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

Definición de una clase: atributos

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

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);
};
```

- Aquí se están **declarando** los métodos, pero no aparecen sus **implementaciones**.

- Por defecto, todos los atributos y métodos 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);
};
```

- Hay tres tipos de modificadores:
  - `public:`
  - `private:`
  - `protected:`

- 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;
};
```

- Hay tres tipos de modificadores:
  - `public:`
  - `private:`
  - `protected:`

- Afectan a los métodos y atributos situados a continuación del modificador.

Implementación de métodos

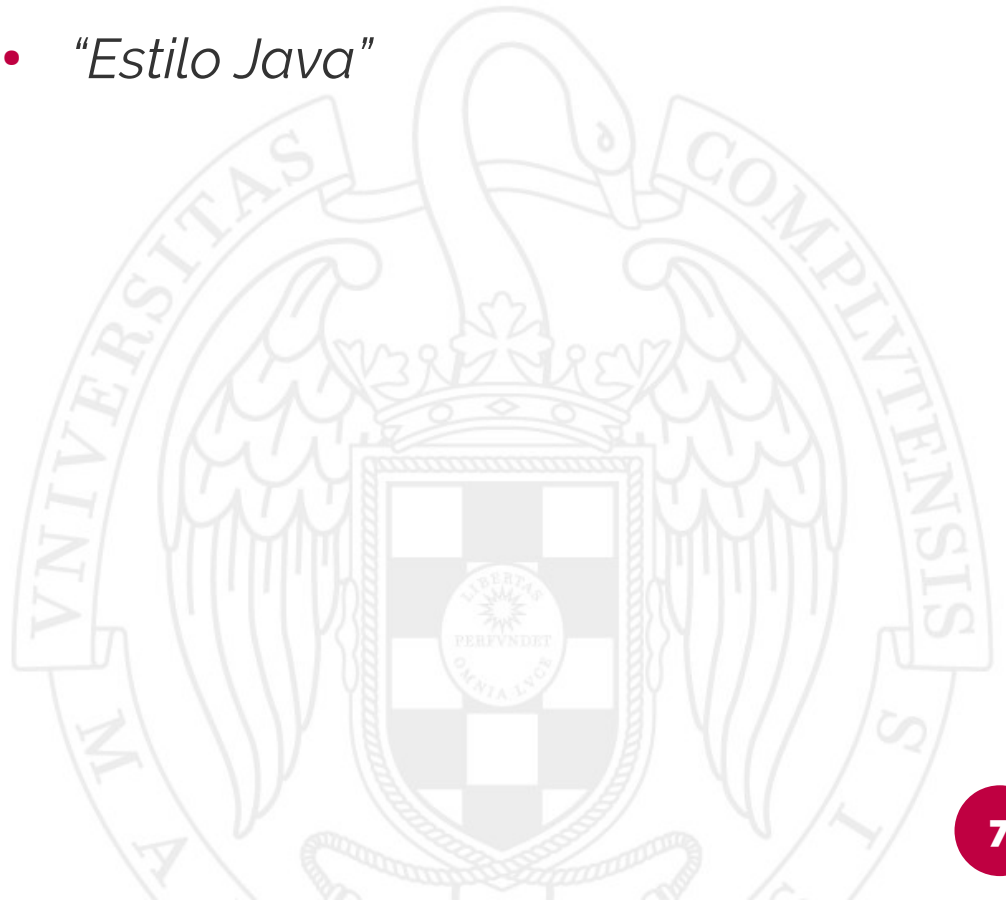
```
class Fecha {
public:
  int get_dia() {
    return dia;
  }

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

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

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

- 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;
}
```

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

¡No son equivalentes!

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

Uso de una clase: instancias

Creación de instancias

```
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;
}
```

```
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;
};
```

Un nuevo método: imprimir

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

Un nuevo método: imprimir

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

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

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
```