ESTRUCTURAS DE DATOS

NOTAS SOBRE C++

Atributos y Métodos

Manuel Montenegro Montes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid

Definición de una clase

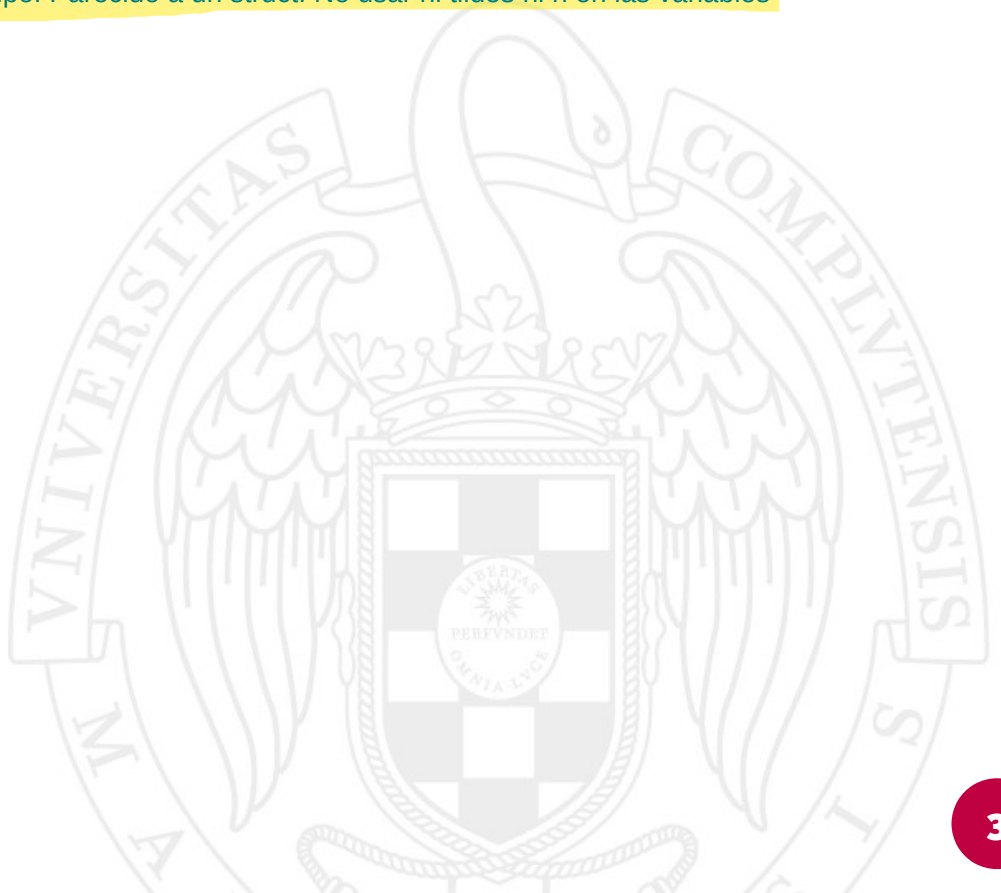
Mezcla de TP y FP vemos clases en c++. C++ tiene muchas cosas que Java no tiene. En esta diapo vemos cosas que ya hemos visto

Definición de una clase: atributos

```
class Fecha {
  int dia;
  int mes;
  int anyo;
};
```

clase con 3 atributos, cada uno con su tipo. Parecido a un struct. No usar ni tildes ni ñ en las variables

Nos falta poner si son privados o publicos.



Definición de una clase: métodos

```
class Fecha {
  int dia;
  int mes;
  int anyo;
  int get_dia();
  void set_dia(int dia);
  int get_mes();
  void set_mes(int mes);
  int get_anyo();
  void set_anyo(int anyo);
};
```

TODOS ESTOS SERÍAN PRIVADOS

- Aquí se están **declarando** los métodos, pero no aparecen sus **implementaciones**.
- Por defecto, todos los atributos y métodos son privados.

Métodos de acceso y modificación de los atributos de una clase.

Si no se pone nada todos son privados.

Modificadores de acceso

```
class Fecha {
  int dia;
  int mes;
  int anyo;

public:
  int get_dia();
  void set_dia(int dia);
  int get_mes();
  void set_mes(int mes);
  int get_anyo();
  void set_anyo(int anyo);
};
```

TODOS SON PRIVATE, NO LO PONEMOS

A los getters creo que faltaría ponerles const.

- Hay tres tipos de modificadores:
  - `public:` VISIBLE DESDE FUERA
  - `private:` VISIBLE DESDE DENTRO DE LA CLASE
  - `protected:` VISIBLE DESDE CLASES Y SUBCLASES
- Afectan a los métodos y atributos situados a continuación del modificador.

Modificadores de acceso

```
class Fecha {
public:
  int get_dia();
  void set_dia(int dia);
  int get_mes();
  void set_mes(int mes);
  int get_anyo();
  void set_anyo(int anyo);

private:
  int dia;
  int mes;
  int anyo;
};
```

Lo público siempre primero

- Hay tres tipos de modificadores:
  - `public:`
  - `private:`
  - `protected:`
- Afectan a los métodos y atributos situados a continuación del modificador.

POR CONVENIO, MEJOR PONER PRIMERO LO PUBLICO Y LUEGO LO PRIVATE.

Implementación de métodos

```
class Fecha {
public:
  int get_dia() {
    return dia;
  }

  void set_dia(int dia) {
    this→dia = dia;
  }

  // Igualmente para mes y anyo
  //  ...

private:
  // ...
};
```

THIS ES UN PUNTERO, ES LO MISMO QUE EN JAVA.

- Posibilidad 1: Implementación dentro de la definición de clase.
- *"Estilo Java"*

Implementación de métodos

```
class Fecha {
public:
  int get_dia();
  void set_dia(int dia);
  //
  //  ...
private:
  //  ...
};

int Fecha::get_dia() {
  return dia;
}

void Fecha::set_dia(int dia) {
  this→dia = dia;
}
```

DEFINICIÓN DENTRO DE LA CLASE.

Ponerlo solamente cuando es fuera de la clase.

IMPLEMENTACIÓN DE LOS MÉTODOS FUERA DE LA CLASE.

- Posibilidad 2: Implementación fuera de la definición de clase.

¡No son equivalentes!

SI SON CORTOS SON MÁS EFICIENTES QUE LOS NO INLINE

- Implementaciones dentro de la clase: se consideran métodos **inline**.

  `https://www.geeksforgeeks.org/inline-functions-cpp/`

  - Son más eficientes, pero incrementan el tamaño del código.

SI SON CORTOS

- **Consejo:**
  - Métodos cortos (p.ej. acceso, modificación) pueden definirse dentro de la clase.

GETTERS Y SETTERS

  - Métodos largos deben definirse fuera de la clase.

Uso de una clase: instancias

COMO HACER UN NEW FECHA EN EL CASO DE JAVA.

Creación de instancias

Se crean igual que los structs

```
int main() {
    Fecha f;
    f.set_dia(28);
    f.set_mes(8);
    f.set_anyo(2019);

    std::cout << "Día: " << f.get_dia() << std::endl;
    std::cout << "Mes: " << f.get_mes() << std::endl;
    std::cout << "Año: " << f.get_anyo() << std::endl;
}
```

es análogo a hacer un new Fecha

llamada a métodos es similar que en java-

```
Día: 28
Mes: 8
Año: 2019
```

Salida con formato

Un nuevo método: imprimir

```
class Fecha {
public:
  int get_dia();
  void set_dia(int dia);
  int get_mes();
  void set_mes(int mes);
  int get_anyo();
  void set_anyo(int anyo);

  void imprimir();

private:
  int dia;
  int mes;
  int anyo;
};
```

Método público

ACORDARNOS!!! PRIMERO ATRIBUTOS Y COSAS PÚBLICAS, Y, POR ÚLTIMO ATRIBUTOS PRIVADOS. AUQNEU BUENO SIEMPRE MEJOR ATRIBUTOS SEAN PRIVADOS.

Un nuevo método: imprimir

```
void Fecha::imprimir() {
    std::cout << dia << "/" << mes << "/" << anyo;
}
```

Imprime las 3 componentes de la fecha separadas por una barra.

IMAGINA QUE QUEREMOS QUE LAS FECHAS SE ESCRIBAN CON DOS DÍGITOS PARA EL DÍA DOS PARA EL MES Y 4 PARA EL AÑO, ES DECIR QUEREMOS OTRO FORMATO.

Un nuevo método: imprimir

```
void Fecha::imprimir() {
    std::cout << std::setfill('0') << std::setw(2) << dia << "/"
              << std::setw(2) << mes << "/"
              << std::setw(4) << anyo;
}
```

MANIPULADOR DE ENTRADA SALIDA

Número de caracteres para imprimir el dato que viene a continuación.

SETFILL INDICA EL CARACTER QUE SE UTILIZA COMO RELLENO.
POR EJEMPLO UNO DE ENERO EN VEZ DE SER 1/1--> 01/01

```
#include <iostream>
#include <iomanip>
```

iomanip es para el uso del setw.

Uso del método imprimir

```
int main() {
    Fecha f;
    f.set_dia(28);
    f.set_mes(8);
    f.set_anyo(2019);


    std::cout << "Fecha: ";
    f.imprimir();
    std::cout << std::endl;
}
```

```
Fecha: 28/08/2019
```