SSC0101 - ICC1 – Teórica

Introdução à Ciência da Computação I

Modularização de Programas
Parte II

Prof. Vanderlei Bonato: vbonato@icmc.usp.br
Prof. Claudio Fabiano Motta Toledo: claudio@icmc.usp.br

Sumário

- Escopo de Variáveis
- Variáveis locais x Variáveis globais
- Passagem de parâmetros por valor
- Passagem de parâmetros por referência

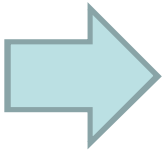
Escopo de Variáveis

- Escopo de uma variável estabelece onde ela poderá ser utilizada em um programa.
- A regra básica envolvendo escopo é que os identificadores são acessados apenas dentro do bloco em que foram declarados.
- Eles não são conhecidos fora dos limites do bloco onde foram declarados.
- Todavia, programadores podem, por diversos motivos, escolher usar um mesmo identificador em diferentes declarações.
- Neste caso, qual objeto está sendo utilizado?

Escopo de Variáveis

- Exemplo: Blocos aninhados

```
{
  int a=2;
  printf(“%d\n”,a); /* 2 é exibido*/
 {
     int a = 5;
     printf(“%d\n”,a); /* 5 é exibido */
 }
  printf(“%d\n”, ++a); /*3 é exibido*/
}
```



```
{
  int a_outer=2;
  printf(“%d\n”,a_outer);
 {
     int a_inner = 5;
     printf(“%d\n”,a_inner);
 }
  printf(“%d\n”, ++a_outer);
}
```

- Exemplo: Blocos aninhados

```
{
   int a=1, b=2, c=3;
   printf("%3d%3d%3d\n",a,b,c);              /* 1   2   3   */
   {
     int b=4;
     float c=5.0;
     printf("%3d%3d%5.1f\n",a,b,c);          /* 1   4   5.0   */
     a=b;
     {
        int c;
        c=b;
        printf("%3d%3d%3d\n",a,b,c);         /* 4   4   4   */
     }
     printf("%3d%3d%5.1f\n",a,b,c);          /* 4   4   5.0   */
   }
 printf("%3d%3d%3d\n",a,b,c);                /* 4   2   3   */
}
```

Escopo de Variáveis

- Exemplo: Blocos em paralelo

```
{
  int a, b;
  ....
  {                          /* bloco interno 1*/
     float b;
     ....                    /* int a é conhecido, mas int b não*/
  }
.....
{                            /* bloco interno 2*/
   float a;
   ....                      /* int b é conhecido, mas int a não */
                             /* ninguém do bloco 1 é conhecido */
}
....
}
```

Variáveis locais x Variáveis globais

- Variáveis Locais
  - Tem como escopo a função onde foi declarada.
  - Os nomes e valores dessas variáveis tem uso restrito à função que declarou estas variáveis.

- Variáveis Globais
  - Os nomes e valores dessas variáveis podem ser acessados em todo o programa.
  - Devem ser declaradas fora do corpo de todos os procedimentos ou funções do programa.
  - O programa pode alterar uma variável global em qualquer ponto, tornando difícil localizar a alteração que possa ter provocado um erro.

- Exemplo:

```
#include <stdio.h>
int a=1, b=2, c=3;              /* variáveis globais */
int f(void);                    /* protótipo da função*/
int main(void)
{
  printf(“%3d\n”, f( ));                  / 12 é exibido */
  printf(“%3d%3d%3d\n”, a,b,c);   / 4 2 3 são exibidos*/
  return 0;
}

inf f(void)
{
  int b, c;              /* b e c são variáveis locais */
  a=b=c=4;
  return (a+b+c);
}
```

Passagem de parâmetros por valor

- Uma cópia dos valores das variáveis é passada à função.
- As modificações executadas nos valores fornecidos estão restritas ao escopo da função.
- O valor original da variável não é alterado.

Exemplo:

```
#include<stdio.h>
int comput_sum(int n);
int main(void) {
        int n=3, sum;
        printf(“%d\n”,n);        /* 3 é exibido*/
        sum = compute_sum(n);
        printf(“%d\n”,n);        /* 3 é exibido */
        printf(“%d\n”,sum); /* 6 é exibido*/
        return 0;
}

int compute_sum(int n){
         int sum = 0;
         for(; n > 0; --n)
              sum+= n;
        return sum;
}
```

Passagem de parâmetros por referência

- O endereço de memória da variável é fornecido à função e não uma cópia do valor da variável.
- Qualquer alteração executada pela função ocorre na posição de memória fornecida.
- Por isso, as alterações permanecem quando a função é encerrada.
- Ponteiros, que serão apresentados em breve, são utilizados nas passagens por referência.

- Exemplo:

```
#include <stdio.h>
void swap(int *, int *);
int main(void){
 int i=3, j=5;
swap(&i, &j);
printf("%d   %d\n", i, j);  /* 5  e 3 são exibidos */
return 0;
}
void swap (int *p, int *q){
   int tmp;
   tmp = *p;
   *p = *q;
    *q = tmp;
}
```

Passagem de parâmetros por referência

- As variáveis i e j são passadas por referência, ou seja, o endereço de memória das variáveis é repassado à função.

swap (&i, &j)

- Os ponteiros *p e *q, declarados no argumento na função swap, passam a referenciar a posição de memória das variáveis i e j.

void swap (int *p, int *q)

```c
#include<stdio.h>
float valorHora = 30;     //VARIÁVEIS GLOBAIS
int totalHorasMes = 160;

float calculaSalario(int, float*, float); //Protótipos

void main(){ //Programa Principal
float salario, salarioReferencia = 1000, gratificacao = 500;
int totalHoraTrabalhada;
char nome[30];

printf("Nome do fucionario:"); gets(nome);
printf("Total de horas trabalhada:"); scanf("%d",&totalHoraTrabalhada);

salario = calculaSalario(totalHoraTrabalhada,&gratificacao, salarioReferencia);
printf("Valor salario: R$%5.2f\n", salario);
printf("*** Valor original da gratificacao FOI alterado: R$%5.2f\n", gratificacao);
}

float calculaSalario(int totalHora, float *gratificacao, float salarioRef){

    float salario; //VARIÁVEIS LOCAIS
    int dif;

    //Ajusta a gratificacao
    dif = totalHora - totalHorasMes;
    if(dif>=0)
        *gratificacao = 1000;
    else if(dif<0)
        *gratificacao = 0;

    printf("*** Valor da gratificacao foi alterado na funcao: R$%5.2f\n", *gratificacao);
    salario = totalHora*valorHora + *gratificacao;

    return salario; //RETORNA O VALOR FINAL DO SALÁRIO (TIPO float)
}
```

```
Nome do fucionario:Maria Silva
Total de horas trabalhada:180
*** Valor da gratificacao foi alterado na funcao: R$1000.00
Valor salario: R$6400.00
*** Valor original da gratificacao FOI alterado: R$1000.00
```

Passagem de parâmetros por referência

Passagem por Valor

Main	valor
totalHoraTrabalhada	180
gratificacao	500
salarioReferencia	1000
salario	

calculaSalario	valor
totalHora	180
gratificacao	~~500~~ 1000
salarioRef	1000
return salario	6400

Passagem por Referência

Main	valor
totalHoraTrabalhada	180
gratificacao	~~500~~ 1000
salarioReferencia	1000
salario	

calculaSalario	valor
totalHora	180
gratificacao	~~500~~ 1000
salarioRef	1000
return salario	6400

Passagem de parâmetros por referência

- Vetores e matrizes
  - São passados sempre como referência.
  - Os colchetes, após o nome do vetor passado a uma função, indicam que o parâmetro é um vetor. Não precisa especificar o tamanho do vetor.

    ```
    int soma_vetor(int vetor[], int elementos)
    ```

  - Exemplo: Formas possíveis de se declarar a função func() para receber o vetor int vet[100]

    ```
    func(int x[100]);
    func (int x[]);
    func(int *x);
    ```

Passagem de parâmetros por referência

- Vetores e matrizes
  - Ao passar uma matriz bidimensional, se for preciso acessar entradas específicas, o número de colunas precisa ser fornecido.

```
int soma_matriz(int matriz[][5], int linhas)
```

  - Se não há necessidade de acessar entradas específicas da matriz, ela poderá ser tratada como um vetor. Nesse caso, o número de elementos da matriz deverá ser fornecido.

```c
#include<stdio.h>
#define LN  2
#define CL  5

int soma_vetor(int[], int);
int soma_matriz(int , int colunas,int matriz[][colunas]);

void main(){
    int vetor[CL]={10, 20, 30, 40, 50};

    int bidim[LN][CL]={{10,20,30,40,50},
                       {60,70,80,90,100} };
    printf("Soma dos valores do vetor:%d\n", soma_vetor(vetor,CL));
    printf("Soma dos valores da matriz:%d\n\n", soma_matriz(LN,CL,bidim));
    printf("Soma dos valores do matriz:%d\n", soma_vetor(bidim,LN*CL));
}

int soma_vetor(int vetor[], int elementos){
    int i,soma;

    soma=0;
    for(i=0; i<elementos; i++)
     soma += vetor[i];

    return soma;
}

int soma_matriz(int linhas, int colunas,int matriz[][colunas]){
    int i,j,soma;

    soma=0;
    for(i=0; i<linhas; i++)
     for(j=0; j<colunas; j++)
       soma += matriz[i][j];

     return soma;
}
```

```
Soma dos valores do vetor:150
Soma dos valores da matriz:550

Soma dos valores do matriz:550
```

Exercícios

1. Escreva um programa que receba três valores reais e retorne o maior deles.

2. Escreva um programa que:
   - Receba valores para as variáveis x1, x2 e x3. Os valores devem ser inteiros positivos. Crie uma função que valide a entrada de dados, ou seja, o usuário só consegue entrar com os valores adequados.
   - Devolva aos programa principal os valores ordenados com x1<x2<x3. Crie uma função que ordene esses valores.

Referências

Ascencio AFG, Campos EAV. Fundamentos de programação de computadores. São Paulo : Pearson Prentice Hall, 2006. 385 p.

Kelley, A.; Pohl, I., *A Book on C*: programming in C. 4ª Edição. Massachusetts: Pearson, 2010, 726p.

Kernighan, B.W.; Ritchie, D.M. C, *A Linguagem de Programação*: padrão ANSI. 2ª Edição. Rio de Janeiro: Campus, 1989, 290p.

FIM Aula 12