Ponteiros e endereços

Considere-se uma variável inteira *i* (*int i*). A expressão:

&i

fornece a **referência** (**ou o endereço**) **dessa variável**; isto é algo que permite identificar e aceder a essa variável. Fisicamente a referência é o número (endereço) da posição de memória que a variável ocupa (e através do qual se lhe pode aceder).

Um **ponteiro** (ou apontador) é uma **variável que contém o endereço de uma outra variável**.

Se pi for um ponteiro para uma variável inteira (*int * pi*) então podemos fazer a atribuição:

pi = &i

significando que colocamos em *pi* um valor que é a referência ou endereço da variável *i*.

A manipulação de variáveis através de ponteiros é possibilitada pelos operadores "&" e "*".

- O operador & aplica-se a uma variável e fornece o endereço dessa variável
- O operador * aplica-se a um endereço e fornece o valor da variável que esse endereço refere.

Exemplo:

```
main()
{
    ...
int x = 9;      /* a variável inteira x é inicializada com o valor 9*/
int * px;       /* px é um ponteiro para inteiro*/

    px = &x;  /* atribui-se a px o valor do endereço da variável x */
    *px = 12  /* x fica com valor 12 */

    ...
}
```

Passagem de parâmetros por referência

Em C todos os **argumentos** de funções **são passados por valor**.

À função que é chamada, é dada uma cópia dos valores dos **argumentos** e ela cria outras variáveis temporárias (os **parâmetros**) para armazenar estes valores. ➔ Uma função chamada não pode alterar **directamente** o valor de uma variável da função que a chama, pode apenas alterar a sua cópia temporária.

Ou seja, uma variável passada por valor a uma função não é modificada pela execução da função.

Os ponteiros têm muitas utilizações na linguagem C. Uma delas é permitirem implementar a passagem de parâmetros a funções por referência, isto é, de forma que a função receba e possa modificar as próprias variáveis passadas como argumento.

Quando um **parâmetro** é **passado** a uma função **por referência**, o parâmetro formal correspondente **recebe a localização na memória da variável actual** em vez do seu valor.

Assim todas as instruções que forem executadas sobre o parâmetro formal dentro da função são, de facto, executadas sobre a variável actual.

Exemplo 1

```
/* Passagem de argumentos por referência */

#include <stdio.h>

/* Função que troca os valores entre dois inteiros passados como
argumento. */
void troca(int *a, int *b)

{
   int temp;
     temp = *a;
     *a = *b;
     *b = temp;
}

/* Exemplo de utilização da função troca. */

int main()
{
   int x = 10, y = 20;

   printf("Antes: X = %d\tY = %d\n", x, y);
   troca(&x, &y);
   printf("Depois: X = %d\tY = %d\n", x, y);
}
```

Resultado da execução:

```
Antes: X = 10   Y = 20
Depois: X = 20  Y = 10
```