MC102 - Algoritmos e Progração de Computador

Prof. Alexandre Xavier Falcão

13° Aula: Cadeias de Caracteres

1 Cadeias de Caracteres

Uma cadeia de caracteres (*string*) é uma seqüência de letras, símbolos (!,?,+,=,%,;, etc.), espaços em branco e/ou dígitos terminada por

```
'\0'
```

Isto é, um vetor de variáveis do tipo char.

```
#include <stdio.h>

int main()
{
  char texto[100]="O professor gost? de alunos estudiosos.";
  int i;

  texto[16]='a'; /* Corrige erro no texto */
  i=0;
  while (texto[i] != '\0'){ /* Imprime caracter por caracter
                               separados por | */
    printf("%c|",texto[i]);
    i++;
  }
  printf("\n%s\n",texto); /* Imprime o texto todo.*/

  return 0;
}
```

2 Lendo da entrada padrão

Até o momento vimos que o comando **scanf** pode ser usado para ler dados da entrada padrão (teclado). Mais adiante veremos que **fscanf** possui sintaxe similar, porém é mais genérico. Pois permite leitura de dados não só da entrada padrão, identificada por **stdin**, como a leitura de dados armazenados em um **arquivo texto** no formato ASCII. A mesma observação se aplica aos comandos **printf** e **fprintf**, com identificador **stdout** para a saída padrão.

```
#include <stdio.h>

int main()
{
  char texto[100];

  fscanf(stdin,"%s",texto); /* Lê cadeias de caracteres até encontrar
                               espaço em branco, nova linha ou EOF (fim de
                               arquivo). Equivalente a scanf("%s",texto);
                               O caracter '\0' é inserido no final
                               do vetor texto após a leitura. */
return 0;
}
```

A mesma relação é válida entre outros comandos de entrada, tais como **gets** e **fgets**, **getc** (ou **getchar**) e **fgetc**, porém o comando **fgets** é mais seguro do que **gets**, pois especifica o número máximo de caracteres que o vetor suporta.

```
#include <stdio.h>

int main()
{
  char texto[100];

  fgets(texto,99,stdin); /* Lê no máximo 99 caracteres, incluindo espaços
                            em branco, até encontrar nova linha ou EOF.
                            Mais seguro que gets(texto);
                            O caracter '\0' é inserido no final
                            do vetor texto após a leitura. */
return 0;
}
```

```
#include <stdio.h>

int main()
{
  char texto[100];
  int i,c;

  i = 0;
  while (((c=fgetc(stdin))!=EOF)&&(i < 100)){/* Lê no máximo 99 caracteres ou
                                                até fim de arquivo especificado
                                                na entrada padrão com o comando
                                                < (e.g. programa < arquivo). */
    texto[i] = (char) c;
    i++;
```

```
  }
  printf("%s\n",texto); /* Imprime o texto, que pode conter várias linhas.*/

  return 0;
}
```

```
Uma variação deste programa seria ler apenas a primeira linha, caso
usássemos '\n' em vez de EOF.
```

3 Convertendo cadeias em números, e vice-versa

Os comandos **sprintf** e **sscanf** funcionam de forma similar aos comandos printf e scanf, porém utilizando uma cadeia de caracteres no lugar da saída/entrada padrão, respectivamente.

```
#include <stdio.h>

int main()
{
  char  v1[10],v2[10],texto[100];
  float v3;
  int v4;

  sprintf(v1,"%2.1f",3.4); /* converte float para string */
  printf("%s\n",v1);

  sprintf(v2,"%2d",12);    /* converte int para string */
  printf("%s\n",v2);

  sscanf(v1,"%f",&v3);     /* converte string para float */
  printf("%2.1f\n",v3);

  sscanf(v2,"%d",&v4);     /* converte string para int */
  printf("%d\n",v4);

  sprintf(v1,"%2.1f %2d",3.4,12); /* grava números em string */
  printf("%s\n",v1);

  sscanf(v1," %f %d",&v3,&v4); /* ler números de strings. Lembre-se do bug
                                  que requer espaço em branco extra */
  printf("%2.1f %2d\n",v3,v4);

  sprintf(texto,"Os valores são %2.1f e %2d\n",v3,v4); /* gera string com
                                                          valores formatados */
  return 0;
}
```