Ponteiros ou Apontadores

Algoritmos e Programação II
2014/2

Profa: Daniela Scherer dos Santos

daniela.santos37@ulbra.edu.br

www.danielascherer.com.br

Ponteiros

- Trata-se de uma variável que guarda o endereço de memória de um dado qualquer;
- Declaração:

Ponteiros

- Trata-se de uma variável que guarda o endereço de memória de um dado qualquer;
- Declaração:

Ponteiros

- Para cada tipo de variável há um ponteiro específico. Isso significa que se você vai armazenar o endereço de uma variável do tipo *int*, deve criar um ponteiro para *int*. Se for *char*, um ponteiro para *char*, etc.
- Exemplos:

```
char* ptA; /* "ptA" é uma variável ponteiro capaz de guardar o
endereço de memória de variáveis do tipo char*/


int* ptB; /* "ptB" é uma variável ponteiro capaz de guardar um
endereço de memória de variáveis do tipo int */


double* ptNota; /* "ptNota" é uma variável ponteiro capaz de
armazenar o endereço de memória de variáveis do tipo double */
```

Ponteiros

- Declarando ponteiros:

Tipo	Variável comum	Ponteiro para o tipo
char	char letra;	char* ptLetra;
int	int numero;	int* ptNumero;
float	float num;	float* ptNum;

Ponteiros

- Operador de endereço &:
  - o operador “&” quando aplicado sobre uma variável retorna o seu endereço;
  - Quando usamos “&” precedendo uma variável, estamos nos referindo ao endereço desta variável;

```
int* ptI;
int i;
ptI = &i;
```

&i indica o endereço da variável “i”

Utilizando Ponteiros

- Atribuindo valores aos ponteiros:
  - Utiliza-se o sinal de atribuição já conhecido "="
- Como ponteiros são utilizados para armazenar o endereço de outras variáveis, então devemos copiar pra ele um endereço de memória de alguma variável existente em nosso programa:

```
int num = 2;

int* ptNum = &num;
```

OBS: A princípio, você não pode atribuir endereços de variáveis que não sejam do mesmo tipo do ponteiro, exemplo:

```
int a = 20;
char *ptA = &a; // Errado, pois "a" não é do tipo char!
```

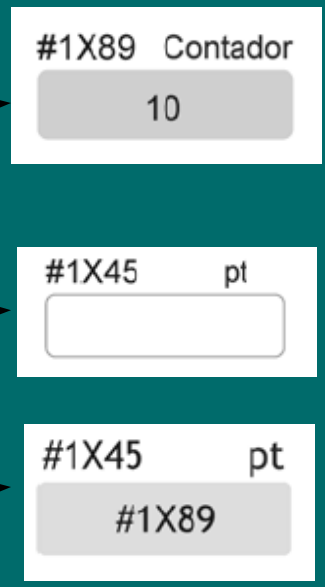
Utilizando Ponteiros

- Atribuindo valores aos ponteiros:

```
int Contador = 10;

int* pt;

pt = &Contador;
```



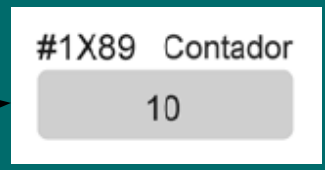
Utilizando Ponteiros

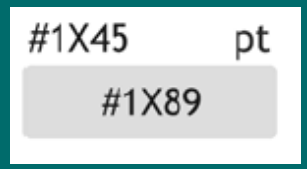
- Atribuindo valores aos ponteiros:

```
int Contador = 10;

int* pt;

pt = &Contador;
```






```
*pt = 20
```

Qual será a consequência desta atribuição?

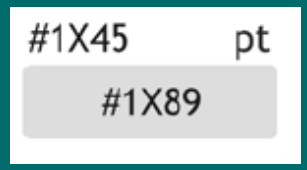
Utilizando Ponteiros

- Atribuindo valores aos ponteiros:

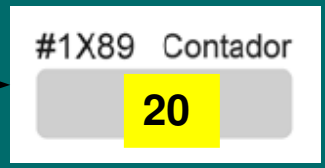
```
int Contador = 10;

int* pt;

pt = &Contador;
```




```
*pt = 20
```



"o conteúdo do endereço apontado por pt (no caso o conteúdo do endereço #1X89) recebe o valor inteiro 20"

Utilizando Ponteiros

- Se você imprimir uma variável ponteiro, ela vai mostrar o endereço de memória que está armazenando:

```
static void Main(string[] args)
{
    unsafe
    {
        int i, j;
        int* ptI;
        ptI = &i;
        *ptI = 6;
        j = i;
        Console.WriteLine("i = " + i);
        Console.WriteLine("j = " + j);
        Console.WriteLine(" *ptI = " + *ptI);
        Console.WriteLine("conteúdo de ptI = {0}", (int) ptI);
        Console.WriteLine("endereço de i = {0}", (int)&i);
    }
}
```

Por padrão C# não possui suporte ao uso de ponteiros. Portanto, é necessário usar a palavra chave *unsafe* para definir um contexto sem proteção no qual ponteiros podem ser utilizados.

Utilizando Ponteiros

- Se você imprimir uma variável ponteiro, ela vai mostrar o endereço de memória que está armazenando:

```
static void Main(string[] args)
{
    unsafe
    {
        int i, j;
        int* ptl;
        ptl = &i;
        *ptl = 6;
        j = i;
        Console.WriteLine("i = " + i);
        Console.WriteLine("j = " + j);
        Console.WriteLine(" *ptl = " + *ptl);
        Console.WriteLine("conteúdo de ptl = {0}", (int) ptl);
        Console.WriteLine("endereço de i = {0}", (int)&i);
    }
}
```

Declara um ponteiro para uma variável do tipo "int"

Atribui o endereço da variável "i" para o ponteiro "ptl"

Atribui o valor "6" para a variável "i"

Imprime o conteúdo da Variável "ptl" que é o endereço da variável "i"

Utilizando Ponteiros

- Ponteiros também têm endereço. Logo, também podemos imprimir o seu endereço, armazená-lo ou utilizá-lo em um outro ponteiro:

```
static void Main(string[] args)
{
    unsafe
    {
        int x;
        int* ptX;
        int** pptX;
        x = 2;
        ptX = &x;
        pptX = &ptX;
        Console.WriteLine("O endereço de ptX é: " + (int)&ptX);
        Console.WriteLine("O conteúdo de pptX é: " + (int)pptX);
    }
}
```

Imprime o endereço do ponteiro "ptX"

Imprime o conteúdo da variável "pptX" que é o endereço do ponteiro "ptX"