Strings (parte 3)

Funções (parte 1)

Profa. Debora Medeiros

Baseado no material de:
Ciro Trindade (Unisantos)

Matrizes de Strings

- Matriz de caracteres
- Por exemplo:
  `char str_vet[30][80];`
- Declara uma matriz de ...
- Para acessar uma *string individual na matriz...*

Matrizes de Strings

- Matriz de caracteres
- Por exemplo:
  `char str_vet[30][80];`
- Declara uma matriz de 30 *strings, cada qual com comprimento máximo de 79 caracteres*
- Para acessar uma *string individual na matriz, basta especificar o 1º índice (linha)*
  `fgets(str_array[2],79,stdin);`
- Faz referência a 3ª *string em* `str_vet`


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

int main() {
    int MAX=100, LEN=80, i, j;
    char texto[100][80];

    printf("Digite uma linha vazia para sair.\n");
    for(i = 0; i < MAX; i++) {
        printf("%03d: ",i+1);
        fgets(texto[i],LEN,stdin);
        if(texto[i][0] == '\n') { // verifica se a linha está em branco
            break;
        }
    }
    // imprime todo o conteúdo da matriz
    for(j = 0; j < i; j++) {
        printf("%03d: %s",j+1,texto[j]);
    }
    system("pause");
    return 0;
}
```



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

int main() {
    int MAX=100, LEN=80, i, j;
    char texto[100][80];

    printf("Digite uma linha vazia para sair.\n");
    for(i = 0; i < MAX; i++) {
        printf("%03d: ",i+1);
        fgets(texto[i],LEN,stdin);
        if(texto[i][0] == '\n') { // verifica se a linha está em branco
            break;
        }
    }
    // imprime todo o conteúdo da matriz
    for(j = 0; j < i; j++) {
        printf("%03d: %s",j+1,texto[j]);
    }
    system("pause");
    return 0;
}
```

```
Digite uma linha vazia para sair.
001: um elefante incomoda muita gente
002: dois elefantes incomodam incomodam muito mais
003: tres elefantes incomodam muita gente
004: quatro elefantes incomodam incomodam incomodam muito mais
005:
001: um elefante incomoda muita gente
002: dois elefantes incomodam incomodam muito mais
003: tres elefantes incomodam muita gente
004: quatro elefantes incomodam incomodam incomodam muito mais
Press any key to continue . . .
```


Exemplo

- Ordenação de uma matriz de strings
  - Quadro...

Inicializando um vetor de strings

- Forma tradicional: matriz bidimensional de caracteres

char naipe[4][8] =
{“Copas”,”Ouros”,”Paus”,”Espadas”};

Funções

Por quê usar funções?

- Evitar a repetição de um conjunto de instruções que ocorre várias vezes no programa
- Modularizar o programa
- Reaproveitamento de código

Estruturas das Funções em C

```
tipo nome-da-função([lista-de-parâmetros])
  {
  corpo da função
  }
```

- `tipo`: especifica o tipo do valor que a função retorna
- `lista-de-parâmetros`: uma lista separada por vírgulas das variáveis que recebem os valores dos argumentos passados quando a função é chamada
- `corpo da função`: comandos delimitados por chaves { }

Exemplo de função

- Imprimir uma linha
- Ex:

Calculadora

Digite a operação desejada:

Exemplo

```
#include <stdio.h>
void linha(); //protótipo da função linha()

int main() {
   linha();
   printf("\nUm programa em C\n");
   linha();
   system("pause");
   return 0;
}

void linha() {
   int i;
   for(i = 1; i <= 80; i++)
      printf("-");
}
```

Exemplo

```
#include <stdio.h>
void linha(); //protótipo da função linha()

int main() {
   linha();
   printf("\nUm programa em C\n");
   linha();
   system("pause");
   return 0;
}

void linha() {
   int i;
   for(i = 1; i <= 80; i++)
      printf("-");
}
```

```
--------------------------------------------------------------------------------
Um programa em C
--------------------------------------------------------------------------------
Press any key to continue . . .
```


Chamando Funções

- Do mesmo modo que chamamos uma função da biblioteca C (`printf()`, `scanf()`, etc.) chamamos nossas próprias funções como `linha()`
- Os parênteses que seguem o nome são necessários para que o compilador possa diferenciar a chamada a uma função de uma variável

Variáveis Locais

- Variáveis declaradas dentro de uma função
  - conhecidas somente dentro do seu próprio bloco
  - Um bloco começa quando o compilador encontra uma chave de abertura ({) e termina quando o compilador encontra uma chave de fechamento (})
  - Variáveis locais existem apenas durante a execução do bloco de instruções onde estão declaradas

Funções que Devolvem um Valor

- O comando **`return`**
  - Causa uma saída imediata da função onde ele está contido
    - ou seja, termina a função
  - Pode ser usado para devolver um valor
  - Todas as funções, exceto aquelas do tipo void, devolvem um valor
  - valor explicitamente especificado pelo comando **`return`**
  - O comando **`return`** faz com que a chamada à função que contém o **`return`** seja substituída pelo valor que sucede este comando

Funções que Devolvem um Valor

- Desde que uma função não seja declarada como **`void`**, ela pode ser usada como um operando em qualquer expressão

```
if (max(x,y) > 100) printf("Maior");
printf("Valor absoluto de %d = %d", x, abs(x));
```

- Entretanto, uma função não pode receber uma atribuição

```
sqrt(x)=100;   // instrução errada
```


Exemplos

- Maior número (exercício das médias da lista)
  - max(x,y)
  - Quadro…