Bubble sort

```
Algoritmo
       declare X[5],n, i, aux: numérico;
       //carregando o vetor
       para i← 0 até 4 faça
       inicio
           escreva (”o “, i+1,”º número: “);
           leia(X[i]);
       fim_para;
       // ordenando de forma crescente
       // laço com a quantidade de elementos
       para n← 1 até 5 faça
       inicio
       //laço que percorre da primeira até a penúltima posição do vetor
            para i← 0 até 3 faca
            inicio
                se (X[i] > X[i + 1]) então
                inicio
                    aux ← X[i];
                    X[i] ← X[i + 1];
                    X[i + 1] ← aux;
                fim_se;
            fim_para;
        fim_para;
        // mostrando o vetor ordenado
        para i←0 até 4 faça
        inicio
            escreva (i+1,"º número: ", X[i]);
        fim_para;
fim_algoritmo.
```

Inserção

```
algoritmo
    declare X[5],j, i, eleito: numérico;
    //carregando o vetor
    para i← 0 até 4 faça
    inicio
       escreva (“Digite o “, i+1, ”º número: “);
       leia (X[i]);
    fim_para;
    // ordenando de forma crescente
    // laço com a quantidade de elementos
    para i←1 até 4 faça
    inicio
       eleito ← X[i];
       j ← i - 1;
       //laço que percorre os elementos
       //à esquerda do número eleito ou até encontrar a posição
       //para relocação do número eleito respeitando a
       //ordenação procurada
       enquanto (j >= 0 .E. X[j] > eleito)
       inicio
           X[j + 1] ← X[j];
           j ← j – 1;;
       fim_enquanto;
       X[j + 1] ← eleito;
    Fim_para;
    // mostrando o vetor ordenado
    para i←0 até 4 faça
    inicio
       escreva(i+1,"º número: ", X[i]);
    fim_para;
fim_algoritmo.
```

Seleção

```
Algoritmo
  declare j, i, eleito, X[5]: numérico;
  //carregando o vetor
  para i← 0 até 4 faça
  inicio
     escreva (“Digite o “, i+1,”º número: “);
     leia (X[i]);
  fim_para
  // ordenando de forma crescente
  // laço que percorre da 1ª posição à penúltima
  // posição do vetor elegendo um número para ser comparado
  para i← 0 até 3 faça
  inicio
     eleito ← X[i];
     // encontrando o menor número à direita do eleito
     //com sua respectiva posição
     //posição do eleito = i
     // primeiro número à direita do eleito
     // na posição = i+1
     menor ← X[i + 1] ;
     pos ← i + 1;
     //laço que percorre os elementos que estão
     //à direita do número eleito, retornando o menor número à direita e
     //sua posição
     para j← i + 1 até 4 faça
     inicio
        se (X[j]<menor) então
        inicio
           menor ← X[j];
           pos ← j;
        fim_se;
     fim_para;
     //troca do número eleito com o número da posição pos
     //o número da posição pos é o menor número à direita
     //do número eleito
     se (menor< eleito) então
     inicio
         X[i] ← X[pos];
         X[pos] ← eleito;
     fim_se;
   fim_para;
   // mostrando o vetor ordenado
   para i← 0 até 4 faça
   inicio
      escreva (i+1,"º número : ", X[i]);
   fim_para;
fim_algoritmo.
```

Quick sort

```
Algoritmo
       declare i, X[10]: numérico;
       //carregando o vetor
       para i← 0 até 9 faça
         inicio
            escreva (“Digite o “, i+1,”º número: “);
            leia (X[i]);
         fim_para;
         // ordenando de forma crescente
         quicksort(X, 0, 9);

         // mostrando o vetor ordenado
          para i← 0 até 9 faça
          inicio
             escreva (i+1,"º número : ", X[i]);
          fim_para;
fim_algoritmo.

função troca(X, i, j)
inicio
   declare aux: numérico;
   aux ← X[i];
   X[i] ← X[j];
   X[j] ← aux;
fim_função_troca.

função particao(X, p, r)
inicio
   declare pivo, i, j: numérico;
   pivo ← X[(p + r) / 2];
   i ← p - 1;
   j ← r + 1;
   enquanto (i < j)
   inicio
      repita
          j ← j - 1;
      até (X[j] <= pivo);
      repita
          i ← i + 1;
      até (X[i] >= pivo);
      se (i < j) então
         troca(X, i, j);
      retorne j;
fim_função_partição.


Função quicksort(X, p, r)
inicio
   declare q: numérico;
   se (p < r) então
   inicio
          q ← particao(X, p, r);
          quicksort(X, p, q);
          quicksort(X, q + 1, r);
   fim_se;
fim_função_quicksort.
```