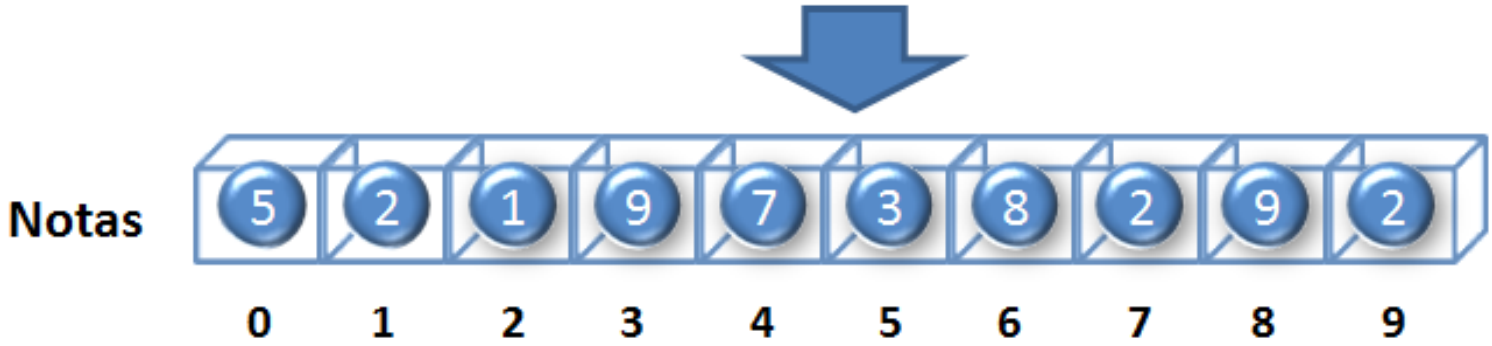
BCC 201 - Introdução à Programação I

Estruturas Homogêneas (Vetores) II

Guillermo Cámara-Chávez
UFOP

Exercícios I



Declaração de um vetor de inteiros

```
int notas[10];
```

Acessando um elemento do vetor: indicamos o nome do vetor e a posição

```
notas[6] = 8;
```

Exercícios II

Leitura via teclado

```
int i, notas[10];
for (i = 0; i < 10; i++){
    cout << "Inserir nota " << i+1;
    cin >> notas[i];
}
...
```

Escrita do vetor

```
int i, notas[10];
...
for (i = 0; i < 10; i++){
    cout << notas[i];
}
...
```

Exercícios I

Escreva um programa que leia 20 valores inteiros e os armazene em um vetor. Depois de ler os 20 valores, o programa deve percorrer o vetor e mostrar na tela apenas os números pares que foram armazenados.

Exercícios II

```
int main(){
    int numVet[20], i;
    for (i = 0; i < 20; i++)
    {
        cout << "Inserir numero " << i+1;
        cin >> numVet[i];
    }
    for (i = 0; i < 20; i++)
    {
        if (numVet[i] % 2 == 0)
            cout << numVet[i];
    }
    return 0;
}
```

Exercícios III

Faça um programa que receba dez números inteiros e armazene-os em um vetor. O programa deve calcular e mostrar dois vetores resultantes, sendo o primeiro com os números pares e o segundo com os números ímpares do vetor lido.

Exercícios IV

```
int main(){
    int vet[10], par[10], impar[10];
    int nPar = 0 nImpar = 0, i;
    for (i = 0; i < 10; i++){
        cout << "Inserir numero " << i+1;
        cin >> vet[i];
    }
    for (i = 0; i < 10; i++){
        if (vet[i] % 2 == 0){
            par[nPar] = vet[i];
            nPar++;
        }
        else{
            impar[nImpar] = vet[i];
            nImpar++;
        }
    }
    ...
}
```

Exercícios V

```
int main(){
    ...
    for (i = 0; i < numPar; i++)
        cout << par[i];
    cout << endl;
    for (i = 0; i < numImpar; i++)
        cout << impar[i];
    return 0;
}
```

Exercícios VI

Faça um programa que leia um vetor de números inteiros de 10 posições. O programa deve calcular e mostrar o maior elemento do vetor e em que posição esse elemento se encontra

Exercícios VII

```
int main(){
    int vet[10], i, maior, pos;
    for (i = 0; i < 10; i++){
        cout << "Inserir numero " << i+1;
        cin >> vet[i];
    }
    pos = 0;
    maior = vet[0];
    for (i = 1; i < 10; i++)
    {
        if (vet[i] > maior)
        {
            maior = vet[i];
            pos = i;
        }
    }
    cout << "O maior elemento " << maior
         << "esta na posicao " << pos;
    return 0;
}
```

Exercícios VIII

Faça um algoritmo que leia um vetor `V[60]`. A seguir, troque o 1° elemento com o 31°, o 2° com o 32°, etc. Mostre no final o vetor modificado.

Exercícios IX

```
int main(){
    int vet[60], i, tmp;
    for (i = 0; i < 60; i++){
        cout << "Inserir numero " << i+1;
        cin >> vet[i];
    }
    for (i = 0; i < 30; i++)
    {
        tmp = vet[i];
        vet[i] = vet[i+30];
        vet[i+30] = tmp;
    }
    for (i = 0; i < 60; i++)
        cin >> vet[i];
    return 0;
}
```

Exercícios X

Fazer um algoritmo que:

1. Leia n valores numéricos e os armazene num arranjo unidimensional v. O valor de n também deve ser lido.
2. Calcule e exiba o valor da série:

$$S = \sum_{i=0}^{n-1} \frac{i+1}{v_i}$$

   onde v_i é o i-ésimo valor armazenado na variável v.
3. Calcule exiba quantos termos da série têm o numerador inferior ao denominador.

Exercícios XI

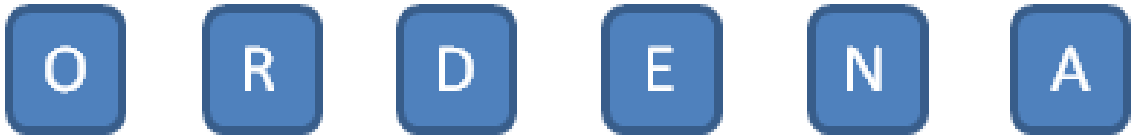
```
int main(){
    int n, i, cont = 0;
    double S = 0, v[100];
    cout << "\n Indicar numero de elementos: ";
    cin >> n;
    cout << "\n Inserir " << n << " numeros";
    for (i = 0; i < n; i++)
        cin >> v[i];
    for (i = 0; i < n; i++)
    {
        if (v[i] != 0){
            S += (i / v[i]);
            if (i < v[i])
                cont++;
        }
    }
    cout << cont << " termos com num inf. ao den";
    return 0;
}
```

Ordenação por BubbleSort (Método da bolha)

- É um dos piores métodos de ordenação.
- É uma ordenação por trocas.
- Implica repetidas comparações e, se necessário, troca de dois elementos adjacentes.

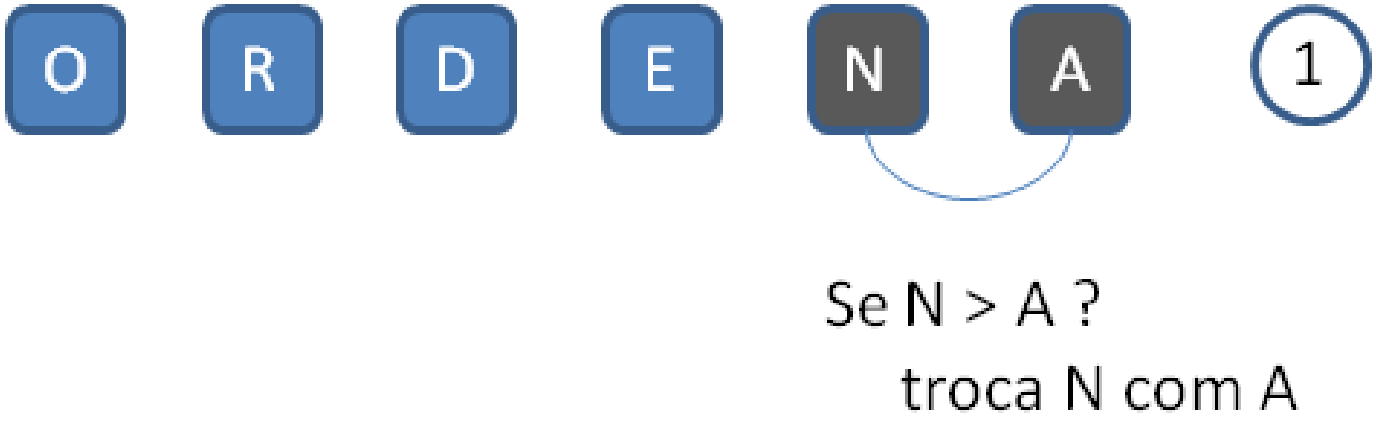
Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

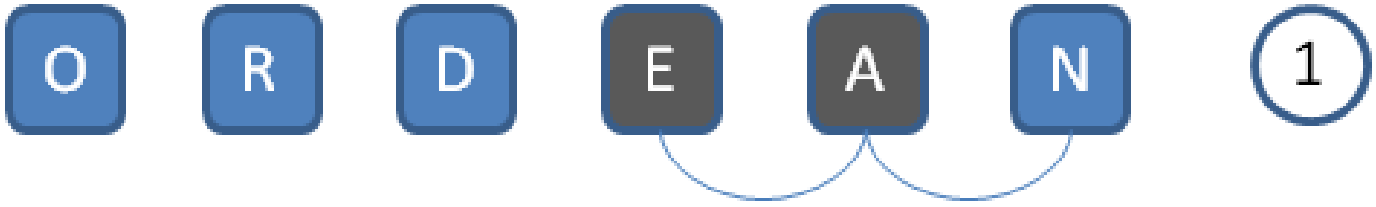


Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Se E > A ?

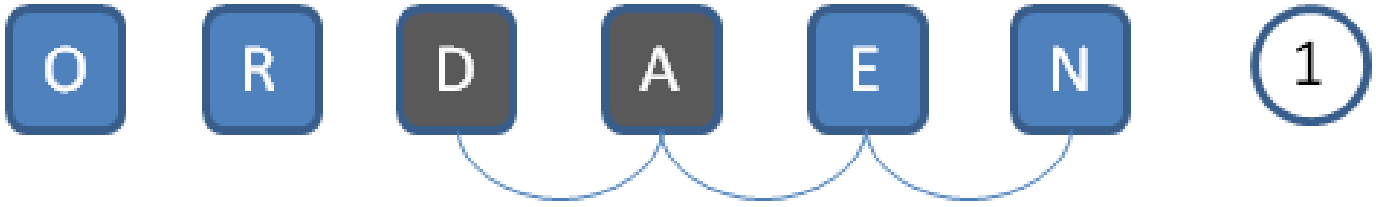
troca E com A

Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

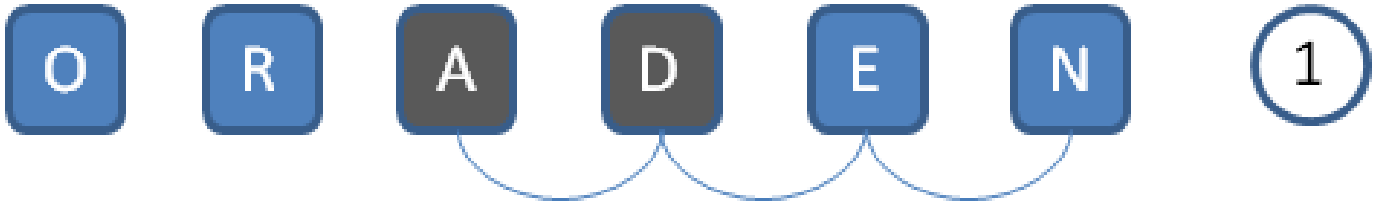


Se D > A ?

troca D com A

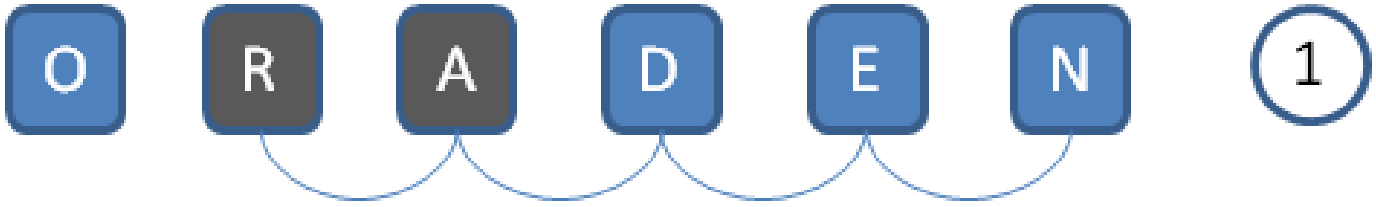
Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

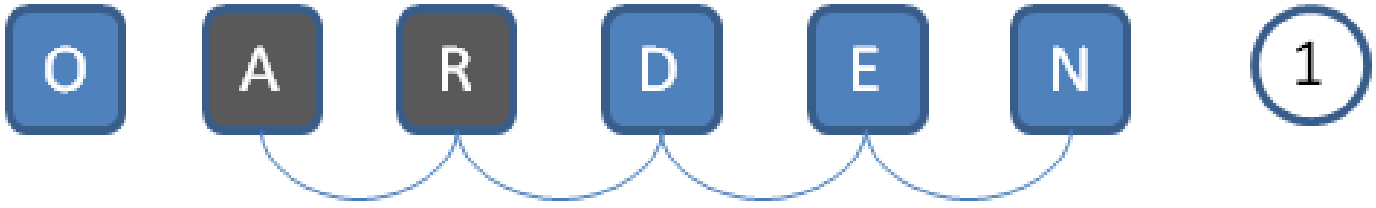


Se R > A ?

troca R com A

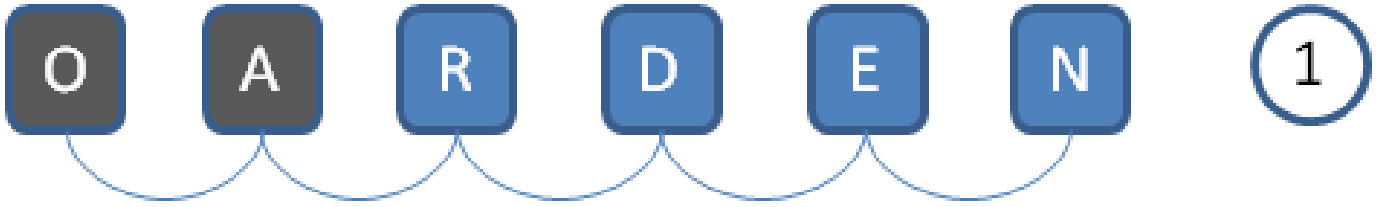
Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

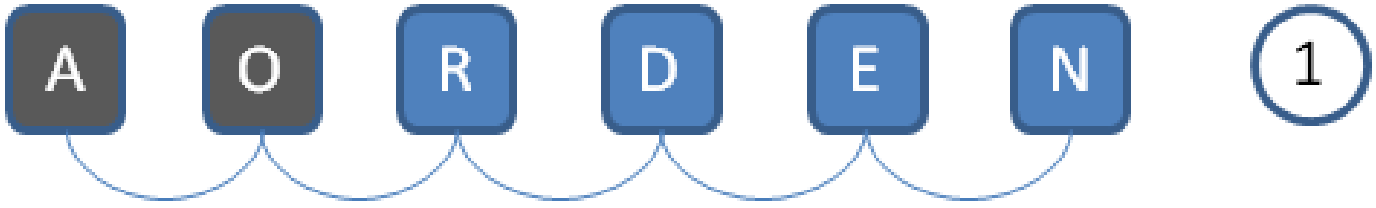


Se O > A ?

troca O com A

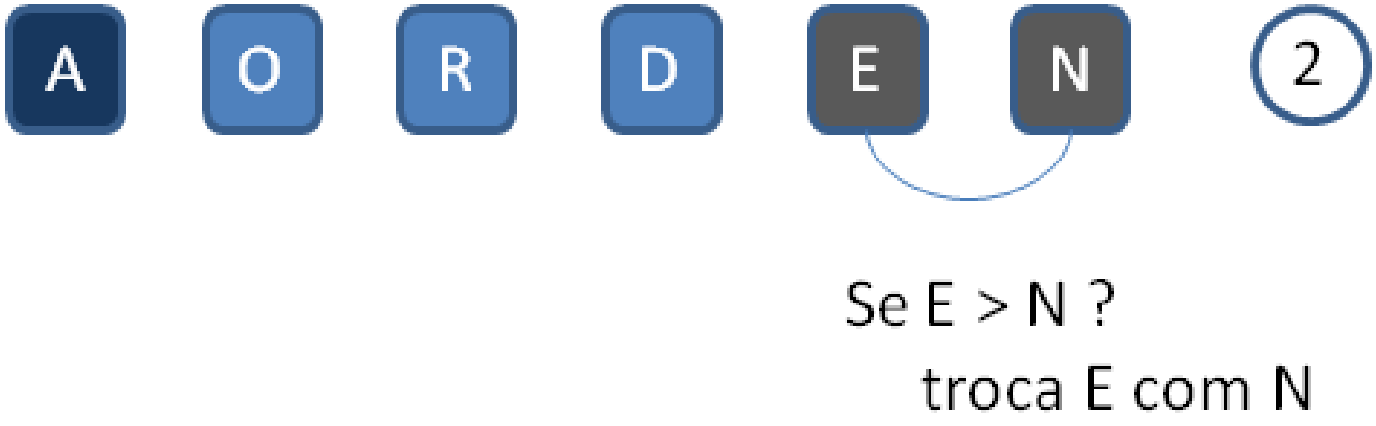
Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



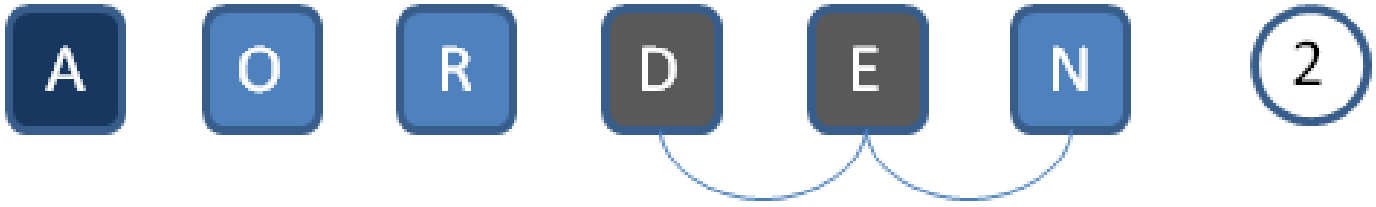
Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

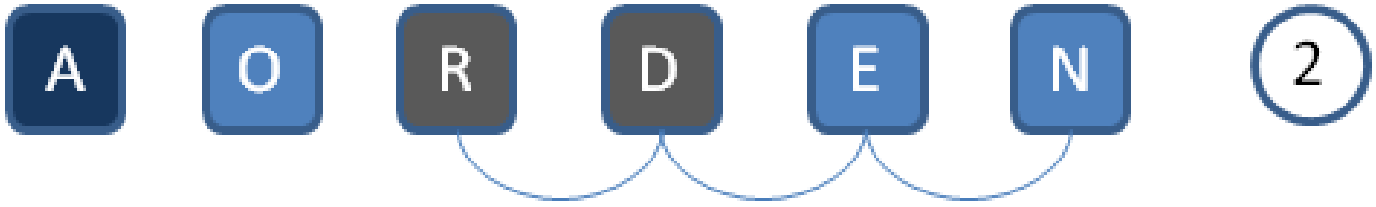


Se D > E ?

troca D com E

Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

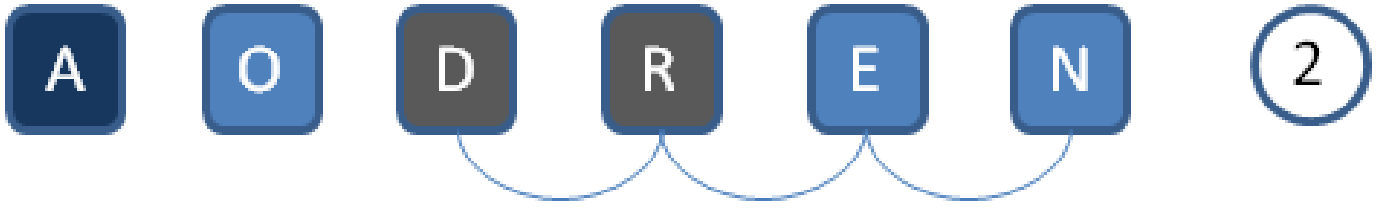


Se R > D ?

troca R com D

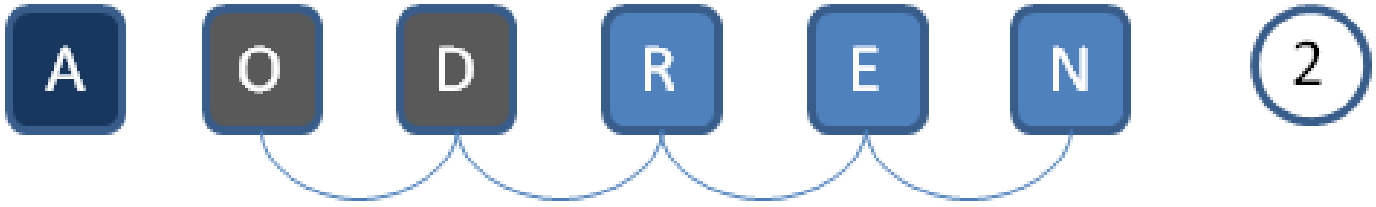
Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

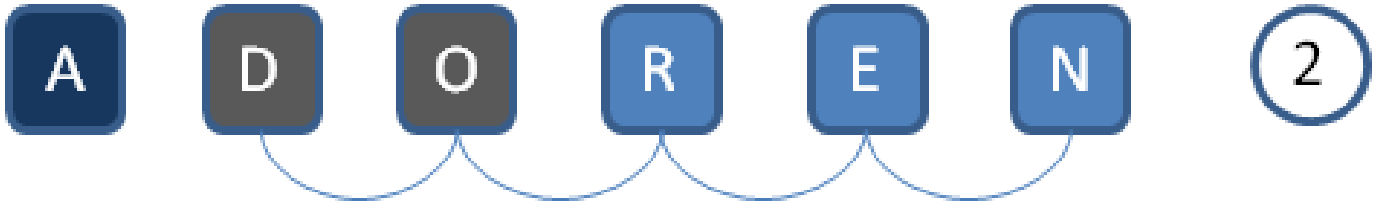


Se O > D ?

troca O com D

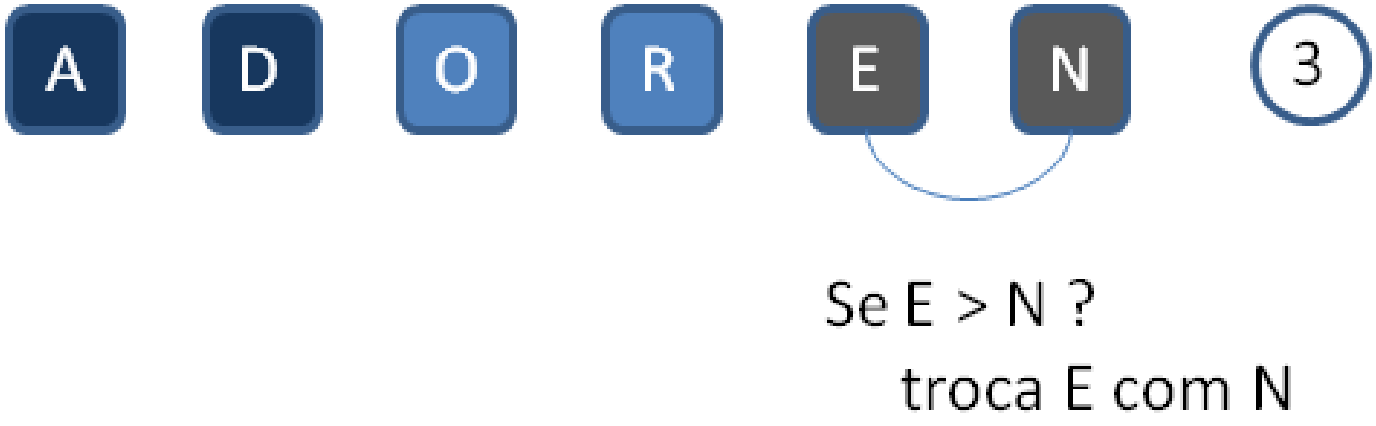
Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



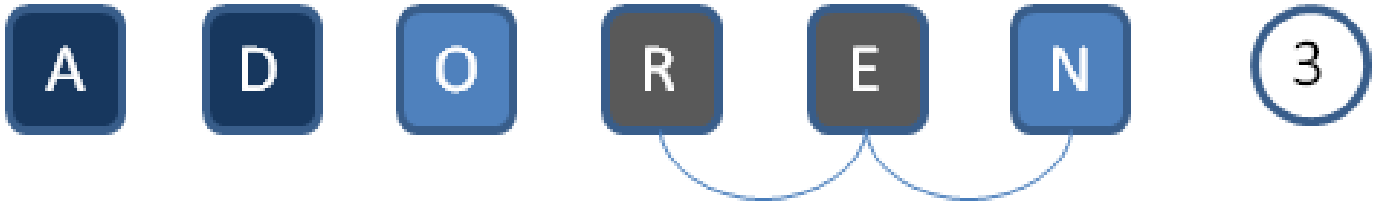
Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

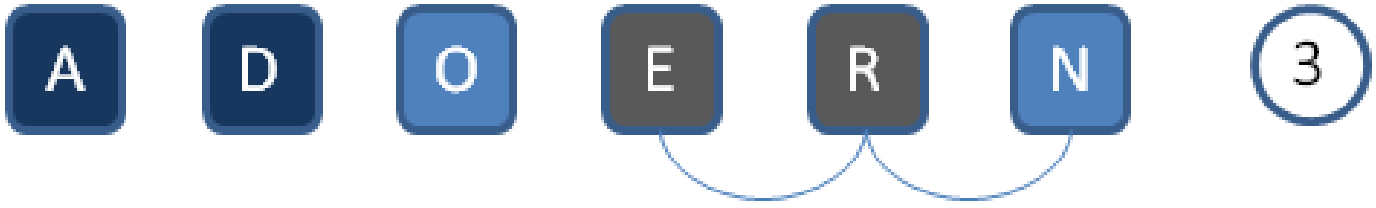


Se R > E ?

troca R com E

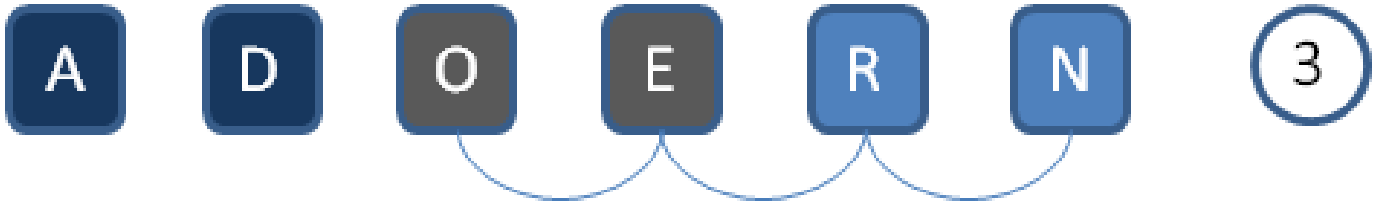
Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

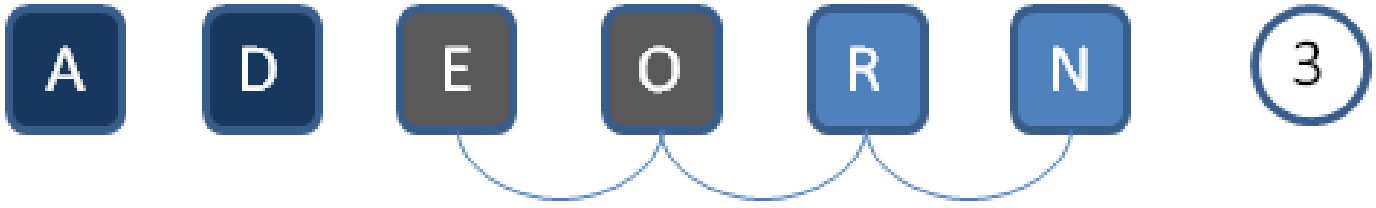


Se O > E ?

troca O com E

Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

Se R > N ?

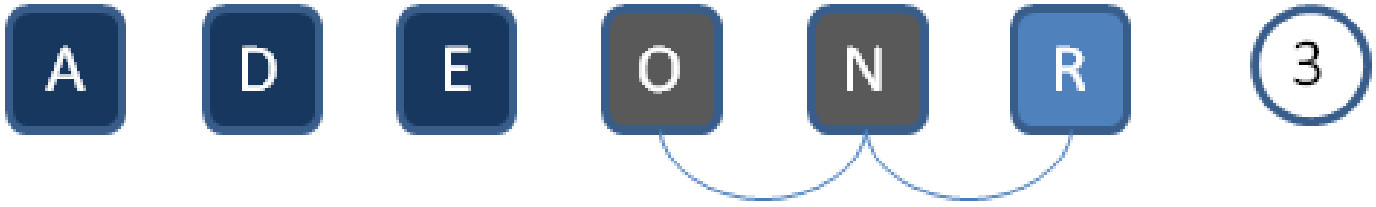
troca R com N

Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

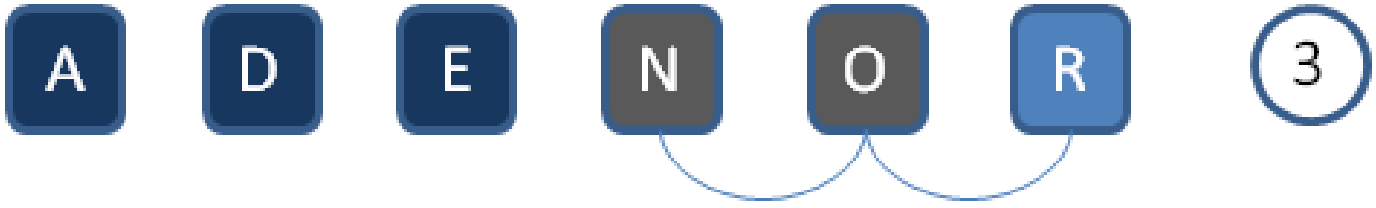


Se O > N ?

troca O com N

Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:



Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

Se O > R ?

troca O com R

Ordenação por BubbleSort (cont.)

O método é ilustrado embaixo:

Ordenação por BubbleSort (Método da bolha)

```
int main(){
    int a, b, temp, A[100], n, i;

    cout << "\n Indicar numero de elementos: ";
    cin >> n;
    cout << "\n Inserir " << n << " numeros ";

    for (i = 0; i < n; i++)
        cin >> A[i];
    ...
}
```

Ordenação por BubbleSort (Método da bolha)

```
int main(){
    ...
    for (a = 1; a < n; a++)
    {
        for (b = n - 1; b >= a; b--)
        {
            if (A[b - 1] > A[b])
            {      // intercambia elementos //
                temp = A[b - 1];
                A[b - 1] = A[b];
                A[b] = temp;
            }
        }
    }
    for (i = 0; i < n; i++)
        cout << A[i];
    return 0;
}
```

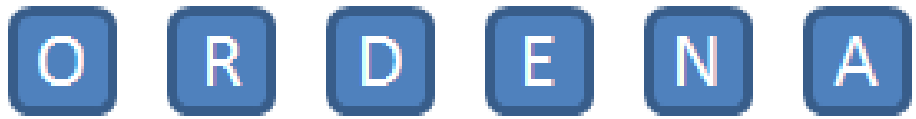
Ordenação por Seleção I

- Um dos algoritmos mais simples.
- Algoritmo:

1. Selecione o menor item do array
2. Troque-o com o item da primeira posição do vetor
3. Repita essas duas operações com os $n - 1$ items restantes, depois com os $n - 2$ items, até que reste apenas um elemento;

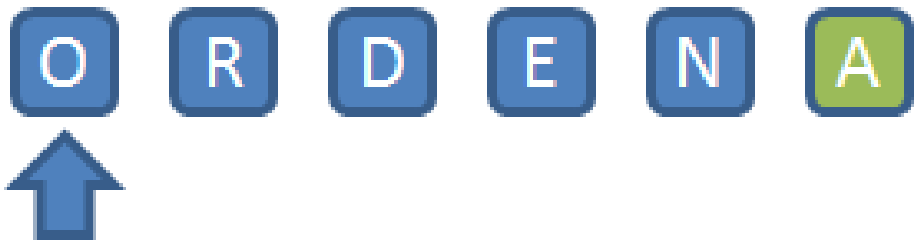
Ordenação por Seleção

- O método é ilustrado a continuação:



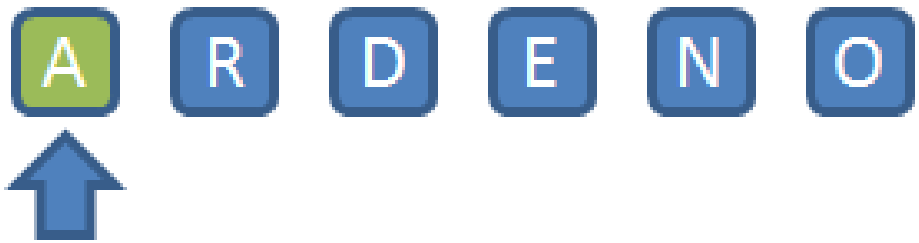
Ordenação por Seleção

- O método é ilustrado a continuação:



Ordenação por Seleção

- O método é ilustrado a continuação:

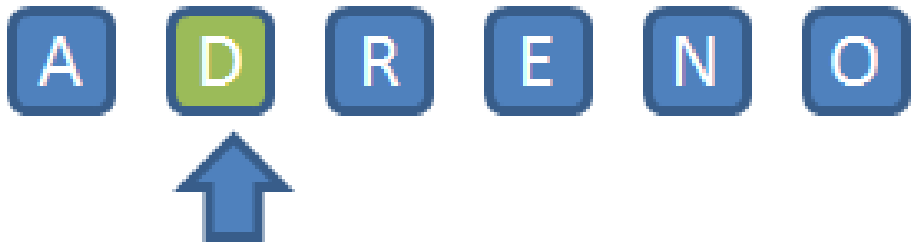


Ordenação por Seleção

- O método é ilustrado a continuação:

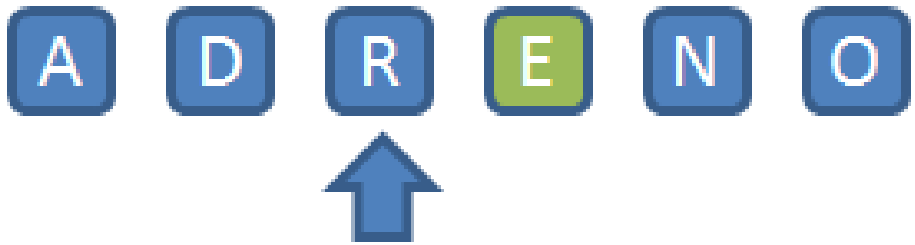
Ordenação por Seleção

- O método é ilustrado a continuação:



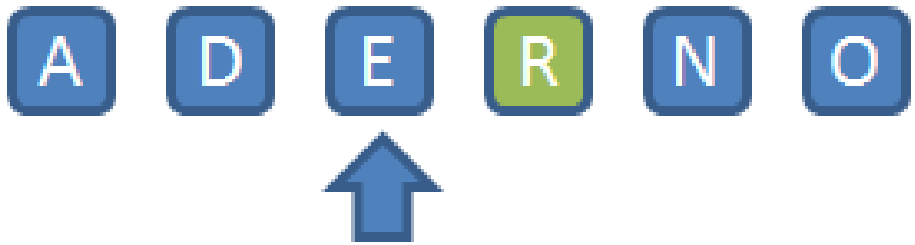
Ordenação por Seleção

- O método é ilustrado a continuação:



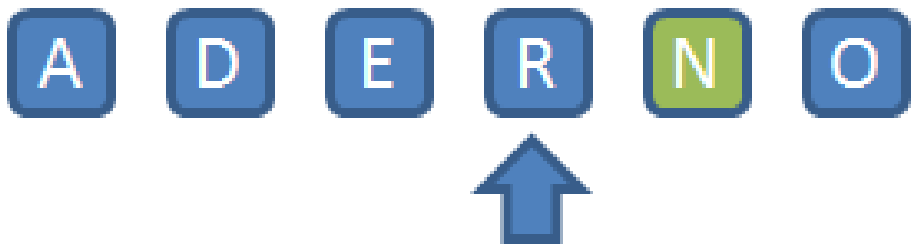
Ordenação por Seleção

- O método é ilustrado a continuação:



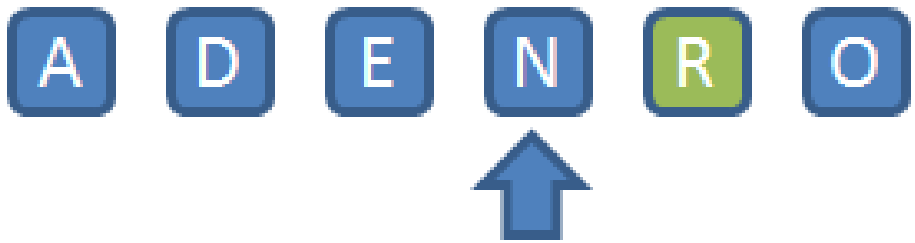
Ordenação por Seleção

- O método é ilustrado a continuação:



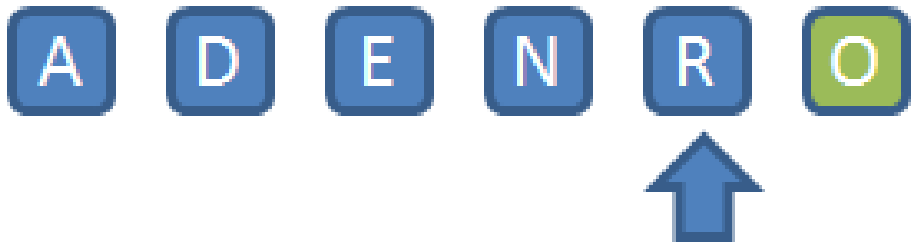
Ordenação por Seleção

- O método é ilustrado a continuação:



Ordenação por Seleção

- O método é ilustrado a continuação:



Ordenação por Seleção

- O método é ilustrado a continuação:

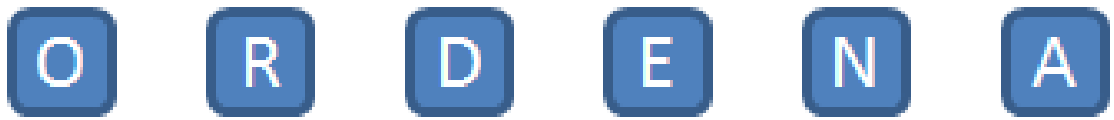
Ordenação por Inserção

- Um dos métodos mais simples de ordenação.
- Método preferido pelos jogadores de cartas.
- Algoritmo:

1. Para todos os elementos a partir de $i = 2$
   - 1.1 Selecione o i-ésimo item da secuencia
   - 1.2 Coloque-o no lugar apropriado na seqüência destino de acordo con o critério da ordenação

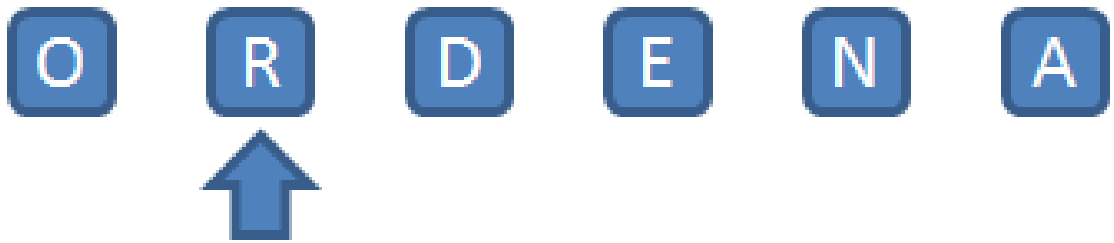
Ordenação por Inserção

- O método é ilustrado a contiução:



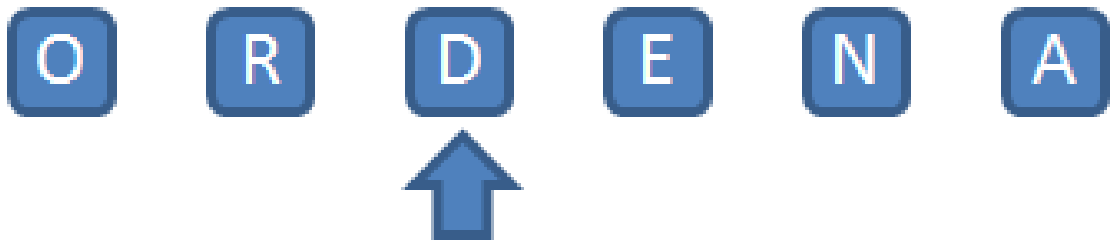
Ordenação por Inserção

- O método é ilustrado a contiução:



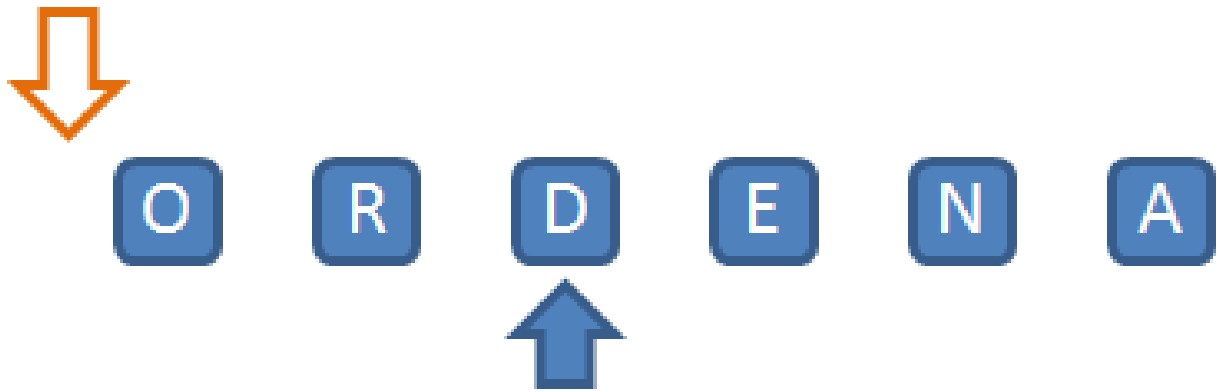
Ordenação por Inserção

- O método é ilustrado a contiução:



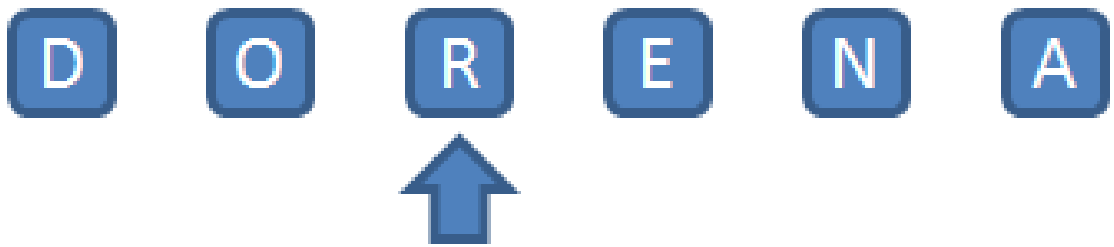
Ordenação por Inserção

- O método é ilustrado a contiução:



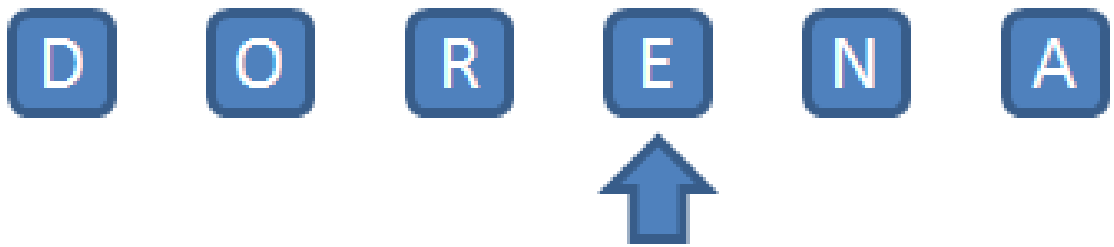
Ordenação por Inserção

- O método é ilustrado a contiução:



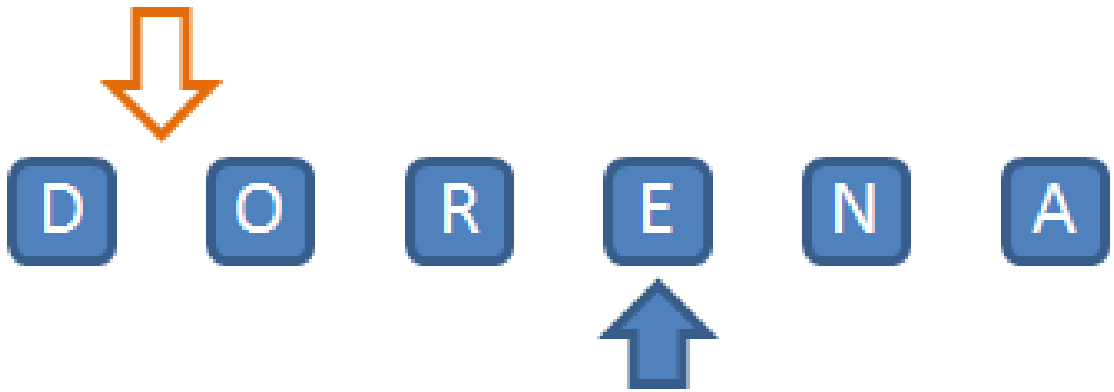
Ordenação por Inserção

- O método é ilustrado a contiução:



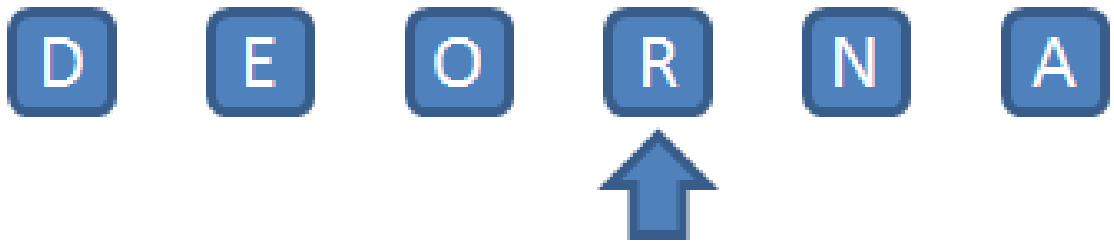
Ordenação por Inserção

- O método é ilustrado a contiução:



Ordenação por Inserção

- O método é ilustrado a contiução:



Ordenação por Inserção

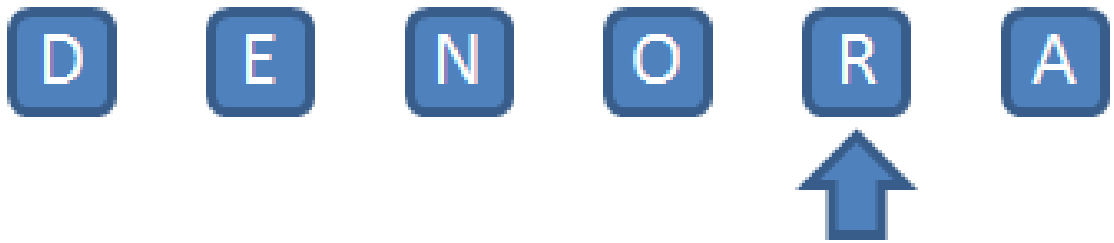
- O método é ilustrado a contiução:

Ordenação por Inserção

- O método é ilustrado a contiução:

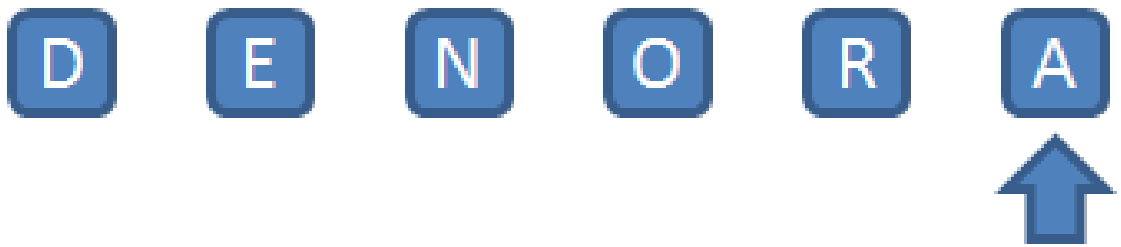
Ordenação por Inserção

- O método é ilustrado a contiução:



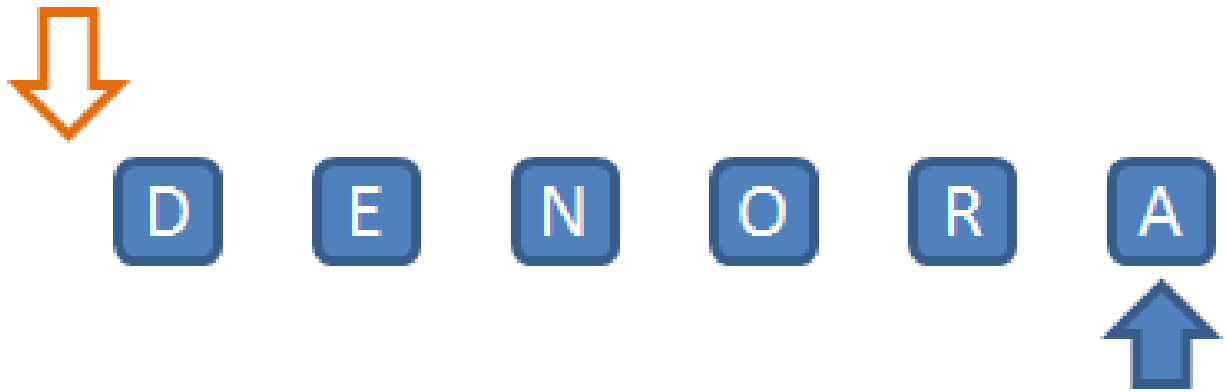
Ordenação por Inserção

- O método é ilustrado a contiução:



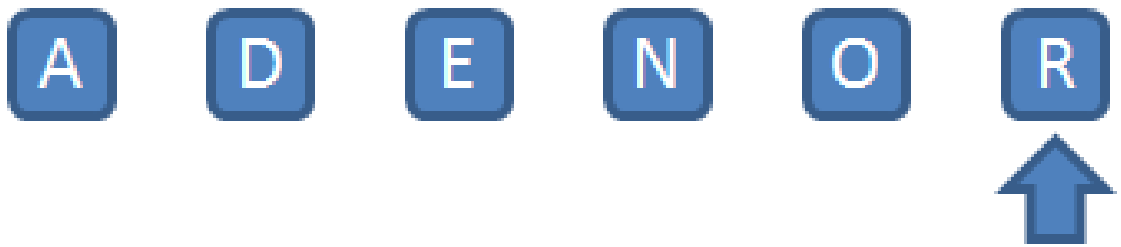
Ordenação por Inserção

- O método é ilustrado a contiução:

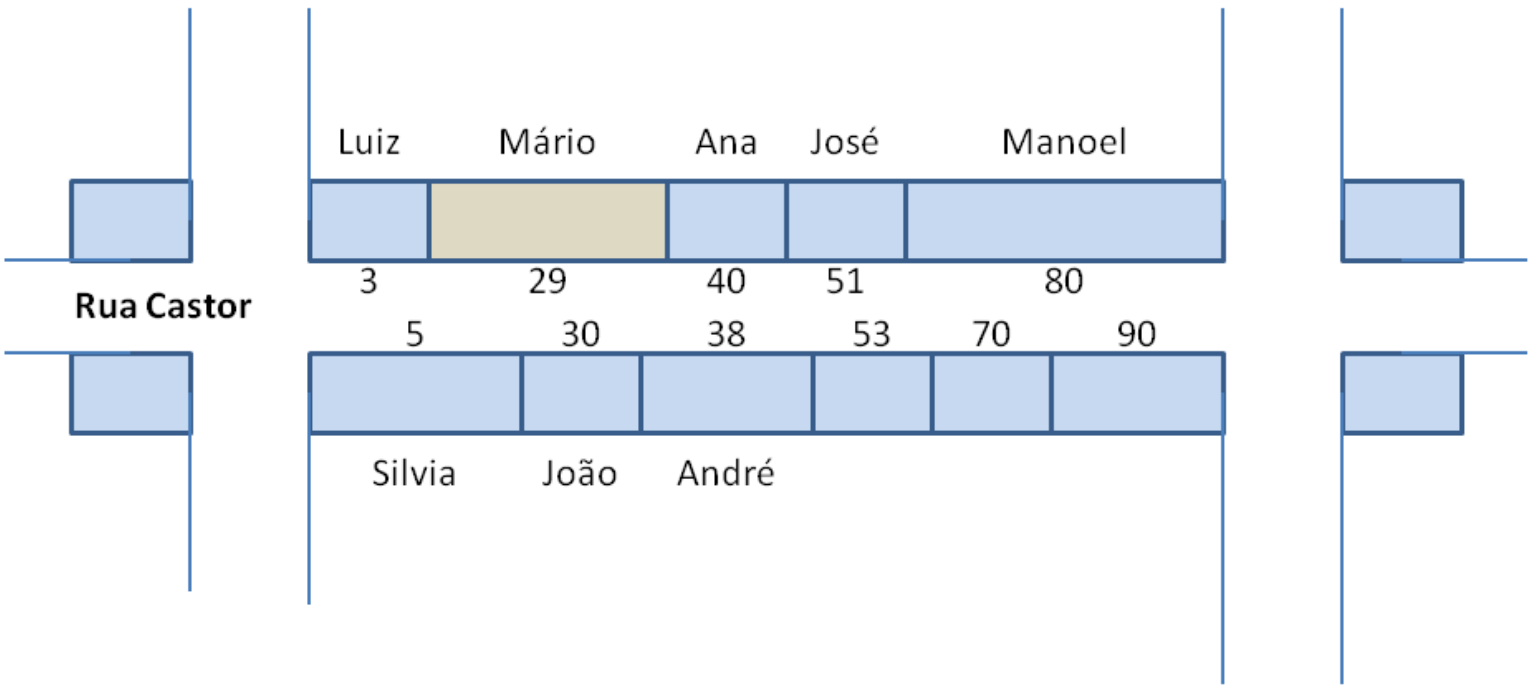


Ordenação por Inserção

- O método é ilustrado a contiução:

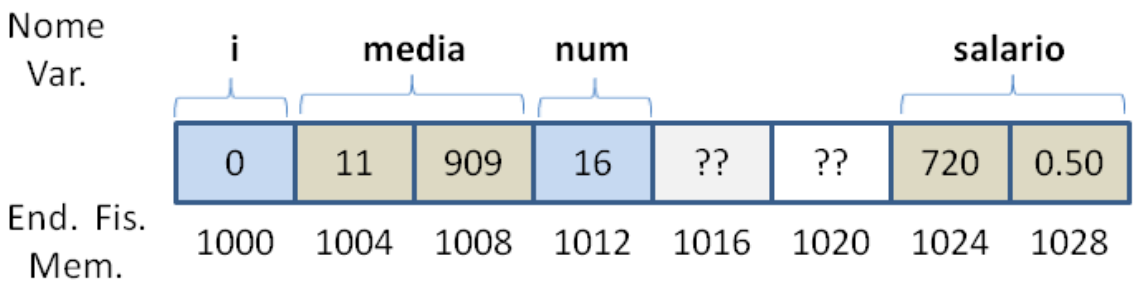


Vetores e Ponteiros I



Vetores e Ponteiros II

```
float salario;
int i = 0;
float media = 11909;
int num;
. . .
Num = 16
Salario = 7200.50;
```



Vetores e Ponteiros III

```
int main()
{
    int v[10], i;
    for (i = 0; i < 10; i++)
        v[i] = 10*i;
    // mostrando
    for (i = 0; i < 10; i++)
        cout << v[i];
    // mostrando 2
    for (i = 0; i < 10; i++)
        cout << *(v+i);
    return 0;
}
```

FIM