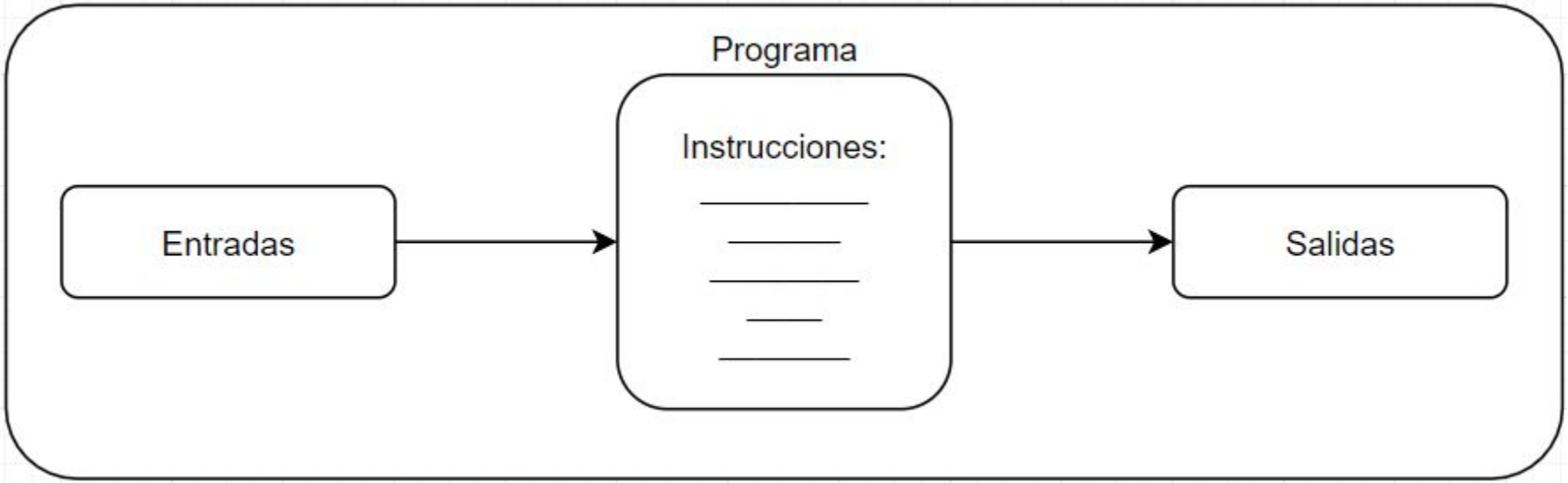
Introducción al cómputo y al desarrollo de software

Clase 2: File I/O, Librerías útiles, arrays, funciones, error handling, recursión

Lic. Agustín Bernardo & MSc. Rodrigo Bonazzola

¿Qué es un programa?



Entradas y salidas - por consola

La clase pasada vimos cómo interactuar con la consola:

cin >> a;

cout << “a imprimir”<< endl;

entradaSalida.cpp

Clase 2 > entradaSalida.cpp

```cpp
#include <iostream>

using namespace std;

int main() {
    int a;

    cout << "Introduzca un número entero." <<endl;
    cin >> a;

    cout << "El número introducido es: " << a <<endl;
    return 0;
}
```

Entradas y salidas - por archivos!

Esta clase, vemos como trabajar con archivos.

#include <fstream>

Hay que abrir el archivo y escribir o leer de él.

myfile >> “writing”

myfile << “reading”

http://www.cplusplus.com/doc/tutorial/files/

fileIO.cpp

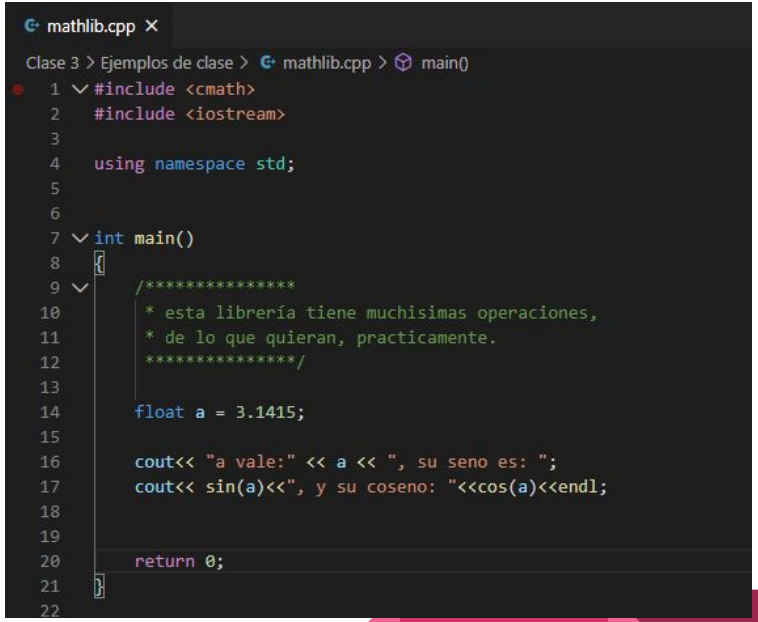
Clase 3 > Ejemplos de clase > fileIO.cpp > main()

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main() {

    ofstream miArchivoAEscribir;
    miArchivoAEscribir.open("NuevoArchivo.txt");
    miArchivoAEscribir<< "Escribiendo a un  archivo!"<<endl;
    miArchivoAEscribir.close();

    ifstream miArchivoALeer;
    miArchivoALeer.open("NuevoArchivo.txt");
    string textoArchivo;
    miArchivoALeer >> textoArchivo;
    cout << textoArchivo<<endl;
    miArchivoALeer.close();

    return 0;
}
```

Librería de matemáticas

Viene con muchísimas funciones que permiten hacer cálculos.

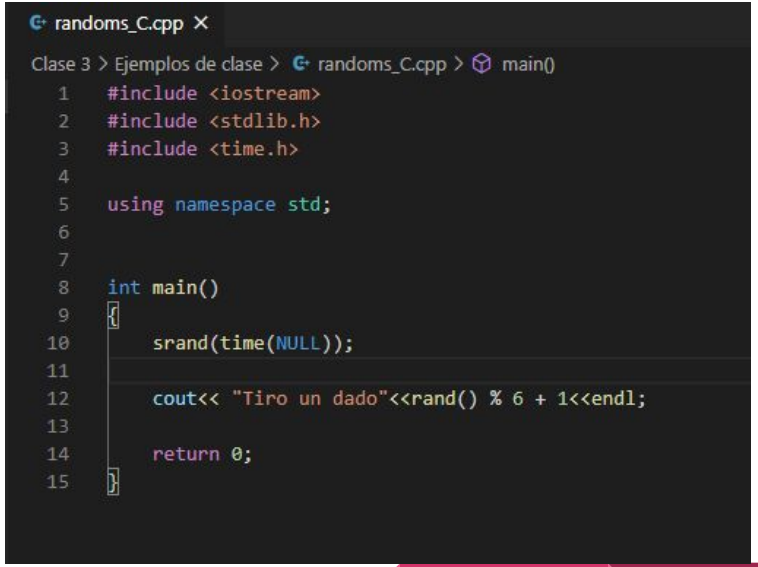
Trigonometría, exponenciales, etcétera.



Números aleatorios

Sirven para tomar decisiones de forma aleatoria.

Utilizaremos la librería <stdlib.h>.



Números aleatorios

Sirven para tomar decisiones de forma aleatoria.

Utilizaremos la librería <random>.

http://www.cplusplus.com/reference/random/

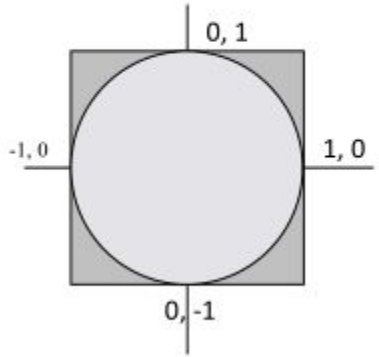
randoms.cpp

Clase 3 > Ejemplos de clase > randoms.cpp > main()

```cpp
#include <iostream>
#include <random>

using namespace std;


int main()
{
    random_device dev;
    mt19937 rng(dev());
    uniform_int_distribution<mt19937::result_type> dist6(1,6);

    cout<<"Numeros aleatorios del 1 al 6"<<endl;

    cout<<dist6(rng)<<endl;

    return 0;
}
```

Números aleatorios

1. Si tiramos puntos al azar en un plano, sobre la región (x=-1, y=-1)(x=1,y=1) y contamos la cantidad de puntos que quedan a una distancia < 1 del origen de coordenadas, estos cumplirán aproximadamente con la relación:



$$N1/N = \pi/4$$

En donde

N1: Número de puntos con distancias <= 1
N: Número total de puntos
π: 3.14159...
4: superficie de la región

Con estos datos, hacer un programa que aproxime el valor de Pi realizando un número de iteraciones definida por el usuario.

Arrays

Un array es un conjunto de variables del mismo tipo, una a continuación de la otra en la memoria.

Podemos acceder al i-ésimo elemento con el operador [].

Aunque hay varios tipos, sólo trabajaremos en esta clase con el array nativo.

arrays.cpp

Clase 3 > Ejemplos de clase > arrays.cpp > main()

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main() {

    int tablaDel2[10] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
    char palabrota[] = "Mequetrefe";

    for (int i = 0; i < 10; i++)
    {
        cout<<tablaDel2[i]<<endl;
        cout<<palabrota[i]<<endl;
        cout<<(char)(tablaDel2[i] + palabrota[i])<<endl;
    }

    return 0;
}
```

Arrays

¿Y entonces, qué hay en la variable tablaDel2?

Una dirección de memoria.

El operador [] es similar al operador contenido.

arrays.cpp

Clase 3 > Ejemplos de clase > arrays.cpp > main()

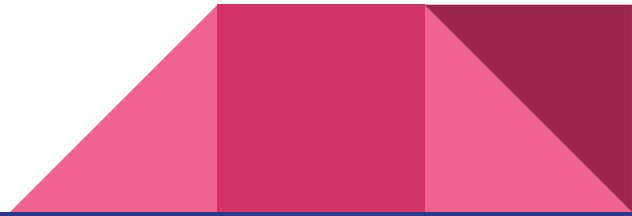
```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main() {

    int tablaDel2[10] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
    char palabrota[] = "Mequetrefe";

    for (int i = 0; i < 10; i++)
    {
        cout<<tablaDel2[i]<<endl;
        cout<<palabrota[i]<<endl;
        cout<<(char)(tablaDel2[i] + palabrota[i])<<endl;
    }

    return 0;
}
```

Arrays

Ejercicio: Cuente la cantidad de letras de una cadena de caracteres.

Ejercicio: Calcule la suma de elementos dentro de un vector.

Ejercicio: Implemente el producto escalar entre dos vectores.



Funciones

Las funciones son piezas de código reutilizable que permiten simplificar el esquema del programa.

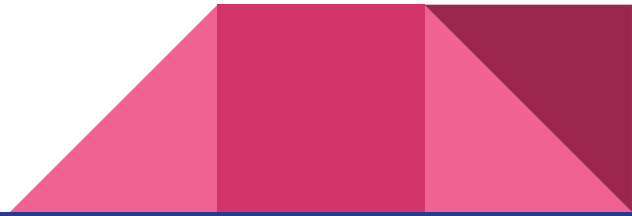
Tienen una **declaración** y una **implementación.**

functions.cpp

Clase 3 > Ejemplos de clase > functions.cpp > ...

```cpp
#include <iostream>

using namespace std;

int esPrimo(int N);

int main()
{

    int M;
    cout << "Ingrese el número hasta el cual desea encontrar los primos." << endl;

    cin >> M;

    for (int i = 0; i < M; i++)
    {
        if (esPrimo(i))
        {
            cout << i << " es primo." << endl;
        }
    }
    return 0;
}

int esPrimo(int N)
{
    for (int i = 0; i < N; i++)
    {
        if (N % i == 0)
        {
            return 0;
        }
    }
    return 1;
}
```

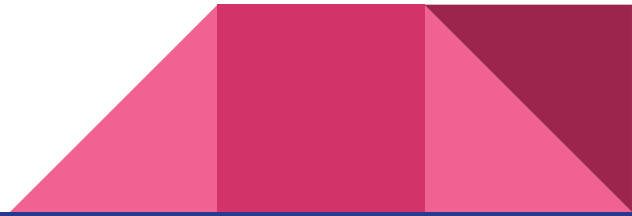
DIVIDE
Y
CONQUISTA



Funciones

Ejercicio: Implemente una función que tome dos componentes de un vector y devuelva su módulo y el ángulo.

Ejercicio: Implemente una función que permita determinar la suma de elementos dentro de un vector.



Error handling

Se utiliza como adición al control de flujo del programa.

Permite evitar situaciones indeseadas.

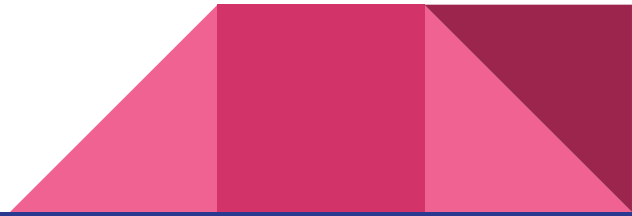
exceptions.cpp

Clase 3 > Ejemplos de clase > exceptions.cpp > ...

```cpp
#include <iostream>
#include <cmath>
#include <stdexcept>

using namespace std;

float mySqrt(float x)
{
    if (x < 0)
    {
        throw invalid_argument("No podes calcular la raiz de un negativo");
    }
    return sqrt(x);
}

int main()
{
    float x;
    cout << "Ingrese el numero cuya raiz quiere calcular" << endl;
    cin >> x;
    try
    {
        float raizX = mySqrt(x)
    }
    catch (exception &e)
    {
        cout << "Error! " << e.what() << endl;
    }
    return 0;
}

```

Error handling

Ejercicio: Implemente una función que intente calcular una división, a partir de dos argumentos, y tire una excepción si el divisor es nulo.

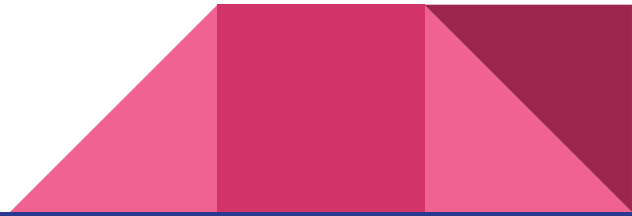


Recursión

Algoritmos recursivos permiten solucionar problemas muy complejos con códigos muy sencillos.

Parecen medio mágicos pero en realidad funcionan bien.

Involucra una solución trivial y una llamada a si mismo.



Recursión

Recursión

fibonacci.cpp

Clase 3 > Ejemplos de clase > fibonacci.cpp > ...

```cpp
#include <iostream>

using namespace std;

int fibonacci(int n)
{
    if (n == 0)
    {
        return 0;
    }
    else if (n == 1)
    {
        return 1;
    }
    else
    {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}

int main()
{
    cout<<"El termino 15 de la serie de fibonacci es: "<<fibonacci(15)<<endl;
}
```

Recursión

Hanoi

El objeto es mover los n discos desde el poste 'from' al poste 'to' utilizando el poste auxiliar 'aux'.

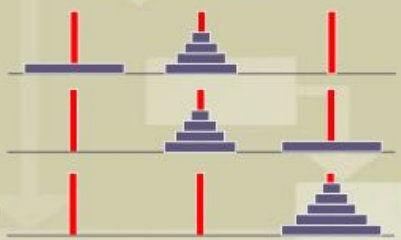
Los discos deben moverse de a uno por vez.

Nunca se debe pasar por la condición de que un disco este arriba de otro de menor diámetro.

Recursión

Hanoi

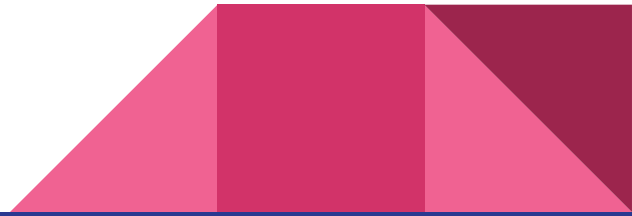

```
void hanoi(int n, int from, int to, int aux) {
    if( n == 1 ) {
        cout << "Mover de " << from << " a " << to << endl;
        return;
    }
    hanoi( n-1, from, aux, to);
    hanoi(   1, from, to, aux);
    hanoi( n-1, aux, to, from);
}

int main() {
    hanoi(3, 1, 3, 2);
    return 0;
}
```



Recursión

Ejercicio: Calcule el factorial de un número de forma recursiva.

Ejercicio: Resuelva el juego de Hanoi.



Más ejercicios en la guía.

¿Preguntas?