CURSO SUPERIOR DE ADS

Strings

Prof. Fernando Marlon Soares Figueiredo

Disciplina: Programação Estruturada

SECTRAS

ADS

Aula de Hoje

- Manipulação de Strings.
- Principais funções da biblioteca string.h

Introdução

- Até agora, as variáveis que utilizamos para programar eram números inteiros e reais. O tipo char não foi muito utilizado pois trabalhar com somente um caractere limita as possibilidades.
- A partir de agora iremos trabalhar com variáveis que armazenam textos para tornar nossos programas mais interessantes.

Conceitos básicos

- Nas linguagens de programação modernas, o tipo de dado específico para texto é chamado de **string**.
- No entanto, em algumas linguagens, incluindo C, strings não fazem parte dos tipos de dados básicos, devendo ser construídos a partir de alternativas.
- No caso da linguagem C, as strings são definidas a partir de um vetor de caracteres (vetores com elementos do tipo char) especiais, que possuem o caractere \0 ao final do vetor, indicando o final da string

Exemplo

- A string contém o texto “azul”, sendo na verdade um vetor composto pelos caracteres ‘a’, ‘z’, ‘u’, ‘l’ e \0.
- Como strings são definidas por vetores, todos os conceitos trabalhados também servem para strings.
- A sintaxe para a declaração de strings consiste em criar um vetor de caracteres:

```
char nome[tamanho + 1]
```

Conceitos Básicos

- Observe que o tamanho do vetor deve possuir um espaço a mais do que o planejado para que contenha o caractere \0.
- Para criar constantes strings, coloca-se o texto entre aspas duplas, conforme é passada a mensagem a ser impressa na função printf.

Conceitos Básicos

- Para **saída de dados** de string, basta utilizar **%s** na função print, da mesma forma como outros tipos de dados.
- Para **entrada de dados**, será utilizada uma nova função, visto que a função scanf não captura corretamente toda string.
- Vamos utilizar a função **fgets**, que armazena tudo o que o usuário digitar até que seja pressionada a tecla enter.

Sintaxe do fgets

A sintaxe da função é:

fgets(nome, tamanho, stdin)

Onde

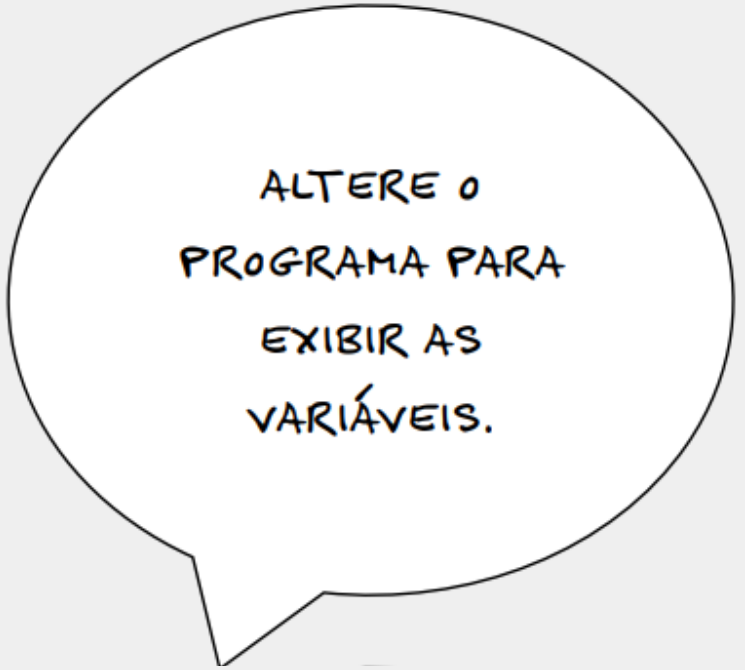
- **nome**: corresponde a variável String.
- **tamanho**: é o número máximo de caracteres da string
- **stdin**: indica que a função deve ler os dados da linha de comando para o teclado. Vai ler todos os caracteres e armazenar na variável quando o caractere **\n** for encontrado ou chegar ao **tamanho-1** caracteres, o que ocorrer primeiro.

O que o programa faz?

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char nomeCompleto[30], cpf[12];
6     int idade;
7
8     printf("Digite seu nome completo:");
9     fgets(nomeCompleto, 30, stdin);
10
11     printf("Digite sua idade:");
12     scanf("%d", &idade);
13     getchar();
14
15     printf("Digite seu cpf:");
16     fgets(cpf, 12, stdin);
17
18     return 0;
19 }
```

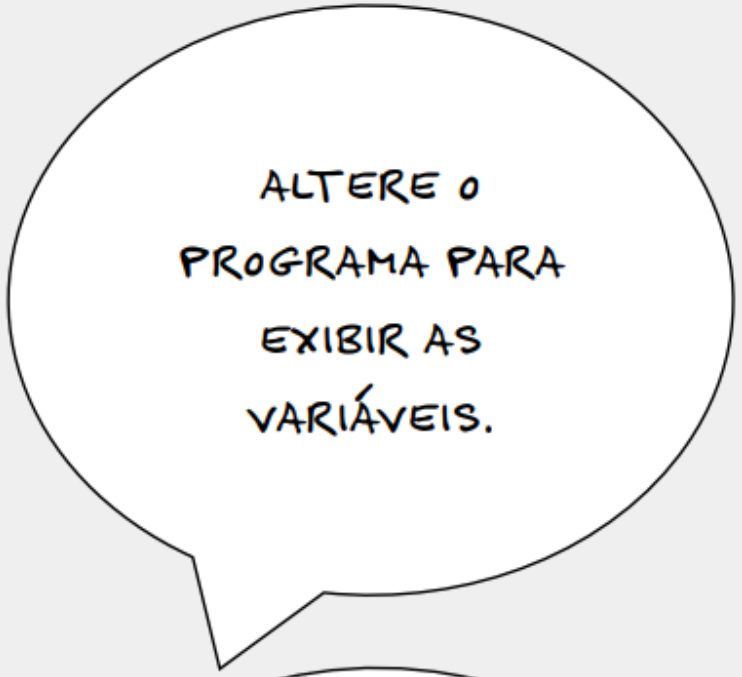
O que o programa faz?

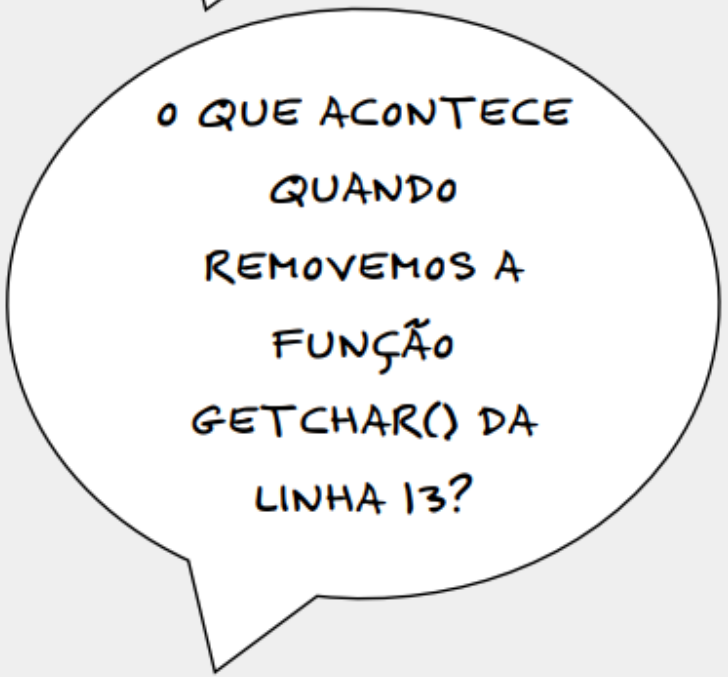
```
1 #include <stdio.h>
2
3 int main()
4 {
5     char nomeCompleto[30], cpf[12];
6     int idade;
7
8     printf("Digite seu nome completo:");
9     fgets(nomeCompleto, 30, stdin);
10
11     printf("Digite sua idade:");
12     scanf("%d", &idade);
13     getchar();
14
15     printf("Digite seu cpf:");
16     fgets(cpf, 12, stdin);
17
18     return 0;
19 }
```



O que o programa faz?

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char nomeCompleto[30], cpf[12];
6     int idade;
7
8     printf("Digite seu nome completo:");
9     fgets(nomeCompleto, 30, stdin);
10
11     printf("Digite sua idade:");
12     scanf("%d", &idade);
13     getchar();
14
15     printf("Digite seu cpf:");
16     fgets(cpf, 12, stdin);
17
18     return 0;
19 }
```





Problema ao usar scanf após o fgets

- A leitura do texto parece ser aparentemente ignorada.
- O scanf lê apenas o dado digitado, deixando o **\n** (o enter) pendente no **stdin**.
- Assim, a função **fgets** pensa que o **\n** é o conteúdo do texto digitado e o programa passa para o próximo comando, quando na verdade a pessoa nem teve a oportunidade de digitar o texto.
- Este problema pode ser resolvido com a introdução da função **getchar()**, que é responsável por ler um caractere do **stdin**, conseguindo remover o **\n** que estava atrapalhando.

Exercício 1

- Leia um texto pela entrada padrão com no máximo 99 caracteres. Em seguida imprima o número de caracteres digitados.

- Dica: percorra o vetor até encontrar o caractere terminador '\0'

Exercício 1 - Resposta

main.c > ...

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

int main(void) {
  char texto[100];

  printf("Digite um Texto: ");
  scanf("%s", texto);

  for(int i = 0; texto[i] != '\0'; i++){
    printf("%d\n", i);
  }

  return 0;
}
```

Exercício 2

- Declare duas strings com capacidade para 20 caracteres.
- Leia pela entrada padrão a primeira string.
- Em seguida, copie o texto da primeira string para a segunda.
- Imprima no final o conteúdo das duas strings.

- Atenção: não utilize a função strcpy.

Exercício 2 - Resposta

C main.c > ...

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

int main(void) {
  char s1[21], s2[21];
  int i;

  printf("Digite uma Palavra: ");
  fgets(s1, 21, stdin);

  for(i = 0; s1[i] != '\0'; i++){
    s2[i] = s1[i];
  }

  s2[i] = '\0';

  printf("S1 = %s\n S2 = %s\n", s1, s2);

  return 0;
}

```

Principais funções da biblioteca string.h

Biblioteca string.h

- As strings não são tipos de dados, são vetores de caracteres. Por isso precisamos de mais linhas de código para resolver problemas simples, como copiar o valor de uma string para outra.
- Este tipo de problema envolve saber quantos elementos a string possui, antes de copiar os elementos para uma outra.
- Felizmente a linguagem C provê uma biblioteca (string.h) para ajudar os programadores a trabalharem com strings.

Função strcpy

- Função que equivale ao operador de atribuição para valores numéricos. Ela copia o valor de uma string para outra.

Sintaxe:

- strcpy (string destino, string fonte)

- A função copia caractere a caractere do vetor da string fonte para o vetor da string destino, até que seja encontrado o caractere \0 indicando o final da string. É encargo do programador controlar os tamanhos máximos dos vetores para que as cópias sejam feitas corretamente.

Função strcmp

- Função que compara dois textos, verificando se os valores são iguais ou diferentes. A função compara caractere a caractere duas strings até que um \0 seja encontrado.
- O valor **zero é retornado caso os valores sejam iguais** e outro valor caso sejam diferentes.

Sintaxe:

- strcmp(string 1, string 2)
- Seu uso costuma ocorrer associado a uma condicional, verificando se o resultado é igual ou diferente de zero.

O que o programa faz?

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int main( )
5 {
6     char fruta1[30], fruta2[30];
7
```

```
8     printf("Digite uma fruta:");
9     fgets(fruta1, 30, stdin);
10     printf("Digite outra fruta:");
11     fgets(fruta2, 30, stdin);
12
13     printf("Fruta1: %s \n", fruta1);
14     printf("Fruta2: %s \n", fruta2);
```

```
16     if(strcmp(fruta1, fruta2) == 0)
17     printf("As duas frutas digitadas sao iguais! \ n");
18
19     strcpy (fruta1, fruta2);
20
21     printf("Fruta1: %s \n", fruta1);
22     printf("Fruta2: %s \n", fruta2);
23
24     return 0;
25 }
```

Função strlen

- Provê uma maneira de descobrir o tamanho atual das strings, ou seja, com quantos caracteres ela está ocupada no momento.
- Seu funcionamento consiste em utilizar uma variável contadora cujo valor é incrementado a cada caractere encontrado. O caractere especial \0 não entra na contagem.

Sintaxe:

- strlen(string)
- O valor retornado deve ser armazenado em uma variável do tipo inteiro, que conterá a quantidade de caracteres atuais da string.

Função strcat

- Função que concatena dois textos, ou seja, emenda um texto logo após o outro. A função copia a segunda string no final da primeira, que tem, por sua vez, seu valor modificado.

Sintaxe:

- strcmp(string destino, string fonte)

O que o programa faz?

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

int main( )
{
    char nome[30];
    int posCaractFinal, numCaracteres, indiceFinal;
    printf("Digite seu nome:");
    fgets(nome, 30, stdin);
    indiceFinal = strlen(nome) -1;
    nome[indiceFinal] = '\0';

    strcat(nome, ", muitas felicidades!");

    numCaracteres = strlen(nome);

    printf("Mensagem: %s \n", nome);
    printf("Caracteres na mensagem: %d", numCaracteres);

    return 0;
}
```

Exercício 3

Leia duas strings. Se as strings forem iguais escreva “strings iguais”. Caso contrário, concatene as duas strings e imprima a string resultante.

Exercício 3 - Resposta

C main.c > ...

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

int main(void) {
  char s1[21], s2[21];


  printf("String 1 = ");
  scanf("%s", s1);
  printf("String 2 = ");
  scanf("%s", s2);

  if (strcmp(s1,s2) == 0){
    printf("Strings Iguais!\n");
  }
  else {
    strcat(s1, s2);
    printf("%s\n",s1);
  }
  return 0;
}

```

Quando usar scanf ou fgets para ler string?

- A escolha entre fgets e scanf em C depende do contexto e dos requisitos do seu programa. Ambas as funções têm finalidades diferentes e são adequadas para situações diferentes.

Leitura de Linhas de Texto:

- fgets é geralmente preferível quando você deseja ler uma linha inteira de texto, incluindo espaços em branco e caracteres especiais. Isso é útil, por exemplo, ao ler nomes completos ou frases.
- scanf lê até o primeiro espaço em branco ou quebra de linha, o que pode resultar em comportamento inesperado ao ler palavras compostas ou frases com espaços.

Exercício 4

Faça um programa que leia um nome e imprima as 4 primeiras letras do nome.

```
PS C:\Users\Desktop\Desktop\PE 2025.2\output> & .\'main.exe'
Digite seu nome: fernando
Nome: fernando
fern
```

Exercício 4 - Resposta

C main.c > ...

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

int main(void) {

    char nome[21];

    printf("Digite seu nome: ");
    fgets(nome, 21, stdin);
    printf("Nome: %s",nome);

    for (int i = 0; i < 4; i++)
    {
        printf("%c", nome[i]);
    }



  return 0;
}

```

Exercício 5

- Faça um programa que conte o numero de 1's que aparecem em um string.
- Exemplo: "0011001" -> 3

```
PS C:\Users\Desktop\Desktop\PE 2025.2\output> cd 'c:\Users\Desktop\Desktop\PE 2025.2\output'
PS C:\Users\Desktop\Desktop\PE 2025.2\output> & .\'main.exe'
Total de 1's: 3
```

Exercício 5 - Resposta

main.c > ...

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

int main(void) {

    char nome[21] = {'1','0','1','0','1','0', '\0'};
    int len = strlen(nome);
    int cont = 0;

    for (int i = 0; i < len; i++)
    {
        if (nome[i] == '1')
        {
            cont++;
        }

    }

    printf("Total de 1's: %d", cont);

  return 0;
}

```

Exercício 6

- Faça um programa que conte a ocorrência de vogais em um texto digitado pelo usuário.
- Imprima o resultado ao final do programa.

```
PS C:\Users\Desktop\Desktop\PE 2025.2\output> cd 'c:\Users\Desktop\Desktop\PE 2025.2\output'
PS C:\Users\Desktop\Desktop\PE 2025.2\output> & .\'main.exe'
Digite sua Frase: Bom dia
Bom dia
Total de Vogais: 3
```

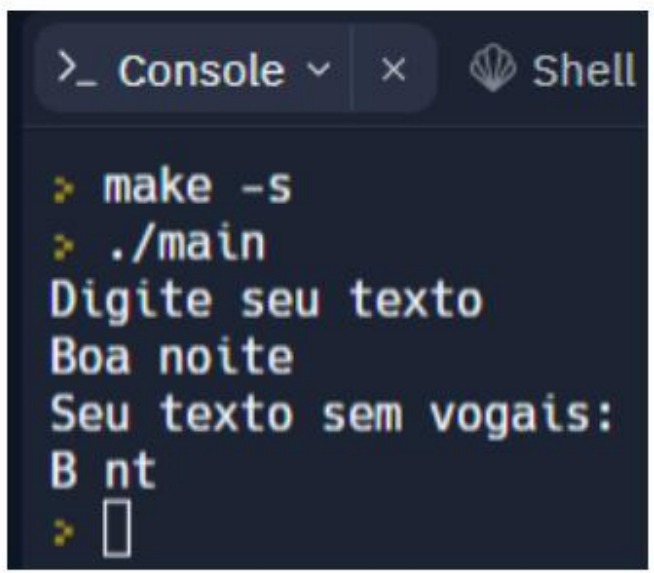
Exercício 6 - Resposta

C main.c > main(void)

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

int main(void) {

    char nome[30];
    int cont = 0;

    printf("Digite sua Frase: ");
    fgets(nome, 30, stdin);

    int len = strlen(nome);
    printf("%s", nome);

    for (int i = 0; i < len; i++)
    {
        if ((nome[i] == 'a') || (nome[i] == 'e') || (nome[i] == 'i') || (nome[i] == 'o') || (nome[i] == 'u'))
        {
            cont++;
        }

    }

    printf("Total de Vogais: %d", cont);

  return 0;
}
```

Exercício 7

Faça um programa que receba do usuário uma string.
O programa imprime a string sem suas vogais.



Exercício 8

Faça um programa que peça ao usuário para digitar uma palavra e em seguida imprima-a de trás pra frente.

Exercício 9

- Faça um programa que implemente e teste uma versão própria das funções strcmp e strcpy.
- Ou seja, o programa deve informar caso sejam iguais ou diferentes e copiar o conteúdo de uma string para outra.
- Assuma que ambas as strings de teste possuem o mesmo tamanho.

Exercício 10

- Faça um programa que implemente e teste uma versão própria das funções strlen e strcat no programa principal.
- Ou seja, o programa deve contar o número de elementos da string e acoplar uma string ao final da outra.
- Assuma que uma string possui o dobro do espaço da outra.

Exercício 11

- A tabela ASCII apresenta os valores verdadeiros dos caracteres utilizados em linguagens de programação, consistindo de 128 símbolos, entre eles caracteres numéricos, letras maiúsculas e minúsculas, entre outros símbolos. Os valores das letras minúsculas ficam entre os números 97 e 122, por exemplo.
- Faça um programa que codifique um texto digitado pelo usuário a partir da soma de duas unidades de cada uma das letras e imprima o texto codificado na tela.
- Ao final, decodifique o texto e imprima seu valor original na tela.

Exercício 12

- Escreva um programa que leia duas palavras e diga qual deles vem primeiro na ordem alfabética.
- Dica: ‘a’ e menor do que ‘b’.