Introdução à Programação

Engenharia Informática

Ana Rosa Pereira Borges

Strings

- Um tipo muito comum de tabelas (arrays) em C são as tabelas de caracteres.
- As strings são tabelas de caracteres “especiais”:
  - Em C, qualquer string deve ser terminada com o caracter nulo => ‘\0’ (Define-se da mesma maneira que qualquer tabela de caracteres, mas tem de se lhe dar espaço para este caracter terminador)
  - Às strings constantes é automaticamente adicionado o caracter ‘\0’
- CUIDADO:
  - Um vector de carecteres pode não conter uma string

Inicialização de Strings

-
  - char string[20]=“Uma string!”;
  - char string[20]={‘U’,’m’,’a’,’ ‘,’s’,’t’,’r’,’i’,’n’,’g’,’!’};
  - char string[]=“Uma string!”;

    - Igual a
      char string[11+1]=“Uma string!”;

    - Mas diferente de
      char string[]={‘U’,’m’,’a’,’ ‘,’s’,’t’,’r’,’i’,’n’,’g’,’!’};
      (tabela de caracteres normal, com 11 caracteres)

Strings - Exemplo *(st1.c)*

```
#include <stdio.h>

void main(void)
{
  char st[80], c;
  int i = 0;

        printf("Diga uma string com menos do que 80 caracteres:");
        while ( (c=getchar())!='\n' )
            st[i++] = c;
        st[i] = '\0';

        for (i=0; st[i]!='\0'; i++)
            printf("%c", st[i]);
        printf("\n");
}
```

⇔ printf(“%s\n”, st);⇔ puts(st);

Strings - Exemplo *(st.c)*

```
#include <stdio.h>

void main(void)
{
  char st[80], c;
  int i = 0;

        printf("Diga uma string com menos do que 80 caracteres:");
        while ( (i<79) && ((c=getchar())!='\n') )
            st[i++] = c;
        st[i] = '\0';

        for (i=0; st[i]!='\0'; i++)       /* (s[i]!='\0') ou (s[i]!=0) ou (s[i]) */
            printf("%c", st[i]);
        printf("\n");
}
```

Strings - Exemplo *(st2.c)*

```
#include <stdio.h>

void main(void)
{
  char st[80];
  char st2[4]="abc"; //{'a','b','c','\0'};

   printf("Diga uma string com menos do que 80 caracteres:");
   gets(st);
   printf("%s\n", st);                    /*  equivalente a puts(st); */
   printf("%s\n", st2);
}
```

CUIDADO: gets(st) não efectua nenhum controlo sobre os limites do array.

Strings - Exemplo *(st3.c)*

```
#include <stdio.h>

void main(void)
{
  char st[80];

                        /* Só lê a primeira palavra!  e …*/
     scanf("%s", &st[0]);        /* ⇔ scanf(“%s”, st); */
      printf("%s\n", st);

      fflush(stdin);

                        /* Lê a frase toda até ENTER */
     gets(st);
     printf("%s\n", st);
}
```

Passagem de Strings para funções

- É exactamente igual à passagem de vectores para funções

Principais funções de manipulação de Strings *(1)*

- Em C as strings não podem ser manipuladas directamente
  (=> as strings não são um tipo básico)
- Em string.h estão definidas funções que permitem manipular strings
  (=> É necessário: #include <string.h>)

  - int strlen(char *s); /* ⇔ int strlen(char s[]); */
    - devolve o número de caracteres existentes na string (sem contar '\0').
      - Exemplo possível de código para esta função:

```
int strlen(char *s)
{
  int i=0;
      while (s[i]!='\0')
          i++;
      return i;
}
```

Principais funções de manipulação de Strings *(2)*

- char *strcpy(char *destino, char *origem);
  - copia uma string (char *origem) para outra (char *destino) e devolve esta última (char *destino).

```
/* poderíamos  "matematicamente" dizer que:
                                        "destino = origem" */
```

- int strcmp(char *st1, char *st2);
  - compara lexicograficamente as strings st1 e st2; devolve o valor 0 se elas forem iguais, <0 se st1 é alfabeticamente menor que st2, >0 se st1 é alfabeticamente maior que st2.

- int stricmp(char *st1, char *st2);
  - compara lexicograficamente as strings st1 e st2 sem "case sensitive".

Principais funções de manipulação de Strings *(3)*

- char *strcat(char *destino, char *origem);

  /* poderíamos matematicamente dizer que:
  "destino = destino + origem" */

  - Coloca a segunda string (char *origem) imediatamente a seguir ao final da primeira string (char *destino) (devolve destino).

    - Exemplo possível de código para esta função:

```
char * strcat(char *destino, char *origem)
{
  int i, len;
      for (i=0, len=strlen(destino); origem[i]!='\0'; i++, len++)
          destino[len]=origem[i];
      destino[len]='\0';
      return destino;
}
```

Strings - Exemplo *(st4.c)*

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

void main(void)
{
    char str1[80], str2[80];          int i;        char c;
      printf("Diga a primeira string:");       gets(str1);
      printf("Diga a segunda string:");       gets(str2);

   /* Calcula comprimento das strings (sem o \0 final) */
      printf("%s tem %d caracteres.\n", str1, strlen(str1));
      printf("%s tem %d caracteres.\n", str2, strlen(str2));

   /* Compara as strings */
      i = strcmp(str1, str2);
      if (!i)  printf("As strings são iguais.\n");
      else
               if (i<0)
                 printf("%s , lexicograficamente menor do que %s.\n", str1, str2);
               else
                 printf("%s , lexicograficamente maior do que %s.\n", str1, str2);
```

Strings - Exemplo *(st4.c – continuação)*

```
/* Concatenação de str2 no final de str1, se houver espaço suficiente */
    if (strlen(str1) + strlen(str2) <80)
    {
      strcat(str1, str2);
      printf("%s\n", str1);
    }

/* Cópia de str2 para str1 */
   strcpy(str1, str2);
   printf(“Primeira = %s\n", str1);
   printf(“Segunda = %s\n", str2);

/* Procura caracter em str1 */
   printf("Diga 1 caracter:");        c=getchar();
   printf("Em \"%s\" %s %c.\n", str1, strchr(str1, c)?"ha":"nao ha", c);

/* Converte str1 a maiúsculas */
   printf("Em maiusculas => \"%s\".\n", strupr(str1) );

/* Converte str2 a minúsculas */
   printf("Em minusculas => \"%s\".\n", strlwr(str2) );
}
```

Bibliotecas Standard de funções

- Existem algumas funções standard (muito úteis) em linguagem C
  - Ver (por exemplo) nas páginas 603-636 de
    - K. N. King,
      C programming: A Modern Approach