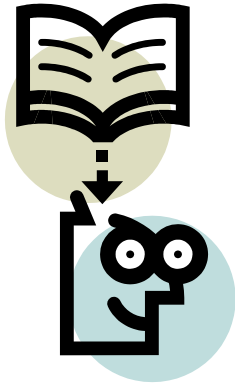
Algoritmo e Estrutura de Dados II
COM-112

Vanessa Souza

Universidade Federal de Itajubá – Instituto de Matemática e Computação

Ordenação

Problema



Ordenação

- Processo bastante utilizado na computação de uma estrutura de dados
- Ordenar significa colocar em ordem, segundo algum critério
- Alterar a ordem na qual os elementos de uma estrutura de dados aparece nessa estrutura
  - Rearranjar a estrutura

Ordenação

- A ordenação de uma coleção de valores é um dos problemas computacionais mais estudados.

- É importante porque muitos problemas definidos sobre coleções de valores se tornam fáceis, se os elementos da coleção estiverem ordenados.

Ordenação

5	3	1	2	4

Ordenação

5	3	1	2	4

1	2	3	4	5

Ordenação

5	3	1	2	4

5	4	3	2	1

Ordenação

5	3	1	2	4

2	4	1	3	5

Por que ordenar?

- Dados ordenados garantem uma melhor performance de pesquisa a uma ED
  - Recuperar Informação

- Busca Sequencial

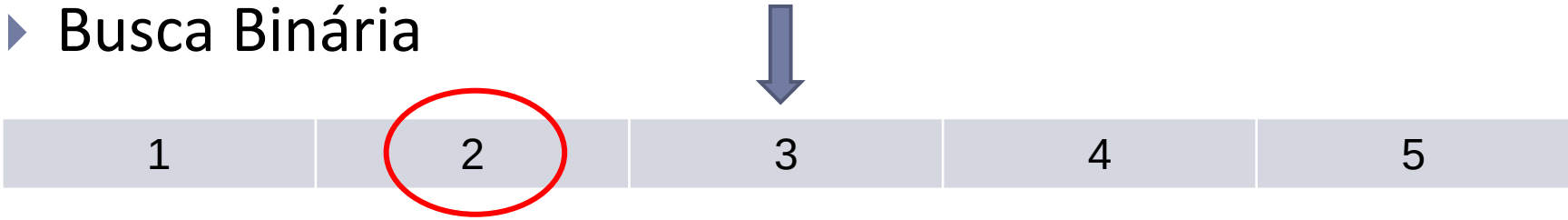
5	3	1	2	4

$$\Theta(n)$$

Por que ordenar?

- Dados ordenados garantem uma melhor performance de pesquisa a uma ED
  - Recuperar Informação

- Busca Binária



Por que ordenar?

- Dados ordenados garantem uma melhor performance de pesquisa a uma ED
  - Recuperar Informação

- Busca Binária

$$\Theta(\log_2 n)$$

Só é possível quando os dados estão ordenado

Ordenação

- Os algoritmos frequentemente usam a ordenação como sub-rotina.

- *"A complexidade da ordenação da ED não deve exceder a complexidade da computação a ser feita na ED sem o processo de ordenação"*

Ordenação

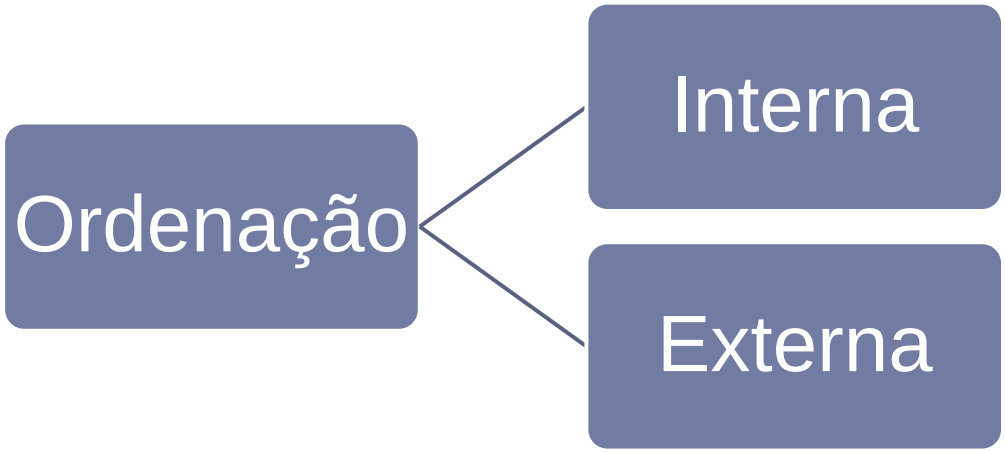
- Tipo de Dado
  - Em geral a ordenação é feita sobre um campo chave de um registro de dados
  - O restante do registro é chamado de dados satélite
  - Na prática, quando um algoritmo permuta uma chave, ele deve permutar também os dados satélites.

```
typedef struct vinho
{
    unsigned int codigo;
    unsigned int safra;
    char tipo[30];
    struct vinho *prox;
}vinho;
```

chave

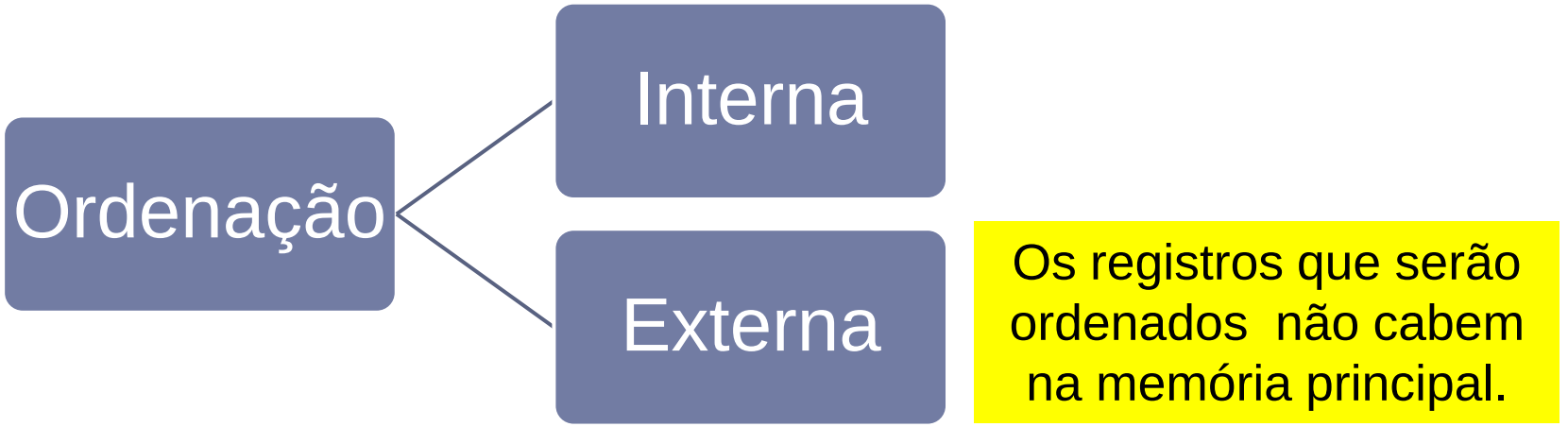
Dados Satélite

Classificação dos Métodos de Ordenação

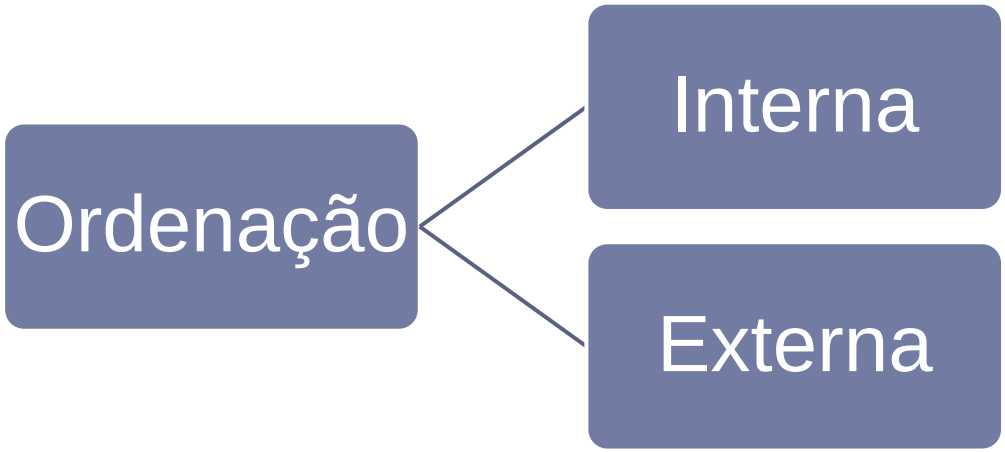


Os registros que serão ordenados cabem todos na memória principal.

Classificação dos Métodos de Ordenação

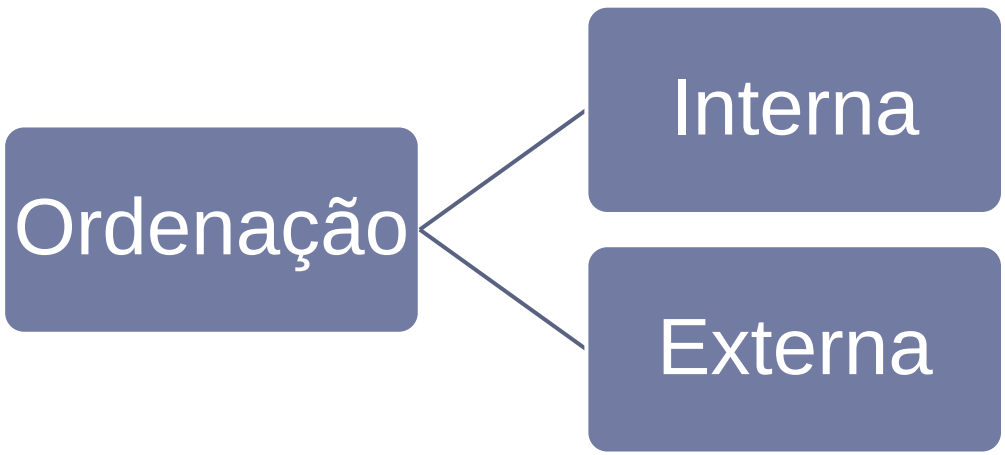


Classificação dos Métodos de Ordenação



Os registros que serão ordenados não cabem na memória principal.

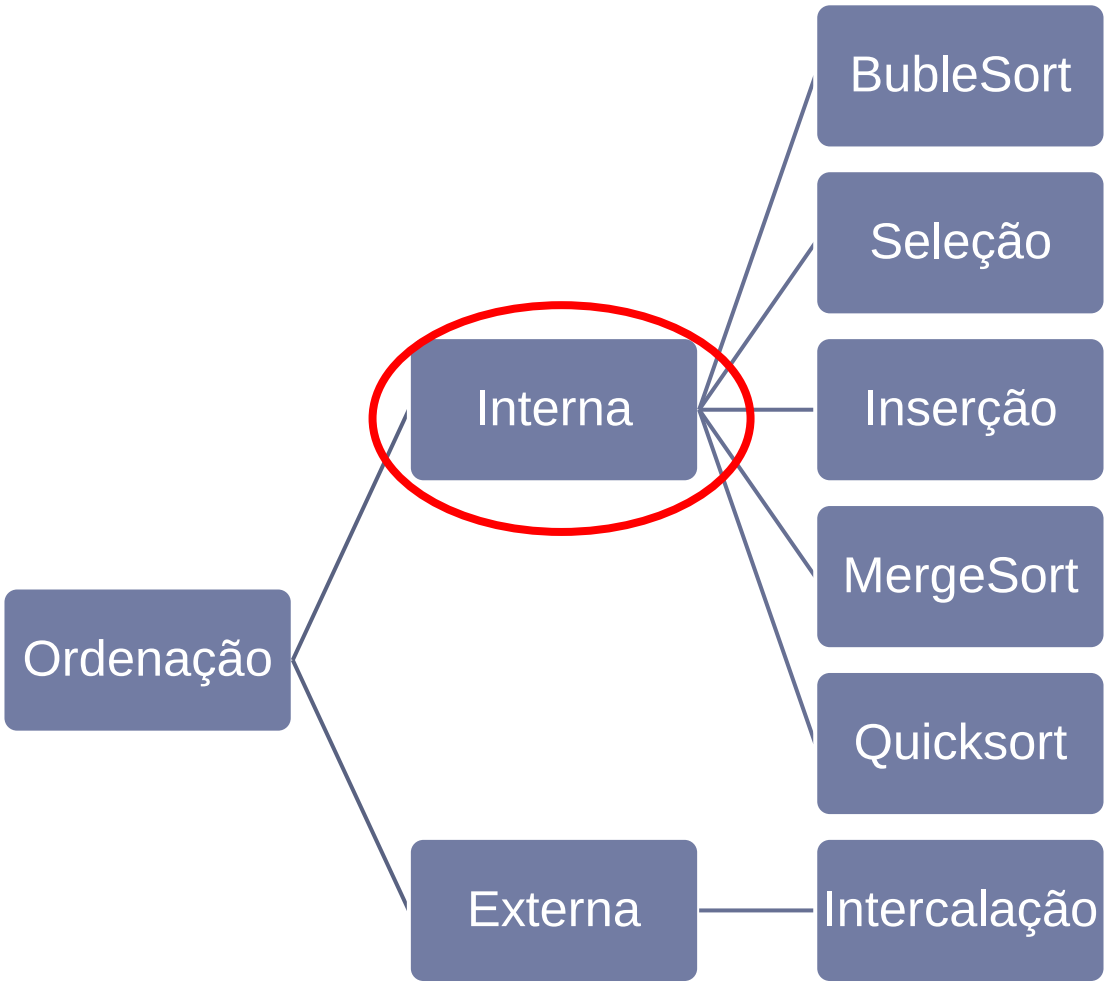
Classificação dos Métodos de Ordenação



Em um método de ordenação interna, qualquer registro pode ser imediatamente acessado.

Na ordenação externa, os registros são acessados sequencialmente ou em grandes blocos.

Classificação dos Métodos de Ordenação



Ordenação Interna

Ordenação Interna

- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.

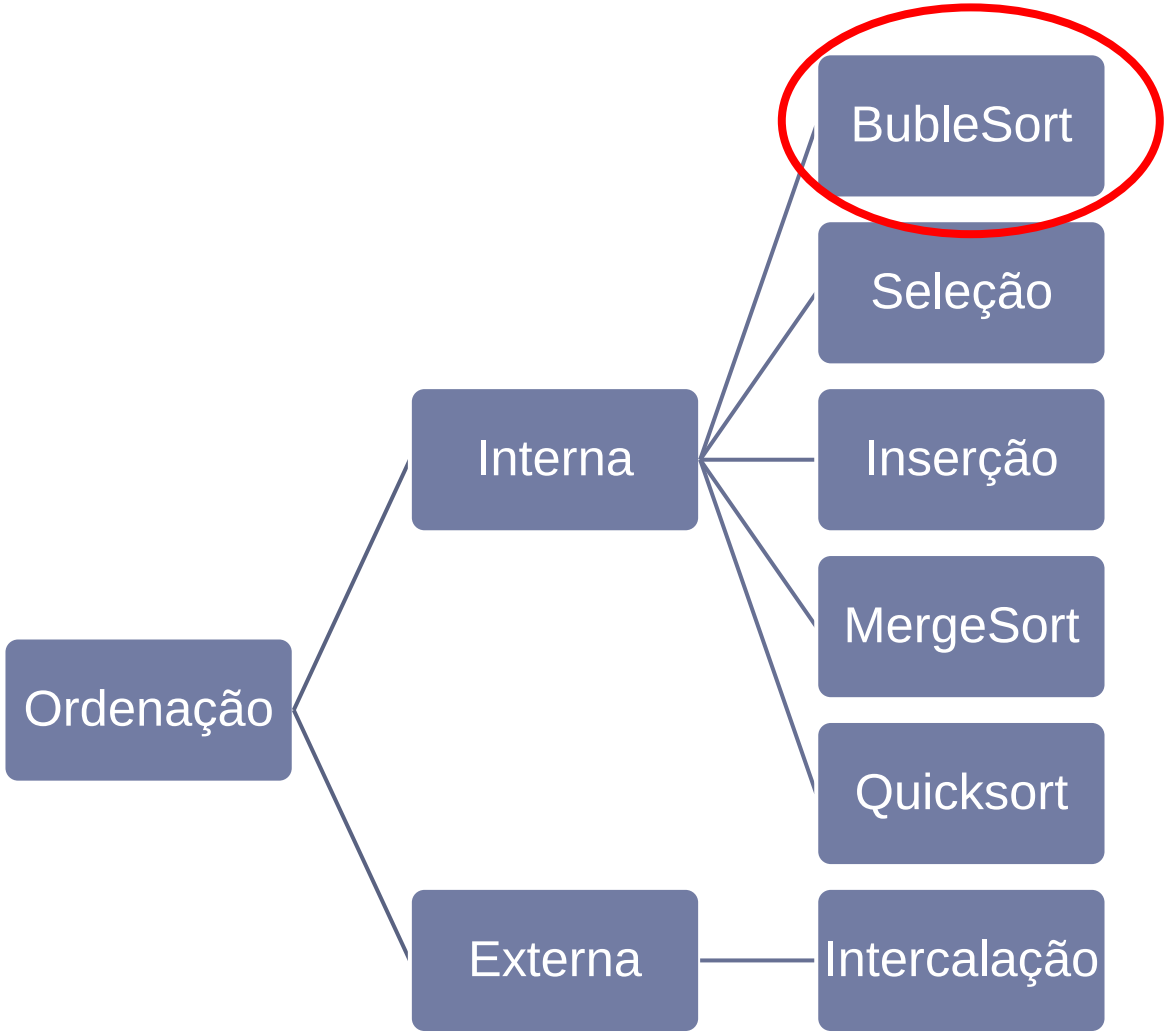
- Sendo *n* o número registros no arquivo, as medidas de complexidade relevantes são:
  - Número de comparações C(n) entre chaves.
  - Número de movimentações M(n) de itens do arquivo.

Ordenação Interna

- Nesse curso, veremos algoritmos de ordenação baseados em comparação de chaves apenas.

- Consideraremos que a coleção é implementada usando a estrutura de dados do tipo vetor.
  - Mas os algoritmos podem ser usados em estruturas do tipo lista também!

Classificação dos Métodos de Ordenação



Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

5	3	1	2	4

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

5	3	1	2	4

Compara o 5 com o 3.
Se for maior, troca

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	5	1	2	4

Compara o 5 com o 3.
Se for maior, troca

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	5	1	2	4

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	5	2	4

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	5	2	4

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	2	5	4

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	2	5	4

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	2	4	5

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	2	4	5

Ao fim da primeira passada, volta ao começo novamente

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	3	2	4	5

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	3	2	4	5

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	2	3	4	5

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	2	3	4	5

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	2	3	4	5

Ao fim da segunda passada, volta ao começo novamente

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	2	3	4	5

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	2	3	4	5

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	2	3	4	5

Repete o procedimento até o fim do vetor

Bolha

- Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	2	3	4	5

Repete o procedimento até o fim do vetor

Bolha

- O BubleSort repete a comparação entre pares *n* vezes.

1	2	3	4	5

Bolha

- Exercício
  - Demonstre passo a passo a ordenação do vetor X pelo método da Bolha.

X = [5,4,3,2,1]

Bolha

- Vamos programar??

Vamos programar??

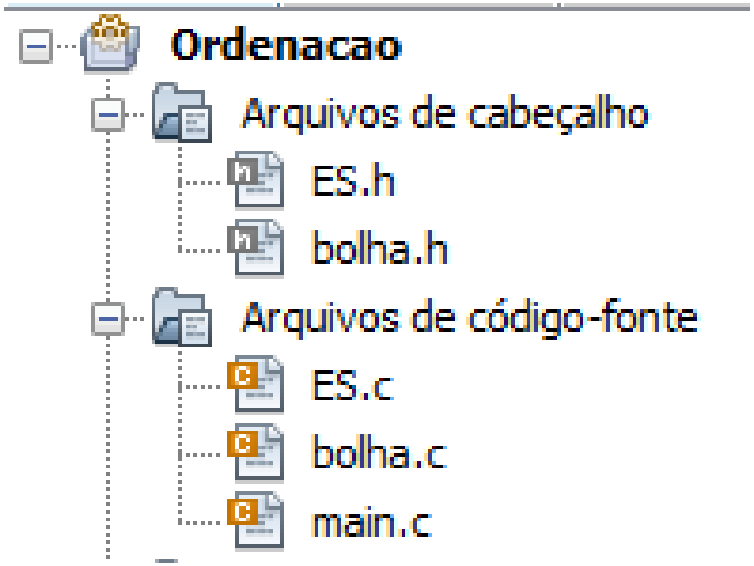
- Iremos agora montar uma estrutura modular de programação, com os seguintes requisitos:
  - O programa deverá apresentar o seguinte menu:

> - *Digite o método de ordenação que você deseja*
>   1. Bolha
>   2. Bolha Inteligente
>   3. Seleção
>   4. Inserção
>   5. MergeSort
>   6. QuickSort
> - *Digite o nome do arquivo com os dados a serem ordenados*

Vamos programar??

- Seu programa deve ter um módulo para leitura e escrita de dados
  - Todos os dados para ordenação devem vir de um arquivo. Depois de ordenados, os dados deverão ser gravados num arquivo com o mesmo nome de entrada, acrescido do sufixo '_ord'.
- A função 'main' deverá apenas apresentar o menu e chamar as demais funções
- Cada algoritmo de ordenação deverá ser implementado em um módulo (arquivos .c e .h)

Vamos programar??



Bolha

```
void bubleSort(int vet[], int tam)
{
    int i, j, aux;
    for (i=0; i<tam;i++)
    {
        for (j=0; j<tam-1; j++)
        {
            if (vet[j] > vet[j+1])
            {
                aux = vet[j];
                vet[j] = vet[j+1];
                vet[j+1] = aux;
            }
        }
    }
    return;
}
```

comparação

movimentação

Bolha

```
Buble sort tradicional

1  2  3  4  5

Executou 5 vezes

1  2  3  4  5
```

```
Buble sort tradicional

5  3  1  2  4

Executou 5 vezes

1  2  3  4  5
```

```
Buble sort tradicional

5  4  3  2  1

Executou 5 vezes

1  2  3  4  5
```

Bolha

```
void bubleSort(int vet[], int tam)
{
    int i, j, aux;
    for (i=0; i<tam;i++)
    {
        for (j=0; j<tam-1; j++)
        {
            if (vet[j] > vet[j+1])
            {
                aux = vet[j];
                vet[j] = vet[j+1];
                vet[j+1] = aux;
            }
        }
    }
    return;
}
```

Esse bolha pode ficar mais inteligente...

Bolha

```
void bubbleSortInteligente(int vet[], int tam)
{
    int i, j, aux;
    int troca = 0, quant = 0;
    for (i=0; i<tam;i++)
    {
        j=0;
        troca = 0;
        while (j<(tam-i))
        {
            if (vet[j] > vet[j+1])
            {
                aux = vet[j];
                vet[j] = vet[j+1];
                vet[j+1] = aux;
                troca = 1;
            }

            j++;
        }
        if (troca == 0)
        {
            printf("\nExecutou %d vezes", quant);
            return;
        }
        quant++;
    }
    printf("\nExecutou %d vezes\n", quant);
    return;
}
```

Bolha

```
Buble sort inteligente

1  2  3  4  5

Executou 1 vezes
1  2  3  4  5
```

```
Buble sort inteligente

5  3  1  2  4

Executou 3 vezes
1  2  3  4  5
```

```
Buble sort inteligente

5  4  3  2  1

Executou 5 vezes
1  2  3  4  5
```

Bolha

Calcular a complexidade assintótica do bolha inteligente

```
void bubbleSortInteligente(int vet[], int tam)
{
    int i, j, aux;
    int troca = 0, quant = 0;
    for (i=0; i<tam;i++)
    {
        j=0;
        troca = 0;
        while (j<(tam-i))
        {
            if (vet[j] > vet[j+1])
            {
                aux = vet[j];
                vet[j] = vet[j+1];
                vet[j+1] = aux;
                troca = 1;
            }

            j++;
        }
        if (troca == 0)
        {
            printf("\nExecutou %d vezes", quant);
            return;
        }
        quant++;
    }
    printf("\nExecutou %d vezes\n", quant);
    return;
}
```

Bolha

```
void bubbleSortInteligente(int vet[], int tam)
{
    int i, j, aux;
    int troca = 0, quant = 0;
    for (i=0; i<tam;i++)
    {
        j=0;
        troca = 0;
        while (j<(tam-i))
        {
            if (vet[j] > vet[j+1])
            {
                aux = vet[j];
                vet[j] = vet[j+1];
                vet[j+1] = aux;
                troca = 1;
            }

            j++;
        }
        if (troca == 0)
        {
            printf("\nExecutou %d vezes", quant);
            return;
        }
        quant++;
    }
    printf("\nExecutou %d vezes\n", quant);
    return;
}
```

Calcular a complexidade assintótica do bolha inteligente

$$\Theta(n^2)$$

Bolha

- Uma das principais vantagens do algoritmo BubbleSort é sua simplicidade.
- O algoritmo tem complexidade assintótica $O(n^2)$ e funciona muito bem em vetores quase ordenados.
- BubbleSort x BubbleSortInteligente

Para Casa

- Estudar e implementar o algoritmo de ordenação Bolha e Bolha Inteligente

- Assistir aos vídeos:
  - http://www.youtube.com/watch?v=_h3aMVBe8k8
  - http://www.youtube.com/watch?v=HEOoceedb7Y
  - Encontrar os erros (há erros?)
  - 0.5 pt extra
  - Em dupla
  - Próxima aula