MC102 Algoritmos e Programação de Computadores

Aula sobre Ponteiros – 2ª Parte

1. Objetivos

- Conceituar e ilustrar a aplicação de variáveis do tipo ponteiro, no contexto de procedimentos.

2. Motivação

- Ponteiros úteis quando trabalhamos com o desenvolvimento de rotinas, em específico, na parte de procedimentos. Usando-se ponteiros para realizar a passagem por referência, temos uma mesma posição de memória sendo acessada pelo programa principal e pela rotina. Além disso, o uso de ponteiros pode melhorar a eficiência de rotinas.

3. Uso de Ponteiros em Procedimentos

A inserção do caracter * **(asterisco)** antes da declaração de uma variável em um procedimento indica que ela é declarada com **Passagem por Referência** e a ausência desse caracter indica que a variável realiza **Passagem por Valor**;

A. Passagem por Valor: a expressão correspondente ao parâmetro é avaliada e apenas seu valor é passado para a variável correspondente ao parâmetro dentro da rotina;

B. Passagem por Referência: o parâmetro que vai ser passado na chamada da rotina deve ser *necessariamente uma variável*. Isto porque não é o valor da variável que é passado no parâmetro, mas sim sua referência. Qualquer alteração de valor no parâmetro correspondente refletirá em mudanças na variável correspondente, externa ao procedimento.

Exemplo: programa com uma rotina chamada Conta, contendo dois parâmetros, um por valor (a) e outro por referência (b).

No algoritmo:

```
programa exemplo;

conta (a : real, *b : real)
início
        a = 2 * a;
        *b = 3 * *b;
        escreva("O valor de a = ",a," e o valor de b = ",*b);
fim;

início
        x, y : real;
        x = 10;
        y = 30;
        conta (x,&y);
        escreva ("O valor de x = ",x," e o valor de y = ",y);
fim.
```

Na linguagem:

```
void conta (float a, float *b)
{
        a = 2 * a;
        *b = 3 * *b;
        printf("O valor de a = %f e o valor de b = %f.\n", a, *b);
}

void main()
{
        float x, y;
        x = 10;
        y = 30;
        conta (x,&y);
        printf ("O valor de x = %f e o valor de y = %f.\n ", x, y);
}
```

- O comando de escrita localizado dentro do procedimento imprime os valores a = 20 e b = 90. No programa principal, depois da chamada do procedimento Conta, são impressos os valores x = 10 e y = 90, já que a passagem do parâmetro correspondente a x é por valor e a passagem do parâmetro correspondente a y é por referência (x tem o mesmo valor de antes da chamada e y tem o valor atualizado pela rotina Conta).

3.1 Considerações sobre a Passagem de Estruturas Grandes como Parâmetros

- Muitas vezes, quando temos passagem de estruturas grandes (como vetores e matrizes) como parâmetros por valor, é preferível recodificar a rotina para que esta seja feita como parâmetro por referência;
- O motivo disto é justamente o fato destas estruturas serem duplicadas na passagem por valor;
- Na passagem de parâmetros por referência, apenas a referência do objeto é transferida, gastando uma quantidade de memória constante para isso.

3.2 Ponteiros e Vetores

Considerando-se o seguinte fragmento:

```
caracter str[80], *p1;
p1 = str;
```

Neste caso, p1 foi inicializado com o endereço do primeiro elemento do vetor str. Para se acessar o 5^{o} elemento em str, teria que ser escrito da seguinte forma:

```
str[4]
```

Ou

```
*(p1+4)
```

Os dois comandos devolvem o quinto elemento. Como as matrizes começam em 0, deve-se usar 4 para indexar str. Também deve-se adicionar 4 ao ponteiro

p1, para acessar o quinto elemento, porque p1 aponta atualmente para o primeiro elemento de str.

No caso da linguagem C, são fornecidos dois métodos para acessar elementos de vetores: aritmética de ponteiros e indexação de vetores. Aritmética de ponteiros pode ser mais rápida que indexação de vetores. Já que velocidade é geralmente uma consideração em programação, programadores em C normalmente usa ponteiros para acessar elementos de vetores.

3.3 Vetores de Ponteiros

Ponteiros podem ser organizados em vetores como qualquer outro tipo de dado. A declaração de uma matriz de ponteiros inteiro, de tamanho 10, é:

inteiro *x[10];

Para atribuir o endereço de uma variável inteira, chamada var, ao terceiro elemento do vetor de ponteiros, deve-se escrever

x[2] = &var;

Para encontrar o valor de var, escreve-se

*x[2]

Se for necessário passar um vetor de ponteiros para uma rotina (procedimento ou função), simplesmente deve-se chamar a rotina com o nome do vetor sem qualquer índice. Por exemplo, uma rotina que recebe o vetor x está ilustrada a seguir:

```
mostra_vetor (inteiro *q[])
início
        inteiro t;
        para t de 1 incr 1 até 10 faça
        início
                escreva(*q[t]);
        fim
fim
```

Exemplo: programa usando procedimentos/funções com cadeias de caracteres.

```
#include <stdio.h>

void copia (char *dest,char *fonte)
  {
  int i=0;
  while (fonte[i] != '\0')
    {
    dest[i] = fonte[i];
    ++i;
    }
  dest[i]='\0';
  }

void copiaboa (char *dest,char *fonte)
  {
  while ((*(dest++) = *(fonte++))!=0)
    ;
  }

int diferente (char *dest,char *fonte)
  {
  while (*dest != '\0' || *fonte != '\0')
    if (*(dest++) != *(fonte++))
      return 1;
  return 0 ;
  }

int main ()
  {
  int i;
  char nome[10] = "joao";

  copia(nome,"fernanda");
  copia(nome,"silva");

  printf("%s %s %s\n",nome, nome, nome);
  printf("tah na pos de mem  %d\n", nome);
```

```
 for (i=0; i < 10; ++i)
   printf("letra[%d] eh %d\n",i, nome[i]);

 if (!diferente(nome,"bbb"))
   printf("O fernanda tah aqui\n");

 getchar();
 }
```

Exemplo: um programa que gere um vetor de três dimensões (X, Y e Z) em que cada posição guarda a soma de suas coordenadas. As dimensões da matriz devem ser determinadas em tempo de execução e o programa deve informar os valores gerados.

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

main(){

 int *matriz, x_max, y_max, z_max, x, y, z;

 puts("ENTRE COM AS DIMENSOES");
 printf("\nDimensao 1 (X): ");
 scanf("%d", &x_max);
 printf("\nDimensao 2 (Y): ");
 scanf("%d", &y_max);
 printf("\nDimensao 3 (Z):");
 scanf("%d", &z_max);

 matriz = (int *)malloc(x_max*y_max*z_max*sizeof(int));

 if(!matriz) puts("Nao consegui alocar memoria.");
 else{

   for(x = 0; x < x_max; x++)
     for(y = 0; y < y_max; y++)
           for(z = 0; z < z_max; z++)
             //atribui os valores as posicoes
             *(matriz + x*(y_max*z_max) + y*(z_max) + z) = (x + y + z);
```

```
   for(x = 0; x < x_max; x++)
     for(y = 0; y < y_max; y++)
          for(z = 0; z < z_max; z++)
            //exibe os valores das posicoes
            printf("\n%d+%d+%d=%2d", x, y, z, *(matriz + x*(y_max*z_max)
+ y*(z_max) + z));
  }

  free(matriz);

  getche();

}
```

Exemplo: programa que lê uma matriz e a imprime. O programa deve ler o número de colunas e linhas do teclado. O programa deve trocar duas linhas da matriz de posição. Os números das linhas a serem trocadas devem ser lidos do teclado.

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

void main(void)
{
          int *matriz;
          int i, j, lin1, lin2;
          int temp, tam;
          int linhas, colunas;
          tam = sizeof(int);
          printf("Entre com o numero de linhas e colunas.\n");
          printf("Linhas? "); scanf("%d", &linhas);
          printf("Colunas? "); scanf("%d", &colunas);
          matriz = malloc(linhas*colunas*tam);
          if (!matriz){
                    printf("Nao consegui alocar a memoria suficiente.\n");
                    exit(1);
          }
```

```
        for(i=0; i<linhas; i++){
                for (j=0; j<colunas; j++){
                        printf("Elemento %d %d = ", i, j);
                        scanf("%d", matriz+(i*colunas+j)*tam);
                }
        }
        for(i=0; i<linhas; i++){
                for (j=0; j<colunas; j++){
                         printf("El%2d,%2d = %2d ", i, j,
                         *(matriz+(i*colunas+j)*tam));

                }
                printf("\n");
        }
        printf("Que linhas quer trocar? "); scanf ("%d %d", &lin1, &lin2);
        for (i=0; i<colunas; i++){
                temp = *(matriz + (lin1*colunas+i)*tam);
                *(matriz + (lin1*colunas+i)*tam) = *(matriz +
                (lin2*colunas+i)*tam);
                *(matriz + (lin2*colunas+i)*tam) = temp;
        }
        for(i=0; i<linhas; i++){
                for (j=0; j<colunas; j++){
                        printf("El%2d,%2d = %2d ", i, j,
                        *(matriz+(i*colunas+j)*tam));
                }
                printf("\n");
        }
        free(matriz);
        printf("Acabou.");
}
```

Referências

Herbert Schildt, "C Completo e Total", Makron Books.
Herbert Schildt, "C Avançado - Guia do Usuário", McGraw-Hill.
Keith Tizzard, "C for Professional Programmers", John Wiley & Sons.