bessel functions is

Bessel Functions J and Y

9.1. Definitions and Elementary Properties

Differential Equation

$$9.1.1 \quad z^2 \frac{d^2w}{dz^2} + z \frac{dw}{dz} + (z^2 - \nu^2)w = 0$$

Solutions are the Bessel functions of the first kind $J_{\pm\nu}(z)$, of the second kind $Y_{\nu}(z)$ (also called Weber's function) and of the third kind $H_{\nu}^{(1)}(z), H_{\nu}^{(2)}(z)$

Ascending Series

$$9.1.10 \quad J_{\nu}(z) = (\frac{1}{2}z)^{\nu} \sum_{k=0}^{\infty} \frac{(-\frac{1}{4}z^2)^k}{k! \Gamma(\nu+k+1)}$$

9.1.11

$$Y_{\nu}(z) = -\frac{(\frac{1}{2}z)^{-\nu}}{\pi} \sum_{k=0}^{n-1} \frac{(n-k-1)!}{k!} (\frac{1}{4}z^2)^k + \frac{2}{\pi} \ln(\frac{1}{2}z) J_n(z) - \frac{(\frac{1}{2}z)^n}{\pi} \sum_{k=0}^{\infty} \{\psi(k+1) + \psi(n+k+1)\} \frac{(-\frac{1}{4}z^2)^k}{k!(n+k)!}$$

where $\psi(n)$ is given by 6.3.2.

9.1.20

$$\begin{aligned} J_{\nu}(z) &= \frac{(\frac{1}{2}z)^{\nu}}{\pi^{\frac{1}{2}} \Gamma(\nu + \frac{1}{2})} \int_0^{\pi} \cos(z \cos \theta) \sin^{2\nu} \theta d\theta \\ &= \frac{2(\frac{1}{2}z)^{\nu}}{\pi^{\frac{1}{2}} \Gamma(\nu + \frac{1}{2})} \int_0^1 (1-t^2)^{\nu-\frac{1}{2}} \cos(zt) dt \quad (\Re \nu > -\frac{1}{2}) \end{aligned}$$

9.1.21

$$\begin{aligned} J_n(z) &= \frac{1}{\pi} \int_0^{\pi} \cos(z \sin \theta - n\theta) d\theta \\ &= \frac{i^{-n}}{\pi} \int_0^{\pi} e^{iz \cos \theta} \cos(n\theta) d\theta \end{aligned}$$

9.1.22

$$\begin{aligned} J_{\nu}(z) &= \frac{1}{\pi} \int_0^{\pi} \cos(z \sin \theta - \nu\theta) d\theta \\ &\quad - \frac{\sin(\nu\pi)}{\pi} \int_0^{\infty} e^{-z \sinh t - \nu t} dt \quad (\arg z < \frac{1}{2}\pi) \end{aligned}$$

Referencias:

- <http://www.cplusplus.com/reference/>
- Numerical Recipes in C, The Art of Scientific Computing, W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Sec. Ed. Cambridge University Press.
- Como programar en C/C++ H. M. Deitel and P. J. Deitel 2 ed. Pearson Prentice Hall.
- A complete guide to programming in C++, Ulla Kirch-Prinz and Peter Prinz, Jones and Bartlett Publishers Inc.
- Métodos Matemáticos de Estadística, H. Cramer, Aguilar, Madrid.
- Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables M. Abramowitz, and I. Stegun. Dover, New York, ninth Dover printing, tenth GPO printing edition, (1964).
- <https://www.gnu.org/software/gsl/>