Linguagem de Programação II

Prof. Mario Bessa

Aula 18

http://mariobessa.info

Cadeias de caracteres (string)

- Strings são vetores de carateres.
- O último caracter da string possui valor ‘\0’.
- Sintaxe: **`char <nome>[tamanho];`**

Inicialização

```
char texto[11]; //declaração
texto[0] = 'B';
texto[1] = 'e';
texto[2] = 'm';
texto[3] = '-';
texto[4] = 'v';
texto[5] = 'i';
texto[6] = 'n';
texto[7] = 'd';
texto[8] = 'o';
texto[9] = '!';
texto[10] = '\0';
printf("%s\n",texto);
```

Representação gráfica:

‘B’	‘e’	‘m’	‘-’	‘v’	‘i’	‘n’	‘d’	‘o’	‘!’	‘\0’
0	1	2	3	4	5	6	7	8	9	10

Inicialização

```
//Inicialização durante a declaração.
char texto[] = {'B','e','m','-','v','i','n','d','o','!','\0'};
// ou
char texto[] = "Bem-vindo!";


printf("%s\n",texto);
```

Representação gráfica:

‘B’	‘e’	‘m’	‘-’	‘v’	‘i’	‘n’	‘d’	‘o’	‘!’	‘\0’
0	1	2	3	4	5	6	7	8	9	10

Leitura de caracteres (string)

- **Lendo da entrada padrão:**
  - `scanf("%s",texto);`
    - Lê cadeias de caracteres até encontrar espaço em branco, nova linha ou **EOF** (fim de arquivo).
  - `gets(texto);`
    - Lê caracteres incluindo espaços em branco, até encontrar nova linha ou **EOF**. Não deve ser usada pois apresenta problemas quando o texto digitado excede o tamanho da string.
  - `fgets(texto, TAM, stdin);`
    - Igual ao **gets** porém mais seguro, lê no máximo **TAM** caracteres.
    - **stdin** dispositivo de entrada padrão (geralmente o teclado).
  - `int getchar(void)`
    - Lê um caracter.
  - **O caracter '\0' é inserido automaticamente no final do vetor `texto` após a leitura em todos os casos.**

Leitura de caracteres (string)

- **Lendo da entrada padrão:**

```
#include <stdio.h>
#define LIM 500

int main(){
  char texto[LIM];

  printf("Digite uma string: ");
  scanf("%s",texto);
//   gets(texto);
//   fgets(texto,LIM-1,stdin); //Mais seguro!

  printf("String: %s\n",texto);

  return 0;
}
```

Leitura de caracteres (string)

- **Lendo da entrada padrão (um caracter por vez):**

```c
#include <stdio.h>

int main(){
  char texto[500];
  int c,i=0;
  while(1){
    c = getchar(); //Lê o próximo caracter.
    if(c==EOF || c=='\n')
      break;
    texto[i] = (char)c;
    i++;
  }
  texto[i] = '\0';
  printf("texto: %s\n",texto);
  return 0;
}
```

Impressão de caracteres (string)

- **Escrevendo na entrada padrão:**
  - `printf("%s",texto);`
    - Imprime, com formatação, cadeias de caracteres até encontrar o **EOF** (fim de arquivo).
  - `puts(texto);`
    - Imprime, sem formatação, cadeias de caracteres até encontrar o **EOF** (fim de arquivo).
  - `int putchar(int c);`
    - Imprime um caracter.

Impressão de caracteres (string)

- **Lendo da entrada padrão (um caracter por vez):**

```c
#include <stdio.h>
int main(){
  char texto[500];
  int c,i=0;
  while(1){
    c = getchar(); //Lê o próximo caracter.
    if(c==EOF || c=='\n')
      break;
    texto[i] = (char)c;
    i++;
  }
  texto[i] = '\0';
  printf("texto: %s\n",texto);
  puts(texto);
  putchar(65); //letra A
  putchar('A'); //letra A
  return 0;
}
```

Funções para manipulação de string

- Existem várias funções em C para manipular strings. Essas função estão na biblioteca <string.h>
- Comparação entre duas strings
  - Não é possível comparar duas strings através do sinal de igualdade (==), pois strings podem ser entendidos como vetores de caracteres em C.
  - Para fazer a comparação é necessário usar a função **`strcmp`**
  - Sintaxe: **`strcmp`** (string1, string2);
  - A função strcmp retorna 0 se as strings forem idênticas e retorna um valor diferente de 0 se não forem idênticas.

Funções de manipulação de string

- #include <string.h>

Função	Operação
strcpy(s1, s2)	Copia string s2 em string s1
strcat(s1, s2)	Concatena string s2 no fim do string s1
strlen(s1)	Returna o tamanho de string s1
strcmp(s1, s2)	• Returna 0 se s1 e s2 são idênticas • Menor que 0 se s1<s2 • Maior que 0 se s1>s2
strchr(s1, c)	Returna um ponteiro para a primeira ocorrência do caracter c em s1
strstr(s1, s2)	Returna um ponteiro para a primeira ocorrência de s2 em s1

Cadeias de caracteres (string)

- **Manipulando strings:** #include <string.h>
- strcpy(s1, s2)

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

int main () {
    char palavra1[10], palavra2[10];
    printf("Digite uma palavra:\n");
    scanf("%s", palavra1);
    printf("Digite outra palavra:\n");
    scanf("%s", palavra2);
    strcpy(palavra2, palavra1);
    printf("Palavra depois da cópia: %s\n", palavra2);
    return 0;
}
```

Cadeias de caracteres (string)

- **Manipulando strings:** #include <string.h>
- strcat(s1, s2)

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

int main() {
    char palavra1[10], palavra2[10];
    scanf("%s", palavra1);
    scanf("%s", palavra2);
    strcat(palavra2, palavra1);
    printf("%s", palavra2);
    return 0;
}
```

Cadeias de caracteres (string)

- **Manipulando strings:** #include <string.h>
- strlen(s1)

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

int main() {
    char palavra1[10];
    printf("Digite uma palavra:\n");
    scanf("%s", palavra1);
    printf("O tamanho da palavra %s é %d.\n", palavra1, strlen(palavra1));
    return 0;
}
```

Cadeias de caracteres (string)

- **Uma possível implementação para strlen:**

```
#include <stdio.h>

int main(){
  char texto[]="Paulo Miranda";
  int  i;

  i = 0;
  while(texto[i]!='\0')
    i++;
  printf("String possui %d caracteres.\n",i);

  return 0;
}
```

Cadeias de caracteres (string)

- **Manipulando strings:** #include <string.h>
- strcmp(s1, s2)

```
#include <stdio.h>

int main() {
        char palavra1[10], palavra2[10];
        printf("Digite uma palavra:\n");
        scanf("%s", palavra1);
        printf("Digite uma palavra:\n");
        scanf("%s", palavra2);
        if (!strcmp(palavra1, palavra2))
                printf("%s e %s são iguais.\n", palavra1, palavra2);
        else
                printf("%s e %s são diferentes.\n", palavra1, palavra2);
        return 0;
}
```

Cadeias de caracteres (string)

- **Manipulando strings:** #include <string.h>
- strchr (s1, c)

```
#include <stdio.h>

int main() {
    char palavra1[10];
    char letra;
    printf("Digite uma palavra:\n");
    scanf("%s", palavra1);
    printf("Digite uma letra:\n");
    scanf("%s",&letra);
    if (strchr(palavra1, letra))
       printf("A letra %c pertence à palavra %s.\n", letra, palavra1);
    else
       printf("A letra %c não pertence à palavra %s.\n", letra, palavra1);
    return 0;
}
```

Tabela ASCII

- ASCII é uma padronização onde cada caracter é manipulado sob forma de código binário.

SIMB	DEC	BINÁRIO
3	51	00110011
4	52	00110100
5	53	00110101
6	54	00110110
7	55	00110111
8	56	00111000
9	57	00111001
:	58	00111010
;	59	00111011

SIMB	DEC	BINÁRIO
<	60	00111100
=	61	00111101
>	62	00111110
?	63	00111111
@	64	01000000
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100

Cadeias de caracteres (string)

- **Manipulando strings:** #include <string.h>

```
#include <stdio.h>
#include <string.h>
int main(){
  char firstname[100]="Paulo";
  char lastname[100]="Miranda";
  char name[100];

  printf("%d\n",strlen(firstname)); //Imprime 5.
  printf("%d\n",strlen(lastname));  //Imprime 7.

  strcpy(name, firstname); //Copia firstname.
  strcat(name, " ");       //Adiciona Espaco em branco.
  strcat(name, lastname);  //Adiciona lastname.
  printf("%d\n",strlen(name)); //Imprime 13.
  printf("name: %s\n",name);   //Imprime nome completo.
  return 0;
}
```

Exercício:

- Faça um programa que receba uma linha de texto do teclado e, lendo um caractere por vez, imprima uma palavra por linha. Exemplo:

Entrada:

```
O        Provedor 123 oferece acesso!
```

Saída:

```
O
provedor
123
oferece
acesso!
```

Solução:

```
#include <stdio.h>

int main(){
  int c = getchar();
  do{
    while(c==' ')  //Remove espaços em branco.
      c = getchar();

    while(c!=' ' && c!='\n' && c!=EOF){
      printf("%c",(char)c);
      c = getchar();
    }
    printf("\n");
  }while(c!=EOF && c!='\n');

  return 0;
}
```

Exercício:

- Faça em programa que conta o número de palavras de um texto terminado com a palavra “end”. Exemplo:

Entrada:

```
um programa que leia as
notícias em um formato especial
chamado XML end
```

Saída:

```
13 palavras
```

Solução:

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

int main(){
  char name[512];
  int c=1;
  do{
    scanf("%s",name);
    if(strcmp(name, "end")==0) break;
    c++;
  }while(1);

  printf("%d palavras\n",c);

  return 0;
}
```

Exercícios propostos:

1. Escreva uma função em C que leia uma frase e retorne o número de vogais dessa frase. A frase digitada pelo usuário terá, no máximo, 100 caracteres e todos serão minúsculos. Não devem ser utilizadas as funções da biblioteca string.h.
   - Para testar a função, escreva um programa que imprima a seguinte mensagem: "A <frase-digitada> tem <quantidade> vogais."
2. Escreva um programa em C para ler duas palavras e verificar se são anagramas.
3. Escreva uma função em C chamada *replace* que aceita uma *string* como argumento e retorna um inteiro. A função substitui todos os espaços em branco do seu parâmetro pelo caracter '-', e retorna o número de substituições feitas.

Exercícios propostos:

4. Escreva um programa em C para ler duas *strings* que representam nomes de cidades. Considere que o tamanho máximo que o nome de cada cidade pode ter é de 20 caracteres. O programa deverá:
   - ter uma função para concatenar os nomes das cidades.
   - ter uma função calcular o tamanho da *string* final concatenada
   - A partir das funções definidas, o programa deverá imprimir:
     - o nome resultante da concatenação.
     - o tamanho final da *string* concatenada.
   - Obs.: Não devem ser utilizadas as funções da biblioteca string.h
5. Escreva um programa em C que receba uma frase e mostre-a do início até a metade, de maneira normal, e na outra linha, do meio até o fim da frase de maneira inversa, por exemplo:
   - `ENTRADA: primeirafrase`
   - `SAÍDA:`
     - `primei`
     - `esarfar`

Solução do exercícios propostos:

```
#include <stdio.h>
#define LIM 101
int contaVogal (char c[]){
    int cont=0;
    for (int i=0; i<LIM; i++) {
        if(c[i]=='a'||c[i]=='e'||c[i]=='i'||c[i]=='o'||c[i]=='u') {
            cont++;
        }
    }
    return cont;
}
int main(){
    char frase[LIM];
    printf("Digite uma frase: ");
    gets(frase);
    printf("\nA frase: %s tem %i vogais.",frase,contaVogal(frase));
    return 0;
}
```

Solução do exercícios propostos:

```
#include <stdio.h>
int check_anagrama(char [], char []);
int main(){
    char a[100], b[100];
    int flag;
    printf("Primeira palavra:\n");
    gets(a);
    printf("Segunda palavra:\n");
    gets(b);
    flag = check_anagrama(a, b);
    if (flag == 1)
        printf("\"%s\" e \"%s\" são anagramas.\n", a, b);
    else
        printf("\"%s\" and \"%s\" não são anagramas.
\n", a, b);
    return 0;
}
```

```
int check_anagrama(char a[], char b[]){
    int primeira[26] = {0}, segunda[26] = {0}, c = 0;
    while (a[c] != '\0'){
        primeira[a[c]-'a']++;
        c++;
    }
    c = 0;
    while (b[c] != '\0'){
        segunda[b[c]-'a']++;
        c++;
    }
    for (c = 0; c < 26; c++){
        if (primeira[c] != segunda[c])
            return 0;
    }
    return 1;
}
```