Aritmética de Ponteiros

Módulo 8 Aula 2

Linguagem C, o Curso Definitivo WR Kits

Autor: Dr. Eng. Wagner Rambo

Conforme vimos, os ponteiros são variáveis como quaisquer outras e por este motivo é possível de se realizar operações aritméticas com os mesmos, tais como incremento, decremento, diferença, comparação.

No caso de incremento e decremento, o ponteiro avança ou retrocede o número de bytes de acordo com o tipo para o qual aponta. Confira o exemplo do Box 1.

```
main()
{
  short var = 10;             /* variável de 2 bytes (short int) */
  short *ptr;                 /* ponteiro para short int */

  ptr = &var;                 /* ptr aponta para o endereço de var */

  printf("%X\n", &var);       /* endereço de var */
  printf("%X\n",  ptr);       /* endereço de var */

  ptr++;                      /* incrementa ptr */

  printf("%X\n",  ptr);       /* endereço 2 bytes acima do
                                 endereço de var */

  ptr--;                      /* decrementa ptr */

  printf("%X\n",  ptr);       /* endereço de var */

  ptr--;                      /* decrementa ptr */

  printf("%X\n",  ptr);       /* endereço 2 bytes abaixo do
                                 endereço de var */

} /* end main */
```

Box 1 - Incremento e decremento de um ponteiro.

Se trocar a declaração da variável e do ponteiro para *long int*, o endereço estará 4 bytes acima após um incremento de ptr ou 4 bytes abaixo após um decremento. E assim de acordo com o tipo de dado. Portanto, dizemos que o ponteiro modificará o endereço *sizeof(tipo)*.

Como o ponteiro modifica o endereço de acordo com o tipo para o qual ele aponta, facilita muito a manipulação de vetores. O exemplo do Box 2 comprova isso.

```
main()
{
  short vec1[3] = {12,15,17};
  long  vec2[3] = {22,17,33};
  short *p1 = NULL;
  long  *p2 = NULL;
  register int i;

  p1 = vec1;
  p2 = vec2;

  for(i=0;i<3;i++)
  {
     *(p1+i) = 2*i;
     *(p2+i) = 3*i;
  }

  printf("Vec1 Vec2\n");

  for(i=0;i<3;i++)
    printf("%2d   %2d\n",vec1[i],vec2[i]);


} /* end main */
```

Box 2 - Ponteiros manipulando vetores de tipos diferentes.

Observe no código do Box 2 que os vetores têm tipos de dados diferentes. O vetor vec1 é do tipo *short int* enquanto que vec2 é do tipo *long int*. Como os ponteiros utilizados para apontar para o endereço inicial de cada vetor apresentam o mesmo tipo de dado do respectivo vetor, a saída apresentará os valores conforme esperado.

O código do Box 3 demonstra uma das formas de utilizar a comparação de ponteiros, onde poderemos verificar qual o maior endereço entre duas variáveis por exemplo.

```
main()
{
  int *p1, *p2;
  int  a1,  a2;

  a1 = 10;
  a2 = 30;

  p1 = &a1;
  p2 = &a2;

  printf("a1 addr: %p\n",&a1);
  printf("a2 addr: %p\n",&a2);

  if(p1>p2) printf("O endereco de a1 \x82 o maior\n");
  else      printf("o endereco de a2 \x82 o maior\n");
}
```

Box 3 - Informa qual variável está localizada no maior endereço.

Para demonstrar uma aplicação prática com aritmética de ponteiros, confira o programa que implementa uma pilha de dados, muito útil no desenvolvimento de sistemas.

Exercício proposto: projete uma pilha de memória em C que contenha 16 níveis, onde cada nível irá armazenar um byte. Para mostrar a pilha, os endereços e dados devem ser apresentados no formato hexadecimal.

Bibliografia: DAMAS, Luís; Linguagem C, décima edição.

Disponível em: https://amzn.to/3nGdIbN