ALGORITMOS

PROFª: SIMONE DOMINICO

FUNÇÕES E PROCEDIMENTOS

até agora

... ATÉ AGORA VIMOS

- Um argumento passado para uma função funciona da mesma forma que uma variável local à função
- Manipular ou alterar o valor dos argumentos no escopo de uma função não altera seus valores originais no programa que chamou a função

Exemplo

EXEMPLO 1: TROCAR OS VALORES DE DUAS VARIÁVEIS

- Escrever uma função que receba dois argumentos inteiros (a e b) e troque o valor das duas variáveis
- Os valores devem “retornar” alterados ao programa que chamou a função
- Lembre que não é possível retornar mais de um valor de uma função

Solução

UMA TENTATIVA FRUSTRADA

```c
void troca(int x, int y)
{
      int temp;
      temp = x;
      x = y;
      y = temp;
      printf("\n*** Na funcao:");
      printf("x=%d y=%d\n",x,y);
}
```

```c
int main()
{
    int a=5, b=10;
    printf("a=%d b=%d\n",a,b);
    troca(a,b);
    printf("\na=%d b=%d\n",a,b);
}
```

```
ALGORITMO TROCAR

PROCEDIMENTO TROCA(INTEIRO X, INTEIRO Y)
INICIO
      INTEIRO TEMP;
      TEMP = X;
      X = Y;
      Y= TEMP;
      ESCREVA("NA FUNÇÃO", X, "E", Y )
FIM
INICIO
      INTEIRO A=5, B =10;
      ESCREVA("A=", A, "E B=",B)
      TROCE(A,B)
      ESCREVA("A:", A, "E B: ", B)
FIM
```

PASSAGEM DE ARGUMENTOS PARA FUNÇÕES

Passagem de parametro

- POR VALOR
  - PASSA UMA CÓPIA DO VALOR DA VARIÁVEL ORIGINAL NO MOMENTO DA CHAMADA DA FUNÇÃO
  - NÃO EXISTE RELAÇÃO ENTRE A VARIÁVEL UTILIZADA NA CHAMADA E DENTRO DA FUNÇÃO
  - AO FINAL DA EXECUÇÃO DA FUNÇÃO O VALOR ORIGINAL DA VARIÁVEL USADA NA CHAMADA PERMANECE O MESMO

- POR REFERÊNCIA
- –PASSA UM **ENDEREÇO** DE MEMÓRIA, O QUAL É ASSOCIADO AO PARÂMETRO, LOCALMENTE
- –ALTERAÇÕES FEITAS NO CONTEÚDO DO ENDEREÇO AFETAM A VARIÁVEL REFERIDA NA CHAMADA
- –PARA PASSAR PARÂMETROS POR REFERÊNCIA EM C PRECISAMOS USAR PONTEIROS

PONTEIROS

- Mecanismo para referenciar algo indiretamente
  - Diferenciar entre o endereço de um assento no show e a pessoa que está sentada no assento específico

Maria está sentada no assento 50 da fila 118
Código do local: a50f118

Endereço

- Toda variável utilizada por um programa reside em determinado endereço de memória
- O acesso ao endereço de uma variável pode ser feito simbolicamente através de seu nome
- Essa referência é simbólica porque o endereço de memória propriamente dito não é especificado

A declaração:

int x;

produz

Variável	Valor da Variável	Endereço de Memória
x	?	AFA1

O comando:

x = 3;

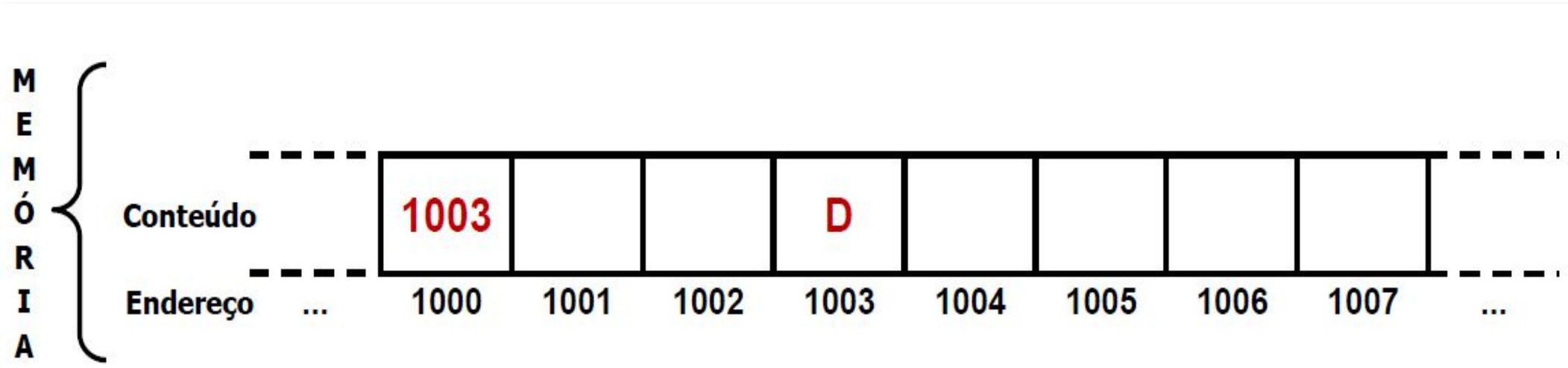
produz

Variável	Valor da Variável	Endereço de Memória
x	3	AFA1

AFA1 = 1010 1111 1010 0001 = 44961

O QUE SÃO PONTEIROS?

- Ponteiros são variáveis que guardam endereços de memória



- ✓ A posição de memória 1000 armazena o endereço de memória 1003, cujo conteúdo é o caractere D
- ✓ O endereço é a posição de uma outra variável na memória
- ✓ Se uma variável **a** armazena o endereço de uma variável **b**, dizemos que **a** aponta para **b**

Declaração

DECLARANDO PONTEIROS

- Sintaxe

```
tipo *nome;
```

Qualquer tipo válido em C

Nome da variável

Asterisco indica que a variável armazena um **endereço de memória**, cujo conteúdo é do tipo especificado

- Exemplos

```
// f é um ponteiro para variáveis do tipo float
float *f;
// p é um ponteiro para variáveis do tipo inteiro
int *p;
// pode declarar junto com variáveis de mesmo tipo
char a, b, *p;
```

Nota: Não confundir o * com o operador aritmético de multiplicação

Operadores

OPERADORES DE PONTEIROS)

- & operador unário, que devolve o endereço de memória de seu operando
- Exemplo:
  - int count, q, *m;
  - m = &count;
- Lê-se: "m recebe o endereço de count"

- * operador unário que devolve o valor da variável localizada no endereço que o segue
- Exemplo:
  - int count, q, *m;
  - q = *m;
- Lê-se: "q recebe o valor contido no endereço m"

Exemplo

VOLTANDO AO SHOW DO U2

Maria está sentada no assento 50 da fila 118 ⇨ Qual o "conteúdo" de a50f118?

***a50f118 = Maria**

João está sentado no assento 43 da fila 77 ⇨ Onde está sentado Joao?

&Joao = a43f77

O ingresso no assento 23 da fila 102 não foi vendido ⇨ Qual o "conteúdo" de a23f102?

***a23f102= ?**

LIXO! Não podemos garantir o conteúdo

ATRIBUIÇÕES COM PONTEIROS

- Como qualquer variável, um ponteiro pode ser usado no lado direito de um comando de atribuição para passar seu valor para um outro ponteiro
- Exemplo:

```
int x = 200, *p1, *p2;
// p1 aponta para x
p1 = &x;
// p2 recebe p1 e também passa a apontar para x
p2 = p1;
```

Exemplo

EXEMPLO DE ATRIBUIÇÃO DE PONTEIROS INTRODUÇÃO À

```c
#include <stdio.h>
int main()
{
    int cont, q, *m;

    m = &cont;     // recebe endereço de count

    cont =  60;
    q = *m;            // recebe valor apontado por m

    printf("m = %p\n", m);  // %p para imprimir ponteiro (endereço apontado)
    printf("q = %d\n", q);
   // conteúdo apontado por m
    printf("conteudo do endereco apontado por m: %d\n\n", *m);
    return 0;
```

Exemplo

OUTRO EXEMPLO

```c
#include <stdio.h>
int main()
{
    int a, *x, *y;
    a = 10;
    x = &a;
    y = x;
    printf("a = %d, *x = %d, *y = %d \n",a,*x, *y);
    printf("a = %d, x = %p, y = %p\n\n", a, x, y);
    *x = 20;
    printf("a = %d, *x = %d, *y = %d \n",a, *x, *y);
    printf("a = %d, x = %p, y = %p\n\n", a, x, y);
    return 0;
}
```

Exemplo

RETOMANDO A FUNÇÃO QUE TROCA OS VALORES DE DUAS VARIÁVEIS

- Escrever uma função que receba dois argumentos inteiros (a e b) e troque o valor das duas variáveis
  - Os valores devem “retornar” alterados ao programa que chamou a função
  - Lembre que não é possível retornar mais de um valor de uma função

Solução

SOLUÇÃO UTILIZANDO PONTEIROS

```
ALGORITMO TROCAR

PROCEDIMENTO TROCA(INTEIRO *X, INTEIRO *Y)
INICIO
      INTEIRO TEMP;
      TEMP = *X;
      *X = *Y;
      *Y= TEMP;
      ESCREVA("NA FUNÇÃO", *X, "E", *Y )
FIM
INICIO
      INTEIRO A=5, B =10;
      ESCREVA("A=", A, "E B=",B)
      TROCE(&A,&B)
      ESCREVA("A:", A, "E B: ", B)
FIM
```

A FUNÇÃO RECEBE DOIS PONTEIROS PARA INTEIROS COMO ARGUMENTOS

NA PASSAGEM DOS ARGUMENTOS USAMOS OS ENDEREÇOS DE MEMÓRIA DAS VARIÁVEIS DECLARADAS NO MAIN

Solução

SOLUÇÃO UTILIZANDO PONTEIROS

```c
// função recebe endereços e troca seus conteúdos
void troca(int *x, int *y){
     int temp;
     temp=*x;
     *x = *y;
     *y = temp;
     printf("\n*** Na funcao:\n");
     printf("Valores: x = %d; y = %d\n", *x,*y);
     printf("Enderecos: x = %p; y = %p\n", x, y);
}
int main(){
     int a=5, b=10;
     printf("\n*** No main:\n");
     printf("Valores: a = %d; b = %d\n",a,b);
     printf("Enderecos: a = %p; b = %p\n",&a,&b);
     troca(&a,&b);
     printf("\n*** De volta ao main:\n");
     printf("Valores: a = %d; b = %d\n",a,b);
     printf("Enderecos: a = %p; b = %p\n",&a,&b);
     return 0;
}
```

A FUNÇÃO RECEBE DOIS PONTEIROS PARA INTEIROS COMO ARGUMENTOS

NA PASSAGEM DOS ARGUMENTOS USAMOS OS ENDEREÇOS DE MEMÓRIA DAS VARIÁVEIS DECLARADAS NO MAIN

Solução

SOLUÇÃO UTILIZANDO PONTEIROS

- Saída do programa implementado!

EXECUÇÃO DA FUNÇÃO

```
*** No main:
Valores: a = 5, b = 10
Enderecos: a = 0028FF0C; b = 0028FF08

*** Na funcao:
Valores: x = 10, y = 5
Enderecos: x = 0028FF0C; y = 0028FF08

*** De volta ao main:
Valores: a = 10, b = 5
Enderecos: a = 0028FF0C; b = 0028FF08
```

EXECUÇÃO DA FUNÇÃO

EXECUÇÃO DA FUNÇÃO

EXECUÇÃO DA FUNÇÃO

EXECUÇÃO DO MAIN

- Os valores foram trocados, pois a passagem de parâmetros foi feita por referência.
- Dentro do subprograma, x e y receberam os endereços de memória de a e b, o seus conteúdos são alterados.

Exemplo

EXEMPLO 2: CALCULAR AS RAÍZES DE UMA EQUAÇÃO

- Solução: usar passagem de parâmetros por referência para repassar os valores das duas raízes
- Estrutura da função: os parâmetros a, b e c devem ser passados por valor, pois não serão alterados
- Os parâmetros x1 e x2 devem ser passados por referência, para que possam ser acessados pelo programa que chama a função!

```
void bhaskara(float a, float b, float c, float *x1, float *x2)
{
...
}
```

Exemplo

EXEMPLO 2: CALCULAR AS RAÍZES DE UMA EQUAÇÃO INTRODUÇÃO

```c
/* Exemplo de rotina que acha raízes reais de uma equação de 2° grau
   Entradas: valores  a,b,c dos coeficientes
   Saídas: valores das raízes x1 e x2 (referências) */
#include<stdio.h>
#include<math.h>

void bhaskara(float, float, float, float*, float*); //protótipo

int main(){
    float r1, r2;
    bhaskara(-1, 3, 9, &r1, &r2);
    printf("x1 = %f -- x2 = %f", r1, r2);
}

// função bhaskara. Entradas a,b,c (passagem por valor) e saídas x1, x2
// Assume-se que as raizes são reais
void bhaskara(float a, float b, float c, float* x1, float* x2){
     float raizdisc;
     raizdisc = sqrt(b*b-4*a*c); // raiz do discriminante
     *x1 = (-b + raizdisc)/(2*a);
     *x2 = (-b - raizdisc)/(2*a);
}
```