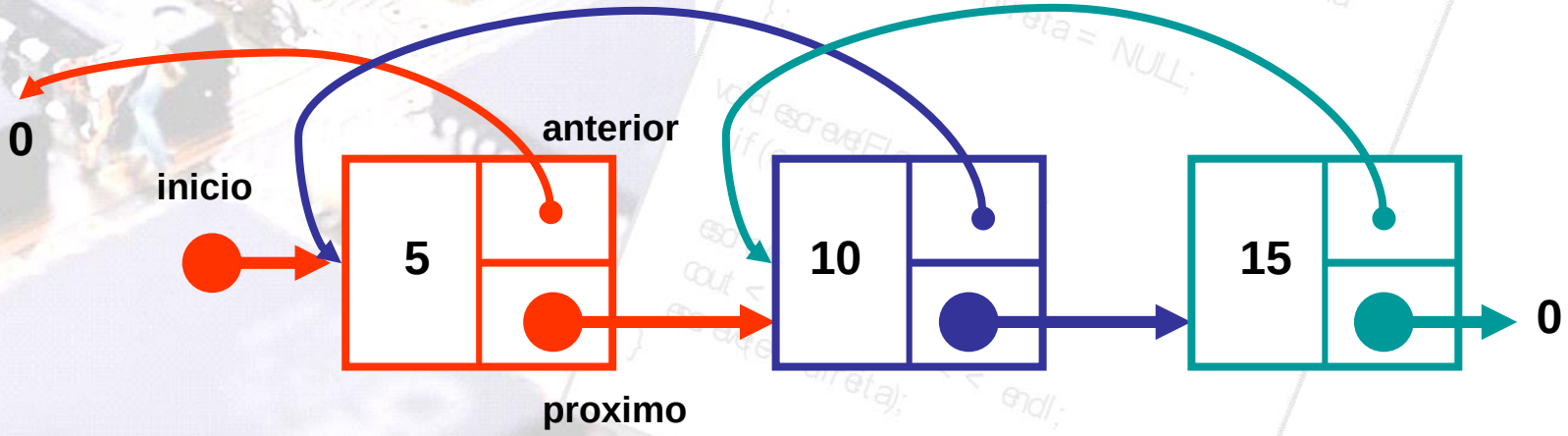
Listas duplamente ligadas

PN & JFig

2004

Listas duplamente ligadas

- Os elementos das listas duplamente ligadas têm, para além do valor e do ponteiro para o próximo elemento, um ponteiro para o elemento anterior da lista.



Listas duplamente ligadas: class Elemento2

- A *class Elemento* utilizada nas listas simplesmente ligadas, tem de sofrer uma ligeira alteração para a sua utilização nas listas duplamente ligadas.
- É necessário acrescentar um novo ponteiro, designado anterior, para indicar a ligação com o elemento anterior da lista.

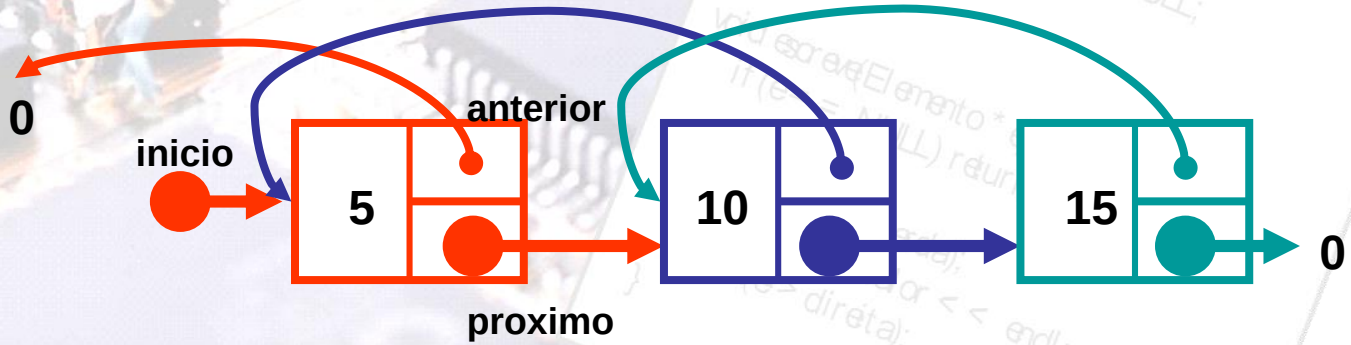
Listas duplamente ligadas:
class Elemento2 - implementação

```
class Elemento {
        public:
                int info;
            Elemento *proximo;
            Elemento(int e) {
                    info = e;
                    proximo = 0;
            }
                Elemento() {
                    proximo = 0;
            }
};
```

```
class Elemento2 {
        public:
            int info;
            Elemento *proximo;
            Elemento *anterior;
            Elemento(int e) {
                    info = e;
                    proximo = 0;
                    anterior = 0;
            }
             Elemento() {
                    proximo = 0;
                    anterior = 0;
            }
};
```

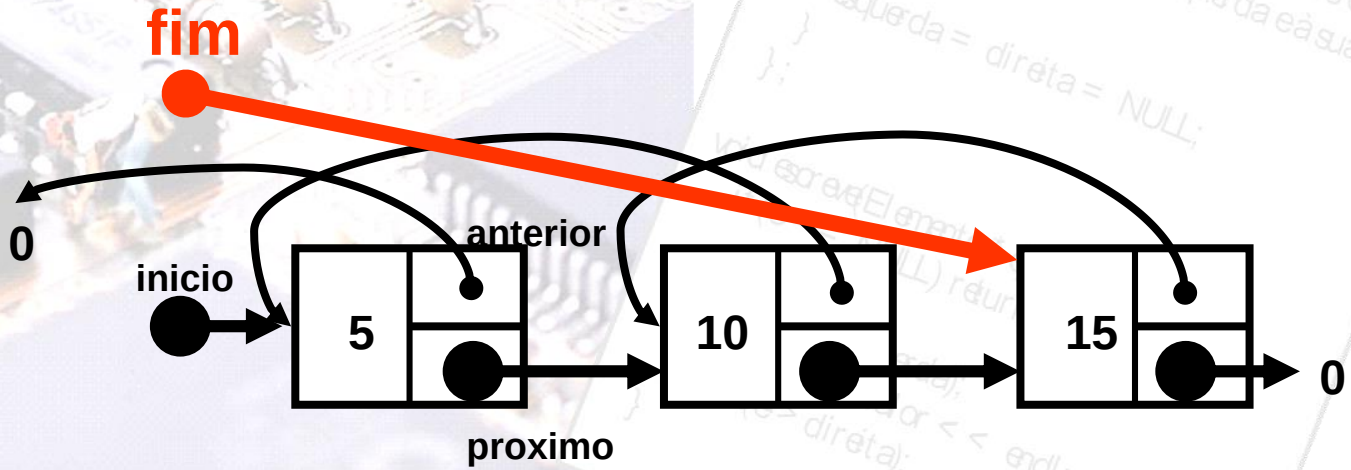
Listas duplamente ligadas:

- Com a utilização deste novo ponteiro (*anterior*), é possível percorrer a lista do inicio para o fim, bem como, do fim para o inicio.



Listas duplamente ligadas:

- Para dar mais versatilidade à lista duplamente ligada, vamos adicionar um ponteiro para o fim da lista.



Lista dupla: implementação

```
class ListaDupla{
        private:
                Elemento2 *inicio;
                Elemento2 *fim;
        public:
                ListaDupla(){
                        inicio = 0;
                        fim = 0;
                }
                void inserirInicio(int valor);
                void inserirFim(int valor);
                int eliminaInicio();
                string verTudo();
                string verTudoInverso();
                Elemento2 * getInicio();
                Elemento2 * getFim();
};
```

Lista dupla: implementação

```
void ListaDupla::inserirInicio(int valor){
        Elemento2 *novo = new Elemento2(valor);
        if (inicio==0){
           fim = novo;
        }  else {
             novo->proximo = inicio;
             inicio->anterior = novo;
        }
        inicio = novo;
}
```

Lista dupla: implementação

```
void ListaDupla::inserirFim(int valor){
      Elemento2 *novo = new Elemento2(valor);
       if (fim==0){
         inicio = novo;
       }
        else{
                novo->anterior = fim;
                fim->proximo = novo;
      }
      fim = novo;
}
```

Lista dupla: implementação

```
string ListaDupla::verTudo(){
        string s("");
        char sx[10];
        Elemento2 *aux=inicio;
        for (;aux;){
                itoa(aux->info,sx,10);
                s += sx;
                s.append(" ");
                aux = aux->proximo;
        }
        return s;
}
```

Lista dupla: implementação

```
string ListaDupla::verTudoInverso(){
        string s("");
        char sx[10];
        Elemento2 *aux = fim;
        for (;aux;){
                itoa(aux->info,sx,10);
                s += sx;
                s.append(" ");
                aux = aux->anterior;
        }
        return s;
}
```

Lista dupla: implementação

```
Elemento2 * ListaDupla::getInicio(){
        return inicio;
}

Elemento2 * ListaDupla::getFim(){
        return fim;
}
```

Elaborar a função para inserir ordenado.

```
void InserirOrdenado(int e)
{
  Elemento * seguinte = inicio;
  while((seguinte != 0) && (seguinte->info < e))
    seguinte = seguinte->proximo;
  Elemento2 * novo = new Elemento2(e);
  novo->proximo = seguinte;
  if (seguinte == NULL) {   // ins fim
    novo->anterior = fim;
    fim = novo;
  } else {
    novo->anterior = seguinte->anterior;
    seguinte->anterior = novo;
  }
  Elemento2 * ant = novo->anterior;
  if (ant == NULL)
    inicio = novo;
  else
    ant->proximo = novo;
}
```