Introdução à Programação #19

Henrique Dias @ Pplware

7 de janeiro de 2015

1 Operações com Apontadores

Uma das principais vantagens da utilização de apontadores é a fácil modificação de outras variáveis e que o C trabalha de forma mais rápida quando são utilizados ponteiros.

Tal como acontece com as outras variáveis, podemos incrementar e decrementar apontadores, ou seja, incrementar ou decrementar ou endereço (valor) que contém o apontador.

Imagine que temos um *array* do tipo *double* com dois elementos. Sabemos também que cada variável deste tipo ocupa, geralmente, 8 *bytes* na memória RAM. De seguida, cria um apontador cujo endereço aponta para esse *array*. O que vai acontecer é que esse endereço apontará para o **primeiro** *byte* do *array* e não para todos eles.

```
1 double numeros[2]; /* Declaração do array com dois elementos. */
2 double *apontador;
3
4 apontador = numeros;
```

Hipoteticamente falando, esse apontador armazena o endereço 4550. Então, sabendo que o *array* ocupa 16 *bytes* (visto que tem capacidade para dois elementos do tipo *double*), podemos dizer que este *array* ocupa todos os *bytes* entre 4550 e 4565 (inclusive).

Se procedermos a uma incrementação (*apontador++*), este apontador deixará de apontar para o endereço 4550, apontando agora para o 4558. Porquê? Porque a próxima variável do *array* localiza-se nessa posição e não na posição 4551 devido ao tamanho ocupado por esse tipo de dados. O mesmo ocorre quando decrementamos um valor.

Imaginando os endereços de variáveis como números decimais (não se esqueça de que são representados em hexadecimal), as duas linhas seguintes seriam equivalentes:

```
1 apontador++;
2 apontador + sizeof(double);
```

Exemplo Para exemplificarmos o uso de operações e apontadores, vamos fazer uma Progressão Aritmética, onde há um termo inicial e uma razão. Cada termo é igual ao termo anterior somado com a razão.

$$n[y] = n[y-1] + r$$

Poderíamos fazer este pequeno programa da seguinte forma (o código abaixo está explicado):

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     int pa[10], razao;
```

```
7       int *pointer;
8
9       /*
10       * Aqui pedímos o termo inicial, ou seja, o primeiro
11       * número da PA (Progressão Aritmética).
12       *
13       * De seguida o apontador é definido para apontar para
14       * o array pa.
15       */
16      printf("Insira o termo inicial: ");
17      scanf("%d", &pa[0]);
18      pointer = pa;
19
20      printf("Insira razão: ");
21      scanf("%d", &razao);
22
23      while(pointer != &pa[9]) {
24          /*
25           * Cada valor da PA é definido somando o valor corrente do apontador com a
    razão.
26           * De seguida, o apontador é incrementado de forma a apontar para o próximo
27           * elemento do array. (Isto acontece enquanto o apontador for diferente do
28           * endereço do último elemento da PA.)
29           */
30          *(pointer + 1) =  *pointer + razao;
31          pointer++;
32      }
33
34      printf("PA");
35
36      for(pointer = pa ; pointer <= &pa[9] ; pointer++) {
37          /*
38           * Aqui todos os elementos do array são percoridos
39           * e imprime-se assim a Progressão Aritmética criada.
40           */
41          printf(" -> %d", *pointer);
42      }
43
44      return 0;
45  }
```