Resumão P2

ESTRUTURAS DE REPETIÇÃO

`for` como estrutura de repetição:

↳ Sintaxe:

* for (inicialização; condição; incremento/decremento) {
    bloco
  }

↳ Usar for com uma variável contadora:

* Quando sabe quantas vezes o loop seja executado

`While(1)` como estrutura de repetição:

↳ sintaxe:

* while (1) {
    bloco
    if (condição de parada){
      break;
    }
  }

↳ Usar while(1) com break:

* Quando não se sabe quantas vezes o loop precisa ser executado

VETORES (ARRAYS)

1) Declaração e Inicialização

↳ Declaração

* tipo nome[tamanho];

↳ Inicialização: atribuir valores

* Estática:
  → int vetor[5] = {1, 2, 3, 4, 5};
* Dinâmica: (mudar valores depois)
  → int vetor[5];
  vetor[0] = 10;
* Quando o usuário escolhe os dados:
  → int vetor[5];
  for (i=0; i<5; i++) {
    printf("Digite um valor");
    scanf("%i", &vetor[i]);
  }

2) Acesso

↳ **Índices:** Começa em zero, ou seja o primeiro elemento é vetor[0]

* Pode ser acessado por: `printf("%i", vetor[0]);`

↳ **Iteração:** Para acessar todos os elementos de um vetor, pode-se usar loops

```
* for (i=0; i<5; i++) {
    printf("%i", vetor[i]);
  }
```

3) Ordenação

↳ **Método Selection Sort:**

Ex.: ordem decrescente: (tamanho do vetor[3])

```
for (contadorA=0; contadorA < 3; contadorA++) {
    for (contadorB= contadorA+1; contadorB < 3; contadorB++); {
        if ( vetor[contadorA] < vetor[contadorB]) {
            maior = vetor[contadorB];
            vetor[contadorB] = vetor[contadorA];
            vetor[contadorA] = maior;
        }
    }
}
```

Ex.: ordem crescente:

```
for (contadorA=0; contadorA < 3; contadorA++) {
    for (contadorB= contadorA+1; contadorB < 3; contadorB++); {
        if ( vetor[contadorA] < vetor[contadorB]) {
            menor = vetor[contadorA];
            vetor[contadorA] = vetor[contadorB];
            vetor[contadorB] = maior;
        }
    }
}
```

só inverter essas linhas

STRINGS

1) Declaração e Inicialização

↳ **Declaração**

* `char nome[tamanho];`

↳ **Inicialização:** atribuir valores

* Estática:
  → `char str[] = "Ola, mundo!";`
* Dinâmica: (mudar valores depois)
  → `Char str[20];`
  `str[0] = 'j';`

2) Funções de manipulação

↳ strlen ( ) ;
- ↳ usada pl calcular o comprimento da string excluindo o caracter nulo (\0)
- ↳ biblioteca <string.h>
- ↳ sintaxe: strlen(nome);
- ↳ bastante utilizada como condição nos loops que acessam todo o vetor de string; Para isso atribui a uma variável. Ex. tamanho = strlen (nome)

↳ strcpy ( );
- ↳ usada pl copiar todos os caracteres de uma string pl outra

↳ strcat ( );
- ↳ usada pl juntar duas strings
- ↳ Ex.: strcat (destino, origem)
  - ↳ a string destino vai receber a string origem

↳ strcmp ( );
- ↳ usada pl comparar duas strings
- ↳ retorna um valor negativo se a primeira string for menor, um valor positivo se a primeira for maior, e zero se as duas forem iguais
- ↳ Ex.:

```
char str1 [] = "hello";
char str2 [] = "world";
int resultado = strcmp (str1, str2);
if (resultado < 0){
    printf("str1 é menor que str2");
} else if (resultado>0) {
    printf("str1 é maior que str2");
  }else {
     printf(" as strings são iguais");
  }
```

↳ isalpha ( );
- ↳ biblioteca <ctype.h>
- ↳ Usada p/ verificar se o caracter é uma letra

↳ ispunct ( );
- ↳ biblioteca <ctype.h>
- ↳ Usada p/ verificar se o caracter é uma pontuação

↳ tolower ( );
- ↳ usada pl converter uma letra maiúscula em minúscula
- ↳ ctype

↳ toupper ( );
- ↳ usada pl converter uma letra minúscula em maiúscula
- ↳ ctype

↳ fgets();
  ↳ comando "leia" p/ strings
  ↳ sintaxe: fgets(string, tamanho, stdin);
    fgets(nome, 8, stdin);

↳ puts();
  ↳ comando "escreva" p/ strings
  ↳ Obs.! Uniliga só quando quero imprimir a string, não substitui o comando printf.

3) Iteração:
↳ Percorrer todos os caracteres da string c/ loops
↳ Ex.: for(i=0; i<strlen(nome); i++) {
    printf("%c", nome[i]);
  }