Ponteiros em C

Disciplina: PROGRAMAÇÃO II

Prof. Jean Eduardo Glazar
Curso de Sistemas de Informação
Campus Colatina-ES

INSTITUTO FEDERAL
Espírito Santo

Definição

- Um ***apontador*** ou ***ponteiro*** é uma variável que armazena o **endereço de memória** de outra variável.

- Sua utilização é vasta em programas por dois motivos:
  1. Algumas computações só são possíveis com a utilização de ponteiros;
  2. A utilização de ponteiros em geral simplifica a programação e aumenta a eficiência do código executável, tornando-o mais rápido.

Declaração

Suponha que a variável **x** tenha sido alocada no endereço 3 (00000011) da memória e **px** na posição 5 (00000101) da memória. Assim, após a execução das linhas abaixo, teremos a seguinte situação:

Endereço	Informação	
00000000		
00000001		
00000010		
00000011	7	x
00000100		
00000101	00000011	px
. . .		

```
int x;
int * px;
x = 7;
px = &x;
```

Onde: o operador **&** fornece o endereço de memória de uma variável.

Operadores * e &

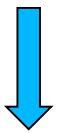
& ➔ fornece o endereço de memória de uma variável.

***** ➔ pega a informação apontada pelo ponteiro

	&	*
Endereço	Informação	
00000000		
00000001		
00000010		
00000011	6	x
00000100		
00000101	00000011	px
. . .		

px = &x;

***px = 6;**



Similar

x = 6;

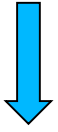
Operadores * e &

& ➔ fornece o endereço de memória de uma variável.

***** ➔ pega a informação apontada pelo ponteiro

	& Endereço	* Informação
	00000000	
y	00000001	20
	00000010	
x	00000011	20
	00000100	
px	00000101	00000011
	. . .	

```
x = 20;
px = &x;
y = *px;
```



Similar

```
x = 20;
y = x;
```

Atribuição de Ponteiros

A expressão ***py = px*** copia o conteúdo de ***px*** (não o conteúdo do endereço apontado por ***px***) para dentro de ***py*** fazendo com que ***py*** passe a apontar para o mesmo local para onde ***px*** aponta.

&	*	
Endereço	**Informação**	
00000000	00000011	py
00000001	15	y
00000010		
00000011	15	x
00000100		
00000101	00000011	px
. . .		

```
int x, y;
int * px, * py;
x = 15;
px = &x;
py = px;
y = *py;
```

Exercícios

- Analise as seguintes sequências de código. Verifique se estão corretas ou incorretas. Caso estejam incorretas, apontem os prováveis erros. Para tanto, considere a declaração das variáveis abaixo:

```
int a, b, *c, *d;
```

a)
```
c = a;
```

b)
```
c = &b;
d = c;
*d = 10;
```

c)
```
c = *a;
d = 10 + *c;
```

d)
```
d = c;
c = &a;
a = 10;
b = *d;
printf("%d", b);
```

e)
```
c = &a;
scanf("%d", &c);
d = c;
b = *d;
```

f)
```
b = 10;
c = &b;
printf("%d, c);
```

g)
```
*c = 20;
```

h)
```
b = 2;
c = &b;
*c += 2;
```

Exercícios (cont.)

```
int a, b, *c, *d;
```

i)
```
b = 1;
c = *b;
*c = a + *c;
```

j)
```
a = 1;
c = &a;
for (b=0; b<10; b++) {
   *c += 1;
}
```

k)
```
c = &b;
b = 10;
printf("%d", *c);
```

l)
```
scanf("%d", c);
```

m)
```
a = 100;
c = &a;
while ( a > 0 ) {
   *c = *c - 1;
   printf("%d\n", a);
}
```

n)
```
*c = 10;
d = c;
*d = *d +10;
printf("%d", *d);
```

Ponteiros x Vetores

Qual a diferença?

(*px)++; ***px++;**

- A resposta desta pergunta pode nos ajudar a compreender a relação entre apontadores e vetores. Mais adiante a pergunta será respondida.

Ponteiros x Vetores

- A definição **int a[10];** define um vetor de tamanho 10. Ou seja, um bloco de 10 objetos consecutivos na memória chamados respectivamente de **a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8]** e **a[9]**.
- Considere **p** um ponteiro para inteiro e **a** um vetor de 10 inteiros. O seguinte programa fará com que **p** receba o endereço da primeira posição do vetor **a**, ou seja, **p** irá apontar para a posição **0** do vetor **a**:

```
int *p;
int a[10];
p = &a[0];
```

Ponteiros x Vetores

- Pela definição da linguagem, se **p** aponta para uma posição de um vetor, **p+1** apontará para a próxima posição do vetor e **p - 1** apontará para a posição anterior do vetor. Sendo assim, o programa abaixo irá armazenar o valor 20 na posição **a[9]** do vetor.

```
int *p;
int a[10];
p = &a[0];
p = p + 9;
*p = 20;
```

Exemplo

```
int v[5], *pi;
pi = v;
*pi = 30;
*pi+1 = 50;
```

O programa acima está incorreto, pois na linha 3, a referência ***pi+1=50** não faz sentido, pois uma referência a uma variável ou posição de vetor é necessária do lado esquerdo da expressão. Uma possível correção seria:

```
*(pi+1) = 50;
```

Exemplo (cont.)

Início

Ender	Info	Var.
0000		pi
0001		v[0]
0010		v[1]
0011		v[2]
0100		v[3]
0101		v[4]
0110		
. . .		

pi = v;

Ender	Info	Var.
0000	0001	pi
0001		v[0]
0010		v[1]
0011		v[2]
0100		v[3]
0101		v[4]
0110		
. . .		

*pi = 30

Ender	Info	Var.
0000	0001	pi
0001	30	v[0]
0010		v[1]
0011		v[2]
0100		v[3]
0101		v[4]
0110		
. . .		

Exemplo (cont.)

*(pi+1) = 50

Ender	Info	Var.
0000	0001	pi
0001	30	v[0]
0010	50	v[1]
0011		v[2]
0100		v[3]
0101		v[4]
0110		
. . .		

*(pi+3) = 70;

Ender	Info	Var.
0000	0001	pi
0001	30	v[0]
0010	50	v[1]
0011		v[2]
0100	70	v[3]
0101		v[4]
0110		
. . .		

Exercícios

- Analise as seguintes sequências de código. Verifique se estão corretas ou incorretas. Caso estejam incorretas, apontem os prováveis erros. Para tanto, considere a declaração das variáveis abaixo:

```
int v[5], *pi, *pa, x, i;
```

a)
```
pi = v;
*pi++ = 200;
pi = pi + 2;
*pi = 100;
```

b)
```
v[0] = 10;
v[4] = 30;
pi = &v[4];
pa = &v[0];
*pa += &pi;
```

c)
```
*v = 10;
v++;
*v = 20;
```

d)
```
scanf("%d", v);
if ( v[0] > 10 ) {
   pi = v+1;
   *pi = 2*(*v);
} else {
   *(pi+1) = v[0];  }
```

e)
```
scanf("%d",v+1);
v++;
x = *v;
if (x <0) {
    pa = &v;
    for (i=0; i<5; i++) {
        *pa+1 = i;
    }
} else {
    pa = &v[4];
    *pa = x + 5;
}
```

Exercícios (cont.)

```
int v[5], *pi, *pa, x, i;
```

f)
```
pi = *v;
for (i=0; i<5; i++) {
   scanf("%d", &(pi+i));
}
if (*pi > *(pi+1) ) {
     x = pi;
     *pi = *(pi+1);
     *(pi+1) = x;
}
```

g)
```
v[0] = 70;
v[1] = 40;
pi = v;
pa = v+1;
if (*pi > *pa) {
   pi = pa;
   pa = v;
   printf("%d %d",*pi,*pa);
}
```

h)
```
pi = &v[4];
v = pi-1;
*(pi-1) = 10;
printf("%d", *v);
```

i)
```
pi = &x;
pa = v;
scanf("%d", &x);
*pa = *pi;
*pa+1 = *pi-1;
*pa+2 = *pi-2;
*pa+3 = *pi-3;
*pa+4 = *pi-4;
```

INSTITUTO
FEDERAL
Espírito Santo