Conceito de ponteiros

Murilo Dantas

Por que usar ponteiros?

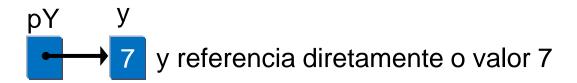
- Dá muito poder à LP (C e C++)
- Aumenta a dinamicidade de estruturas
 - Podem crescer ou diminuir com o tempo.
 - ► Ex.: vetores, matrizes, árvores, listas, pilhas, filas, etc.
- Permite a passagem por referência
 - Alterar dados indiretamente.

Declaração e atribuição

- Uma variável guarda um valor
 - Acessa o valor <u>diretamente</u>.
- Um ponteiro guarda um endereço
 - Acessa o valor indiretamente.
 - ▶ Referenciar um valor por um ponteiro é frequentemente chamado de indireção.

Declaração e atribuição

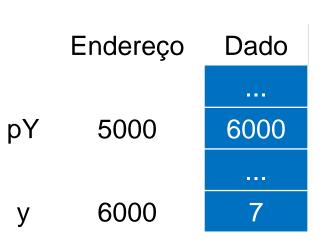
int y; //declarando uma variável qualquer int *pY; //declarando um ponteiro para um valor inteiro pY = &y; //atribui o endereço de y a pY



pY referencia indiretamente o valor 7

Representação na memória

- Caso hipotético
 - Variável y do tipo int no endereço 6000 da memória, armazenando 7.
 - Variável pY do tipo int no endereço 5000 da memória, armazenando o endereço de y.



Operador *

- É o operador de referenciação
 - Retorna um link (alias) ao objeto para o qual seu operando está apontando.
- Exemplo comparativo

```
printf("%d",y); //imprime o valor de y
printf("%d",*pY); //imprime o valor de y
*pY = 9; //é o mesmo que y=9
printf("%d",*pY * *pY); //imprime o valor de y²
```

Erros comuns de programação

- Não identificar um ponteiro rapidamente
 - Inclua sempre "p" ou "ptr" nos nomes das variáveis ponteiro.
- Esquecer do *
 - ▶ Cada ponteiro tem o seu * junto ao nome na declaração e referenciação.

Erros comuns de programação

- Não inicializar ponteiros
 - Sempre inicialize ponteiros para evitar acessos indevidos à memória.
 - Desreferenciar ponteiros não inicializados pode alterar dados importantes e causar erros em tempo de execução.

Erros comuns de programação

- Tentar desreferenciar uma variável comum
 - Se não for um ponteiro, haverá um erro de compilação.
- Desreferenciar ponteiro nulo
 - Normalmente, é um erro fatal em tempo de execução.

Perguntas?