Linguagem C: Funções Parte 2*

Joice Otsuka

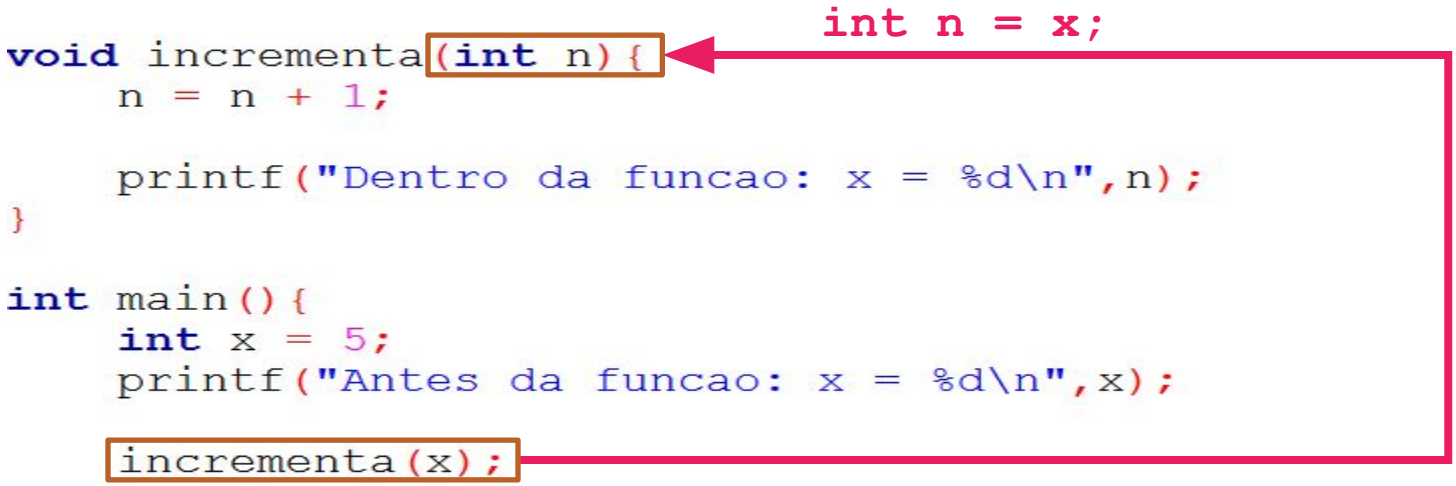
*Baseado no livro: Linguagem C Completa e Descomplicada, de André Backes

Passagem de Parâmetros

- Na linguagem C, os parâmetros de uma função geralmente são passados por ***valor***, ou seja, uma **cópia** do valor do parâmetro é feita e passada para a função.

- Mesmo que esse valor mude dentro da função, nada acontece com o valor de fora da função.

Passagem por valor



Saída:

```
Antes da funcao: x = 5
Dentro da funcao: x = 6
Depois da funcao: x = 5
```

Passagem por referência

- Quando se quer que o valor da variável mude dentro da função, usa-se passagem de parâmetros **por *referência***.
- Neste tipo de chamada, não se passa para a função o valor da variável, mas a sua ***referência*** (seu endereço na memória).

Passagem por referência

- Utilizando o endereço da variável, qualquer alteração que a variável sofra dentro da função será refletida fora da função.
- Ex: função **scanf()**

```
int n;
scanf (“%d”, &n);
```

Passagem por referência

- Para passar um parâmetro por referência, utiliza-se um ponteiro:

```
//passagem de parâmetro por referência
void incrementa(int *n);
```

- Ao chamar a função, é necessário utilizar o operador “&”, para passar um endereço (ponteiro):

```
//passagem de parâmetro por referência
int x = 10;
incrementa(&x);
```

Passagem por referência

- No corpo da função, é necessário usar o operador * (de-referência) sempre que se desejar acessar o conteúdo no endereço (ponteiro) passado como parâmetro.

```
//passagem de parâmetro por valor
void incrementa(int n){
    n = n + 1;
}
//passagem de parâmetro por referência
void incrementa(int *n){
    *n = *n + 1;
}
```

Passagem por referência

```c
#include <stdio.h>
void incrementa(int *n){
    *n=*n+1;
    printf("x dentro da função: %d\n",*n);
}

int main()
{
    int x;
    scanf("%d", &x);
    printf("x na main antes da função: %d\n",x);
    incrementa(&x);
    printf("x na main depois da função: %d\n",x);

    return 0;
}
```

Exercício

- Crie uma função que troque o valor de dois números inteiros passados por referência.

Por valor

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

void Troca(int a,int b){
  int temp;
  temp = a;
  a = b;
  b = temp;
  printf(''Dentro: %d e %d\n'',a,b);
}

int main(){
  int x = 2;
  int y = 3;
  printf(''Antes: %d e %d\n'',x,y);
  Troca(x,y);
  printf(''Depois: %d e %d\n'',x,y);

  return 0;
}
```

Por referência

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

void Troca(int*a,int*b){
  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
  printf(''Dentro: %d e %d\n'',*a,*b);
}

int main(){
  int x = 2;
  int y = 3;
  printf(''Antes: %d e %d\n'',x,y);
  Troca(&x,&y);
  printf(''Depois: %d e %d\n'',x,y);

  return 0;
}
```

Por valor

Saída
Antes: 2 e 3
Dentro: 3 e 2
Depois: 2 e 3

Por referência

Saída
Antes: 2 e 3
Dentro: 3 e 2
Depois: 3 e 2

Exercícios: passagem por valor e por referência - AVA

Arrays como parâmetros

- Quando passamos um array por parâmetro, independente do seu tipo, o que é de fato passado é **o endereço do primeiro elemento do array (referência)**
- É necessário declarar um **segundo parâmetro** (em geral uma variável inteira) para passar para a função o **tamanho do array** separadamente.

Arrays como parâmetros

- Na passagem de um array como parâmetro de uma função podemos declarar a função de diferentes maneiras, todas equivalentes:

```
void imprime(int * v, int n);
void imprime(int v[], int n);
void imprime(int v[10], int n);
```

Arrays como parâmetros

- Exemplo:
  - Função que imprime um array

```
void imprime(int *m, int n){
    int i;
    for (i=0; i< n;i++)
        printf ("%d \n", m[i]);
}

int main (){
    int vet[5] = {1,2,3,4,5};
    imprime(vet,5);

    return 0;
}
```

Memória

posição	variável	conteúdo
119		
120		
121	int vet[5]	123
122		
123	vet[0]	1
124	vet[1]	2
125	vet[2]	3
126	vet[3]	4
127	vet[4]	5
128		

Exercícios

1. Escreva uma função que receba um vetor de n elementos como parâmetro e o preencha com valores aleatórios de 0 a 99.
2. Escreva uma função que imprima um vetor de n inteiros.
3. Escreva a função **pares** que, dado um vetor de n inteiros, retorne o número de elementos pares nesse vetor.
4. Escreva a função **indice_maximo** que, dado um vetor de n inteiros, retorne o índice do maior elemento do vetor.

Exercícios 1 e 2 da lista do AVA (aula 2)

Matrizes como parâmetros

- Para arrays com mais de uma dimensão, é necessário especificar o tamanho de todas as dimensões, exceto a primeira

```
void imprime (int m[][5], int n);
```

Matrizes como parâmetros

- Na passagem de um array para uma função, o compilador precisa saber o tamanho de cada elemento
- Uma matriz pode ser interpretada como um array de arrays.
  - **int m[4][5]**: array de 4 elementos onde cada elemento é um array de 5 posições inteiras.
  - **int m[][5]**: array, onde cada elemento é outro array de 5 posições inteiras.
- Isso é necessário para que o programa saiba que o array possui mais de uma dimensão e mantenha a notação de um conjunto de colchetes por dimensão.

Exemplo: passagem de matriz como parâmetro

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

Exercícios

Exercícios 3 e 4 da Lista do AVA

Matriz como parâmetro com alocação dinâmica

- Para matrizes alocadas dinamicamente, dependerá da abordagem utilizada na alocação da matriz
  - Array unidimensional (ponteiro único, navegação com 1 colchete)
  - Array de array (ponteiro para ponteiro, navegação com 2 colchetes)

Observações

- Na linguagem C não é possível retornar array alocado estaticamente em funções

- Mas é possível retornar o endereço (ponteiro) de um array alocado dinamicamente em uma função. A função que fez a chamada fica responsável por liberar a memória alocada (free)

Exemplo no AVA

Um professor mantém as notas dos alunos das turmas em que leciona em tabelas, onde as linhas correspondem aos alunos e as colunas correspondem às avaliações. O número de alunos e número de avaliações não são conhecidos previamente e podem variar a cada turma.

Solicite o tamanho da turma e número de avaliações e implemente as funções a seguir:

- maior_nota: retorna a maior nota;
- media_turma: retorna a média das notas obtidas pela turma.

Obs: a tabela de notas dos alunos da turma deve ser gerada aleatoriamente com números de 0 a 10.0

Exercícios

Exercícios 5,6 e 7 no AVA

Struct como parâmetro

- Podemos passar uma struct por valor ou por referência
- Temos duas possibilidades
  - Passar por parâmetro toda a struct
  - Passar por parâmetro apenas um campo específico da struct

Struct como parâmetro

- Passar por parâmetro apenas um campo específico da struct
  - Valem as mesmas regras vistas até o momento
  - Cada campo da struct é como uma variável independente. Ela pode, portanto, ser passada individualmente por ***valor*** ou por ***referência***

Struct como parâmetro

- Passar por parâmetro toda a struct
- Passagem por valor
  - Valem as mesmas regras vistas até o momento
  - A struct é tratada com uma variável qualquer e seu valor é copiado para dentro da função
- Passagem “por referência”
  - Valem as regras de uso do operador de-referência “*” e operador de endereço “&”
  - Devemos acessar o conteúdo da struct para somente depois acessar os seus campos e modificá-los.
  - Uma alternativa é usar o ***operador seta “->”***

Struct como parâmetro por referência

Usando “*”

```
struct ponto {
    int x, y;
};

void atribui(struct ponto *p){
    (*p).x = 10;
    (*p).y = 20;
}

struct ponto p1;

atribui(&p1);
```

Usando “->”

```
struct ponto {
    int x, y;
};

void atribui(struct ponto *p){
    p->x = 10;
    p->y = 20;
}

struct ponto p1;

atribui(&p1);
```

Exercício

Considere a definição a seguir.

```
struct vinho {
int cod_produtor ,
cod_uva ;
double preco ;
};
```

Escreva:

- Um procedimento para ler e armazenar os dados de uma garrafa de vinho (passagem por referência);
- Uma função para retornar o vinho mais caro entre dois vinhos;
- Um procedimento para, dados dois registros de vinhos, trocar o conteúdo de um registro com o de outro.

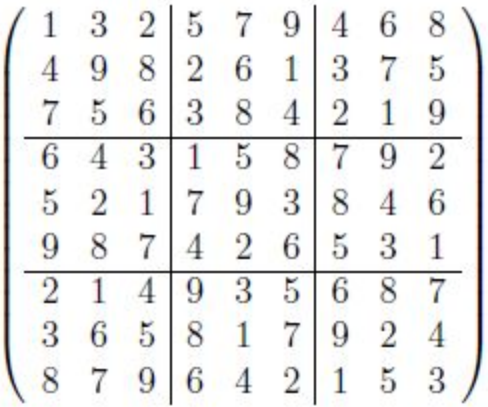
Escreva um programa simples para testar suas rotinas.

Exercício no AVA:
Lista 6: Funções (aula 2)

DESAFIO: SUDOKU

Beecrowd
Problema
1383

```
int main() {
    int n,i;
    int m[9][9];
    int sudoku;
    scanf("%d",&n);
    for(i=0;i<n;i++){
        leMatriz(m,9);
        sudoku = verifica_linhas(m,9) &&
                 verifica_colunas(m,9) &&
                 verifica_submatrizes(m,9);
        printf("Instancia %i\n",i+1);
        printf("%s\n\n",(sudoku?"SIM":"NAO"));
    }
    return 0;
}
```



Referências

- BACKES, André. **Linguagem C:** completa e descomplicada. Rio de Janeiro: Elsevier, 2013. 371 p. ISBN 978-85-352-6855-3.Disponível na Biblioteca.