

# MC102 – Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Primeiro Semestre de 2016

# Roteiro

- 1 Introdução
- 2 Vetores
- 3 Exemplos com vetores
- 4 Strings
- 5 Exemplos com strings
- 6 Informações extras sobre strings
- 7 Exercícios

## Armazenando 3 notas

```
#include <stdio.h>

int main() {
    float nota1, nota2, nota3;

    printf("Nota do aluno 1: ");
    scanf("%f", &nota1);

    printf("Nota do aluno 2: ");
    scanf("%f", &nota2);

    printf("Nota do aluno 3: ");
    scanf("%f", &nota3);

    ...

    return 0;
}
```

## Armazenando 100 notas

```
#include <stdio.h>

int main() {
    float nota1, nota2, nota3, ..., nota100;

    printf("Nota do aluno 1: ");
    scanf("%f", &nota1);

    printf("Nota do aluno 2: ");
    scanf("%f", &nota2);
    ...
    printf("Nota do aluno 100: ");
    scanf("%f", &nota100);

    ...

    return 0;
}
```

## Definição:

Vetor é uma coleção de variáveis do mesmo tipo referenciada por um nome comum.

- Acesso por meio de índice inteiro.
- Posições contíguas na memória.
- Tamanho pré-definido (usando constantes).
- Uso de índices fora dos limites pode causar que o programa tenha comportamento inesperado.

## Declaração de um vetor

```
<tipo> identificador[<tamanho_do_vetor>;
```

### Exemplos:

```
#define MAX 100
```

```
...
```

```
float notas[MAX];
```

```
double medias[50];
```

```
int idades[MAX];
```

```
char nome[20];
```

- É considerada uma boa prática de programação usar constantes pré-definidas (com define) para indicar o tamanho dos vetores.

## Usando um vetor

Após declarada uma variável do tipo vetor, podemos acessar uma determinada posição do vetor utilizando um valor inteiro como índice.

```
identificador [<posição>]
```

- O acesso a um vetor em uma posição específica tem o mesmo comportamento que uma variável simples.
- Em C, a primeira posição de um vetor tem índice 0.
- A última posição de um vetor tem índice `<tamanho_do_vetor> - 1`.

### Exemplo:

```
int nota[10], a;  
nota[5] = 9;  
a = nota[5] - 2;
```

## Usando um vetor

- Podemos usar valores inteiros, tanto constantes quanto variáveis, para acessar uma posição do vetor.

### Exemplo:

```
#define MAX 100
```

```
...
```

```
int i, v[MAX];
```

```
for (i = 0; i < MAX; i++)
```

```
    v[i] = i + 1;
```



# Vetores

- Declaração de variáveis:

```
int a;
```

```
int vetor[5];
```

```
int b;
```

Nome	a	vetor					b
Índice	-	0	1	2	3	4	-
Valor							

# Vetores

- Acessos válidos:

```
a = 3;
```

```
vetor[a] = 2;
```

```
vetor[1] = vetor[a] + 5;
```

Nome	a	vetor					b
Índice	-	0	1	2	3	4	-
Valor	3		7		2		

# Vetores

- Acessos inválidos:

```
vetor[5] = 4; /* Erro: alterou o valor de b */  
vetor[-1] = 6; /* Erro: alterou o valor de a */  
vetor[10] = 8; /* Erro: segmentation fault */
```

Nome	a	vetor					b
Índice	-	0	1	2	3	4	-
Valor	6		7		2		4

## Armazenar $n$ ( $\leq 100$ ) notas

```
#include <stdio.h>

#define MAX 100

int main() {
    int n, i;
    float nota[MAX];

    printf("Numero de alunos: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {

    }

    return 0;
}
```

## Armazenar $n$ ( $\leq 100$ ) notas

```
#include <stdio.h>

#define MAX 100

int main() {
    int n, i;
    float nota[MAX];

    printf("Numero de alunos: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Nota do aluno %d: ", i + 1);
        scanf("%f", &nota[i]);
    }

    return 0;
}
```

## Armazenar $n$ ( $\leq 100$ ) notas

```
#include <stdio.h>

#define MAX 100

int main() {
    int n, i;
    float nota[MAX];

    do {
        printf("Numero de alunos: ");
        scanf("%d", &n);
    } while ((n < 0) || (n > MAX));

    for (i = 0; i < n; i++) {
        printf("Nota do aluno %d: ", i + 1);
        scanf("%f", &nota[i]);
    }

    return 0;
}
```

## Inicialização de um vetor

- Em algumas situações é necessário declarar e já atribuir um conjunto de valores constantes a um vetor.
- Em C, isto é feito atribuindo-se uma lista de elementos para o vetor na sua criação, da seguinte forma:

```
<tipo> identificador[] = {<elementos separados por vírgulas>;
```

### Exemplos:

```
float vet1[] = {2.3, 3.4, 4.5, 5.6};  
int vet2[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

- Note que automaticamente é criado um vetor com o tamanho igual ao número de elementos da lista de inicialização.

# Média de valores

- Ler um vetor com 10 valores inteiros e computar a média dos valores.
- Quais tipos de variáveis usar?



# Média de valores

```
#include <stdio.h>

int main() {
    int i, valores[10], soma = 0;

    for (i = 0; i < 10; i++) {
        printf("Digite valor %d: ", i + 1);
        scanf("%d", &valores[i]);
    }

    for (i = 0; i < 10; i++)
        soma += valores[i];

    printf("Media: %.1f\n", soma / 10.0);

    return 0;
}
```

## Produto interno de dois vetores

- Ler dois vetores de tamanho 5 e computar o produto interno destes.
- Quais tipos de variáveis usar?

# Produto interno de dois vetores

```
#include <stdio.h>

int main() {
    int i;
    double vetor1[5], vetor2[5], resultado;

    for (i = 0; i < 5; i++) {
        printf("Entre com valor %d para vetor 1: ", i + 1);
        scanf("%lf", &vetor1[i]);
    }

    for (i = 0; i < 5; i++) {
        printf("Entre com valor %d para vetor 2: ", i + 1);
        scanf("%lf", &vetor2[i]);
    }

    resultado = 0.0;
    for (i = 0; i < 5; i++)
        resultado = resultado + (vetor1[i] * vetor2[i]);

    printf("Produto interno: %f\n", resultado);
    return 0;
}
```

# Elementos iguais

- Ler dois vetores com 5 inteiros cada.
- Identificar quais elementos do segundo vetor são iguais a algum elemento do primeiro vetor.

# Elementos iguais

```
#include <stdio.h>

int main() {
    int i, j, vetor1[5], vetor2[5];

    for (i = 0; i < 5; i++) {
        printf("Entre com o valor de vetor1[%d]: ", i);
        scanf("%d", &vetor1[i]);
    }

    for (i = 0; i < 5; i++) {
        printf("Entre com o valor de vetor2[%d]: ", i);
        scanf("%d", &vetor2[i]);
    }

    for (i = 0; i < 5 ; i++)
        for (j = 0; j < 5; j++)
            if (vetor1[i] == vetor2[j])
                printf("vetor1[%d] = vetor2[%d] = %d\n", i, j, vetor1[i]);

    return 0;
}
```

# Habemos Papam

- As eleições dos papas são realizadas por Conclaves, reuniões que reúnem todos os cardeais com menos de 80 anos, onde todos são elegíveis e têm direito a voto.
- O Conclave de 2013 contou com a presença de 114 cardeais.
- Para um cardeal ser eleito papa, ele precisa obter mais de  $2/3$  dos votos, ou seja, 77 votos no caso do Conclave de 2013.
- Supondo que os cardeais foram numerados de 1 a 114 (e o número 0 foi usado para representar votos brancos ou nulos), dada a lista de votos, como determinar se um novo papa foi eleito?

# Habemos Papam

```
#include <stdio.h>

int main() {
    int i, cardeal[115], voto, papa = 0;

    for (i = 0; i <= 114; i++)
        cardeal[i] = 0;

    for (i = 1; i <= 114; i++) {
        printf("Qual o cardeal indicado?\n");
        scanf("%d", &voto);
        cardeal[voto]++;
    }

    for (i = 1; i <= 114; i++)
        if (cardeal[i] > cardeal[papa])
            papa = i;

    if ((papa != 0) && (cardeal[papa] >= 77))
        printf("Habemus Papam: cardeal %d\n", papa);

    return 0;
}
```

# Habemos Papam

```
#include <stdio.h>

int main() {
    int i, cardeal[115], voto, papa = 0;

    for (i = 0; i <= 114; i++)
        cardeal[i] = 0;

    for (i = 1; i <= 114; i++) {
        printf("Qual o cardeal indicado?\n");
        scanf("%d", &voto);
        cardeal[voto]++;
    }

    for (i = 1; i <= 114; i++)
        if (cardeal[i] >= 77)
            papa = i;

    if (papa != 0)
        printf("Habemus Papam: cardeal %d\n", papa);

    return 0;
}
```



# Habemos Papam

```
#include <stdio.h>

int main() {
    int i, cardeal[115], voto, papa = 0;

    for (i = 0; i <= 114; i++)
        cardeal[i] = 0;

    for (i = 1; i <= 114; i++) {
        printf("Qual o cardeal indicado?\n");
        scanf("%d", &voto);
        cardeal[voto]++;
    }

    for (i = 1; (i <= 114) && (papa != 0); i++)
        if (cardeal[i] >= 77)
            papa = i;

    if (papa != 0)
        printf("Habemus Papam: cardeal %d\n", papa);

    return 0;
}
```

# Strings

- A linguagem C não possui explicitamente um tipo string, entretanto, podemos considerar um vetor de caracteres como uma string.
- Em C, uma string sempre deve ser terminada pelo caractere especial `'\0'` (equivalente ao número zero).
- Portanto, sempre declare uma string com um caractere a mais do que você pretende usar.
- Se, por exemplo, pretender usar uma string de 10 caracteres, declare:  

```
char string[11];
```
- Para ler (usando `scanf`) ou imprimir (usando `printf`) uma string, usamos o formato `"%s"`.
- Para strings, não é utilizado o símbolo `&` no comando `scanf`.

# Strings

```
#include <stdio.h>

int main() {
    char palavra[21];
    int numero;

    printf("Entre com uma palavra: ");
    /* Para ler uma string, nao usar o simbolo & no scanf */
    scanf("%s", palavra);

    printf("Entre com um numero: ");
    scanf("%d", &numero);

    printf("Palavra = %s\n", palavra);
    printf("Numero = %d\n", numero);

    return 0;
}
```

# Strings

```
#include <stdio.h>

#define ERRO "Mensagem de erro padrao"

int main() {
    char string1[81] = "Primeira string";
    char string2[] = "Segunda string";
    char string3[81];

    scanf("%s", string3);

    printf("%s\n", ERRO);
    printf("%s, %s, %s\n", string1, string2, string3);

    return 0;
}
```

## Determinar o tamanho de uma string

- Ler e salvar uma sequência de até 80 caracteres (palavra).
- Contar e imprimir o número de caracteres da sequência.

## Determinar o tamanho de uma string

```
#include <stdio.h>
#include <string.h>

int main() {
    char string[81];
    int n;

    printf("Digite uma string: ");
    scanf("%s", string);

    /* determina o tamanho da string */
    n = strlen(string);

    printf("Tamanho da string = %d\n", n);

    return 0;
}
```

## Determinar o tamanho de uma string

```
#include <stdio.h>

int main() {
    char string[81];
    int n = 0;

    printf("Digite uma string: ");
    scanf("%s", string);

    /* determina o tamanho da string */
    while (string[n] != '\0')
        n++;

    printf("Tamanho da string = %d\n", n);

    return 0;
}
```

## Determinar o tamanho de uma string

```
#include <stdio.h>

int main() {
    char string[81];
    int n = 0;

    printf("Digite uma string: ");
    scanf("%s", string);

    /* determina o tamanho da string */
    while (string[n] != 0)
        n++;

    printf("Tamanho da string = %d\n", n);

    return 0;
}
```



## Determinar o tamanho de uma string

```
#include <stdio.h>

int main() {
    char string[81];
    int n = 0;

    printf("Digite uma string: ");
    scanf("%s", string);

    /* determina o tamanho da string */
    while (string[n])
        n++;

    printf("Tamanho da string = %d\n", n);

    return 0;
}
```

## Determinar o tamanho de uma string

```
#include <stdio.h>

int main() {
    char string[81];
    int n;

    printf("Digite uma string: ");
    scanf("%s", string);

    /* determina o tamanho da string */
    for (n = 0; string[n]; n++);

    printf("Tamanho da string = %d\n", n);

    return 0;
}
```

# Inverter uma string

- Ler e salvar uma sequência de até 80 caracteres (palavra).
- Usando uma string auxiliar, obter a palavra invertida.
- Imprimir a palavra invertida (string auxiliar).

## Inverter uma string

```
#include <stdio.h>

int main() {
    char palavra[81], inversa[81];
    int tam, i;

    printf("Digite uma palavra: ");
    scanf("%s", palavra);

    for (tam = 0; palavra[tam]; tam++);

    for (i = 0; i < tam; i++)
        inversa[tam - i - 1] = palavra[i];
    inversa[tam] = '\0';

    printf("Inversa: %s\n", inversa);

    return 0;
}
```

## Inverter uma string

```
#include <stdio.h>

int main() {
    char palavra[81], inversa[81];
    int tam, i;

    printf("Digite uma palavra: ");
    scanf("%s", palavra);

    for (tam = 0; palavra[tam]; tam++);

    for (i = 0; i < tam; i++)
        inversa[i] = palavra[tam - i - 1];
    inversa[tam] = '\0';

    printf("Inversa: %s\n", inversa);

    return 0;
}
```

# Inverter uma string

- Ler e salvar uma sequência de até 80 caracteres (palavra).
- Sem usar uma string auxiliar, alterar a string original de tal forma a obter a palavra invertida.
- Imprimir a palavra invertida (usando a string original).

## Inverter uma string

```
#include <stdio.h>

int main() {
    char palavra[81], temp;
    int tam, i;

    printf("Digite uma palavra: ");
    scanf("%s", palavra);

    for (tam = 0; palavra[tam]; tam++);

    for (i = 0; i < tam/2; i++) {
        temp = palavra[tam - i - 1];
        palavra[tam - i - 1] = palavra[i];
        palavra[i] = temp;
    }

    printf("Inversa: %s\n", palavra);

    return 0;
}
```

# Inverter uma string

```
#include <stdio.h>

int main() {
    char palavra[81], temp;
    int tam, i;

    printf("Digite uma palavra: ");
    scanf("%s", palavra);

    for (tam = 0; palavra[tam]; tam++);

    for (i = 0; i < tam/2; i++) {
        temp = palavra[i];
        palavra[i] = palavra[tam - i - 1];
        palavra[tam - i - 1] = temp;
    }

    printf("Inversa: %s\n", palavra);

    return 0;
}
```



# Anagramas

- Um anagrama é o resultado do rearranjo dos caracteres de uma palavra ou frase para produzir outras palavras, utilizando todas os caracteres originais exatamente uma vez.
- Exemplos: “america” e “iracema”, “porta” e “tropa”, “celia” e “alice”, “missa” e “assim”, “pedro” e “poder”, “argentino” e “ignorante”.
- Escreva um programa que leia duas palavras e verifique se elas são anagramas (uma em relação à outra).

# Anagramas

```
#include <stdio.h>

int main() {
    char palavra1[21], palavra2[21];
    int i, caracteres[128], anagramas = 1;

    printf("Entre com duas palavras: ");
    scanf("%s %s", palavra1, palavra2);

    /* inicializa o vetor de frequencia de caracteres */
    for (i = 0; i < 128; i++)
        caracteres[i] = 0;

    /* conta os caracteres encontrados na palavra1 */
    for (i = 0; palavra1[i]; i++)
        caracteres[(int) palavra1[i]]++;

    /* desconta os caracteres encontrados na palavra2 */
    for (i = 0; palavra2[i]; i++)
        caracteres[(int) palavra2[i]]--;
```

# Anagramas

```
...

/* verifica se as palavras tem a mesma frequencia de caracteres */
for (i = 0; i < 128; i++)
    if (caracteres[i] != 0)
        anagramas = 0;

if (anagramas)
    printf("As duas palavras sao anagramas\n");
else
    printf("As duas palavras nao sao anagramas\n");

return 0;
}
```

# Anagramas

```
...

/* verifica se as palavras tem a mesma frequencia de caracteres */
for (i = 0; (i < 128) && anagramas; i++)
    if (caracteres[i] != 0)
        anagramas = 0;

if (anagramas)
    printf("As duas palavras sao anagramas\n");
else
    printf("As duas palavras nao sao anagramas\n");

return 0;
}
```

# Problemas com a leitura de cadeia de caracteres

```
#include <stdio.h>
```

```
int main() {  
    char mensagem[21];  
  
    printf("Digite uma mensagem: ");  
    scanf("%s", mensagem);  
  
    printf("Mensagem: %s\n", mensagem);  
  
    return 0;  
}
```

```
$ gcc -ansi -Wall -pedantic -Werror mensagem.c -o mensagem
```

```
$ ./mensagem
```

```
Digite uma mensagem: Bom dia, mundo!
```

```
Mensagem: Bom
```

```
$ ./mensagem
```

```
Digite uma mensagem: UmTesteComUmaMensagemMaisLongaPodeCausarUmErro
```

```
Mensagem: UmTesteComUmaMensagemMaisLongaPodeCausarUmErro
```

```
Segmentation fault
```

## Comando fgets

- Ao usar o comando `scanf` para ler uma string, você deve garantir que foi alocada uma string de tamanho suficiente para armazenar todos os caracteres.
- Caso o programa tente ler mais caracteres do que o tamanho alocado, um erro ocorrerá durante a execução do programa.
- O comando `scanf` não é adequado para ler strings contendo espaços em branco.
- Uma alternativa para ler strings é o comando `fgets()`:

```
fgets(identificador, tamanho, stdin);
```

onde `identificador` é o nome da variável usada para armazenar a string e `tamanho` é um inteiro indicando o número máximo de caracteres que devem ser lidos (até `tamanho-1` caracteres são de fato lidos e um caractere extra é reservado para o `'\0'`).

- Note que o comando `fgets` lê inclusive o caractere `'\n'`.

# Inverter uma string

```
#include <stdio.h>
```

```
int main() {  
    char string[81], temp;  
    int tam, i;  
  
    printf("Digite uma string: ");  
    fgets(string, 81, stdin);  
  
    for (tam = 0; string[tam] && (string[tam] != '\n'); tam++);  
  
    for (i = 0; i < tam/2; i++) {  
        temp = string[tam - i - 1];  
        string[tam - i - 1] = string[i];  
        string[i] = temp;  
    }  
  
    printf("Inversa: %s", string);  
  
    return 0;  
}
```

## Cuidados com fgets

```
#include <stdio.h>

int main() {
    char string[81];
    int n;

    printf("Digite um numero: ");
    scanf("%d", &n);
    printf("Numero digitado: %d\n", n);

    printf("Digite um texto: ");
    fgets(string, 81, stdin); /* string nao eh lida corretamente */
    printf("Texto digitado: %s", string);

    return 0;
}
```



## Cuidados com fgets

- O comando `scanf` não lê o caractere `'\n'` que representa o final de linha (*enter*).
- Este caractere fica armazenado no *buffer* da entrada padrão.
- A seguir, o comando `fgets` lê este caractere do *buffer*, o que automaticamente encerra a leitura da string.
- Como resolver este problema?
- Podemos ler e descartar espaços em branco (`' '`, `'\n'` ou `'\t'`) usando o seguinte comando:

```
scanf(" ");
```

# Cuidados com fgets

```
#include <stdio.h>

int main() {
    char string[81];
    int n;

    printf("Digite um numero: ");
    scanf("%d", &n);
    printf("Numero digitado: %d\n", n);

    scanf(" "); /* descarta espacos em branco */

    printf("Digite um texto: ");
    fgets(string, 81, stdin);
    printf("Texto digitado: %s", string);

    return 0;
}
```

## Cuidados com fgets

```
#include <stdio.h>

int main() {
    char string[81];
    int n;

    printf("Digite um numero: ");
    scanf("%d ", &n); /* descarta espaços em branco */
    printf("Numero digitado: %d\n", n);

    printf("Digite um texto: ");
    fgets(string, 81, stdin);
    printf("Texto digitado: %s", string);

    return 0;
}
```

## Cópia de uma string

```
#include <stdio.h>
#include <string.h>

int main() {
    char string1[81], string2[81];

    printf("Digite uma string: ");
    fgets(string1, 81, stdin);

    /* copia o conteudo da string1 para a string2 */
    strcpy(string2, string1);

    printf("%s", string2);

    return 0;
}
```

# Cópia de uma string

```
#include <stdio.h>

int main() {
    char string1[81], string2[81];
    int i = -1;

    printf("Digite uma string: ");
    fgets(string1, 81, stdin);

    do {
        i++;
        string2[i] = string1[i];
    } while (string1[i]);

    printf("%s", string2);

    return 0;
}
```

# Cópia de uma string

```
#include <stdio.h>

int main() {
    char string1[81], string2[81];
    int i = 0;

    printf("Digite uma string: ");
    fgets(string1, 81, stdin);

    do {
        string2[i] = string1[i];
    } while (string1[i++]);

    printf("%s", string2);

    return 0;
}
```

# Cópia de uma string

```
#include <stdio.h>

int main() {
    char string1[81], string2[81];
    int i;

    printf("Digite uma string: ");
    fgets(string1, 81, stdin);

    for (i = 0; string1[i]; i++)
        string2[i] = string1[i];
    string2[i] = '\0';

    printf("%s", string2);

    return 0;
}
```

# Concatenação de duas strings

```
#include <stdio.h>
#include <string.h>

int main() {
    char string1[81], string2[161];

    printf("Digite uma string: ");
    fgets(string1, 81, stdin);

    printf("Digite outra string: ");
    fgets(string2, 81, stdin);

    /* concatena o conteudo da string1 na string2 */
    strcat(string2, string1);

    printf("%s", string2);

    return 0;
}
```



# Concatenação de duas strings

```
#include <stdio.h>

int main() {
    char string1[81], string2[161];
    int i, j;

    printf("Digite uma string: ");
    fgets(string1, 81, stdin);

    printf("Digite outra string: ");
    fgets(string2, 81, stdin);

    for (i = 0; string2[i]; i++);

    for (j = 0; string1[j]; j++)
        string2[i + j] = string1[j];
    string2[i + j] = '\0';

    printf("%s", string2);

    return 0;
}
```

# Palíndromo

- Um palíndromo é uma palavra ou frase que é igual quando lida da esquerda para a direita ou da direita para a esquerda, desconsiderando-se os espaços em brancos.
- Exemplos de palíndromos: “radar”, “reviver”, “mirim”, “a sacada da casa”, “a mala nada na lama”, “anotaram a data da maratona”, “a torre da derrota”, “o galo ama o lago” e “a cara rajada da jararaca”.
- Escreva um programa que leia uma string de até 80 caracteres e teste se ela é um palíndromo.

# Palíndromo

```
#include <stdio.h>

int main() {
    char string[81];
    int tam, i, j, palindromo = 1;

    printf("Digite uma string: ");
    fgets(string, 81, stdin);

    for (tam = 0; string[tam] && (string[tam] != '\n'); tam++);

    for (i = 0, j = tam - 1; palindromo && (i < j); i++, j--) {
        while ((string[i] == ' ') && (i < j)) i++; /* pula espaços a esquerda */
        while ((string[j] == ' ') && (i < j)) j--; /* pula espaços a direita */

        palindromo = (string[i] == string[j]);
    }

    if (palindromo)
        printf("Palindromo\n");

    return 0;
}
```

## Exercícios

- Escreva um programa que, dados  $n$  números reais ( $1 \leq n \leq 100$ ), determine a moda deste conjunto de dados, ou seja, determine qual é o número mais frequente. Note que a moda pode ser composta por mais de um número, neste caso, basta indicar um deles.
- Escreva um programa que, dados  $n$  números reais ( $1 \leq n \leq 100$ ), determine quais são os números que diferem de mais de um desvio padrão da média deste conjunto.
- Escreva um programa que, dada uma sequência de  $n$  números reais ( $1 \leq n \leq 100$ ), determine quais são os máximos e mínimos locais. Um número é dito máximo (mínimo) local de uma sequência se ele for maior (menor) tanto do elemento que o precede quanto do que o sucede na sequência. Por definição, não existem máximos ou mínimos locais nos extremos das sequências.

# Exercícios

- Escreva um programa que, dadas duas strings A e B, verifique se A é um anagrama de B, sem usar vetores auxiliares (além daqueles usados para armazenar A e B).
- Escreva um programa que, dadas duas strings A e B, descubra qual é o primeiro caractere da string A que também pertence à string B, ou imprima uma mensagem de erro caso não exista tal caractere. Por exemplo, o primeiro caractere de “bola” que também é um caractere de “amor” é a letra 'o'. Não use a função `strpbrk` da biblioteca `string.h`.

# Exercícios

- Escreva um programa que, dadas duas palavras, verifique qual é a menor em termos lexicográficos, ou seja, qual deveria aparecer primeiro num dicionário. Por exemplo, “casamento” é menor do que “catapulta”, mas é maior do que “casa”. Não use a função `strcmp` da biblioteca `string.h`.
- Escreva um programa que, dadas duas strings A e B, verifique se uma delas é substring da outra. Por exemplo, “asa” é uma substring de “casamento”, mas não de “assado”. Não use a função `strstr` da biblioteca `string.h`.

# Moda

```
#include <stdio.h>

#define MAX 100

int main() {
    double vetor[81];
    int n, i, j, cont, max = 0, moda;

    scanf("%d", &n);

    for (i = 0; i < n; i++)
        scanf("%lf", &vetor[i]);

    ...
}
```

# Moda

...

```
for (i = 0; i < n; i++) {
    for (j = 0, cont = 0; j < n; j++)
        if (vetor[i] == vetor[j])
            cont++;

    if (cont > max) {
        max = cont;
        moda = vetor[i];
    }
}

printf("Moda = %d\n", moda);
return 0;
}
```