Algoritmos II

MANIPULAÇÃO DE CADEIAS DE CARACTERES (STRINGS)

Conceito

- Uma string é uma seqüência de caracteres utilizada para o armazenamento de texto.
- Na linguagem C strings são vetores de caracteres que possuem um caracter que indica o término de seu conteúdo, o caracter nulo ‘\0’.

Declaração de strings

- Como a string possui o caracter nulo para delimitar o final do seu conteúdo, o tamanho da string deve ser definido com um caracter a mais do que será efetivamente necessário.

char identificador-da-string [tamanho+1];

- Exemplo:
  - char vetc [6];
    - Vetor de caracteres (string) de tamanho 6.

Inicialização de strings

- Uma string pode ser inicializada na sua declaração com uma sequência de caracteres entre chaves e separadas por virgula.
  - char vetc[6]= {'T', 'e', 'x', 't', 'o', '\0'};
- Uma string pode também ser inicializada por uma sequência de caracteres entre aspas duplas.
  - Neste caso, não é necessário o uso de aspas simples e virgulas, o compilador C coloca automaticamente o '\0' no final.
    - char vetc[6] = “Texto”;

Inicialização de strings

- Assim como vetores e matrizes, na inicialização de uma string o seu tamanho pode ser omitido. O compilador vai verificar e considerar o tamanho declarado na inicialização.
  - char vetc[ ] = “Texto”; /* vetor não-dimensionado, o compilador coloca automaticamente o ‘\0’ no final */

Funções de entrada e saída (<stdio.h>)

- **gets(s)** - Lê uma string do dispositivo de entrada padrão e armazena esta string em s. Não é uma função segura, pois o tamanho da string não é especificado.
- **fgets(s, TAM, stdin)** - Lê uma string de tamanho TAM do dispositivo de entrada padrão e armazena esta string em s.
- **puts(s)** - Imprime a string s no dispositivo de saída padrão.
- **sscanf(s, "expressão-controle", end1, end2, ...)** - Faz a leitura formatada em uma string s
- **sprintf(s, "expressão-controle", arg1, arg2, ...)** - Faz a escrita formatada em uma string s

Funções de entrada e saída (<stdio.h>)

```
char string[ ]= "2 20 200"; /* inicializa a string str */
sscanf(string, "%d %d %d", &i, &j, &k); /* armazena 2 em i, 20 em j e 200 em k */
```

```
char string[10];
int i=2;
sprintf(string,"%d", i); /*armazena 2 em str */
```

- Observe que foram usados exemplos numérico como valores, neste caso o compilador interpreta os numeros como sendo uma string. Obcerve as notações abaixo.
  - 1 para o compilador poder ser um numero inteiro ou real.
  - '1' para o compilador é o alfanumerico (caracter) char.
  - "11..." para o compilador é uma string de conteudo 11...
  - logo deve-ser ter muita atenção ao uso de aspas para manipular strings.

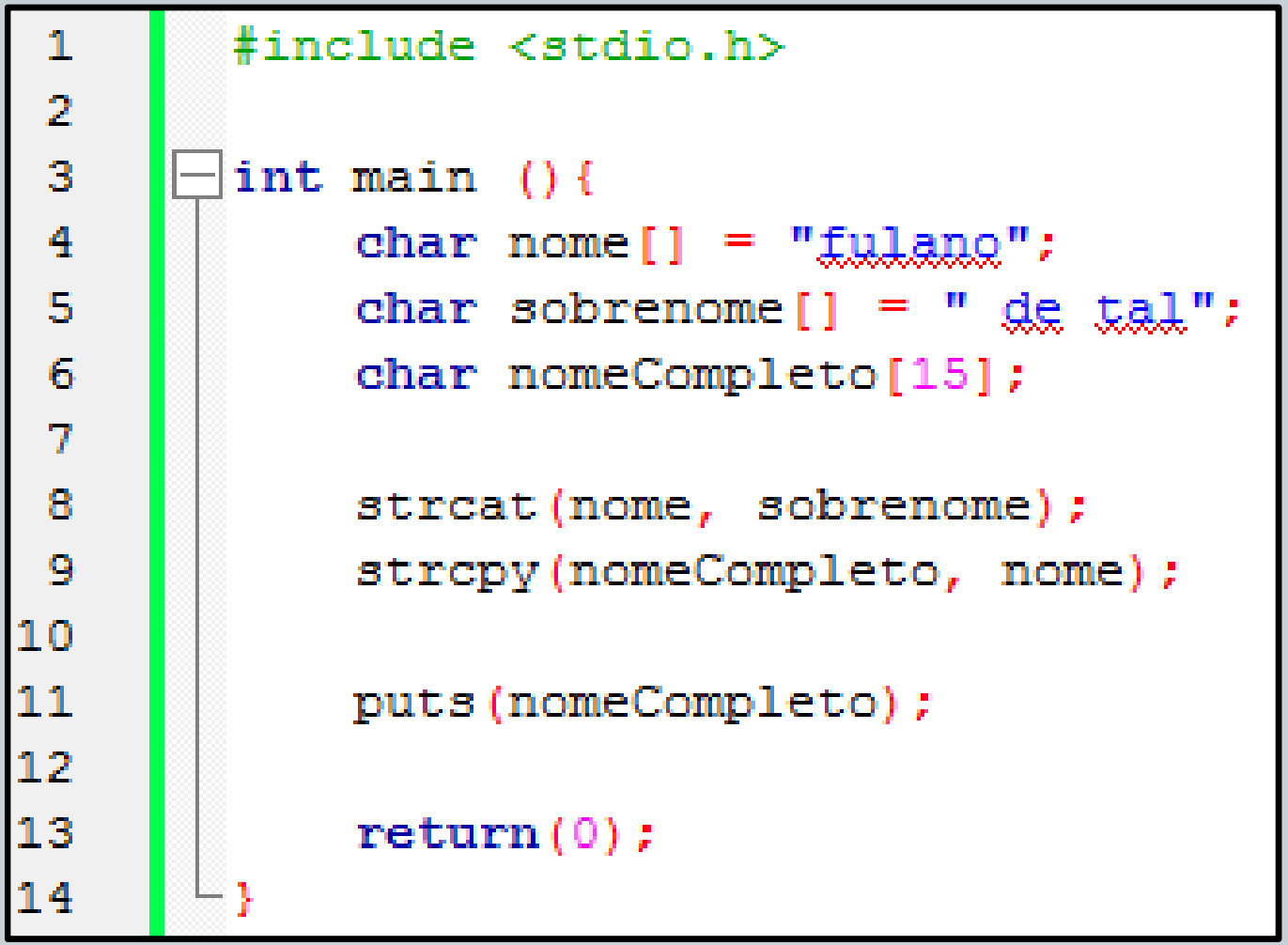
Funções de manipulação de strings <string.h>

- Strings não podem ser comparadas com o operador de comparação padrão (==), neste caso deve-se usar função strcmp() ou a função stricmp().
  - strcmp(s1,s2) – Retorna 0 se s1 e s2 são iguais; menor que 0 se s1<s2; maior que 0 se s1>s2 (comparação alfabética).
  - stricmp(s1,s2) – Retorna 0 se s1 e s2 são iguais; menor que 0 se s1<s2; maior que 0 se s1>s2 (comparação alfabética). Essa função considera letras maiúsculas ou minúsculas como símbolos iguais.

Funções de manipulação de strings <string.h>

- Strings não podem ser atribuídas com o operador de atribuição (=), para uma atribuição usa-se a função strcpy( ).
  - strcpy(s1,s2) – Copia s2 em s1.
- Strings não podem ser concatenadas com o operador (+), para tal usa-se a função strcat().
  - strcat(s1,s2) – Concatena s2 ao final de s1.
  - strlen(s) – Retorna o número de caracteres em s (sem contar o caracter nulo (/0)).
  - strchr(s,c) – Retorna um ponteiro na primeira ocorrência do caracter c na string s.
  - strstr(s1,s2) - Retorna um ponteiro na primeira ocorrência se s2 em s1.
  - strrev(s) – Inverte a string s sobre ela mesma.

Exemplo



```c
#include <stdio.h>

int main (){
    char nome[] = "fulano";
    char sobrenome[] = " de tal";
    char nomeCompleto[15];

    strcat(nome, sobrenome);
    strcpy(nomeCompleto, nome);

    puts(nomeCompleto);

    return(0);
}
```