ESTRUCTURAS DE DATOS

NOTAS SOBRE C++

Los tipos `pair` y `tuple`

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

Ejemplo

- Calcular el elemento mínimo de un array.

```
int min(int *array, int longitud) {
  int min = std::numeric_limits<int>::max();

  for (int i = 0; i < longitud; i++) {
    min = std::min(min, array[i]);
  }

  return min;
}
```

¿Y si quiero devolver también el máximo?

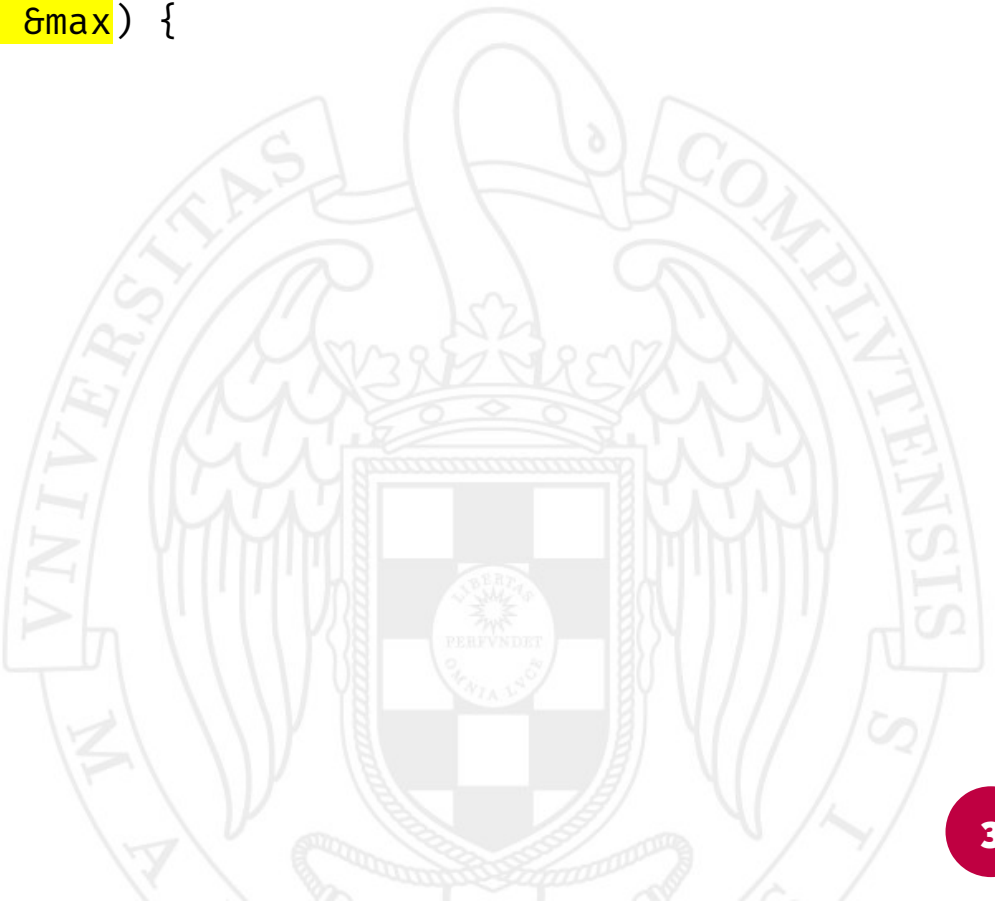
Ejemplo

- Calcular el elemento mínimo y máximo de un array.

```
int min_max(int *array, int longitud, int &max) {




}
```



Ejemplo

- Calcular el elemento mínimo y máximo de un array.

```
void min_max(int *array, int longitud, int &min, int &max) {
  min = std::numeric_limits<int>::max();
  max = std::numeric_limits<int>::min();

  for (int i = 0; i < longitud; i++) {
    min = std::min(min, array[i]);
    max = std::max(max, array[i]);
  }
}
```

- ¿Son parámetros de salida o de E/S?
- Llamadas a función:

```
int min, max;
min_max(arr, longitud, min, max);
```

Múltiples resultados

- ¿Cómo podemos especificar varios valores de retorno para una función, sin tener que recurrir a parámetros de salida?

Tipo específico

```
struct MinMaxResult {
  int min;
  int max;
};

MinMaxResult min_max(int *array, int longitud) {
  MinMaxResult res;
  res.min = std::numeric_limits<int>::max();
  res.max = std::numeric_limits<int>::min();

  for (int i = 0; i < longitud; i++) {
    res.min = std::min(res.min, array[i]);
    res.max = std::max(res.max, array[i]);
  }

  return res;
}
```

Tipo específico

- Llamada a la función:

```
MinMaxResult r = min_max(arr, longitud);
std::cout << "Min = " << r.min << " | Max = " << r.max;
```

- Problema: tener que definir un tipo específico.

Pares – `std::pair`

La clase `pair`

- Denota un par de elementos *(x, y)*, que pueden ser de distinto tipo.
- Definida en el fichero de cabecera `<utility>`

```
template <typename T1, typename T2>
class pair {
public:
  T1 first;
  T2 second;

  pair(const T1 &first, const T2 &second){ ... };
  ...
};
```

La clase pair

```
std::pair<int, int> min_max(int *array, int longitud) {
  int min = std::numeric_limits<int>::max();
  int max = std::numeric_limits<int>::min();

  for (int i = 0; i < longitud; i++) {
    min = std::min(min, array[i]);
    max = std::max(max, array[i]);
  }

  return std::pair<int, int>(min, max);
}
```

La clase pair

```
std::pair<int, int> min_max(int *array, int longitud) {
  int min = std::numeric_limits<int>::max();
  int max = std::numeric_limits<int>::min();

  for (int i = 0; i < longitud; i++) {
    min = std::min(min, array[i]);
    max = std::max(max, array[i]);
  }

  return {min, max};
}
```

La clase `pair`

- Llamada a la función:

```
std::pair<int, int> p = min_max(arr, longitud);
std::cout << "Min = " << p.first << " | Max = " << p.second;
```

- Sintaxis abreviada (*structured binding declaration*) de C++17.

```
auto [min, max] = min_max(arr, longitud);
std::cout << "Min = " << min << " | Max = " << max << std::endl;
```

  - En Visual Studio 2019 es necesario activar la opción `/std:c++17` o `/std:c++latest`.

La clase `pair`

- Hace explícitos los valores de salida.
- No requiere declarar ninguna clase.
- Pero… conviene documentar el significado de las componentes:

```
// Devuelve un par de enteros.
// - La primera componente es el valor mínimo del array
// - La segunda componente es el valor máximo del array

std::pair<int, int> min_max(int *array, int longitud) {
  …
}
```

¿Y si la función devuelve más de dos valores?

Tuplas – `std :: tuple`

La clase `tuple`

- Definida en el fichero de cabecera `<tuple>`

```
// Devuelve una tupla con tres componentes:
// - La primera componente es el valor mínimo del array
// - La segunda componente es el valor máximo del array
// - La tercera componente es la suma de los valores del array

std::tuple<int, int, int> min_max_sum(int *array, int longitud) {
  int min = std::numeric_limits<int>::max();
  int max = std::numeric_limits<int>::min();
  int sum = 0;

  for (int i = 0; i < longitud; i++) {
    min = std::min(min, array[i]);
    max = std::max(max, array[i]);
    sum += array[i];
  }

  return {min, max, sum};
}
```

La clase `tuple`

- Llamada:

```
auto [min, max, sum] = min_max_sum(arr, longitud);
std::cout << "Min = " << min << " | Max = " << max << " | Sum = " << sum;
```