MC102
Algoritmos e Programação de Computadores

Aula de Laboratório 08

Instituto de Computação
Primeiro Semestre de 2012

23 de abril de 2012

Conteúdo

Escopo

- O escopo de uma variável determina onde ela pode ser acessada;
- Variáveis locais e globais:
  - Uma variável local é declarada dentro de uma função;
    - São visíveis apenas na função onde foram declaradas;
  - Uma variável global é declarada fora de qualquer função;
    - São visíveis por todas as funções;

Escopo

Exemplo

```
1  #include<stdio.h>
2  void fun1();
3  int fun2(int local_b);
4  int global;
5  int main() {
6    int local_main;
7    /*Neste ponto sao visiveis global e local_main */
8  }
9  void fun1() {
10   int local_a;
11   /*Neste ponto sao visiveis global e local_a */
12 }
13 int fun2(int local_b){
14   int local_c;
15   /*Aqui sao visiveis global, local_b e local_c*/
16 }
```

Escopo

Exemplo

```
1  #include <stdio.h>
2  void fun1();
3  void fun2();
4  int x = 1;
5  int main(){
6    x=2;
7    fun1();
8    fun2();
9    printf("%d\n", x);
10 }
11 void fun1(){
12   x = 3;
13   printf("\n%d",x);
14 }
15 void fun2(){
16   int x = 4;
17   printf("\n%d",x);
18 }
```

Vetores em 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;
- Os valores de um vetor são alterados dentro de uma função!

Ex. com variável simples

```
1  #include <stdio.h>
2  void imprime(int a){
3     a = a + 4;
4     printf("a: %d\n",a);
5  }
6  int main(){
7     int b = 2;
8     imprime(b);
9     printf("b: %d\n",b);
10 }
```

Ex. com vetor

```
1  #include <stdio.h>
2  void imprime(int a[]){
3     a[0] = a[0] + 4;
4     printf("a[0]: %d\n",a[0]);
5  }
6  int main(){
7     int b[] = {2,3,4};
8     imprime(b);
9     printf("b[0]: %d\n",b[0]);
10 }
```

Vetores em funções

Problema

Programa que lê 10 valores em um vetor, sendo que para ler e imprimir deve-se utilizar funções passando o vetor por parâmetro.

Vetores em funções

Problema

Programa que lê 10 valores em um vetor, sendo que para ler e imprimir deve-se utilizar funções passando o vetor por parâmetro.

Solução

```
1  #include <stdio.h>
2  void leVet(int vet[], int tam){
3    int i;
4    for(i = 0; i < tam; i++){
5      printf("Digite numero: ");
6      scanf("%d",&vet[i]);
7    }
8  }
9  void escreveVet(int vet[], int tam){
10   int i;
11   for(i=0; i< tam; i++)
12     printf("vet[%d] = %d\n",i,vet[i]);
13 }
14 int main(){
15   int vet[10];
16   printf(" ------ Vetor --------\n");
17   leVet(vet,10);
18   printf(" ------ Vetor --------\n");
19   escreveVet(vet,10);
20 }
```

Matriz em funções

Exemplo

```
1  #include<stdio.h>
2  void mostra_matriz(int mat[][5], int n_linhas) {
3    int i, j;
4    for (i = 0; i < n_linhas; i++) {
5      for (j = 0; j < 5; j++)
6              printf("%2d ", mat[i][j]);
7          printf("\n");
8    }
9  }
10 int main() {
11   int mat[][5]={{0, 1, 2, 3, 4},
12                 {10, 11, 12, 13, 14},
13                 {20, 21, 22, 23, 24}};
14     mostra_matriz(mat, 3);
15   return 0;
16 }
```

Arquivos cabeçalhos

- Permite a inclusão de códigos no seu código
- Diretiva #include <stdio.h>. O que acontece?
- Eu posso incluir código? Mas deve utilizar: #include "funcoes.h"
- Assim pode-se aproveitar uma função em vários programas

Ex.: principal.c

```
1  #include<stdio.h>
2  #include "funcoes.h"
3
4  int main(){
5     float a, b, res;
6     printf("Digite um valor: ");
7     scanf("%f",&a);
8     printf("Digite outro valor: ");
9     scanf("%f",&b);
10    printf("\nSoma: %.2f",soma(a, b));
11    printf("\nMultip.: %.2f\n", multiplica(a, b));
12 }
```

Arquivos cabeçalhos

Ex.: funcoes.h

```
1 float soma(float a, float b);
2 float multiplica(float a, float b);
```

Ex.soma.c

```
1 float soma(float a, float b){
2   return (a+b);
3 }
```

Ex. multiplica.c

```
1 float multiplica(float a, float b){
2   return (a * b);
3 }
```

Ponteiros

- São tipos especiais de dados que armazenam endereços de memória;
- **Ex.:** `int *end_a; float *end_b;`
  - **end_a** armazena endereço de memória de variáveis do tipo int;
  - **end_b** armazena endereço de memória de variáveis do tipo float;
- Existe dois operadores em ponteiros:
  1. Operador **&** retorna o endereço de memória de uma varíavel;
     Ex.:
     ```
     int *end_a;
     int a = 90;
     end_a = &a;
     ```
  2. Operador * retorna o conteúdo do endereço indicado pelo ponteiro;
     Ex.:
     ```
     printf("%d", *end_a);
     ```

Ponteiros

- Recordando: **`scanf("%d",&a);`**

Exemplo

```
1 #include <stdio.h>
2 int main(void){
3    int a;
4    int *end_a;
5    a = 10;
6    end_a = &a;
7    *end_a = 11;
8    printf("\nValor: %d\n",a);
9 }
```

Ponteiros

Exemplo2

```
1  #include <stdio.h>
2  int main(void){
3     int b, a;
4     int *c;
5     b = 10;
6     c = &a;
7     *c = 11;
8     printf("\nValor *c: %d", *c);
9     a = b * (*c);
10    printf("\nValor  a: %d", a);
11    printf("\nValor *c: %d\n", *c);
12 }
```

Passagem por valor

- Ao passar parâmetros a uma função, os valores fornecidos são **copiados para as variáveis de parâmetros** da função. **Esse processo é chamado de passagem por valor**.
- Alterações nos parâmetros dentro da função não alteram os valores que foram passados:

Exemplo

```
1  #include<stdio.h>
2
3  void quadrado(int x){
4      x = x * x;
5  }
6
7  int main(){
8      int x=4;
9      quadrado(x);
10     printf("\nValor de x: %d\n",x);
11 }
```

Passagem por referência

- E se quisessemos que o valor passado fosse modificado.
- Em C podemos utilizar ponteiros para esse fim, assim podemos passar o endereço da variável;
- **Esse processo é chamado de passagem por referência**

Exemplo

```
1  #include<stdio.h>
2
3  void quadrado(int *x){
4      *x = (*x) * (*x);
5  }
6
7  int main(){
8      int x=4;
9      quadrado(&x);
10     printf("\nValor de x: %d\n",x);
11 }
```

Passagem de parâmetros

Problema

Faça um programa que passe um vetor (10 valores) a uma função que encontre o maior e o menor valor do vetor. **Utilize passagem por referência para as variáveis maior e menor**.

Passagem de parâmetros

Solução

```
1  #include<stdio.h>
2  void maxMin(int vet[], int *max, int *min){
3      int i;
4      *max = vet[0];
5      *min = vet[0];
6      for(i=1; i<10; i++){
7          if(vet[i] > *max)
8              *max = vet[i];
9          if(vet[i] < *min)
10             *min = vet[i];
11     }
12 }
13 int main(){
14     int vet[10] = {10,80,5,-10,45,-20,100,160,10,15};
15     int max, min;
16     maxMin(vet, &max, &min);
17     printf("\nMaior: %d e Menor: %d\n", max, min);
18 }
```

Questões?

Obrigado!

Para informação:

Página dos Laboratórios (Tarefas): *http://susy.ic.unicamp.br:9999/mc102ab*

Página do Curso: *http://www.lrc.ic.unicamp.br/∼geraldoms/mc102*

E-mail:

geraldoms[at]lrc.ic.unicamp.br

brhenrique.fischer[at]gmail.com