Ponteiros

Introdução

Um recurso poderoso da linguagem C é a possibilidade de referenciar e manipular os endereços onde as variáveis estão armazenadas na memória. E essa possibilidade existe por meio da existência de ponteiros na linguagem C.

Vejamos a seguinte declaração:

int x = 2;

Ao encontrar essa linha, o compilador irá proceder do seguinte modo:

1. Alocará na memória RAM um espaço de memória correspondente ao tipo "int". No caso, 4 bytes[1].
2. Armazenará nesse espaço de 4 bytes o valor 2.
3. Fará com que a palavra "x" seja o nome pelo qual se pode modificar e alterar os valores armazenados nesse espaço de memória de 4 bytes.

Esquematicamente, supondo que esse espaço de memória alocado para "x" seja o endereço 1000:

Endereço	RAM
1000	2
1004	

No esquema anterior, pode-se observar que foi armazenado na memória, na posição 1000, o valor 2. Só que, ao invés de no programa eu utilizar diretamente o valor 1000 como forma de manipular o valor 2, a linguagem C (e qualquer outra linguagem de alto nível), permite que eu defina um nome como "sinônimo" desse valor 1000. No caso, foi definido o nome "x".

Só que e se eu quisesse, manipular esse endereço (valor 1000) diretamente? A resposta é direta: usando ponteiros.

Por exemplo, se eu acrescentasse as seguintes linhas:

```
int *p;
p = &x;
```

Ao encontrar essa linha, o compilador irá proceder do seguinte modo:

1. Alocará na memória RAM um espaço de memória correspondente ao tipo "int *". No caso, 4 bytes.

[1] 4 bytes supondo uma máquina de 32 bits. Se fosse em uma máquina de 64 bits, o tamanho do campo de endereços corresponderia a 8 bytes, sendo portanto, alocada essa quantidade de bytes.

2. Fará com que a palavra “p” seja o nome pelo qual se pode modificar e alterar os valores armazenados nesse espaço de memória de 4 bytes.
3. Atribuirá a posição de memória onde se encontra “x” como sendo o valor da variável “p”.

Esquematicamente, supondo que esse espaço de memória alocado para “p” seja o endereço 1004 (conforme foi dito, “p” é uma espécie de sinônimo do número 1004):

Endereço	RAM
1000	2
1004	1000

No esquema anterior, pode-se observar que foi armazenado na memória, na posição 1004, o valor 1000. Ou seja, o valor de “p” contém o endereço da área da memória denominada “x” pelo compilador C. Logo, **“p” é o que se chama de ponteiro na linguagem C**!

Exercícios:

1. Para cada uma das linhas de código seguintes, diga quantos bytes de memória são alocados:
   a. int i = 2;
   b. char c = ‘B’;
   c. int *i;
   d. char *c;
2. Para cada uma das declarações de variáveis seguintes, declare uma variável que permita armazenar o endereço da variável declarada:
   a. float f; **Ex.: float *pf; pf = &f;**
   b. char c;
   c. int x;
   d. unsigned int x;
3. Analise cada uma das linhas de código seguintes e diga o que cada uma delas faz:
   a. char c = ‘A’; printf(“%i”, c);
   b. char c = ‘A’; printf(“%c”, c);
   c. char c = ‘A’; printf(“%p”, &c);
   d. char c = ‘A’, *pf = &c; printf(“%p”, &pf);
   e. char c = ‘A’, *pf = &c; printf(“%p”, pf);
   f. char c = ‘A’, *pf = &c; printf(“%i”, sizeof(pf));
   g. char c = ‘A’, *pf = &c; printf(“%p”, sizeof(c));
   h. char c = ‘A’, *pf = &c; printf(“%p”, sizeof(&c));
   i. char c = ‘A’, *pf = &c; printf(“%p”, sizeof(&pf));
4. Qual é o maior endereço de memória que consigo referenciar em uma máquina com arquitetura de 32 bits? Conseqüentemente, quanto de memória essa máquina suporta?