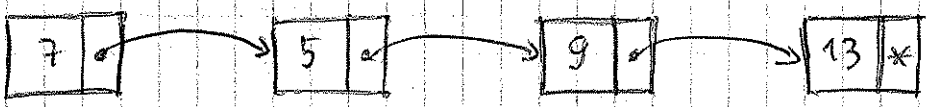
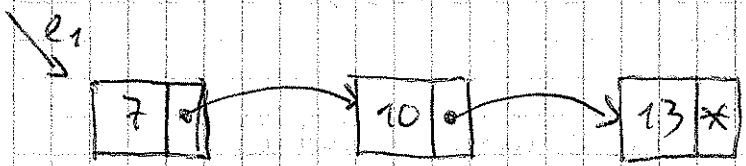


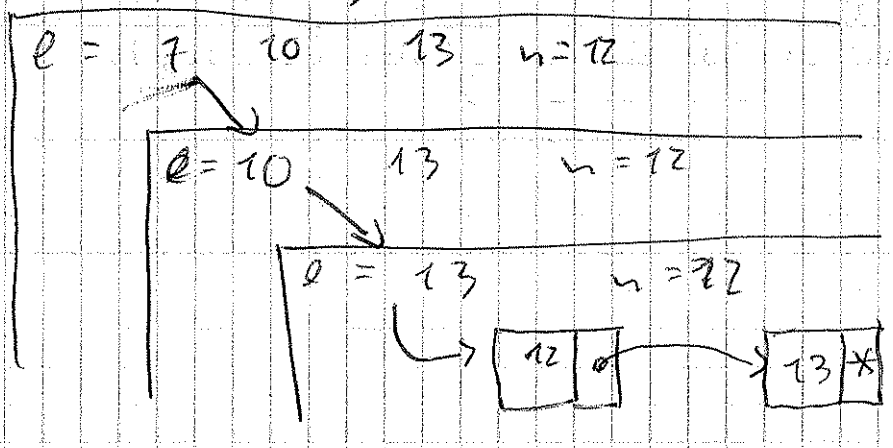

```
LInt inserir (LInt l, int n)
{
  LInt aux;
  aux = ((LInt) malloc (sizeof (Node)));
  aux -> valor = n;
  aux -> seg = l;
  return aux;
}

typedef struct sLInt
{     int valor;
      struct sLInt  *seg;
} * LInt, Node;

LInt insord (LInt l, int n)
{
  LInt aux;
  if (!l || l -> valor > n) {
        aux = (LInt) malloc (sizeof (Nodo));
        aux -> valor = n;
        aux -> seg = l;
        return aux;
    }
  else {
      l -> seg = insord (l -> seg, n);
    }
  return (l);
}
```



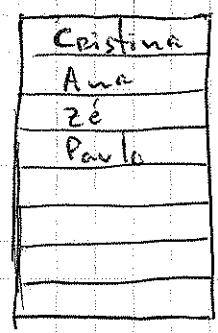

```
l1 = insord (l1, 12);
```



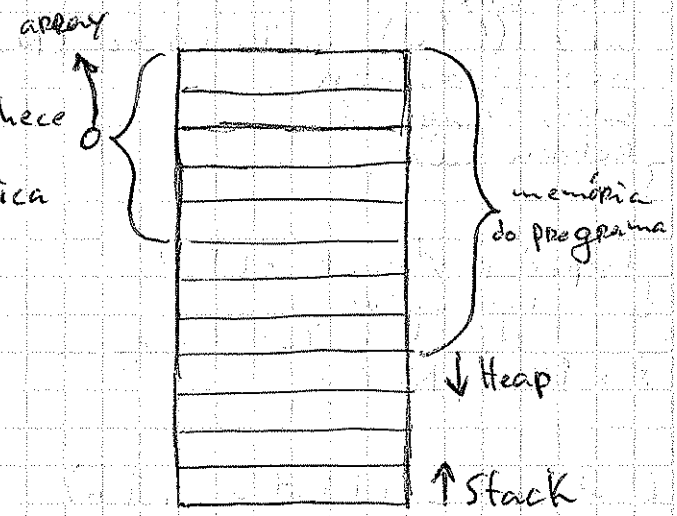
15-04-2010 LEI-T-(23)

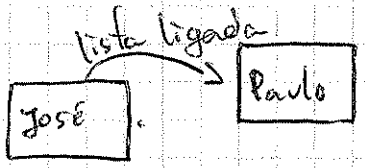
Pedro Meleipe de Nápoles - 54798

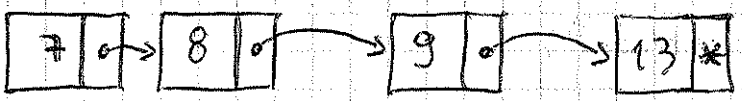
Gestão de alunos



Quando não se conhece o limite:
⇒ Memória dinâmica







Cada uma das estruturas é dividida em duas partes, neste caso, um inteiro e um apontador para a estrutura seguinte

```
typedef struct sLInt
{
    int valor;
    struct sLInt seg
} *LInt;
```

LInt l1;] → apontador para uma struct

alocar espaço na memória ← void *malloc(int nbytes);
libertar espaço na memória ← void free(void*);

```
int main () {
    LInt l1 = NULL;
    l1 = inserir (l1, 13);
    l1 = inserir (inserir (l1, 13), 9);
    ...
    listar (l1);
}
```

```
void listar (LInt l) {
    if (l)
    {printf ("%d ", l->valor);
     listar (l->seg);
    }
}
```