



Computação 2

Aula 7

Ponteiros

Prof^a. Fabiany
fabiany1@utfpr.edu.br

O que são Ponteiros?

Um ponteiro é uma variável que contém um endereço de memória. Este endereço é normalmente a posição de uma outra variável na memória. Se uma variável contém o endereço de um outra, então a primeira variável é dita apontar para a segunda [Schiltd, 1997].

Endereço na memória	Variável na memória
1000	1003
1001	1
1002	1
1003	7
1004	2

The diagram illustrates a pointer variable. A table shows memory addresses and their corresponding values. The value at address 1000 is 1003, which is the address of the cell containing the value 7. An arrow points from the value 1003 to the cell at address 1003, and another arrow points from the value 7 to the cell at address 1003. A small asterisk is next to the value 1003.

Ponteiros

- Variáveis ponteiros:

Se uma variável irá conter um ponteiro, ela deve ser declarada como tal. Uma declaração de ponteiro consiste no tipo, um * e o nome da variável. A forma geral para declarar um ponteiro é:

```
<tipo> *nome_variável;
```

onde <tipo> é qualquer tipo válido em C.

O tipo base do ponteiro define que tipo de variáveis o ponteiro pode apontar.

Os operadores de Ponteiros

- Existem dois operadores especiais para ponteiros: * e &.
- O & é um operador unário que devolve o endereço na memória do seu operando (um operador unário requer apenas um operando). Por exemplo:

```
m = &count;
```

coloca em m o endereço de memória que contém a variável count. O operador & pode ser imaginado como retornando “o endereço de”. Assim, esse comando de atribuição significa “**m** recebe o endereço de **count**”.

Os operadores de Ponteiros

- O operador `*` também é um operador unário que devolve o valor da variável localizada no endereço que o segue. Por exemplo, se `m` contém o endereço da variável `count`:

`q = *m;`

coloca o valor de `count` em `q`. O operador `*` pode ser imaginado como “no endereço”. Nesse caso, o comando anterior significa “`q` recebe o valor que está no endereço `m`”

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
```

```
int main()
{
```

```
    int x, y;
    x = 1;
```

```
    int* p;
```

```
// Declaração de um ponteiro para inteiro.
// Isto significa que p poderá apontar para um inteiro.
// Um ponteiro é declarado com auxílio do *
```

```
    p = &x;
```

```
// Aqui o ponteiro recebe o endereço da variável x na memória.
// Para um ponteiro receber o endereço de uma variável é
// necessário utilizar o operador & antes da variável.
```

```
    y = *p;
```

```
// Aqui a variável y recebe o valor da variável apontada pelo
// ponteiro p. Na prática, isto significa que y terá o valor
// da variável x.
// Para uma variável receber o valor da variável apontado por
// um ponteiro, é necessário utilizar o operador *.

// Assim sendo, o operador * serve para diversas funções em C:
// - serve para declarar um ponteiro.
// - serve para informar o valor de uma variável apontada.
// - serve para multiplicar dois números (sua função primordial).
// Portanto, o uso do * depende do contexto.
```

```
// ...
```

Continuação do exemplo

```
// ...
```

```
printf ("O valor da variável x : %i. \n", x);
```

```
printf ("O endereço da variável x : %p. \n", &x);  
printf ("\n");
```

```
printf ("O valor da variável x via o ponteiro p : %i. \n", *p);
```

```
printf ("O endereço da variável x via o ponteiro p: %p. \n", p);
```

```
printf ("O endereço do ponteiro p : %p. \n", &p);  
printf ("\n");
```

```
printf ("O valor da variável y, adquirido de x, via p: %i. \n", y);
```

```
printf ("O endereço da variável y : %p. \n", &y);  
printf ("\n");
```

```
// O %p serve para expressar o valor de um endereço de memória.
```

```
system("Pause");  
return 0;
```

```
}
```

Exemplo 2

```
#include <stdlib.h>
#include <stdio.h>
```

```
int main()
{
```

```
    int x, y;
    x = 1;
```

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

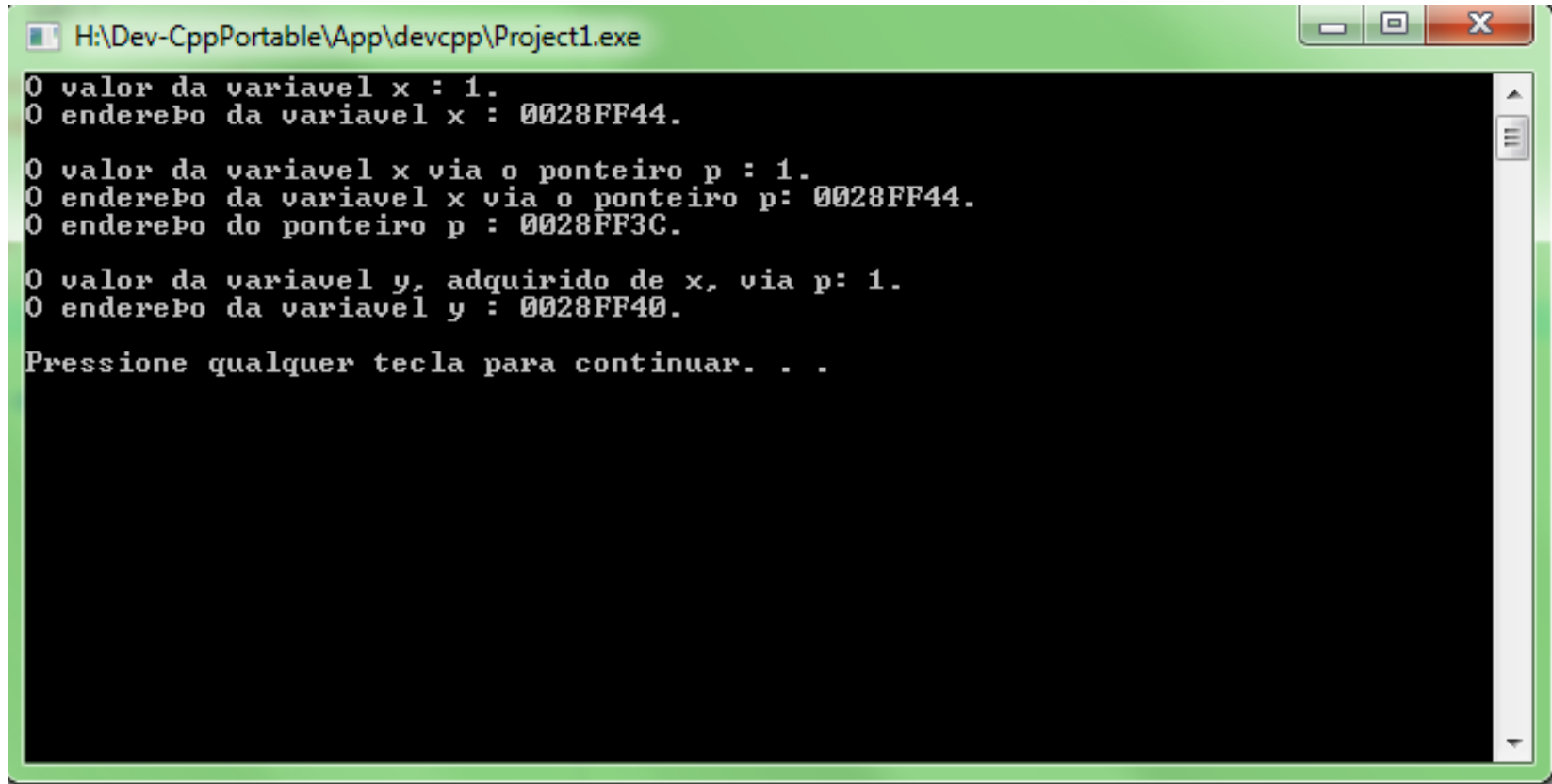
```
    printf ("O valor da variável x : %i. \n", x);
    printf ("O endereço da variável x : %p. \n", &x);
    printf ("\n");
    printf ("O valor da variável x via o ponteiro p : %i. \n", *p);
    printf ("O endereço da variável x via o ponteiro p: %p. \n", p);
    printf ("O endereço do ponteiro p : %p. \n", &p);
    printf ("\n");
    printf ("O valor da variável y, adquirido de x, via p: %i. \n", y);
    printf ("O endereço da variável y : %p. \n", &y);
    printf ("\n");
```

// O %p serve para expressar o valor de um endereço de memória.

```
    system("Pause");
    return 0;
```

```
}
```


Exemplo 2

A screenshot of a Windows command prompt window with a green title bar. The title bar text is "H:\Dev-CppPortable\App\devcpp\Project1.exe". The window contains the following text:

```
0 valor da variavel x : 1.  
0 endereco da variavel x : 0028FF44.  
  
0 valor da variavel x via o ponteiro p : 1.  
0 endereco da variavel x via o ponteiro p: 0028FF44.  
0 endereco do ponteiro p : 0028FF3C.  
  
0 valor da variavel y, adquirido de x, via p: 1.  
0 endereco da variavel y : 0028FF40.  
  
Pressione qualquer tecla para continuar. . .
```

Exemplo 3

```
#include <stdlib.h>
#include <stdio.h>
int main()
{   int x, y;   x = 1;

    int *p;
    p = &x;
    y = *p;

    printf ("O valor da variável x : %i. \n", x);
    printf ("O endereço da variável x : %p. \n \n", &x);

    printf ("O valor da variável x via o ponteiro p : %i. \n", *p);
    printf ("O endereço da variável x via o ponteiro p: %p. \n", p);
    printf ("O endereço do ponteiro p : %p. \n \n", &p);

    printf ("O valor da variável y, adquirido de x, via p: %i. \n", y);
    printf ("O endereço da variável y : %p. \n \n", &y);

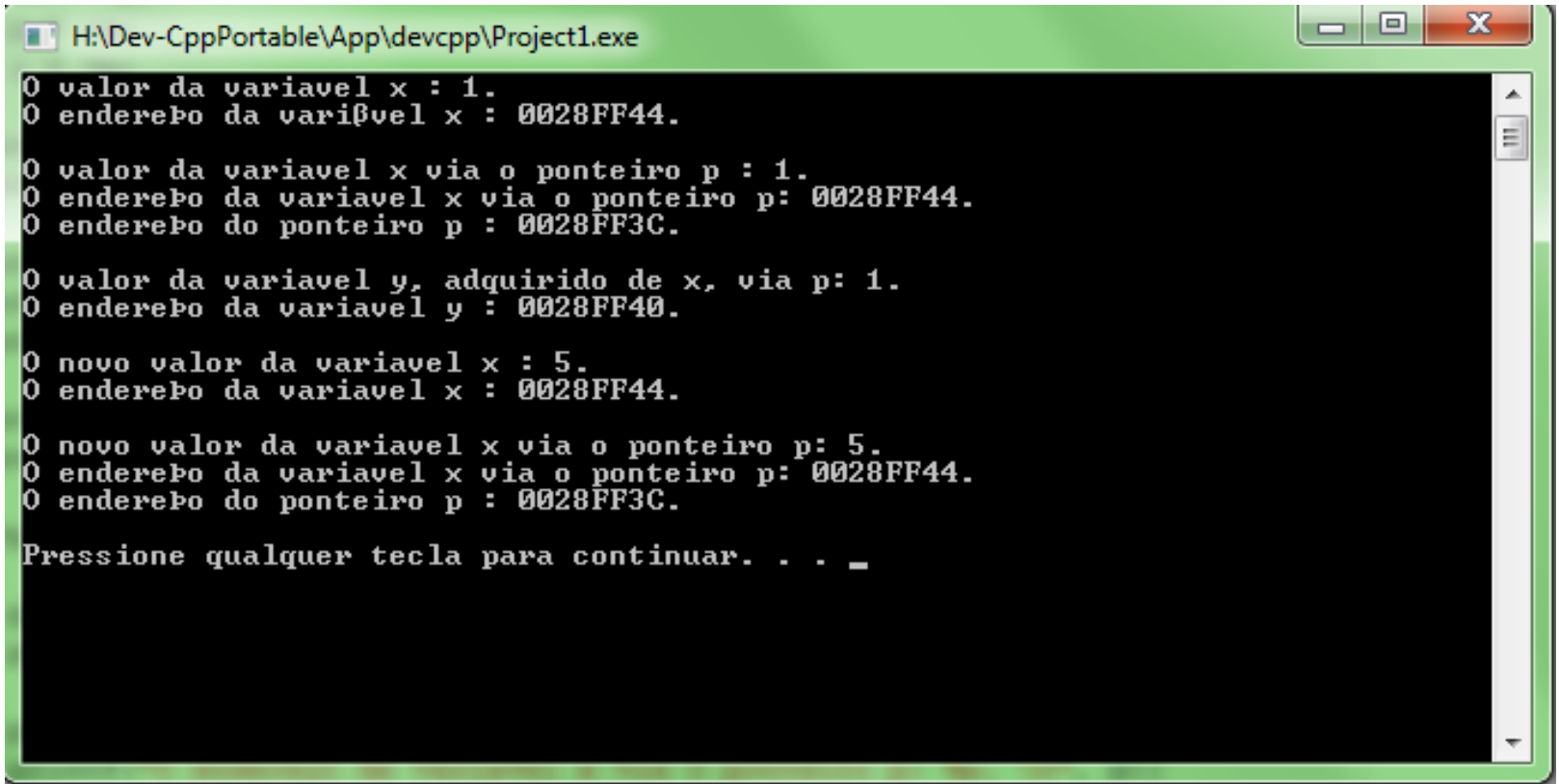
    *p = 5;           // Aqui muda-se o valor de x via *p.

    printf("O novo valor da variável x : %i. \n", x);
    printf("O endereço da variável x : %p. \n \n", &x);

    printf("O novo valor da variável x via o ponteiro p: %i. \n", *p);
    printf("O endereço da variável x via o ponteiro p: %p. \n", p);
    printf("O endereço do ponteiro p : %p. \n \n", &p);

    ...
}
```

Exemplo 3



```
H:\Dev-CppPortable\App\devcpp\Project1.exe
0 valor da variavel x : 1.
0 endereco da variavel x : 0028FF44.

0 valor da variavel x via o ponteiro p : 1.
0 endereco da variavel x via o ponteiro p: 0028FF44.
0 endereco do ponteiro p : 0028FF3C.

0 valor da variavel y, adquirido de x, via p: 1.
0 endereco da variavel y : 0028FF40.

0 novo valor da variavel x : 5.
0 endereco da variavel x : 0028FF44.

0 novo valor da variavel x via o ponteiro p: 5.
0 endereco da variavel x via o ponteiro p: 0028FF44.
0 endereco do ponteiro p : 0028FF3C.

Pressione qualquer tecla para continuar. . . _
```

Atribuição de Ponteiros.

```
#include <stdio.h>
#include <stdlib.h>

void main ()
{
    int x = 3;

    int *p1, *p2;

    p1 = &x;

    p2 = p1;

    printf (" O endereço da variável x é %p. \n", p2 );

    system ("Pause");

    return 0;
}
```

Aritmética de Ponteiro

```
void main ()  
{  
  char *CH;  
  CH = 3000;  
  printf (" O endereço apontado por ch é %p. \n", ch );  
  CH++;  
  printf (" O endereço apontado por ch é %p. \n", ch );  
  CH++;  
  printf (" O endereço apontado por ch é %p. \n", ch );  
}
```

```
void main ()  
{  
  int *INTEIRO;  
  INTEIRO = 3000;  
  printf (" O endereço apontado por INTEIRO é %p. \n", INTEIRO );  
  INTEIRO++;  
  printf (" O endereço apontado por INTEIRO é %p. \n", INTEIRO );  
  INTEIRO++;  
  printf (" O endereço apontado por INTEIRO é %p. \n", INTEIRO );  
}
```

Endereços de Memória.

CH	3000	INTEIRO
CH + 1	3001	
CH + 2	3002	INTEIRO + 1
CH + 3	3003	
CH + 4	3004	INTEIRO + 1
CH + 5	3005	

Variáveis em C

Tipo de dados	Variação	Total de Bytes Utilizados
char	0 a 255	1
int	-32.768 a 32.767	2
short int	-128 a 127	1
unsigned int	0 a 65.535	2
long int	-4.294.967.296 a 4.294.967.296	4
float	Aproximadamente 6 dígitos de precisão	4
double	Aproximadamente 10 dígitos de precisão	8
long double	Aproximadamente 10 dígitos de precisão	10
void	-	0

Vetores e Ponteiros

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ // Define e inicializa uma string.
  char str [80] = "Universidades";
  // Um ponteiro chamando Pont para caractere.
  char *Pont;
  // O ponteiro Pont recebe o endereço da primeira posição da str.
  // O nome de um string sozinho sempre é o endereço da primeira posição.
  // Obs.: Neste caso, não é necessário &.
  Pont = str;

  // Enquanto o conteúdo do ponteiro não for \0.
  while ( *Pont != '\0')
  {
    putchar (*Pont);
    // Aritmética de ponteiros
    Pont++;
  }
  printf("\n");

  Pont = str;

  // O ponteiro que aponta para o primeiro elemento de uma string pode ainda se
  // comportar como um vetor!!!!
  int idx = 0;
  while( Pont[idx] != '\0' )
  {
    putchar( Pont[idx] );
    idx++;
  }
  printf("\n");
  return 0;
}
```

```

#include <stdio.h>
#include <stdlib.h>
int main()
{ // Define e inicializa uma string.
  char str[80] = "Universidades";
  // Um ponteiro chamando Pont para caractere.
  char *Pont;
  // O ponteiro Pont recebe o endereço da primeira posição da str.
  // O nome de um string sozinho sempre é o endereço da primeira posição.
  // Obs.: Neste caso, não é necessário &.
  Pont = str;

  // Enquanto o conteúdo do ponteiro não for \0.
  while ( *Pont )
  {
    putchar (*Pont);
    // Aritmética de ponteiros
    Pont++;
  }
  printf("\n");

  Pont = str;

  // O ponteiro que aponta para o primeiro elemento de uma string pode ainda se
  // comportar como um vetor!!!!
  int idx = 0;
  while( Pont[idx] )
  {
    putchar( Pont[idx] );
    idx++;
  }
  printf("\n");
  system("Pause");
  return 0;
}

```


Referências Bibliográficas

- Baseado nos slides do Professor Jean Simão disponível em:
<http://www.pessoal.utfpr.edu.br/jeansimao/Fundamentos1/Fundamentos1.htm>
- **SHILDT**, H. C, Completo e Total, 3a edição, rev. e atual. Ed. Makron. São Paulo, c1997.