MC102 – Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Segundo Semestre de 2013

Roteiro

Variáveis locais e variáveis globais

- Uma variável é chamada local se ela foi declarada dentro de uma função. Nesse caso, ela existe somente dentro daquela função e, após o término da execução da mesma, a variável deixa de existir. Variáveis utilizadas como parâmetros de funções também são locais.
- Uma variável é chamada global se ela for declarada fora de qualquer função. Essa variável é visível em todas as funções. Qualquer função pode alterá-la e ela existe durante toda a execução do programa.

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

Protótipos de funções;
Declaração de Variáveis Globais;

int main() {
  Declaração de variáveis locais;
  Comandos;
}

int funcao1(parâmetros) { /* Parâmetros também são considerados locais */
  Declaração de variáveis locais;
  Comandos;
}

int funcao2(parâmetros) { /* Parâmetros também são considerados locais */
  Declaração de variáveis locais;
  Comandos;
}

...
```

Escopo de variáveis

- O escopo de uma variável determina em quais partes do código pode-se ter acesso a ela.
- A regra de escopo em C é bem simples:
  - As variáveis globais são visíveis por todas as funções.
  - As variáveis locais são visíveis apenas na função onde foram declaradas.

Escopo de variáveis

```
#include <stdio.h>

void funcao1();
int funcao2(int local_b);

int global_a;

int main() {
  int local_main;
  /* Neste ponto são visíveis global_a e local_main */
}

void funcao1() {
  int local_a;
  /* Neste ponto são visíveis global_a e local_a */
}

int funcao2(int local_b) {
  int local_c;
  /* Neste ponto são visíveis global_a, local_b e local_c */
}
```

Escopo de variáveis

- É possível declarar variáveis locais com o mesmo nome de variáveis globais.
- Nesta situação, a variável local "esconde" a variável global.

```
int nota = 10;

void a() {
  int nota;

  ...

  /* Altera o valor da variavel local,
     sem afetar a variavel global */
  nota = 5;
}
```

Escopo de variáveis

```
#include <stdio.h>
int x = 1;

void funcao1() {
  x = 3;
  printf("%d ", x);
}

void funcao2() {
  int x = 4;
  printf("%d ", x);
}

int main() {
  x = 2;
  funcao1();
  funcao2();
  printf("%d\n", x);
  return 0;
}
```

O que será impresso?

```
3 4 3
```

Vetores em funções

- Vetores também podem ser passados como parâmetros em funções.
- Ao contrário dos tipos simples, vetores têm um comportamento diferente quando usados como parâmetros de funções.
- Quando uma variável simples é passada como parâmetro, seu valor é atribuído para uma nova variável local da função.
- No caso de vetores, não é criado um novo vetor.
- Isto significa que os valores de um vetor podem ser alterados dentro de uma função.

Vetores em funções

```
#include <stdio.h>

void funcao(int vet[], int tam) {
  int i;
  for (i = 0; i < tam; i++)
    vet[i] = 5;
}

int main() {
  int x[10];
  int i;

  for (i = 0; i < 10; i++)
    x[i] = 8;

  funcao(x, 10);
  for (i = 0; i < 10; i++)
    printf("%d\n", x[i]);
  return 0;
}
```

O que será impresso? O programa imprimirá o valor 5 dez vezes, cada valor em uma linha.

Vetores em funções

- Vetores não podem ser devolvidos por funções.

```
#include <stdio.h>

int[] leVetor(int tam) {
  int i, vet[100];

  for (i = 0; i < tam; i++) {
    printf("Digite um numero: ");
    scanf("%d", &vet[i]);
  }

  return vet;
}
```

  O código acima não compila, pois não podemos retornar um `int[]`.

- Entretanto, podemos fazer algo semelhante usando o fato de que vetores são alterados dentro de funções.

Vetores em funções

- Como um vetor é alterado dentro de uma função, podemos criar a seguinte função:

```
#include <stdio.h>

void leVetor(int vet[], int tam) {
  int i;
  for (i = 0; i < tam; i++) {
    printf("Digite numero: ");
    scanf("%d", &vet[i]);
  }
}

void escreveVetor(int vet[], int tam) {
  int i;
  for (i = 0; i < tam; i++)
    printf("vet[%d] = %d\n", i, vet[i]);
}
```

Vetores em funções

```
int main() {
  int vet1[10], vet2[20];

  printf("----- Vetor 1 -----\n");
  leVetor(vet1, 10);
  printf("----- Vetor 2 -----\n");
  leVetor(vet2, 20);

  printf("----- Vetor 1 -----\n");
  escreveVetor(vet1, 10);
  printf("----- Vetor 2 -----\n");
  escreveVetor(vet2, 20);

  return 0;
}
```

Vetores multidimensionais e funções

- Ao passar um *vetor simples* como parâmetro, não é necessário fornecer o seu tamanho na declaração da função.
- Quando o *vetor é multidimensional*, a possibilidade de não informar o tamanho na declaração se restringe à primeira dimensão apenas.

```
void mostra_matriz(int mat[][10], int linhas) {
   ...
}
```

Vetores multidimensionais e funções

- Pode-se criar uma função deixando de indicar a primeira dimensão:

```
void mostra_matriz(int mat[][10], int linhas) {
  ...
}
```

- Ou pode-se criar uma função indicando todas as dimensões:

```
void mostra_matriz(int mat[5][10], int linhas) {
  ...
}
```

- Não se pode deixar de indicar outras dimensões (exceto a primeira):

```
void mostra_matriz(int mat[5][], int linhas) {
  /* Este programa nao funciona */
  ...
}
```

Vetores em funções

```
#include <stdio.h>

void mostra_matriz(int linhas, int colunas, int matriz[8][10]) {
  int i, j;
  for (i = 0; i < linhas; i++) {
    for (j = 0; j < colunas; j++)
      printf("%2d ", matriz[i][j]);
    printf("\n");
  }
}

int main() {
  int matriz[8][10] = { { 0,  1,  2,  3,  4,  5,  6,  7,  8,  9},
                        {10, 11, 12, 13, 14, 15, 16, 17, 18, 19},
                        {20, 21, 22, 23, 24, 25, 26, 27, 28, 29},
                        {30, 31, 32, 33, 34, 35, 36, 37, 38, 39},
                        {40, 41, 42, 43, 44, 45, 46, 47, 48, 49},
                        {50, 51, 52, 53, 54, 55, 56, 57, 58, 59},
                        {60, 61, 62, 63, 64, 65, 66, 67, 68, 69},
                        {70, 71, 72, 73, 74, 75, 76, 77, 78, 79} };
  mostra_matriz(8, 10, matriz);
  return 0;
}
```

Exercício

- Escreva uma função para ler matrizes de dimensões até 100×100.
- Sua função deve receber como parâmetro uma matriz 100×100 e inteiros numLinhas e numColunas que indicam quantas linhas e colunas desta matriz serão utilizadas.