Introdução à Programação #18

Henrique Dias @ Pplware

31 de dezembro 2014

1 Apontadores

Os apontadores, como já foi referido anteriormente, são um tipo de variáveis que armazenam endereços.

1.1 Declaração de apontadores

A declaração deste tipo de variáveis é bastante simples: para declarar um apontador basta colocar um * (asterisco) antes do nome da variável.

Sintaxe Veja então a sintaxe da declaração de ponteiros.

```
tipo *nome;
```

Tal como nas restantes variáveis, temos que colocar o tipo de dados na declaração da variável devido ao facto do espaço ocupado por cada tipo de dados ser diferente.

Quando não é dado um valor a um apontador, geralmente este assume o valor da constante NULL, ou seja, não aponta para endereço nenhum.

A constante NULL está contida na biblioteca stdlib, ou seja, standard library e o seu valor é, na maioria das vezes, 0.

1.2 Inicialização de apontadores

Para inicializar um apontador, devemos igualar o seu valor ao endereço de uma outra variável do mesmo tipo.

Exemplo 1 Para exemplificarmos a relação entre uma variável comum e um ponteiro, iremos declarar o seguinte código:

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

int main()
{
    int numero = 5;
    int *ponteiro = &numero;

    printf("%d e %d", numero, *ponteiro);

    return 0;
}
```

Dentro da função principal, declaramos duas variáveis: a primeira, chamada *numero*, é uma variável do tipo de números inteiros e conta com o número 5.

De seguida, declaramos um ponteiro cujo nome é, exatamente, ponteiro e este ponteiro **aponta** para o endereço da variável *numero*. Podemos então dizer que "ponteiro aponta para numero"ou "ponteiro é o endereço de numero".

Depois imprimimos o valor de *numero* e do endereço que para o qual aponta *ponteiro*. Isto irá imprimir "5 e 5", pois é o mesmo dizer **numero** e ***ponteiro** visto que o ponteiro *ponteiro* aponta para a variável *numero*.

Exemplo 2 O segundo exemplo é minimamente bizarro: efetuar a soma de duas variáveis utilizando apenas apontadores que apontem para estas mesmas variáveis. Ora veja:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <windows.h>
4
5  int main()
6  {
7      SetConsoleOutputCP(65001);
8
9      int a, b, c;
10     int *p, *q;
11
12     a = 5;
13     b = 4;
14
15     p = &a;
16     q = &b;
17     c = *p + *q;
18
19     printf("A soma de %d e %d é %d.", a, b, c);
20     return 0;
21 }
```

Se está a ter problemas com a codificação dos caracteres no Windows, inclua a biblioteca "Windows"e altere a página de codificação da linha de comandos como está na linha 7.

No exemplo anterior pode visualizar que são declaradas cinco variáveis: a *a*, **b** e *c* que são do tipo número inteiro e, de seguida, dois apontadores, *p* e *q*. Posteriormente é atribuído o valor 5 à variável *a* e o valor 4 à variável *b*.

Agora, para efetuar a soma utilizando os operadores, igualamos o apontador *p* ao endereço da variável *a* e o apontador *q* ao endereço da *b*. Finalmente, a soma (que estará contida na variável *c*) é igual à soma do conteúdo dos endereços que para os quais os apontadores **p** e **q** estão a apontar.

Exemplo 3 O exemplo anterior aparenta ser inútil, pois poderíamos fazer o mesmo com menos linhas de código. Mas, e se precisar de, por algum motivo, criar uma função que troque o valor de duas variáveis? Poderíamos, inicialmente, pensar no seguinte:

```
1  int trocaDeValores( int x, int y)
2  {
3      /* Esta função está errada */
4      int temp;
5
6      temp = x;
7      x = y;
8      y = temp;
9
10     return 0;
11 }
```

Mas a função apresentada não vai realmente alterar os valores das variáveis, pois apenas recebe os seus valores e "coloca-os"numas míseras variáveis locais. A função correta teria que ser a seguinte:

```
1  int trocaDeValores( int *p, int *q)
```

```
{
    int temp;

    temp = *p;
    *p = *q;
    *q = temp;

    return 0;
}
```

Com esta função sim, poderíamos trocar os valores de duas variáveis, pois ao termos acesso aos seus endereços podemos facilmente efetuar as mudanças.