



Técnicas de Programação 1

11ª parte

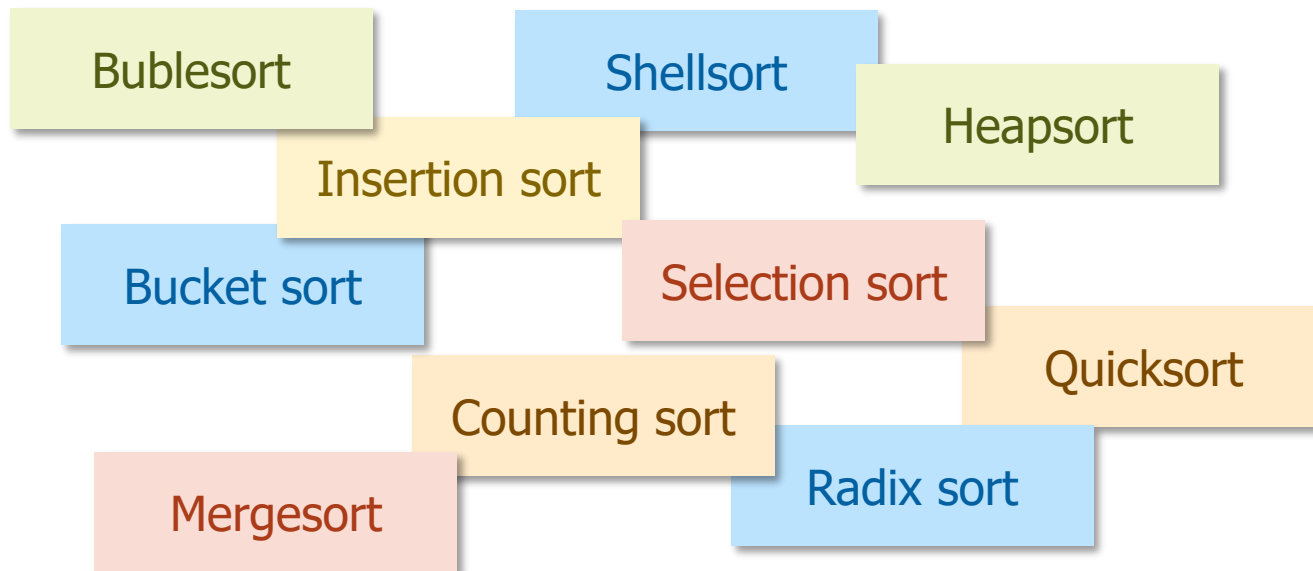
Ordenação e Busca

Prof. Jobson Massollar

jobson@uniriotec.br

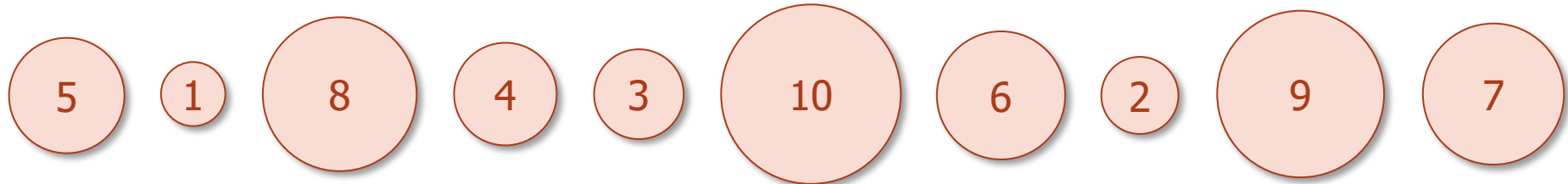


Existem diversos algoritmos que tratam o problema da ordenação:



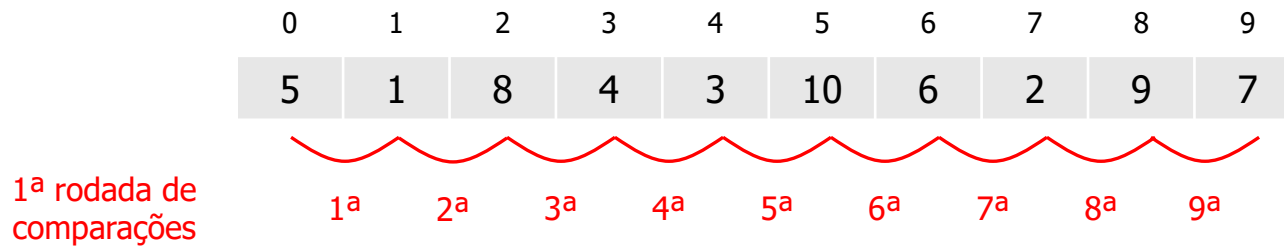


O Bubblesort é um algoritmo bastante simples de ordenação onde, a cada iteração, os **maiores elementos** vão sendo deslocados para as posições **mais altas** do vetor.





O Bubblesort ordena os elementos de um vetor comparando os elementos adjacentes dois a dois. Ele verifica se o 1º elemento é maior que o 2º. Se sim, ele troca um com o outro. Em seguida, compara o 2º com o 3º, depois o 3º com o 4º e assim por diante até o fim do vetor.





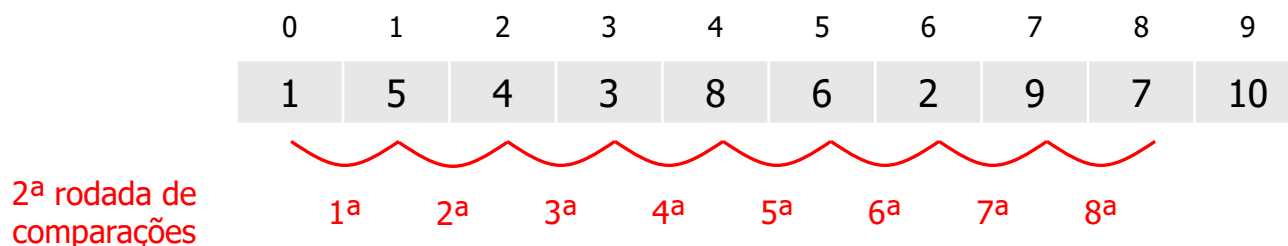
Bubblesort

0	1	2	3	4	5	6	7	8	9	
5	1	8	4	3	10	6	2	9	7	troca
1	5	8	4	3	10	6	2	9	7	não troca
1	5	8	4	3	10	6	2	9	7	troca
1	5	4	8	3	10	6	2	9	7	troca
1	5	4	3	8	10	6	2	9	7	não troca
1	5	4	3	8	10	6	2	9	7	troca
1	5	4	3	8	6	10	2	9	7	troca
1	5	4	3	8	6	2	10	9	7	troca
1	5	4	3	8	6	2	9	10	7	troca
1	5	4	3	8	6	2	9	7	10	

Ao final da 1ª rodada o maior valor do vetor estará na última posição.



O processo de comparação é repetido, mas descartando-se o **último elemento** do vetor, porque ele é o maior de todos e já está na posição correta.





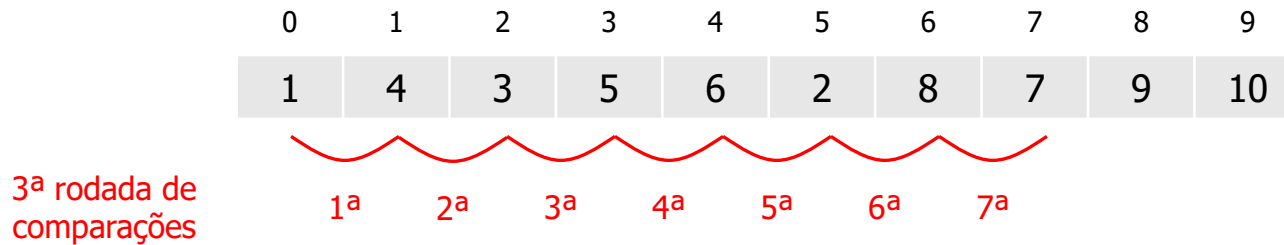
Bubblesort

0	1	2	3	4	5	6	7	8	9	
1	5	4	3	8	6	2	9	7	10	não troca
1	5	4	3	8	6	2	9	7	10	troca
1	4	5	3	8	6	2	9	7	10	troca
1	4	3	5	8	6	2	9	7	10	não troca
1	4	3	5	6	8	2	9	7	10	não troca
1	4	3	5	6	8	2	9	7	10	troca
1	4	3	5	6	2	8	9	7	10	não troca
1	4	3	5	6	2	8	9	7	10	troca
1	4	3	5	6	2	8	7	9	10	

Ao final da 2ª rodada os dois maiores valores do vetor estarão ordenados nas 2 últimas posições.



Novamente o processo de comparação é repetido, mas descartando-se o **dois últimos elementos** do vetor, porque eles são os dois maiores e já estão nas posições corretas.





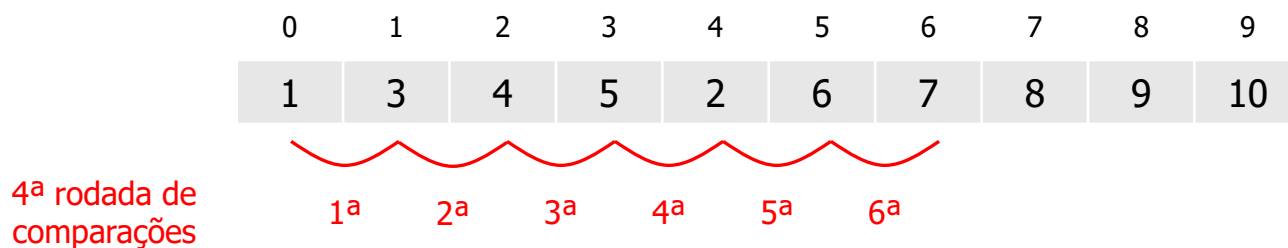
Bubblesort

0	1	2	3	4	5	6	7	8	9	
1	4	3	5	6	2	8	7	9	10	não troca
1	4	3	5	6	2	8	7	9	10	troca
1	3	4	5	6	2	8	7	9	10	não troca
1	3	4	5	6	2	8	7	9	10	não troca
1	3	4	5	6	2	8	7	9	10	troca
1	3	4	5	2	6	8	7	9	10	não troca
1	3	4	5	2	6	8	7	9	10	troca
1	3	4	5	2	6	7	8	9	10	

Ao final da 3ª rodada os três maiores valores do vetor estarão ordenados nas 3 últimas posições.



O processo de comparação continua descartando os últimos elementos a cada rodada.





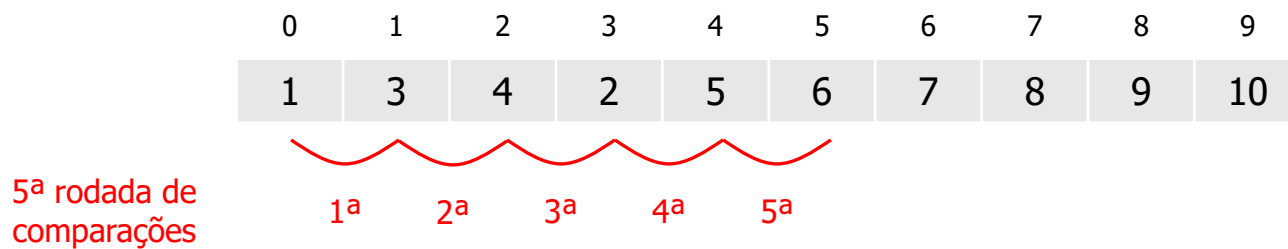
Bubblesort

0	1	2	3	4	5	6	7	8	9	
1	3	4	5	2	6	7	8	9	10	não troca
1	3	4	5	2	6	7	8	9	10	não troca
1	3	4	5	2	6	7	8	9	10	não troca
1	3	4	5	2	6	7	8	9	10	troca
1	3	4	2	5	6	7	8	9	10	troca
1	3	4	2	5	6	7	8	9	10	não troca
1	3	4	2	5	6	7	8	9	10	

Ao final da 4ª rodada os quatro maiores valores do vetor estarão ordenados nas 4 últimas posições.



Bubblesort





Bubblesort

0	1	2	3	4	5	6	7	8	9	
1	3	4	2	5	6	7	8	9	10	não troca
1	3	4	5	2	6	7	8	9	10	não troca
1	3	4	5	2	6	7	8	9	10	não troca
1	3	4	5	2	6	7	8	9	10	troca
1	3	4	2	5	6	7	8	9	10	não troca
1	3	4	2	5	6	7	8	9	10	

Ao final da 5ª rodada os cinco maiores valores do vetor estarão ordenados nas 5 últimas posições.



6ª rodada de
comparações





0	1	2	3	4	5	6	7	8	9	
1	3	4	2	5	6	7	8	9	10	não troca
1	3	4	2	5	6	7	8	9	10	não troca
1	3	4	2	5	6	7	8	9	10	troca
1	3	2	4	5	6	7	8	9	10	não troca
1	3	2	4	5	6	7	8	9	10	

Ao final da 6ª rodada os seis maiores valores do vetor estarão ordenados nas 6 últimas posições.



7ª rodada de
comparações

0	1	2	3	4	5	6	7	8	9
1	3	2	4	5	6	7	8	9	10

1ª 2ª 3ª



Bubblesort

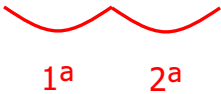
0	1	2	3	4	5	6	7	8	9	
1	3	2	4	5	6	7	8	9	10	não troca
1	3	2	4	5	6	7	8	9	10	troca
1	2	3	4	5	6	7	8	9	10	não troca
1	2	3	4	5	6	7	8	9	10	

Ao final da 7ª rodada os sete maiores valores do vetor estarão ordenados nas 7 últimas posições.



0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10

8ª rodada de
comparações





Bubblesort

0	1	2	3	4	5	6	7	8	9	
1	2	3	4	5	6	7	8	9	10	não troca
1	2	3	4	5	6	7	8	9	10	não troca
1	2	3	4	5	6	7	8	9	10	

Como não houve nenhuma troca nessa rodada o vetor está ordenado!



Do exemplo anterior podemos depreender que:

1. Em um vetor com n posições podem ocorrer, no máximo, $n-1$ rodadas de comparação;
2. A 1ª rodada de comparação tem $n-1$ comparações, a 2ª rodada tem $n-2$ comparações, e assim por diante até a última rodada que tem apenas 1 comparação;
3. Se em uma rodada de comparação **não ocorrerem trocas**, o processo de ordenação pode ser **interrompido**.



ALGORITMO Bubblesort

VARIÁVEIS

K, Trocou, Aux

INICIO

$K \leftarrow \text{tamanho do vetor} - 1$

Trocou \leftarrow verdadeiro

ENQUANTO $K > 0$ E Trocou FAÇA

Trocou \leftarrow falso

PARA $I \leftarrow 0$ ATÉ $K-1$ FAÇA

SE $\text{Vetor}[I] > \text{Vetor}[I+1]$ ENTÃO

Aux \leftarrow Vetor[I]

Vetor[I] \leftarrow Vetor[I+1]

Vetor[I+1] \leftarrow Aux

Trocou \leftarrow verdadeiro

FIM-SE

FIM-PARA

$K \leftarrow K - 1$

FIM-ENQUANTO

FIM

O loop externo controla a quantidade de rodadas e se houve troca.



ALGORITMO Bubblesort

VARIÁVEIS

K, Trocou, Aux

INICIO

$K \leftarrow \text{tamanho do vetor} - 1$

Trocou \leftarrow verdadeiro

ENQUANTO $K > 0$ E Trocou FAÇA

Trocou \leftarrow falso

PARA $I \leftarrow 0$ **ATÉ** $K-1$ **FAÇA**

SE $\text{Vetor}[I] > \text{Vetor}[I+1]$ ENTÃO

Aux \leftarrow Vetor[I]

Vetor[I] \leftarrow Vetor[I+1]

Vetor[I+1] \leftarrow Aux

Trocou \leftarrow verdadeiro

FIM-SE

FIM-PARA

$K \leftarrow K - 1$

FIM-ENQUANTO

FIM

O loop interno realiza as comparações e trocas de uma rodada.



A implementação em C pode ser feita com **while** ou **for**:

```
void bubble_sort(int v[], int n) {
    bool trocou = true;
    int k = n-1;

    while (k > 0 && trocou)
    {
        trocou = false;
        for (int i = 0; i < k; i++)
            if (v[i] > v[i+1]) {
                int aux = v[i+1];
                v[i+1] = v[i];
                v[i] = aux;
                trocou = true;
            }
        k--;
    }
}
```

```
void bubble_sort(int v[], int n) {
    bool trocou = true;

    for (int k = n-1; k > 0 && trocou; k--)
    {
        trocou = false;
        for (int i = 0; i < k; i++)
            if (v[i] > v[i+1]) {
                int aux = v[i+1];
                v[i+1] = v[i];
                v[i] = aux;
                trocou = true;
            }
    }
}
```



11.1) Leia n nomes e, em seguida, imprima-os em ordem crescente. Dicas:

- a) Crie uma matriz para armazenar as palavras, onde cada linha armazena uma palavra e a quantidade máxima de caracteres é a quantidade de colunas-1.

	0	1	2	3	4	5	6	7	8	9
0	J	O	A	O	\0					
1	A	N	A	\0						
2	R	E	N	A	T	O	\0			
3	B	E	A	T	R	I	Z	\0		
4	P	E	D	R	O	\0				
5	C	A	R	L	O	S	\0			

- b) Adapte o algoritmo do Bubblesort e use a função `strcmp` para comparar duas strings e definir a sua ordem.



O problema da busca pode ser definido como:

Dada uma coleção com n elementos deseja-se saber se um dado elemento x pertence a essa coleção.

Esse é um dos problemas mais básicos da Computação e pode ser encontrado em diversas situações do dia a dia:

- ✓ Saber se um aluno está matriculado em uma turma.
- ✓ Saber se um produto está disponível no estoque de uma loja.
- ✓ Saber se uma palavra pertence a um dicionário.



Serão vistos dois algoritmos de busca assumindo que a coleção de dados está armazenada em um **vetor**.

- ✓ Busca Sequencial
- ✓ Busca Binária



O algoritmo de busca mais trivial é a Busca Sequencial, já apresentada quando estudamos vetores:

- ✓ Percorra o vetor sequencialmente, da primeira posição até a última.
- ✓ Se o valor procurado está na posição i do vetor, então informe que encontrou e pare a busca.
- ✓ Se chegar ao final do vetor sem sucesso, então informe que o valor procurado não existe.



Busca Sequencial

```
bool busca(int v[], int n, int k)
{
    for (int i = 0; i < n; i++)
        if (v[i] == k)
            return true;

    return false;
}
```

v é o vetor, **n** é o tamanho do vetor e **k** é o valor a ser procurado.



Busca Sequencial

```
bool busca(int v[], int n, int k)
{
    for (int i = 0; i < n; i++)
        if (v[i] == k)
            return true;

    return false;
}
```

O algoritmo varre **sequencialmente** o vetor e verifica, elemento a elemento, se este é o valor desejado.

Se encontrar retorna **true**.

Se não encontrar retorna **false**.



Busca Sequencial

```
bool busca(int v[], int n, int k)
{
    for (int i = 0; i < n; i++)
        if (v[i] == k)
            return true;

    return false;
}
```

Ok, mas esse algoritmo não informa em que posição está o valor **k** no vetor **v**!

Então podemos fazer uma pequena modificação!



Foto criada por freepik - br.freepik.com



Busca Sequencial

```
int busca(int v[], int n, int k)
{
    for (int i = 0; i < n; i++)
        if (v[i] == k)
            return i;

    return -1;
}
```

O algoritmo varre **sequencialmente** o vetor e verifica, elemento a elemento, se este é o valor desejado.

Se encontrar retorna a posição de **k** em **v** (0, 1, 2, ...).

Se não encontrar retorna **-1**.



Busca Sequencial

```
int busca(int v[], int n, int k)
{
    for (int i = 0; i < n; i++)
        if (v[i] == k)
            return i;

    return -1;
}
```

Note que o valor **-1** retornado pela função quando $k \notin v$ é uma convenção, pois -1 é uma posição **inválida**.



Foto criada por [katemangostar - br.freepik.com](https://br.freepik.com)



Ok, mas se o vetor for muito grande, digamos um milhão de elementos, o algoritmo terá que fazer um milhão de comparações, na pior das hipóteses. Isso não é ruim?

Sim. Existe uma forma mais eficiente de fazer essa busca, que é a Busca Binária. Entretanto, o vetor precisa estar ordenado!



Foto criada por freepik - br.freepik.com





O algoritmo da Busca Binária pode ser resumido da seguinte forma:

1. Se índice final $<$ índice inicial, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$.
3. Se o valor procurado está na posição central, então informe que encontrou e pare a busca.
4. Se o valor procurado é menor que o elemento da posição central então repita a busca na primeira metade do vetor.
5. Se o valor procurado é maior que o elemento da posição central então repita a busca na segunda metade do vetor.



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início									fim



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35




↑ início

↑ fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.






Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início									fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$






Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início									fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 11, então informe que encontrou e pare a busca.






Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início					fim				

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 11, então informe que encontrou e pare a busca.
4. Se o valor procurado < 11 , então repita a busca na primeira metade do vetor (início até central-1).



Exemplo 1: buscar 21


0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início					fim				


1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 11, então informe que encontrou e pare a busca.
4. Se o valor procurado < 11 , então repita a busca na primeira metade do vetor (início até central-1).
5. Se o valor procurado > 11 , então repita a busca na segunda metade do vetor (central+1 até fim).



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35




 início

 fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.



Exemplo 1: buscar 21




0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
					início				fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35




 início   fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 26, então informe que encontrou e pare a busca.



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35



 início   fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 26, então informe que encontrou e pare a busca.
4. Se o valor procurado < 26, então repita a busca na primeira metade do vetor (início até central-1).



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35




 
início fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35




 
início fim


1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35




 
início fim


1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 14, então informe que encontrou e pare a busca.



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35




 
início fim


1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 14, então informe que encontrou e pare a busca.
4. Se o valor procurado < 14, então repita a busca na primeira metade do vetor (início até central-1).



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35


 
início fim


1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 14, então informe que encontrou e pare a busca.
4. Se o valor procurado < 14 , então repita a busca na primeira metade do vetor (início até central-1).
5. Se o valor procurado > 14 , então repita a busca na segunda metade do vetor (central+1 até fim).



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35


início & fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35



início & fim



1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$



Exemplo 1: buscar 21

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35

↑
início & fim

↑

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 21, então informe que encontrou e pare a busca.



Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35




↑ início

↑ fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.






Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início									fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$



Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início									fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 11, então informe que encontrou e pare a busca.





Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
↑				↑					↑
início									fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 11, então informe que encontrou e pare a busca.
4. Se o valor procurado < 11 , então repita a busca na primeira metade do vetor (início até central-1).






Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
 início			 fim						

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.






Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início			fim						

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$






Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início			fim						

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 5, então informe que encontrou e pare a busca.






Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início			fim						

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 5, então informe que encontrou e pare a busca.
4. Se o valor procurado < 5, então repita a busca na primeira metade do vetor (início até central-1).



Exemplo 2: buscar 6



0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35
									
início			fim						

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 5, então informe que encontrou e pare a busca.
4. Se o valor procurado < 5, então repita a busca na primeira metade do vetor (início até central-1).
5. Se o valor procurado > 5, então repita a busca na segunda metade do vetor (central+1 até fim).



Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35

 
início fim

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.



Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35

↑ ↑
início fim

↑

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$



Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35

↑ ↑
início fim

↑

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 7, então informe que encontrou e pare a busca.



Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35

↑ ↑
início fim

↑

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.
2. Encontre a posição central do vetor: $(\text{início} + \text{fim}) / 2$
3. Se o valor procurado = 7, então informe que encontrou e pare a busca.
4. Se o valor procurado < 7, então repita a busca na primeira metade do vetor (início até central-1).



Exemplo 2: buscar 6

0	1	2	3	4	5	6	7	8	9
2	5	7	8	11	14	21	26	32	35

↑ ↑
fim início

1. Se $\text{fim} < \text{início}$, então informe que não existe e pare a busca.



Busca Binária

```
int busca_binaria(int v[], int n, int k)
{
    int inicio = 0;
    int fim = n-1;

    while (inicio <= fim) {
        int meio = (inicio + fim) / 2;
        if (v[meio] == k)
            return meio;
        else if (v[meio] > k)
            fim = meio - 1;
        else
            inicio = meio + 1;
    }

    return -1;
}
```

Essa implementação
retorna a posição de
k em **v**, ou **-1**
quando $k \notin v$.



Foto criada por katemangostar - br.freepik.com



11.2) Complemente o exercício 11.1 para que, ao final da impressão, o programa leia um nome e informe a posição desse nome na lista ou se ele não existe. O programa deve terminar quando o usuário digitar string vazia. A busca na lista de nomes deverá ser feita usando pesquisa binária sem considerar a diferença entre maiúsculas e minúsculas.

Dica use a função `stricmp` para comparar duas strings.