Algoritmos e programação estruturada

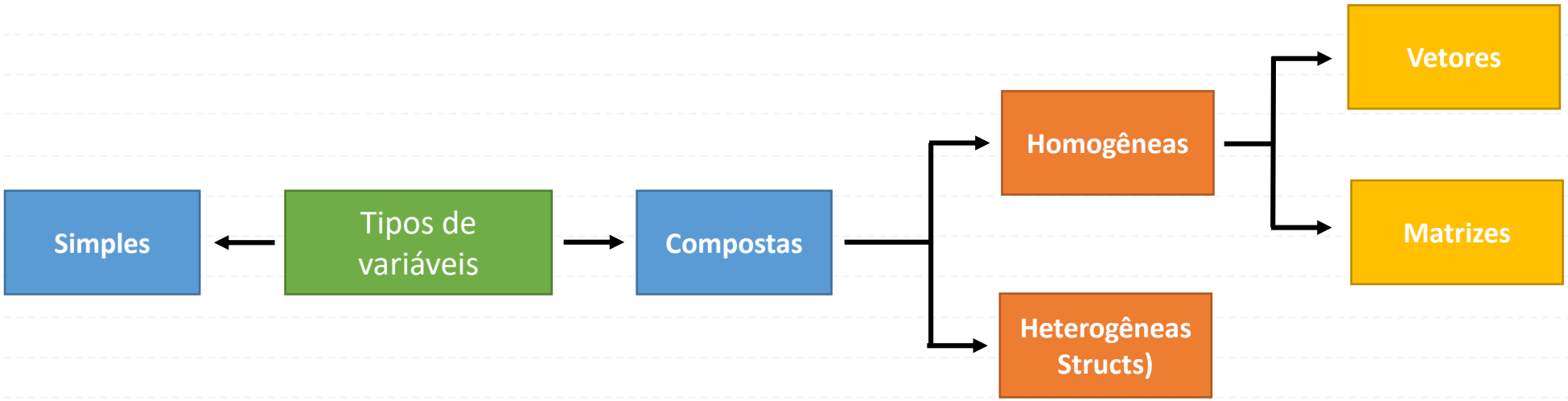
Vetores e matrizes

11/02/2025

Professor Marcelo Eustáquio

Vetores

Variáveis compostas são estruturas de dados que permitem armazenar múltiplos valores em uma única variável e podem ser classificadas em homogêneas ou heterogêneas.



Vetores são estruturas de dados que armazenam uma coleção de elementos do mesmo tipo em locais consecutivos de memória em que cada elemento pode ser acessado por meio de um índice.

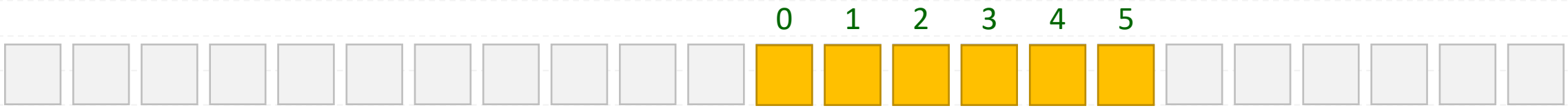
Vetores

Vetores (ou **arrays**) são estruturas de dados que armazenam uma coleção de elementos do mesmo tipo em locais consecutivos de memória, em que cada elemento pode ser acessado individualmente por meio de um índice, que é um número inteiro.

Como declarar um vetor de inteiros com 6 posições?

```
int V[6];
```

São reservadas, na memória RAM, o espaço para armazenar 6 valores inteiros ($6 \cdot 4 = 24\ bytes$), que serão acessados pelos índices 0, 1, 2, 3, 4 e 5.



Vetores

Identificando os endereços de memória reservados para um vetor.

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    int V[6];
    for (int i = 0; i < 6; i++) printf("%d %d %x \n", i, &V[i], &V[i]);
}
```

Observações:

- O caractere **&** acessa o **endereço de memória** de uma variável.
- O uso de **%x** faz com que um inteiro seja exibido na forma de um **número hexadecimal**.
- Para mostrar um inteiro na **forma octal**, deve ser feito o uso de **%o**.

Vetores

Vetores (ou **arrays**) são estruturas de dados que armazenam uma coleção de elementos do mesmo tipo em locais consecutivos de memória, em que cada elemento pode ser acessado individualmente por meio de um índice, que é um número inteiro.

Como inicializar os elementos de um vetor no momento da declaração?

```
char Letra[5] = {'T', 'e', 's', 't', 'e'};
```

Agora, são reservados espaços para 5 caracteres na memória RAM ($5 \cdot 1 = 5\ bytes$), que são inicializados com as letras da palavra “teste” e serão acessados a partir dos índices 0, 1, 2, 3 e 4.

Vetores

Imprimindo alguns elementos de um vetor de caracteres...

```c
#include <stdio.h>

int main(int argc, char *argv[]) {

    char V[12] = {'M', 'e', ' ', 'l', 'e', 'm', 'b', 'r', 'o', 'u', '\0'};
    printf("%c%c%c%c%c%c%c\n", V[3], V[4], V[5], V[6], V[7], V[8], V[9]);

}
```

Observações:

- O tamanho do vetor não precisa ser inicializado, mas é bom fazê-lo.
- O caracteres **\0** (nulo) é usado para encerrar uma string (vetor de caracteres).
- Para imprimir um **caractere**, deve ser usado **%c**.

Vetores

Em C, os vetores (arrays) são armazenados na memória de forma contígua, e o compilador precisa saber o tamanho do array para alocar o espaço necessário. Como o C não possui um tipo de dado embutido para "vetores dinâmicos" (diferente de linguagens como **Python** ou **JavaScript**), o tamanho do array precisa ser conhecido em **tempo de compilação** para garantir a correta alocação.

Observações:

- O tamanho do vetor deve ser um número **inteiro**.
- Não pode ser usada uma variáveis para especificar o tamanho do vetor (pode ser **constante**).

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    int Tamanho = 5;
    char Verso[Tamanho];
}
```

```c
#include <stdio.h>
#define Tamanho 5

int main(int argc, char *argv[]) {
    char Verso[Tamanho];
}
```

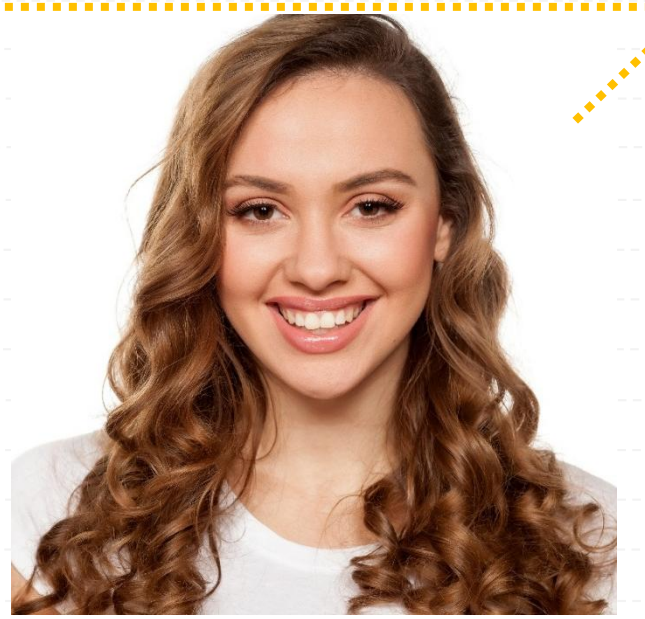
```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    int Tamanho = 5;
    char Verso[Tamanho];
}
```

*Usando **variável** para indicar tamanho do vetor.*

```c
#include <stdio.h>
#define Tamanho 5

int main(int argc, char *argv[]) {
    char Verso[Tamanho];
}
```



*Usando **constante** para indicar tamanho do vetor.*

Vetores

Sobre a diretiva **define**:

- É uma diretiva de pré-processamento que define uma constante;
- A constante é substituída pelo valor correspondente no processo de compilação.
- O uso de constantes torna o código mais legível e programas mais escaláveis.

```c
#include <stdio.h>
#define TAMANHO 8

int main() {

    int V[TAMANHO] = {32, 27, 64, 18, 95, 14, 90, 70};
    for (int i = 0; i < TAMANHO; i++) printf("%d\t%d\n", i, V[i]);
    return 0;
}
```

Vetores

Considerando o vetor definido por `int P[6] = {2, 4, 6, 7, 9, 2};`, determine o valor de $P[2] + P[4] * P[1]$.

Vetores

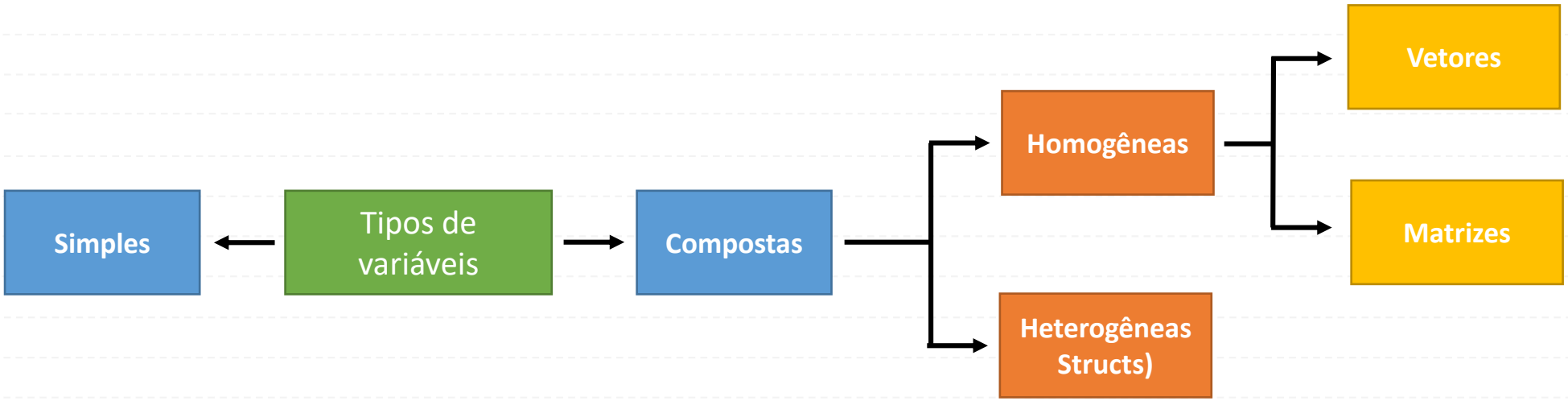
O que estudamos até o momento...

- Vetores são estruturas de dados que consistem em itens de dados do **mesmo tipo**.
- Vetores são estruturas estáticas, ou seja, tem o **mesmo tamanho** durante a execução do programa.
- O espaço alocado para o vetor é liberado assim que os blocos que as contém forem finalizados.
- Vetores são armazenados em posições contínuas de memória.

Ponteiros e vetores são termos sinônimos?

Strings

Em C, existem 5 (cinco) tipos básicos de dados: *caractere*, *inteiro*, *ponto flutuante*, *double* e *void*. Todos os outros tipos de dados são baseados em um desses tipos. Note que não existe o tipo **string**.



E então, o que é uma string?

E então, o que é uma string?

String é um vetor de caracteres encerrados com caractere '**\0**' (caractere nulo).

```c
#include <stdio.h>

int main() {

    char minhaStr[] = "Eis uma string de exemplo.";
    int C = 0;

    // Calcula o comprimento da string

    while (minhaStr[C] != '\0') {
        C++;
    }

    printf("Posição do final da string: %d", C);
    return 0;
}
```

Strings

Em C, uma string é declarada como um array de caracteres (char) com um caractere nulo **'\0'** no final para indicar o término da string. Existem duas maneiras comuns de declarar uma string:

- Usando um vetor de caracteres (**char array**)

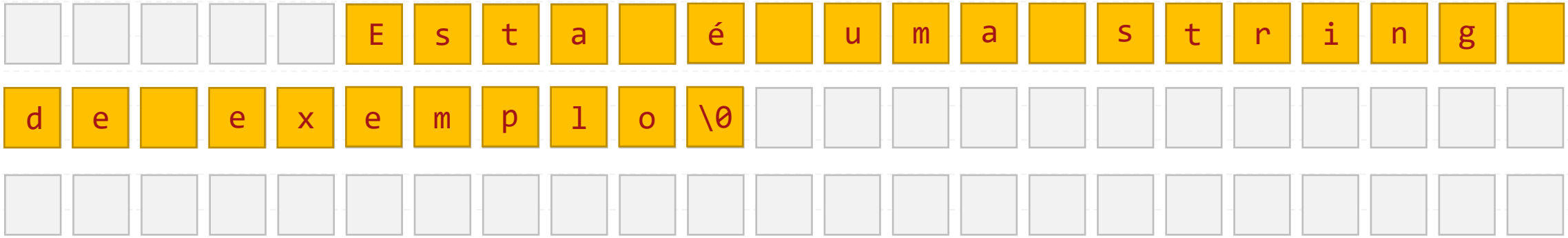
```c
char minhaString[] = "Esta é uma string de exemplo";
```

Neste caso, o tamanho da string é dimensionado com base no tamanho da string literal atribuída a ela. Além disso, a string será armazenada na memória como uma sequência de caracteres, e o caractere nulo **'\0'** será adicionado automaticamente pelo compilador no final da string.

Strings

```
char minhaString[] = "Esta é uma string de exemplo";
```

Neste caso, o tamanho da string é dimensionado com base no tamanho da string literal atribuída a ela. Além disso, a string será armazenada na memória como uma sequência de caracteres, e o caractere nulo **'\0'** será adicionado automaticamente pelo compilador no final da string.



Strings

Em C, uma string é declarada como um array de caracteres (char) com um caractere nulo **'\0'** no final para indicar o término da string. Existem duas maneiras comuns de declarar uma string:

- Usando a função (**strcpy**):

```
char minhaString[50];
strcpy(minhaString, "Esta é uma string de exemplo");
```

Agora, um array de caracteres minhaString com espaço para 50 caracteres é declarada. Em seguida, a função strcpy é usada para copiar a string para a matriz.

Strings

```
char minhaString[50];
strcpy(minhaString, "Esta é uma string de exemplo");
```

Agora, assim como na declaração anterior, a variável `minhaString` é inicializada com `"Esta é uma string de exemplo“`, encerrada com o caractere nulo **‘\0’**.

Strings

Em C, uma string é declarada como um array de caracteres (char) com um caractere nulo **'\0'** no final para indicar o término da string. Existem duas maneiras comuns de declarar uma string:

- Inicializando caractere por caractere

```c
#include <stdio.h>

int main() {
    char minhaString[20];
    minhaString[0] = 'E';
    minhaString[1] = 's';
    // restante da string (...)
    minhaString[20] = '\0';
    printf("A string é: %s\n", minhaString);
    return 0;
}
```

Strings

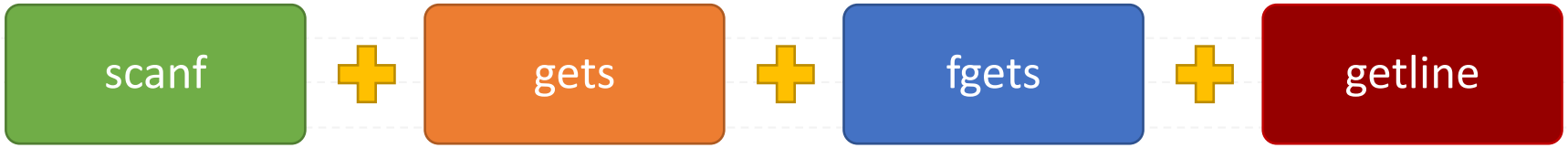
Considerando a string Música definida a seguir:

```
char Musica[] = "Um pé na soleira e um pé na calçada, um pião\nUm passo na estrada e um pulo no mato\nUm pedaço de pau\nUm pé de sapato e um pé de moleque\nLéo\n\nUm pé de moleque e um rabo de saia, um serão\nAs sombras da praia e o sonho na esteira\0Uma alucinação\0Uma companheira e um filho no mundo\nLéo\n\nUm filho no mundo e um mundo virado, um irmão\nUm livro, um recado, uma eterna viagem\nA mala de mão\nA cara, a coragem e um plano de vôo\nLéo\n\nUm plano de vôo e um segredo na boca, um ideal\nUm bicho na toca e o perigo por perto\nUma pedra, um punhal\nUm olho desperto e um olho vazado\nLéo\n\nUm olho vazado e um tempo de guerra, um paiol\nUm nome na serra e um nome no muro\nA quebrada do sol\nUm tiro no escuro e um corpo na lama\nLéo\n\nUm nome na lama e um silêncio profundo, um pião\nUm filho no mundo e uma atiradeira\nUm pedaço de pau\nUm pé na soleira e um pé na calçada\n\0";
```

Escreva, usando a linguagem C, o código-fonte para contar quantas vezes a palavra “pé” está presente na música. A seguir, modifique o programa escrito para mostrar as posições da string em que “pé” aparece.

Strings

Existem quatro formas principais para se ler uma string e armazená-la em um vetor de caracteres: **scanf**, **gets**, **fgets** e getline.



Observações:

- **gets** foi removida do padrão C porque é propensa a vulnerabilidades, como buffer overflow.
- **getline** é usado sistema POSIX (como Linux), pode usar getline para ler strings de tamanho variável.
- As opções mais comuns (e recomendadas) são **scanf** e **fgets**.

Strings

O método mais básico é o **scanf**, mas ele tem limitações de parar na leitura do primeiro espaço em branco.

```c
#include <stdio.h>

int main() {
    char str[100]; // Define o tamanho máximo da string

    printf("Digite uma string: ");
    scanf("%99s", str); // Lê até 99 caracteres e garante espaço para o '\0'
    printf("Você digitou: %s\n", str);

    return 0;
}
```

O **%s** do **scanf** para de ler ao encontrar um espaço ou quebra de linha e o tamanho da string deve ser limitado para evitar buffer overflow.

Strings

O **fgets** é a melhor opção para strings com espaços e para melhor controle de tamanho.

```c
#include <stdio.h>

int main() {
    char str[100]; // Define o tamanho máximo da string

    printf("Digite uma string: ");
    fgets(str, sizeof(str), stdin); // Lê até 99 caracteres, incluindo espaços e '\n'
    printf("Você digitou: %s\n", str);

    return 0;
}
```

O **fgets** permite ler string com espaços e garante que não será lido mais do que o tamanho do buffer (que no exemplo acima é de 100 caracteres).

Strings

Escreva código-fonte em C que faz a leitura de uma string até que seja identificado um #.

```c
#include <stdio.h>

int main() {
    char str[100]; // Define o tamanho máximo da string

    printf("Digite uma string (será lida até o '#'):\n");
    scanf("%99[^#]", str);
    printf("String lida: %s\n", str);

    return 0;
}
```

Explique o funcionamento de `scanf("%99[^#]", str);`.

Algoritmos e programação estruturada

Vetores e matrizes

Exercícios

Professor Marcelo Eustáquio

Q01

Escreva código-fonte me C para inicializar um vetor com os 12 primeiros números ímpares e positivos. A seguir, determine a média aritmética dos elementos desse vetor.

A média aritmética dos elementos de um conjunto é dada por $\frac{1}{n} \cdot \sum_1^n a_i = \frac{1}{n} \cdot (a_1 + a_2 + \cdots + a_n)$

Q02

Escreva código-fonte em C inicializando dos 8 elementos de um vetor com os números 32, 27, 64, 18, 95, 14, 90 e 70 para, a seguir, exibi-los na forma de uma tabela, conforme mostrado a seguir.

```
PS C:\Users\Administrador\Desktop\output> cd 'c:\Users\Administrador\Desktop\output'
PS C:\Users\Administrador\Desktop\output> & .\'teste.exe'
├ìndice Valor
------  -----
0       32
1       27
2       64
3       18
4       95
5       14
6       90
7       70
PS C:\Users\Administrador\Desktop\output>
```

Vetores

Algumas dicas úteis...

- Todos os elementos podem ser inicializados com 0 assim: `int n[10] = {0};`
- Essa declaração inicializa o primeiro elemento com 0 e copia esse valor para os demais.
- Arrays não são iniciados automaticamente em zero.
- O tamanho do vetor pode ser omitido.

O tamanho do vetor `int V[] = {32, 27, 64, 18, 95, 14, 90, 70};` pode ser determinado usando a instrução `int T = sizeof(V)/sizeof(V[0]);`

Q03

Escreva, em C, um algoritmo para ler (do teclado) as temperaturas registradas nos últimos 7 dias e armazená-las em um vetor de float’s. Em seguida, calcule e imprima a média das temperaturas desses 7 dias e indique também os dias em que a temperatura foi maior do que a média.

Q04

Escreva, em C, um algoritmo para preencher um vetor de 10 elementos inteiros colocando -1 nas posições ímpares e 0 nas posições pares.

Q05

Escreva, em C, um algoritmo para determinar o maior elemento de um vetor de 8 elementos inteiros lidos a partir do dispositivo de entrada padrão.

Q06

Escreva um programa, na linguagem C, que leia 15 números inteiros em um vetor e conte quantos são pares e quantos são divisíveis por 3. A seguir, imprima o resultado.

Q07

Escreva um programa que leia 7 números inteiros de um vetor e imprima os elementos na ordem inversa em relação à entrada.

Q08

Escreva, usando a linguagem C ANSI, um programa que cria dois vetores de 5 elementos cada para, a seguir gerar um terceiro vetor que intercale os dois vetores originais. Por exemplo, se os vetores forem $A = \{1, 3, 5, 7, 9\}$ e $B = \{2, 4, 6, 8, 10\}$, o vetor resultante será $C = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

Q09

Escreva um programa que leia um vetor de 5 números inteiros e rotacione seus elementos uma posição para a direita. O último elemento deve se tornar o primeiro.

Exemplo:

Se a entrada for $A = \{1, 2, 3, 4, 5\}$, a saída será $R = \{5, 1, 2, 3, 4\}$.

Q10

Escreva um programa que leia 12 números inteiros em um vetor e remova os valores duplicados, deixando apenas os números únicos. Imprima o vetor resultante sem duplicatas.

Q11

Escreva código-fonte m C para inverter uma string dada. Assim, caso a string de entrada seja declarada por `char A[] = "ABCD"`, a string invertida será equivalente a `char B[] = “DCBA"`.

Não existe, em **string.h**, uma função que inverte uma string.

Q12

(Concatenando strings) Escreva código-fonte, na linguagem C, que concatene as strings A e B para gerar a string R. Por exemplo, caso A e B sejam declaradas como em `char A[] = "Meu caminho "` e `char B[] = "é de pedra ..."`, a string R será `char R[] = "Meu caminho é de pedra ... "`.

Q13

Considerando a string Música definida a seguir:

```
char Musica[] = "Um pé na soleira e um pé na calçada, um pião\nUm passo na estrada e um pulo no mato\nUm pedaço de pau\nUm pé de sapato e um pé de moleque\nLéo\n\nUm pé de moleque e um rabo de saia, um serão\nAs sombras da praia e o sonho na esteira\0Uma alucinação\0Uma companheira e um filho no mundo\nLéo\n\nUm filho no mundo e um mundo virado, um irmão\nUm livro, um recado, uma eterna viagem\nA mala de mão\nA cara, a coragem e um plano de vôo\nLéo\n\nUm plano de vôo e um segredo na boca, um ideal\nUm bicho na toca e o perigo por perto\nUma pedra, um punhal\nUm olho desperto e um olho vazado\nLéo\n\nUm olho vazado e um tempo de guerra, um paiol\nUm nome na serra e um nome no muro\nA quebrada do sol\nUm tiro no escuro e um corpo na lama\nLéo\n\nUm nome na lama e um silêncio profundo, um pião\nUm filho no mundo e uma atiradeira\nUm pedaço de pau\nUm pé na soleira e um pé na calçada\n\0";
```

(removendo espaços) Escreva código-fonte para remover todos os espaços da string Música. Observação: o resultado deve ser armazenado na string MúsicaSE.

Q14

Escreva, em C, um algoritmo para comparar duas strings (indicadas por A e B) e, caso sejam iguais, será mostrada uma mensagem no dispositivo de saída padrão.

```c
#include <stdio.h>

int main() {
    char A[100];
    char B[100];

    printf("Digite a string A: ");
    fgets(A, sizeof(A), stdin); // Lê até 99 caracteres, incluindo espaços e '\n'
    printf("Digite a string B: ");
    fgets(B, sizeof(B), stdin); // Lê até 99 caracteres, incluindo espaços e '\n'

    // Comparar as strings A e B

    return 0;
}
```

Q15

Escreva o código-fonte, em C, que que converte todos os caracteres de X para caixa alta. Por exemplo, se X for “Brasil”, o resultado da execução da função fará com que X passe a ser “BRASIL”.

Q16

Escreva o código-fonte em C que verifica se uma string é um palíndromo (lê-se igual de trás para frente, desconsiderando espaços e maiúsculas/minúsculas).

Um palíndromo é uma palavra, frase, número ou outra sequência de caracteres que pode ser lida da mesma forma de trás para frente, ignorando espaços, acentuação e pontuação. Em outras palavras, a ordem dos caracteres permanece a mesma, independentemente de ser lida da esquerda para a direita ou vice-versa. São exemplos de palíndromos:

arara, osso, reviver, a base do teto desaba, anotaram a data da maratona,
121 e 20302

Q17

Implemente um programa (na linguagem C) que leia uma frase e duas palavras. O programa deve substituir todas as ocorrências da primeira palavra pela segunda na frase.

Exemplo de entrada:

```
Digite uma frase: Eu adoro programação!
Digite a palavra a ser substituída: adoro
Digite a nova palavra: amo
```

Exemplo de saída:

```
Frase modificada: Eu amo programação!
```

Algoritmos e programação estruturada

Vetores e matrizes

Trabalho prático

Professor Marcelo Eustáquio

Escreva o código fonte para criptografar uma string usando o Algoritmo de Cesar, considerando chave de codificação 3. Ou seja, o A é codificado em D, o B em E, e assim por diante.

O Algoritmo de César (ou Cifra de César) é uma das formas mais simples de criptografia e foi usado por Júlio César na Roma Antiga para enviar mensagens secretas. O princípio básico é deslocar cada letra do texto original um número fixo de posições no alfabeto para criar o texto criptografado.

O funcionamento consiste em escolher um número fixo N chamado chave (ou deslocamento); e, para cada letra do texto é substituída pela letra que está N posições à frente no alfabeto, sendo que, ao final do alfabeto, ele "volta ao início". Espaços, números e caracteres especiais podem ser mantidos sem alteração (depende da implementação).