Tabela de Símbolos

- Coleção de pares de chave-valor
- Mecanismo abstrato para armazenar informações que podem ser acessadas através de uma chave
- Dicionários e maps
- Chaves duplicadas: em muitas aplicações não são permitidas
- Dificuldade:
  - Definir uma estrutura que represente uma tabela de símbolos capaz de armazenar uma grande quantidade de dados (informações + chaves)
  - Definir formar de recuperar essas informações eficientemente

Busca binária em vetores ordenados

- Paradigma da divisão e conquista
  - Dividir o vetor no meio
  - Procurar o elemento na esquerda: elemento procurado seja menor que o elemento central
  - Procurar o elemento na direita: elemento procurado seja maior que o elemento central
  - Repetir, recursivamente, até o elemento procurado ser o elemento central (ou não - falha na busca)
- Complexidade: até $\lfloor lgN \rfloor + 1$ comparações (acerto ou falha)

Busca binária em vetores ordenados

```
1   #define key(A) (A.chave)
2
3   typedef int Key;
4   typedef struct data Item;
5   struct data { Key chave; char info[100]; };
6
7   Item binary_search(Item *v, int l, int r, Key k)
8   {
9       if(l >= r) return NULL;
10
11      int m = (l+r)/2; //l+(r-l)/2
12      if(k == key(v[m])) return v[m];
13      if(k < key(v[m]))
14          return binary_search(v, l, m-1, k);
15
16      return binary_search(v, m+1, r, k);
17  }
18
```

Busca binária em vetores ordenados

Interpolation search

```
1  Item binary_search(Item *v, int l, int r, Key k)
2  {
3      if(l >= r) return NULL;
4
5      int m = l + (r-l)*((k-key(v[l]))/(key(v[r])-(key(v[l
       ]))));
6
7      if(k == key(v[m])) return v[m];
8      if(k < key(v[m]))
9          return binary_search(v, l, m-1, k);
10
11     return binary_search(v, m+1, r, k);
12 }
```

- Interessante para muitas chaves, mas ...
- é altamente dependente da boa distribuição das chaves