Cadeia/Vetores de Caracteres: Strings

Prof. Gabriel Barbosa da Fonseca
Email: gbfonseca@sga.pucminas.br

Introdução

Strings (cadeias de caracteres) são muito utilizadas para guardar:

- nomes de arquivos
- nomes de usuários
- qualquer informação baseada em caracteres.

String nada mais é que uma estrutura de dados para representar “texto” ao invés de simplesmente caracteres separados.

Introdução

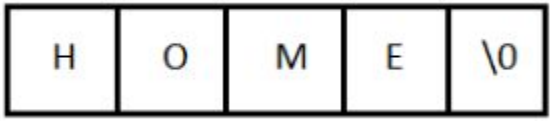
A linguagem C utiliza **vetores de char** para armazenar uma cadeia de caracteres, onde cada posição representa um caractere.

A diferença básica entre strings e outros vetores é que a linguagem de programação C **indica o fim do vetor de strings através do acréscimo do caractere NULL (‘\0’) no final do String.**

Definição

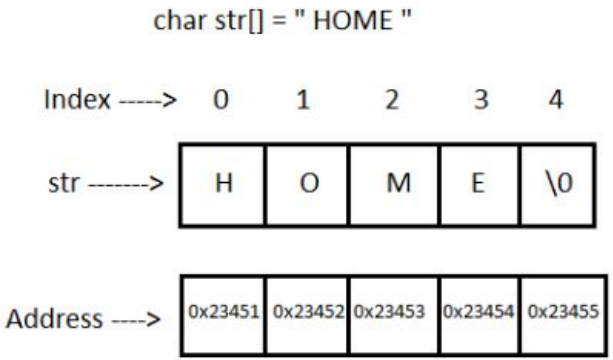
String é uma sequência de caracteres tratada como um único item de dados e terminada por um caractere nulo '\0'.

Uma **string** é, na verdade, um **vetor unidimensional de caracteres** em linguagem C.



Introdução

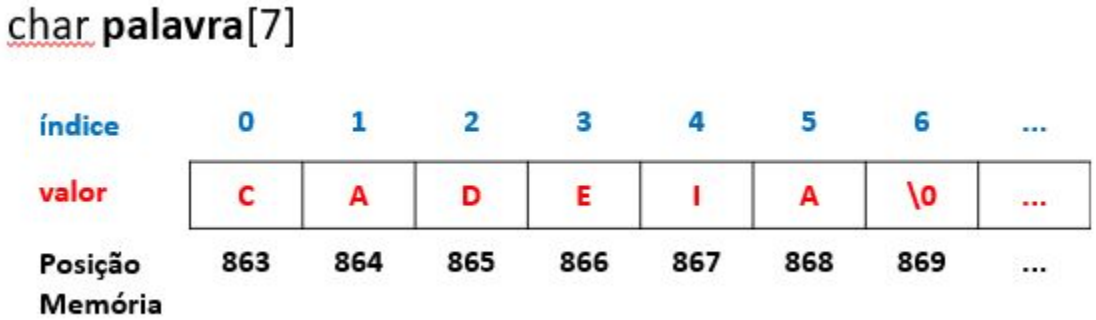
OBS: Deve-se declarar sempre o vetor **com uma posição a mais** para armazenar o caractere nulo (‘\0’), que não precisa ser armazenado manualmente. Isso é feito automaticamente pelo compilador (se utilizarmos as funções pré implementadas na linguagem)



Introdução

A variável **palavra**, quando é declarada pode ocupar qualquer posição na memória.

Entretanto, todas as posições do vetor ocupam espaços de memória adjacentes, sendo que cada caractere ocupa 1 byte.



Inicialização

Inicialização no momento da declaração:

```
//Inicialização de uma string, com atribuição de valor
char nome1[ ]= {'P', 'r', 'o', 'g', 'r', 'a', 'm', 'a', '\0'};
//Imprimindo a string nome1 na tela
printf("%s",nome1);
```

A variável nome1 recebeu as letras separadamente (inclusive o caractere nulo). Por isso, cada uma das letras está envolvida por apóstrofos (' ‘). Essa é a maneira de identificar um caractere isoladamente.

Inicialização

Inicialização alternativa no momento da declaração:

```
int main(void) {
  char nome2[ ]= "Programa";
  printf("%s", nome2);
}
```

A variável nome2 recebeu uma palavra entre aspas (" "), recebendo automaticamente o caractere nulo. Esta é a maneira de identificar uma cadeia de caracteres.

Exemplo

```
int main(void) {
  char nome1[] = {'P', 'r', 'o', 'g', 'r', 'a', 'm', 'a', '\0'};
  int vec[] = {1, 2, 3, 5, 6, 4, 3};
  printf("%s", nome1);
  printf("\nTamanho de vec: %d"
    "\nTamanho de nome1:
%d",sizeof(vec),sizeof(nome1));
}
```

Saída:
Programa
Tamanho de vec: 28
Tamanho de nome1: 9

Exemplo

```
int main(void) {
  char nome1[] = {'P', 'r', 'o', 'g', 'r', 'a', 'm', 'a', '\0'};
  int vec[] = {1, 2, 3, 5, 6, 4, 3};
  printf("%s", nome1);
  printf("\nTamanho de vec: %d"
    "\nTamanho de nome1:
%d",sizeof(vec),sizeof(nome1));
}
```

Saída:
Programa
Tamanho de vec: 28
Tamanho de nome1: 9

Exemplo

```
// valid
char name[13] = "StudyTonight";
char name[10] = {'c','o','d','e','\0'};

// Illegal
char ch[3] = "hello";
char str[4];
str = "hello";
```

Manipulação

Inicialização por meio da atribuição (depois da declaração):

```
char vet1[10], vet2[5];
strcpy(vet1, “Programa”);
```

A variável vet1 recebeu um valor constante (a palavra Programa).

Usando duas strings, uma será copiada na outra. Ou seja, o conteúdo da variável vet2 foi copiado na variável vet1.

```
char vet1[10], vet2[5];
strcpy(vet1, vet2);
```

Manipulação

Lendo uma string com **scanf**

```
char frase[100];
printf(“Digite um texto: ”);
scanf(“%s”, frase);
```

O comando scanf consegue armazenar valores vindos do teclado na variável frase. No caso de uma cadeia de caracteres , esse comando consegue armazenar todos os símbolos digitados até a primeira ocorrência do **espaço em branco OU <ENTER>**.

Solução: ditar conjunto de código de conversão %[..] que pode ser usado para ler uma linha até o ENTER, contendo uma variedade de caracteres, incluindo espaços em branco.

```
char str[20];
printf("Enter a string");
scanf("%[^\n]", &str);
printf("%s", str);
```

Manipulação

Inicialização

```
char frase[100];
printf(“Digite um texto: ”);
scanf(“%s”, frase);
```

A função gets() armazena na variável frase todos os símbolos digitados até a ocorrência do ENTER.
Esta função exige a utilização da biblioteca stdio.h.

Exibição

```
char frase[100];
printf(“\n Digite um texto: ”);
gets(frase);
printf(“\n A frase digitada foi: ”);
puts(frase);
```

A função puts() é usada para imprimir uma cadeia de caracteres inicializada com o uso da função gets().

Manipulação

Exibição
Também é possível imprimir a cadeia de caracteres com o comando printf, nesse caso, vamos precisar usar o %s (string)

```
char frase[100];
printf(“\n Digite um texto: ”);
gets(frase);
printf(“\n A frase digitada foi: %s”, frase);
```

Manipulação: STRING.H

Como todas as cadeias de caracteres são variáveis compostas homogêneas (vetor ou matriz) deve-se utilizar funções específicas.
Essas funções fazem parte da biblioteca **string.h**
Algumas delas:

strlen( )
strcpy( )
strcat( )
strcmp( )
strupr( )
strlwr( )

STRLEN

A função **strlen (string length)** retorna para a variável Tamanho o número de caracteres da cadeia str1.

```
int Tamanho;
char str1[20];
Tamanho = strlen(str1);
```

STRCPY

A função **strcpy (string copy)** copia a cadeia str2 na cadeia str1, inclusive o ‘\0’ (NULL). Sendo assim, a cadeia str1 será substituída pela cadeia str2.

```
strcpy (str1, str2);
```

STRNCPY

A função **strncpy** copia os n primeiros caracteres da cadeia str2 para a cadeia str1 sem incluir o ‘\0’ (NULL).

```
strncpy (str1, str2, n);
```

STRCMP

Compara duas cadeias de caracteres e retorna um número inteiro para a variável Resultado, que pode ser:

Zero: se as duas cadeias forem iguais

Um número menor que 0: se a cadeia cadeia1 for alfabeticamente menor que cadeia2

Um número maior que 0: se a cadeia cadeia1 for alfabeticamente maior que cadeia2

Essa função **strcmp** considera letras maiúsculas como sendo símbolos diferentes de letras minúsculas.

Resultado = strcmp (cadeia1, cadeia2);

Essa função **stricmp** considera letras maiúsculas e minúsculas como sendo símbolos iguais.

Resultado = stricmp (cadeia1, cadeia2);

Outras funções

Outras Funções em (string.h):

- **toupper**(caracter) – converte o caracter em maiúsculo.
- **strupr**(string) – converte a string para maiúsculo
- **tolower**(caracter) – converte o caracter em minúsculo.
- **strlwr**(string) – converte a string para minúsculo

Para casa!

Entrar no link:

https://pt.wikipedia.org/wiki/Stdlib.h

e estudar as funções de **conversão de tipos** presentes nesta biblioteca.

Ler a documentação **não oficial** da biblioteca string.h em:

http://linguagemc.com.br/a-biblioteca-string-h/

E estudar as funções presentes no site.