Algoritmos de ordenação

- Ordenar crescentemente (decrescentemente) significa alterar a ordem pela qual surgem os elementos de uma sequência, tal que o primeiro seja menor (maior) do que o segundo, o segundo menor (maior) do que o terceiro, e assim sucessivamente.
- A utilidade destes algoritmos está na resolução de problemas como:
  - ✔ pesquisa (lista telefónica);
  - ✔ agrupar elementos repetidos:

    (7, 1, 2, 7, 2, 5) $\rightarrow$ (1, 2, 2, 5, 7, 7) ou (7, 7, 5, 2, 2, 1).
- Dado um vector V de N inteiros (caso mais simples), pretende-se construir um algoritmo que, após a sua execução, deixe o vector V ordenado por ordem crescente: V[0] ≤ V[1] ≤ V[2] ≤ ... ≤ V[N-1].

Algoritmos de ordenação

Ordenação por selecção

- 1ª posição (V[0]) o menor elemento,
- 2ª posição (V[1]) o 2º menor elemento,
- ...
- k-ésima posição (V[k-1]) o k-ésimo menor elemento,
- ...
- N-ésima (última) posição (V[N-1]) o maior elemento.

Algoritmos de ordenação

Ordenação por selecção (Algoritmo – versão 1)

➔ Para a 1ª posição (V[0])

- ✔ Det. a posição do menor elemento (pos_menor) de V[0] a V[N-1]
- ✔ Trocar o elemento da *pos_menor* com o elemento da 1ª posição

➔ Para a 2ª posição (V[1])

- ✔ Det. a posição do menor elemento (pos_menor) de V[1] a V[N-1]
- ✔ Trocar o elemento da *pos_menor* com o elemento da 2ª posição

➔ . . .

➔ Para a k-ésima posição (V[k])

- ✔ Det. a posição do menor elemento (*pos_menor*) de V[k] a V[N-1]
- ✔ Trocar o elemento da *pos_menor* com o elemento da k-ésima posição

➔ . . .

Algoritmos de ordenação

Ordenação por selecção (Algoritmo – versão 1)

➔ . . .

➔ Para a (N-1)-ésima posição (V[N-2])

- ✔ Det. a posição do menor elemento (*pos_menor*) de V[N-2] a V[N-1]
- ✔ Trocar o elemento da *pos_menor* com o elemento da (N-1)-ésima posição

➔ Para a N-ésima posição (V[N-1])

- ✔ Det. a posição do menor elemento (*pos_menor*) de V[N-1] a V[N-1]. Como só falta um elemento para ordenar, V[N-1], este encontra-se automaticamente ordenado (é o maior elemento de todos).

Algoritmos de ordenação

Ordenação por selecção (Algoritmo – versão 2)

Para k desde 0 até N-2 fazer:

pos_menor ← k

Para kk desde k+1 até N-1 fazer:

Se (V[kk] < V[pos_menor]) então

pos_menor ← kk

Trocar V[pos_menor] com V[k], se necessário, da seguinte forma:

aux ← V[pos_menor]

V[pos_menor] ← V[k]

V[k] ← aux

Algoritmos de ordenação

Ordenação por selecção (função em C)

```
void Ordenar_Seleccao  (int  V[],  int  N)
{
   int  k, kk, pos_menor, aux;
   for (k = 0; k < N-1; k++)
   {
      pos_menor = k;
      for (kk = k+1; kk < N; kk++)
         if (V[kk] < V[pos_menor])  pos_menor = kk;
      if (pos_menor != k) {
         aux = V[pos_menor];
         V[pos_menor] = V[k];
         V[k] = aux; }
   }
}
```

Algoritmos de ordenação

Ordenação por borbulhagem ("bubble sort")

- Consiste em comparar 2 elementos consecutivos e, se estiverem desordenados, trocá-los entre si.
- Desta forma, os maiores elementos tendem a deslocar-se para a direita e os menores para a esquerda do vector.
- O vector fica ordenado quando após várias passagem pelo vector com pelo menos uma troca, não há qualquer troca na actual passagem.

Algoritmos de ordenação

Ordenação por borbulhagem (Exemplo)

➔ (7, 5, 2, 6)

➔ Primeira passagem:

- ✔ Troca do 1º elemento com o 2º (7 $\leftrightarrow$ 5) V = (5, 7, 2, 6)
- ✔ Troca do 2º elemento com o 3º (7 $\leftrightarrow$ 2) V = (5, 2, 7, 6)
- ✔ Troca do 3º elemento com o 4º (7 $\leftrightarrow$ 6) V = (5, 2, 6, 7)

➔ Segunda passagem:

- ✔ Troca do 1º elemento com o 2º (5 $\leftrightarrow$ 2) V = (2, 5, 6, 7)

➔ Terceira passagem:

- ✔ Sem qualquer troca $\Rightarrow$ V = (2, 5, 6, 7) está ordenado.

Algoritmos de ordenação

Ordenação por borbulhagem (Algoritmo)

Fazer:

Num_trocas $\leftarrow$ 0;

Para k desde 0 até N-2 fazer: { k = N-2 $\Rightarrow$ k+1 = N-1 }

Se (V[k] > V[k+1]) então

Trocar V[k] com V[k+1]

Num_trocas $\leftarrow$ Num_trocas + 1

Enquanto (Num_trocas $\neq$ 0)

Algoritmos de ordenação

Ordenação por borbulhagem (função em C)

```
void Ordenar_Borbulhagem  (int  V[],  int  N)
{
   int  k, Num_trocas, aux;
   do{
      Num_trocas = 0;
      for (k = 0; k < N-1; k++)
         if (V[k] > V[k+1])
         {
            aux = V[k];
            V[k] = V[k+1];
            V[k+1] = aux;
            Num_trocas++;
         }
   } while (Num_trocas != 0);
}
```

Algoritmos de ordenação

Ordenação por borbulhagem (função em C optimizada)

```
void Ordenar_Borbulhagem_2  (int  V[],  int  N)
{
   int  k, kk, fim = N-1, aux;
   do {
      kk = 0;
      for (k = 0; k < fim; k++)
         if (V[k] > V[k+1]) {
            aux = V[k];
            V[k] = V[k+1];
            V[k+1] = aux;
            kk = k;
         }
      fim = kk;
   } while (kk != 0);
}
```