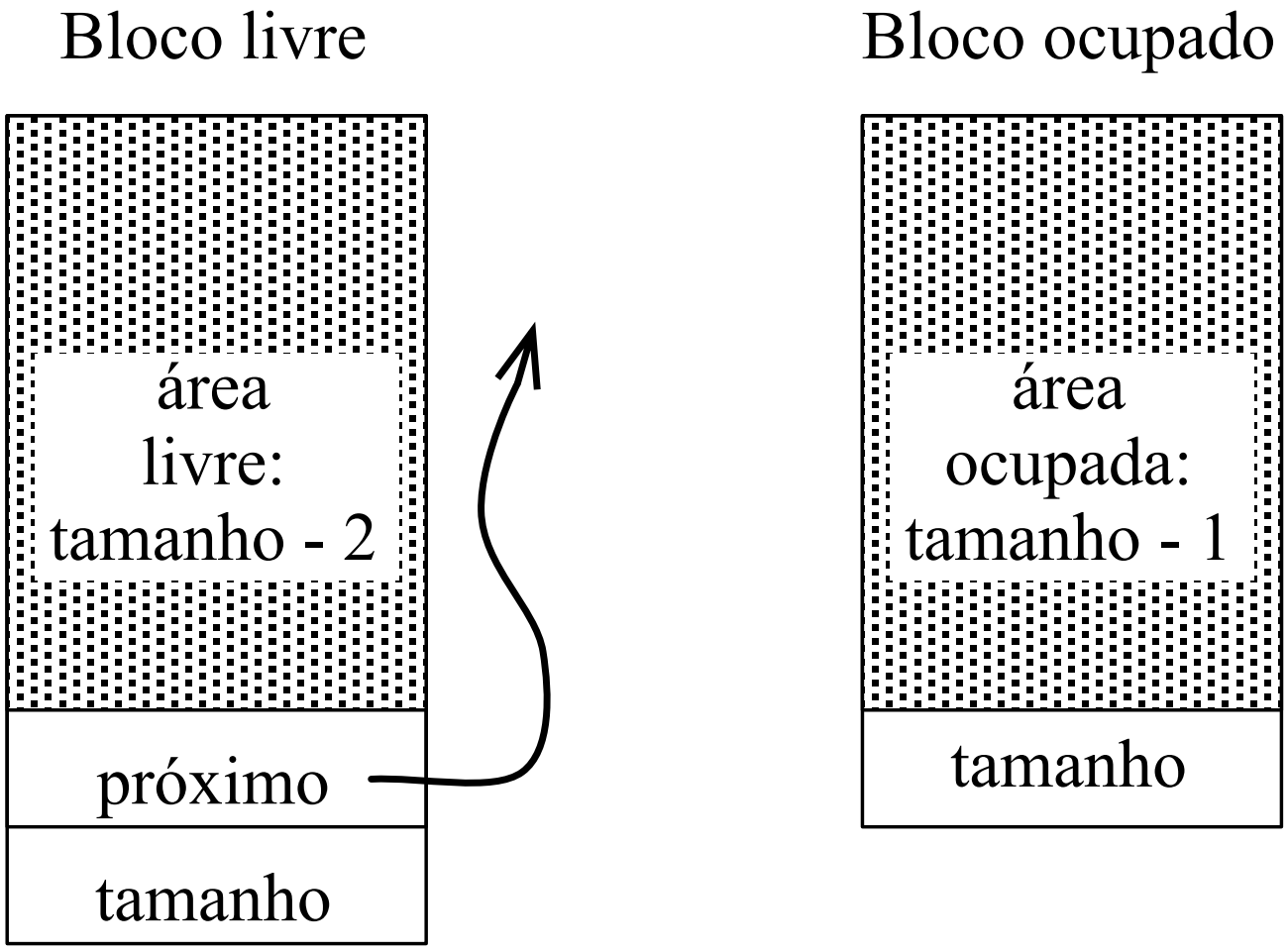
Gerenciamento de Memória em Linguagens OO

Fernando Magno Quintão Pereira

16 de Novembro de 2010

Os elementos que compõem o heap



O algoritmo de alocação

```
@ allocate(int size) {
  @ aux = FL;
  @ lag = NULL;
  while (aux != NULL && *aux < size + 1) {
    lag = aux;
    aux = *aux + 1;
  }
  if (aux == NULL) throw new OutOfMemoryError();
  @ nextFree = aux + size + 1;
  *nextFree = *aux - size - 1;
  *(nextFree + 1) = *(aux + 1);
  if (lag == NULL)
    FL = nextFree;
  else
    *(lag + 1) = nextFree;
  *aux = size + 1;
  return aux + 1;
}
```

O algoritmo de dealocação

```
deallocate(@p) {
    @aux = p - 1;
    *p = FL;
    FL = aux;
}
```

O que acontecerá nesta sequência, sendo size = 10?

```
@ allocate(int size) {
  @ aux = FL;
  @ lag = NULL;
  while (aux != NULL && *aux < size + 1) {
    lag = aux;
    aux = *aux + 1;
  }
  if (aux == NULL) throw new OutOfMemoryError();
  @ nextFree = aux + size + 1;
  *nextFree = *aux - size - 1;
  *(nextFree + 1) = *(aux + 1);
  if (lag == NULL)
    FL = nextFree;
  else
    *(lag + 1) = nextFree;
  *aux = size + 1;
  return aux + 1;
}
```

```
p1 = m.allocate(4);
p2 = m.allocate(2);
m.deallocate(p1);
p3 = m.allocate(1);
```

O que acontecerá nesta sequência, sendo size = 10?

```
@ allocate(int size) {
  @ aux = FL;
  @ lag = NULL;
  while (aux != NULL && *aux < size + 1) {
    lag = aux;
    aux = *aux + 1;
  }
  if (aux == NULL) throw new OutOfMemoryError();
  @ nextFree = aux + size + 1;
  *nextFree = *aux - size - 1;
  *(nextFree + 1) = *(aux + 1);
  if (lag == NULL)
    FL = nextFree;
  else
    *(lag + 1) = nextFree;
  *aux = size + 1;
  return aux + 1;
}
```

```
p1 = m.allocate(4);
p2 = m.allocate(2);
m.deallocate(p1);
m.deallocate(p2);
p3 = m.allocate(7);
```

O que acontecerá nesta sequência, sendo size = 10?

```
@ allocate(int size) {
  @ aux = FL;
  @ lag = NULL;
  while (aux != NULL && *aux < size + 1) {
    lag = aux;
    aux = *aux + 1;
  }
  if (aux == NULL) throw new OutOfMemoryError();
  @ nextFree = aux + size + 1;
  *nextFree = *aux - size - 1;
  *(nextFree + 1) = *(aux + 1);
  if (lag == NULL)
    FL = nextFree;
  else
    *(lag + 1) = nextFree;
  *aux = size + 1;
  return aux + 1;
}
```

```
p1 = m.allocate(4);
p2 = m.allocate(1);
m.deallocate(p1);
p3 = m.allocate(5);
```