ESTRUCTURAS DE DATOS

TIPOS ABSTRACTOS DE DATOS ARBORESCENTES

Implementando recorridos en profundidad (*DFS*)

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

Tipos de recorridos

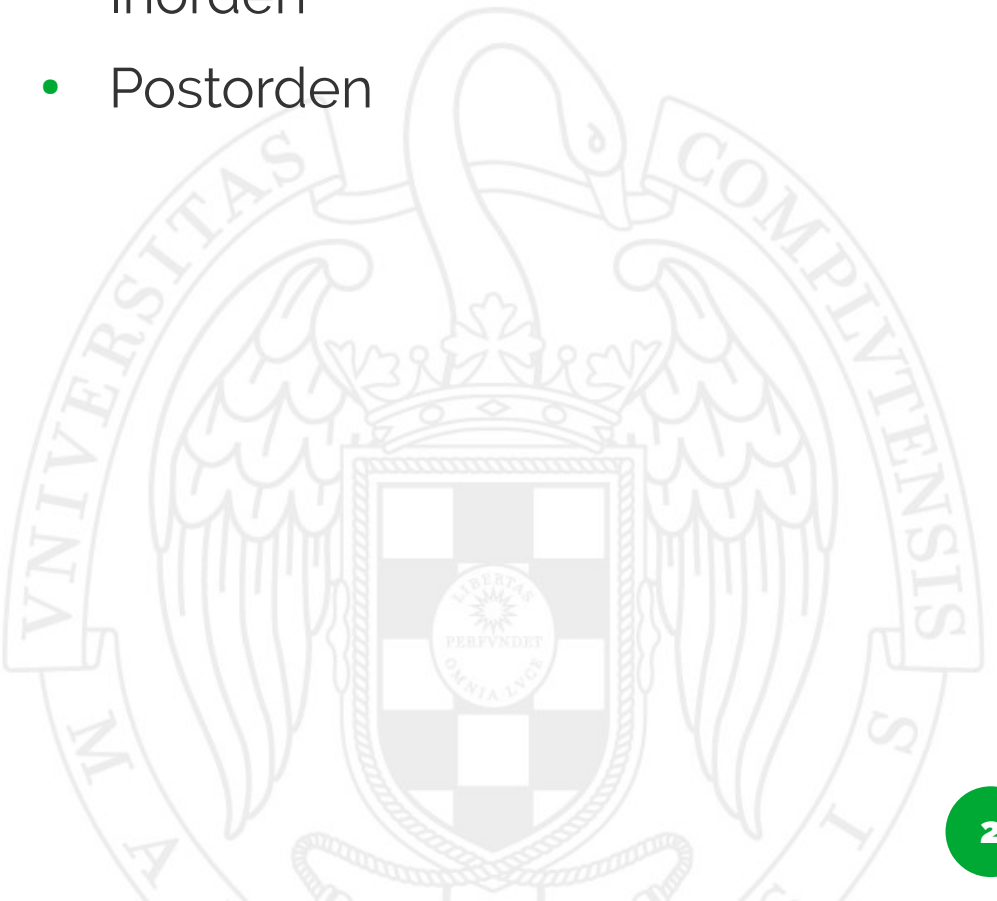
- Recorrido en profundidad

  Depth First Search (DFS)

  - Preorden
  - Inorden
  - Postorden

- Recorrido en anchura

  Breadth First Search (BFS)



Recordatorio: interfaz de BinTree<T>

```
template<class T>
class BinTree {
public:
  BinTree();
  BinTree(const T &elem);
  BinTree(const BinTree &left, const T &elem, const BinTree &right);

  const T & root() const;
  BinTree left() const;
  BinTree right() const;
  bool empty() const;

private:
  ...
};
```

Recordatorio: interfaz de `BinTree<T>`

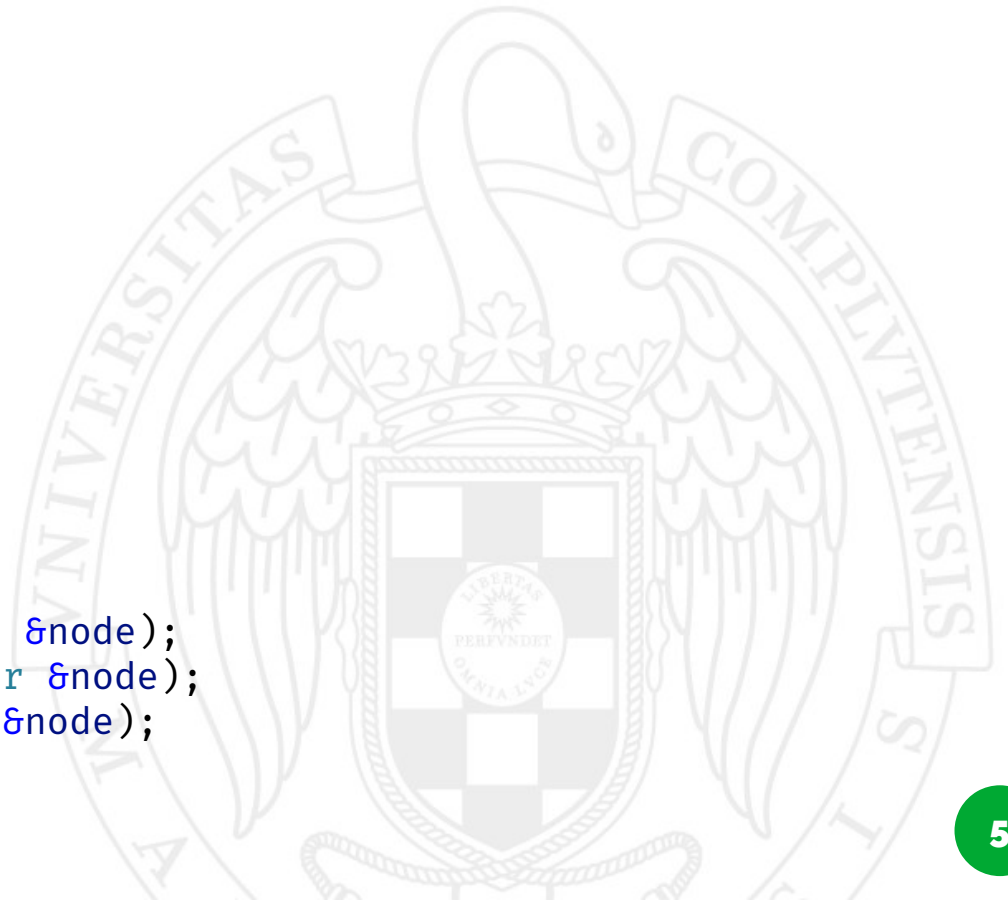
```
template<class T>
class BinTree {
public:
  // ...
  void preorder() const;
  void inorder() const;
  void postorder() const;

private:
  ...
};
```

- Añadimos tres nuevos métodos a `BinTree<T>`.

Recordatorio: interfaz de `BinTree<T>`

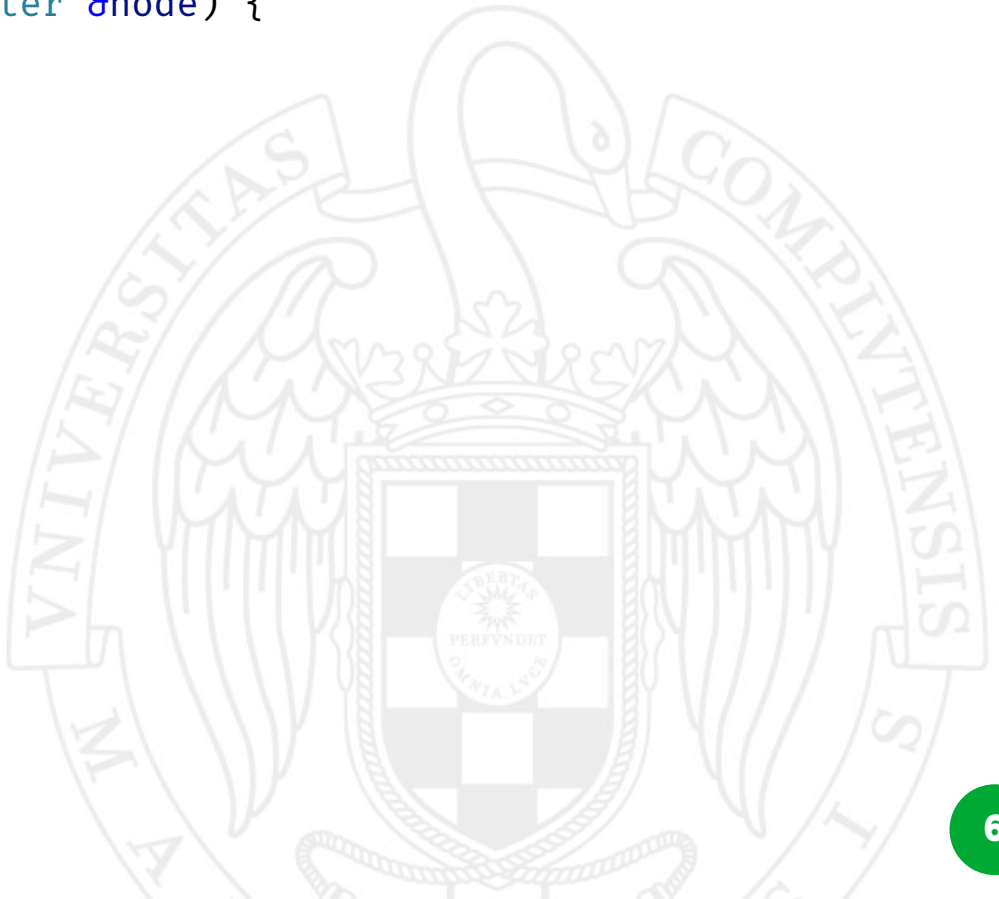
```
template<class T>
class BinTree {
public:
  // ...
  void preorder() const {
    preorder(root_node);
  }

  void inorder() const {
    inorder(root_node);
  }

  void postorder() const {
    postorder(root_node);
  }

private:
  static void preorder(const NodePointer &node);
  static void postorder(const NodePointer &node);
  static void inorder(const NodePointer &node);
...
};
```

- Estos métodos harán uso de otros tres métodos privados auxiliares.



Método auxiliar preorder

```
template<typename T>
void BinTree<T>::preorder(const NodePointer &node) {
  if (node ≠ nullptr) {
    std::cout << node→elem << " ";
    preorder(node→left);
    preorder(node→right);
  }
}
```

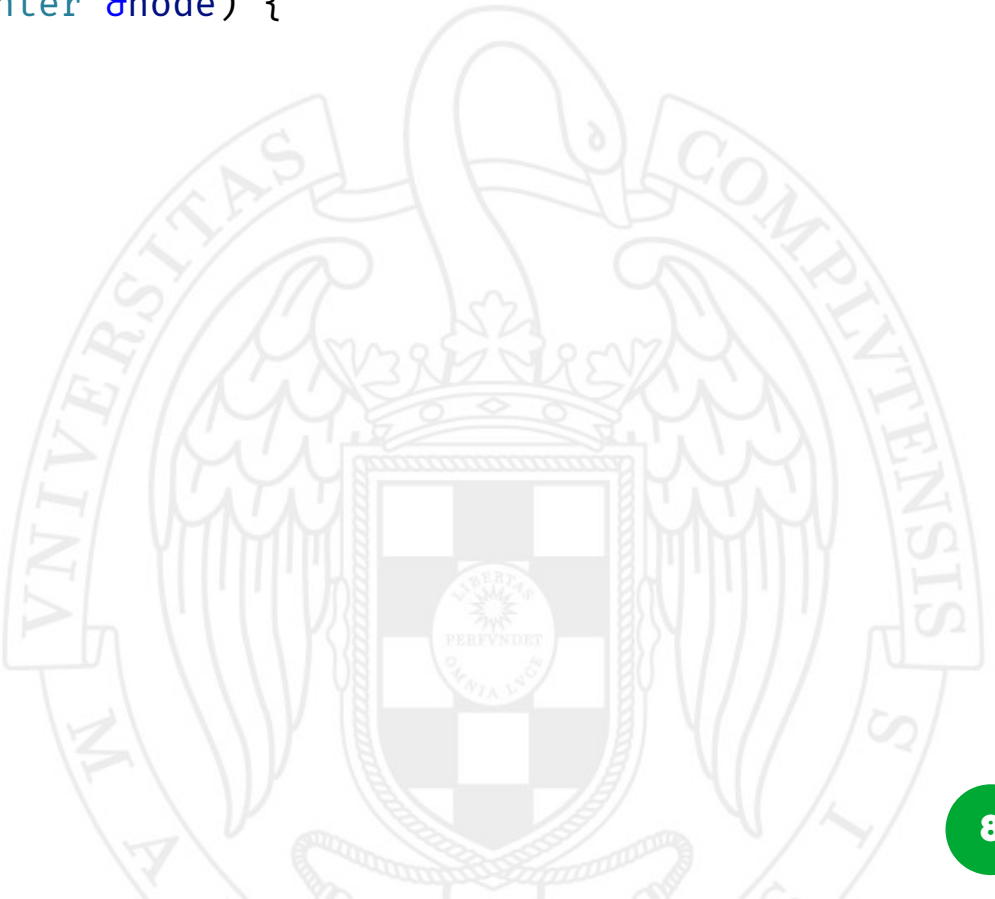


Método auxiliar `inorder`

```
template<typename T>
void BinTree<T>::inorder(const NodePointer &node) {
  if (node ≠ nullptr) {
    inorder(node→left);
    std::cout << node→elem << " ";
    inorder(node→right);
  }
}
```

Método auxiliar postorder

```
template<typename T>
void BinTree<T>::postorder(const NodePointer &node) {
  if (node ≠ nullptr) {
    postorder(node→left);
    postorder(node→right);
    std::cout << node→elem << " ";
  }
}
```



Ejemplo

```
int main() {
  BinTree<int> tree = {{{ 9 }, 4, { 5 }}, 7, {{ 10 }, 4, { 6 }}};

  std::cout << "Recorrido en preorden: " << std::endl;
  tree.preorder();
  std::cout << std::endl;

  std::cout << "Recorrido en inorden: " << std::endl;
  tree.inorder();
  std::cout << std::endl;

  std::cout << "Recorrido en postorden: " << std::endl;
  tree.postorder();
  std::cout << std::endl;

  return 0;
}
```

```
Recorrido en preorden:
7 4 9 5 4 10 6
Recorrido en inorden:
9 4 5 7 10 4 6
Recorrido en postorden:
9 5 4 10 6 4 7
```