

PC1 - Aula 24 - 22/05/2014

Ponteiros

Hebert Coelho

Instituto de Informática
Universidade Federal de Goiás

Roteiro

- Ponteiros

Roteiro

- Ponteiros
- Exemplos

Ponteiros

Definição:

Um *ponteiro* é uma variável que contém um endereço de memória.

Ponteiros

Definição:

Um *ponteiro* é uma variável que contém um endereço de memória.

- Esse endereço é normalmente a posição de uma outra variável na memória;

Ponteiros

Definição:

Um *ponteiro* é uma variável que contém um endereço de memória.

- Esse endereço é normalmente a posição de uma outra variável na memória;
- Se uma variável contém o endereço de uma outra, então dizemos que a primeira variável *aponta* para a segunda;

Ponteiros

Definição:

Um *ponteiro* é uma variável que contém um endereço de memória.

- Esse endereço é normalmente a posição de uma outra variável na memória;
- Se uma variável contém o endereço de uma outra, então dizemos que a primeira variável *aponta* para a segunda;
- *PONTEIRO* \longrightarrow *VARIÁVEL*;

Ponteiros

Declaração:

Uma declaração de ponteiro consiste no tipo de base, um * e o nome da variável.

Ponteiros

Declaração:

Uma declaração de ponteiro consiste no tipo de base, um * e o nome da variável.

Sintaxe:

```
tipo *nome;
```

Ponteiros

Declaração:

Uma declaração de ponteiro consiste no tipo de base, um * e o nome da variável.

Sintaxe:

```
tipo *nome;
```

Exemplos:

```
int *valor;  
double *numero;  
char *letra;
```

Ponteiros

Operadores: * e &.

& é um operador unário que devolve o endereço na memória.

& pode ser lido como “o endereço de”;

O * é um operador unário que devolve o valor da variável localizada no endereço que segue.

Ponteiros

Operadores: * e &.

& é um operador unário que devolve o endereço na memória.

& pode ser lido como “o endereço de”;

O * é um operador unário que devolve o valor da variável localizada no endereço que segue.

Exemplo:

```
int a; int *p; p = &a;
```

Ponteiros

Operadores: * e &.

& é um operador unário que devolve o endereço na memória.

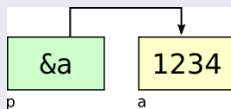
& pode ser lido como “o endereço de”;

O * é um operador unário que devolve o valor da variável localizada no endereço que segue.

Exemplo:

```
int a; int *p; p = &a;
```

O que ocorre na memória:



Aritmética de Ponteiros

- igualar dois ponteiros. Ex: **p1=p2**, onde **p1** e **p2** são variáveis ponteiro. Neste caso, **p1** aponta para o mesmo lugar que **p2**.

Aritmética de Ponteiros

- igualar dois ponteiros. Ex: **p1=p2**, onde **p1** e **p2** são variáveis ponteiro. Neste caso, **p1** aponta para o mesmo lugar que **p2**.
- incremento de ponteiros. Ex: **p1++**. **p1** passa a apontar para o próximo valor do mesmo tipo para o qual o ponteiro aponta.

Aritmética de Ponteiros

- igualar dois ponteiros. Ex: **p1=p2**, onde **p1** e **p2** são variáveis ponteiro. Neste caso, **p1** aponta para o mesmo lugar que **p2**.
- incremento de ponteiros. Ex: **p1++**. **p1** passa a apontar para o próximo valor do mesmo tipo para o qual o ponteiro aponta.
- o decremento funciona de forma semelhante.

Aritmética de Ponteiros

- igualar dois ponteiros. Ex: **p1=p2**, onde **p1** e **p2** são variáveis ponteiro. Neste caso, **p1** aponta para o mesmo lugar que **p2**.
- incremento de ponteiros. Ex: **p1++**. **p1** passa a apontar para o próximo valor do mesmo tipo para o qual o ponteiro aponta.
- o decremento funciona de forma semelhante.

incremento do valor da variável apontada por p

```
(*p)++;
```

Treinando

Qual o valor de y ao final do algoritmo?

```
1 int main(){
2     int y, *p, x;
3     y = 0;
4     p = &y;
5     x = *p;
6     x = 4;
7     (*p)++;
8     x--;
9     (*p) = (*p) + x;
10    printf ("y = %d \n", y);
11    return 0;
12 }
```

Vetores e Matrizes

Considere a declaração:

```
tipo_da_variável nome_da_variável [tam1][tam2] ...  
[tamN];
```

- o compilador C calcula o tamanho, em bytes, necessário para armazenar esta matriz em um espaço livre de memória;

Vetores e Matrizes

Considere a declaração:

```
tipo_da_variável nome_da_variável [tam1][tam2] ...  
[tamN];
```

- o compilador C calcula o tamanho, em bytes, necessário para armazenar esta matriz em um espaço livre de memória;
- O nome da variável que você declarou é na verdade um ponteiro para o tipo da variável da matriz e aponta para o primeiro elemento da matriz.

Vetores e Matrizes

Qual o significado de:

```
nome_da_variavel[índice]
```

Vetores e Matrizes

Qual o significado de:

`nome_da_variavel[índice]`

Significado:

$\ast(\textit{nome_da_variavel} + \textit{indice})$

Vetores e Matrizes

Qual o significado de:

`nome_da_variavel[índice]`

Significado:

$*(nome_da_variavel + indice)$

Conclusão:

`*nome_da_variável` é equivalente a `nome_da_variável[0]`

Vetores e Matrizes

Como ficaria este código com ponteiros?

```
1 int main (){  
2     int vet[10]={1,2,3,4,5,6,7,8,9,10};  
3     int j;  
4     for(j=0;j<10;j++){  
5         printf("%d\n",vet[j]);  
6     }  
7     return 0;  
8 }
```


Vetores e Matrizes

Varrendo vetores através de ponteiros!

```
1  int main (){  
2  int vet[10]={1,2,3,4,5,6,7,8,9,10};  
3  int j;  
4  for(j=0;j<10;j++){  
5      printf("%d\\n",*(vet+j));  
6  }  
7  return 0;  
8 }
```

Vetores e Matrizes

Tarefa para casa: Faça o mesmo com matrizes!!!!