MAC0110 - Introdução à Programação

Prof. Roberto Hirata Jr.

Lista de ponteiros – Data de entrega: 30/5/2016

- Fundamentos - um ponteiro é uma variável que armazena um endereço de memória.
- Declaração - `<tipo> * <variável> ;`
- Observação 1 - O asterísco antes da variável indica que ela é um ponteiro.
- Observação 2 - Em geral, o nome da variável deve começar com a letra `p`, para indicar que a variável é um ponteiro.
- Exemplos:

```
int *pi ; /* Define um ponteiro para um inteiro */
float *pf ; /* Define um ponteiro para um float */
```

- Impressão - para imprimir o conteúdo de um ponteiro usa-se o formato `%p`, por exemplo: `printf(``%p\n'',pi) ;`. Note que o asterisco não foi usado.
- Erro comum: **usar um ponteiro sem ter sido inicializado.**
- Inicialização: a atribuição de um valor a um ponteiro pode ser feita de três maneiras.

  Atribuindo-se ao ponteiro um endereço previamente conhecido, por exemplo:

```
pi = 0x1234h ;
```

  Usando o operador `&`, que toma o endereço da variável à qual ele está operando, por exemplo: `pf = &meuPi ;`, onde `meuPi` foi declarado anteriormente com sendo uma variável do tipo `float`. Após a atribuição, `pf` conterá o endereço da variável `meuPi`, que pode, ou não, ter sido inicializada.

  Através de uma função cujo valor de retorno é um endereço.

- Acesso ao conteúdo da memória apontada por uma variável do tipo ponteiro. Usa-se o operador `*`, por exemplo: `*pf = 3.141592654 ;`; altera o conteúdo da variável `meuPi`.

  Observação: O fato de, na declaração do ponteiro, usar-se o mesmo símbolo que o operador para dar acesso ao conteúdo da memória apontada pelo ponteiro é um potencial causador de confusões, principalmente quando se usa duplo direcionamento, isto é, ponteiro para ponteiro.

- Observação: os exercícios abaixo devem ser feitos em linguagem ANSI C e entregues através do site da disciplina.

E1. Declare uma variável inteira `n`, um ponteiro para uma variável inteira `pn` e inicialize o ponteiro para o apontar para a variável `n`. Num loop até que a entrada seja igual a -99999, leia um inteiro, coloque em `n`, imprima o valor de `n` e de `*pn`.

E2. Declare duas variáveis `double x` e `y` e um ponteiro para uma variável `double pd`. Inicialize `x` $= \frac{1}{3}$ e `y` $= \frac{1}{5}$. Imprima o valor de `x` e `y`. Inicialize o ponteiro `pd` com o endereço de `x` e imprima o conteúdo do endereço apontado por `pd` (`*pd`), o valor do endereço `pd` (`pd`) e o endereço de `pd` (`&pd`). Para imprimir endereços, use o formato de impressão de ponteiros, o `%p`.

E3. Usando o exercício **E2**, mude o valor de `pd` para apontar para a variável `y` e , novamente, imprima o conteúdo do endereço apontado por `pd`, o valor de `pd` e o endereço de `pd`. Finalmente, faça a multiplicação de `*pd` por `x`, coloque em `*pd` e imprima o conteúdo de `*pd`, o valor de `pd`, o endereço de `pd`, o valor de `x` e de `y`.

E4. Declare três variáveis inteiras `m`, `n` e `p` e um ponteiro para uma variável inteira `pz`. Inicialize `m` com 10, `n` com 20 e `p` com 30. Inicialize `pz` com o endereço de `m`. Imprima `m`, `pz` e `*pz`. Mude o valor de `pz` para apontar para o endereço de `n` e imprima o conteúdo de `*pz` e de `pz`. Mude mais uma vez o valor de `pz` para apontar para o endereço de `p` e imprima o conteúdo de `*pz` e de `pz`.

E5. Implemente uma função que retorna o maior valor dentre duas variáveis `m` e `n` e troca os valores de `m` e `n` se `m` $>$ `n`. Na função `main`, chame a função de duas maneiras diferentes: (1) declare duas variáveis inteiras no `main`, inicialize-as e passe o **endereço** delas para a função; (2) além das variáveis inteiras, declare dois ponteiros na função `main`, que serão inicializados para apontar para essas variáveis inteiras. Em seguida, passe esses ponteiros para a função.

E6. Repita o exercício anterior mas agora use uma função troca (que recebe ponteiros para duas variáveis inteiras e troca o valor dessas variáveis) dentro da função que retorna o maior valor dentre as duas entradas. Preste atenção em como passar os parâmetros para a função troca.

E7. Na função `main`, crie um vetor inteiro de 100 posições e inicialize cada posição com o dobro do valor de seu índice. Crie um ponteiro para um inteiro `pz` e inicialize-o com o valor de `V`, isto é, `pz = V`. Imprima o valor de `V[0]` e de `*pz`. Verifique se eles são iguais.

E8. Na função `main`, crie um vetor inteiro de 100 posições e inicialize cada posição com o dobro do valor de seu índice. Crie um ponteiro para um inteiro `pz` e inicialize-o com o valor de `V`, isto é, `pz = V`. Imprima o valor de `V[1]`, de `*(pz+1)` e de `*pz+1`.

E9. Na função `main`, crie um vetor inteiro de 100 posições e inicialize cada posição com o dobro do valor de seu índice. Crie dois ponteiros para inteiros `pz` e `pw` e inicialize-os com o valor de `V+1` e de `&V[1]`, respectivamente. Imprima o valor de `V[1]`, de `*pz`, de `*pw`, de `pz` e de `pw`. Verifique se eles são iguais.

E10. Na função `main`, crie um vetor inteiro de 100 posições e inicialize cada posição com o dobro do valor de seu índice. Crie um ponteiro para um inteiro `pz` e inicialize-o com o endereço do primeiro elemento do vetor. Finalmente, imprima o valor das primeiras 10 posições do vetor usando o ponteiro.

E11. Repita o exercício anterior, fazendo a impressão através de uma função:

`void imprimeVetor(int *p, int n)`,

que recebe o vetor e seu tamanho como parâmetros.