Strings - Cadeia de Caracteres

Apresentação

A maioria das linguagens diferenciam um caractere de uma string - ou cadeia de caracteres. Um caractere é um símbolo único, número, letra, pontuação ou algum caractere especial. Uma string é um conjunto de caracteres que representam uma palavra, frase ou texto.

Ainda, a maioria das linguagens possuem um tipo básico (já definido na própria linguagem) para representar strings. O que **não** é o caso de C.

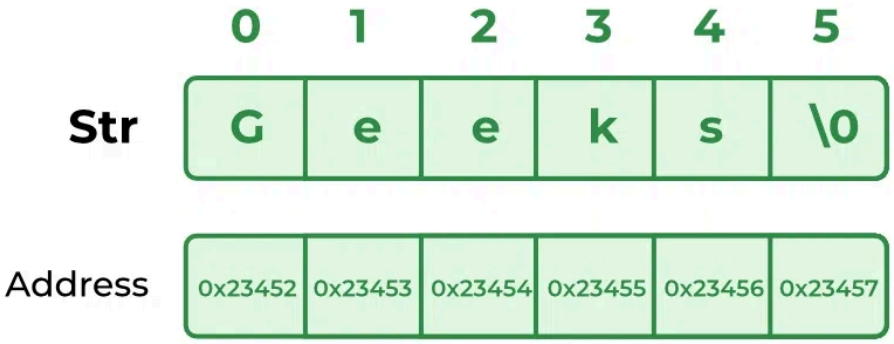
Em C, esta representação é bem *rústica.* Uma string é uma cadeia de caracteres e na linguagem C elas são implementadas assim, um array de char.

Para criar uma string, podemos declarar um array de char e depois ler esta string da entrada padrão, usando scanf, gets ou fgets - lembrando que a stream de entrada é processada separando os dados por espaços ‘ ‘ e devemos cuidar isso.

Novamente, uma string é um array. Cada posição possui um caractere (char). Assim como todo tipo de dado, precisamos reservar memória para a string. Ao criar um array de 50 posições, poderemos então armazenar até 49 - sim, um a menos - caracteres.

A string possui um caractere especial que delimita o final dela. Por exemplo, digamos que reservei 100 caracteres para guardar um nome do usuário. O usuário digita seu nome, com 5 letras. Assim, como saberei até onde na memória reservada está armazenado o nome do usuário? Após o nome, um caractere de final de string é armazenado (pela entrada, como o gets ou pelas funções de manipulação de string). Este caractere especial permite então saber quais caracteres da memória reservada compõem o nome. O restante dos caracteres pode ter valor aleatório, assim como todas variáveis criadas na linguagem C.

O caractere que simboliza o final da string é o “Null character” com o código ASCII 0. Ele é adicionado ao final do texto ‘válido’. https://www.ascii-code.com/0



Por ser um array, podemos então acessar cada índice da string, alterá-lo ou mesmo o imprimir. No código abaixo, lemos uma string e imprimimos cada caractere separadamente.

```c
#include <stdio.h>
int main(){

    char nome[10]; //string com no maximo 10-1 caracteres.

    printf("Informe o primeiro nome: ");
    scanf("%s", nome);

    for(int i = 0; i < 10; i++){
        printf("Char na posicao %d : %c (%d)\n", i, nome[i], nome[i]);
    }
}
```

Neste caso, a leitura com o scanf funciona se não forem inseridos espaços na string. Não é necessário usar o & pois a string é um array e ele é passado como referência, ou seja, o que acontece no escopo interno da função reflete fora dela. Para leitura de string, recomenda-se o gets (que não é seguro) ou o fgets que é mais seguro.

O printf(“%s”, …) imprime a string até o seu delimitador (caractere null), assim, o que está armazenado nas posições que sobram não vão para a tela. Podemos alterar caractere por caractere de uma string, através de seu índice no array:

```c
...
scanf("%s", nome);
```

```
nome[0] = 'A';
nome[1]= 'A';
for(int i = 0; i < 10; i++){
...
```

Mas a linguagem C já fornece algumas funções para manipulação de strings. É recomendado utilizar as funções já definidas na linguagem pois provavelmente estão implementadas da maneira mais otimizada, ou seja, o programa terá uma melhor performance. Além do mais, não é tão interessante ficar “reinventando a roda”, a não ser que seja para fins de estudo.

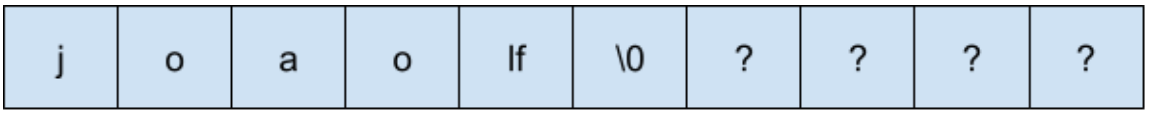
O código abaixo mostra como ler strings com fgets, saber seu tamanho e concatenar (unir) duas strings.

O fgets inclui antes do caractere null (0) um caractere de valor 10, que significa 'fim da linha'. Este caractere é armazenado. uma maneira de tratar isso é remover ele do final, a 'força bruta':

Importante: Não confundir o caractere null (\0) com o caractere do dígito 0. O código do caractere \0 é ‘0’, enquanto o código ASCII do dígito 0 é 48!

```
fgets(nome, 9,stdin);
```

Digamos que o nome lido com o fgets seja 'joao'. Este ficaria armazenado no vetor:

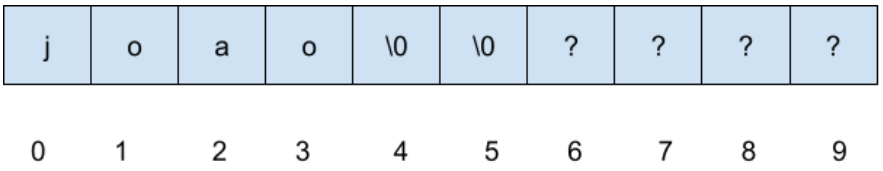


0 1 2 3 4 5 6 7 8 9

Ou seja, teremos o fim de linha (lf) e depois o fim de string (\0). o strlen retornará 5, contando o lf. Veja bem: o tamanho da string é 5, mas os índices começam em 0, então, o final da string está na posição tamanho-1, no exemplo, a **4**. Removemos o ‘lf’ desta posição e adicionamos o caractere nulo (0):

```
nome[strlen(nome)-1] = '\0';
```

Após isso, a string terá duas marcas de final, mas a considerada é sempre a primeira (que aparece no menor índice):



Outra função para manipulação de strings é a strcat(*destino*, *outra_string*). Ela concatena (adiciona *outra_string*) ao final de destino.

```c
#include <stdio.h>
#include <string.h>
int main(){

    char nome[10]; //string com no maximo 10-1 caracteres.
    char sobreNome[10];

    char nomeCompleto[20] ="\0";

    printf("Informe o primeiro nome: ");
    fgets(nome, 9,stdin);
    for (int i = 0; i < 10; i++){
        printf("char %d = %c (%d) \n", i, nome[i], nome[i]);
    }
    //o fgets inclui antes do caractere null (0) um caractere de valor 10, que significa 'fim da linha'.
    //este caractere é armazenado. uma maneira de tratar isso é remover ele do final, a 'força bruta':

    // digamos que o nome seja 'joao'. ficara armazenado no vetor:
    //'j' 'o' 'a' 'o' '\lf' '\0' '?' '?' '?' '?'
    //ou seja, teremos o fim de linha e depois o fim de string.
    nome[strlen(nome)-1] = 0;

    printf("----------------------\n");
    for (int i = 0; i < 10; i++){
        printf("char %d = %c (%d) \n", i, nome[i], nome[i]);
    }

    printf("Nome de familia: ");
    fgets(sobreNome, 9,stdin);


    printf("Seu nome possui %d letras\n", strlen(nome));
```

```
    printf("Seu sobrenome possui %d letras\n", strlen(sobreNome));

    strcat(nomeCompleto, nome); // concatena 'nome' ao final de 'nomeCompleto
    printf("::: %s\n",nomeCompleto);

    strcat(nomeCompleto, " ");//adiciona um espaco...
    printf("::: %s\n",nomeCompleto);

    strcat(nomeCompleto, sobreNome);//adiciona um espaco...
    printf("::: %s\n",nomeCompleto);

}
```

A função strcpy é útil para copiar uma string para outra. Em determinadas situações, podemos precisar alterar o valor de uma string, e, ao invés de definir uma posição do array por vez, podemos copiar uma string completamente para outra.

A linguagem C, na biblioteca string.h possui várias funções. Uma delas é a função de comparação de strings, strcmp. Esta função recebe como parâmetro duas strings e informa qual delas é maior lexicograficamente, ou seja, em ordem alfabética.

A função tem a forma strcmp(string1, string2) e retorna 0 caso as duas sejam iguais, um valor maior que zero caso a string1 seja maior do que a string2, ou menor que zero caso a string1 seja menor que string2.

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

int main(){

    char string_1[25]; //inicializa vazia.

    char string_2[25];

    strcpy(string_1, "Xalor inicial");

    strcpy(string_2, "Xalor inicial");
```

```
    int comp = strcmp(string_1, string_2);

    if(comp < 0){
        printf("A string '%s' eh 'menor' que '%s'",string_1, string_2);
    }

    if (comp == 0) {
    printf("A string '%s' eh igual a '%s'",string_1, string_2);
    }

    if (comp > 0){
        printf("A string '%s' eh 'maior' que '%s'",string_1, string_2);
    }
}
```

Exercícios

1. Crie uma função que recebe uma string como parâmetro e escreve ela ao contrário.
2. Crie uma função que recebe uma string como parâmetro e converte todos seus caracteres para maiúsculo. https://www.w3schools.com/c/c_ref_ctype.php
3. Crie um programa que escreve a letra de uma música, linha por linha. O usuário deve pressionar uma tecla (<enter>) para que a próxima linha seja exibida. Você pode usar uma “matriz”: char letra[NUMERO_LINHAS][COLUNAS].
4. Crie uma função que “codifica” uma string, trocando letras por números e símbolos. Por exemplo, “SAPATO” poderia ficar “$4P4T0”.