Ponteiros

Jacson Luiz Matte
Jacson.matte@uffs.edu.br

Ponteiros

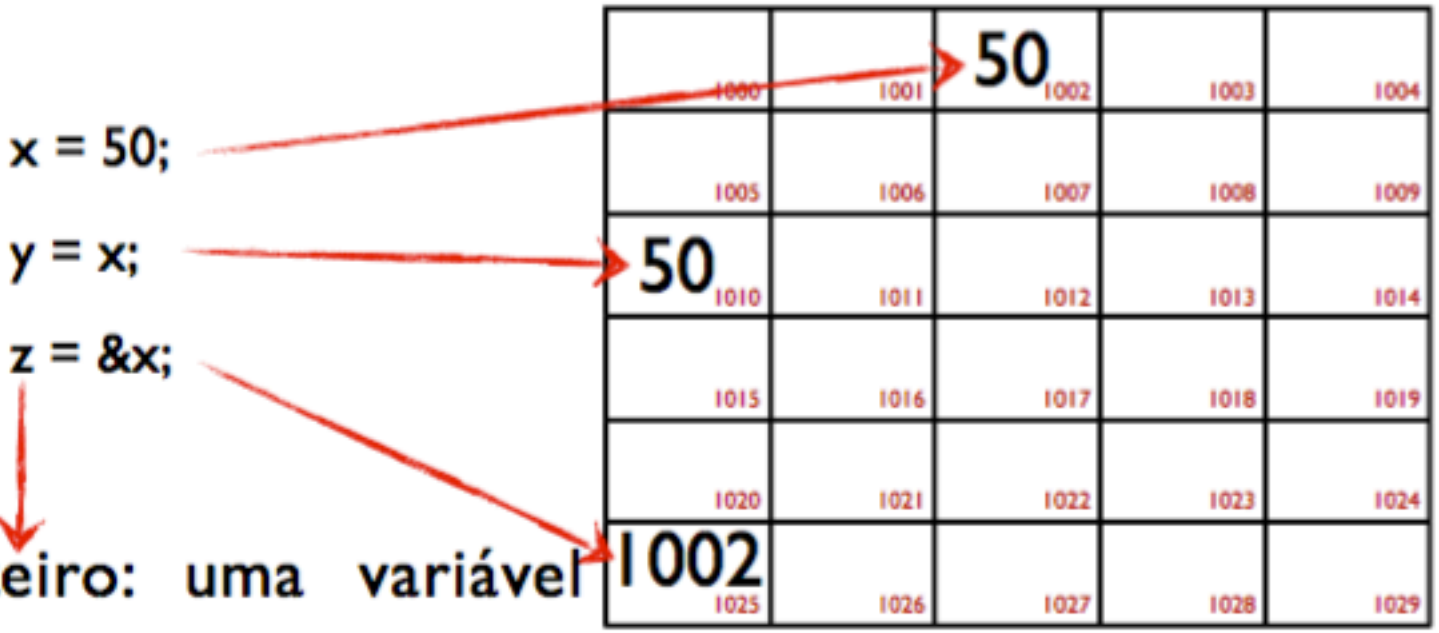
(apontadores, referências)

- Quando declaramos uma variável, uma quantidade de memória é alocada em algum lugar específico (endereço de memória);
- O programador não decide em qual endereço uma variável será armazenada;
- Por vezes, pode ser necessário conhecer o endereço de memória de uma variável;
- O endereço de memória de uma variável é uma referência para a variável;

Ponteiros

(apontadores, referências)

- O endereço de uma variável pode ser obtido usando o operador de referência, &, antes do nome da variável;

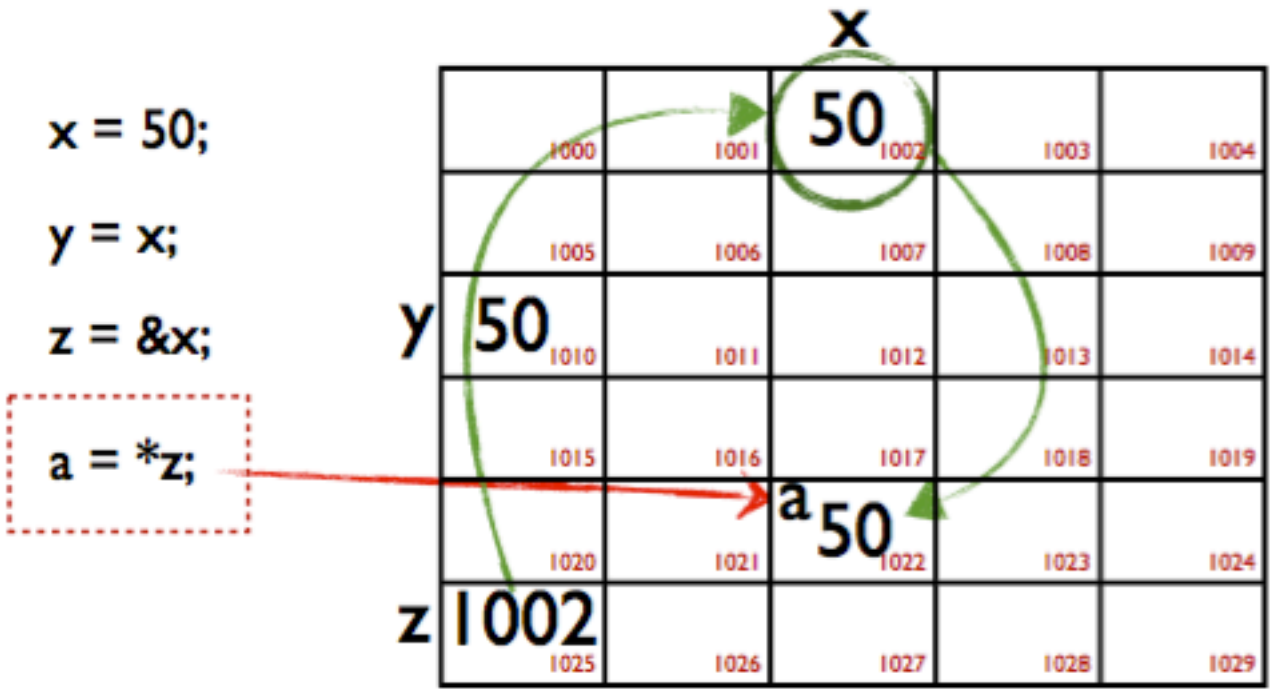


Ponteiro: uma variável que armazena uma referência para outra variável.

Ponteiros

(apontadores, referências)

- Usando um ponteiro podemos obter o valor armazenado na variável que é referenciada (apontada);
- Para tal, devemos preceder o ponteiro com o operador de dereferência, *, (valor apontado por...);



Ponteiros

(apontadores, referências)

- Qual é a diferença entre z e *z?

```
a = z;
a = *z;
```

a recebe o valor de z (1002)

a recebe o valor apontado por z (50)

Ponteiros

(declaração de variáveis do tipo ponteiro)

- A forma geral da declaração de ponteiros é:

```
tipo * nome;

int * v;
float * vmedio;
char * letra;
```

- Ponteiros com tipos diferentes usam a mesma quantidade de memória;

Ponteiros

(exemplo 1)

```
1  //ex1Ponteiros.c
2
3  #include <stdio.h>
4
5  int main()
6  {
7   int valor1, valor2;
8   int * ponteiro;
9
10  ponteiro = &valor1;
11  *ponteiro = 10;
12
13  ponteiro = &valor2;
14  *ponteiro = 20;
15
16  printf("Valor 1 é %d.\n", valor1);
17  printf("Valor 2 é %d.\n", valor2);
18
19  return 0;
20 }
```

Ponteiros

- Um ponteiro pode receber vários valores ao longo do programa;

```
1  //ex2Ponteiros.c
2
3  #include <stdio.h>
4
5  int main()
6  {
7   int valor1 = 5, valor2 = 15;
8   int * p1, * p2;
9
10  p1 = &valor1;// p1 = endereço de valor1
11  p2 = &valor2;// p2 = endereço de valor2
12  *p1 = 10;    // endereço apontado por p1 = 10
13  *p2 = *p1;   // end. apontado por p2 = valor apontado por p1
14  p1 = p2;     // p1 = p2 (valor do ponteiro é copiado)
15  *p1 = 20;    // valor apontado por p1 = 20
15
16  printf("Valor 1 é %d.\n", valor1);
17  printf("Valor 2 é %d.\n", valor2);
18
19  return 0;
20 }
```

Ponteiros

(e vetores)

- O identificador de um vetor equivale ao endereço do seu primeiro elemento;
- Um ponteiro equivale ao endereço do primeiro elemento para o qual aponta; (mesmo conceito?)

```
int vnum[10];
int * p;
p = vnum;
```

p e vnum são equivalentes. Valor de p pode ser alterado. Valor de vnum sempre apontará para o primeiro elemento do vetor (ponteiro constante).

`vnum = p;` declaração inválida!

Ponteiros

```
//ex3Ponteiros.c

#include <stdio.h>

int main()
{
  int i, vnum[5];
  int * p1;
  p1 = vnum;  *p1 = 10;
  p1++;  *p1 = 20;
  p1 = &vnum[2];  *p1 = 30;
  p1 = vnum + 3;  *p1 = 40;
  p1 = vnum;  *(p1+4) = 50;

  for (i = 0; i < 5; i++){
    printf("%d\t", vnum[i]);
  }

  return 0;
}
```

Ponteiros

(e vetores)

- Em vetores (ou matrizes) usamos colchetes, [ ], para especificar o índice de um elemento;
  - Colchetes são operadores de dereferência;
  - Eles dereferenciam a variável da mesma forma que * faz em ponteiros, mas indicam o endereço na estrutura a ser dereferenciada;

```
vnum[3] = 0;
*(vnum+3) = 0;
```

Ponteiros

- É possível inicializar um ponteiro com o valor para o qual aponta;

```
char * p1 = "uffs";
```

- Neste caso é reservado um espaço de memória para armazenar "uffs" e o endereço do primeiro elemento deste bloco de memória é atribuído ao ponteiro p1;

'u'	'f'	f'	's'	'\0'
1000	1001	1002	1003	1004

p1 | 1000

- O ponteiro p1 aponta para uma sequência de caracteres e pode ser lido como um vetor;
  - ex.: `*(p1+3);` ou `p1[3];` para obter a letra 's'.

Ponteiros void

- Ponteiros que podem ser utilizados com qualquer tipo de dados;

```
void * p;
```

```
/*
ex5PonteirosVoid.c
*/

#include <stdio.h>

int main()
{
  int v1 = 10;
  float v2 = 2.50;

  void *p; // ponteiro genérico

  p = &v1; // aponta para um inteiro
  printf("%d\n", *(int *)p);

  p = &v2; // aponta para um float
  printf("%0.2f\n", *(float *)p);

  return 0;
}
```

Ponteiros NULL

- Ponteiro de qualquer tipo que tem um valor especial que indica que não aponta para qualquer endereço válido na memória;

```
int * p;
p = NULL;
```

p é um ponteiro de valor nulo