Universidade Federal Rural de Pernambuco
(UFRPE)
- Unidade Acadêmica de Garanhuns-

Ponteiros (Apontadores)

Introdução a Linguagem C Parte V

Prof. Priscilla Kelly Machado Vieira

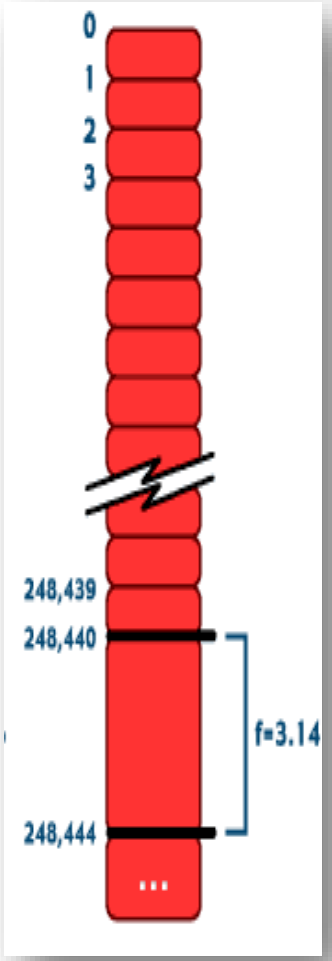
2019.1 – Aula 6

Apresentação do Capítulo

- Introdução
- Definição de Ponteiros
- Tipos de Ponteiros
- Operadores de Ponteiros
- Expressões com Ponteiros
- Vantagens e Desvantagens
- Exercícios

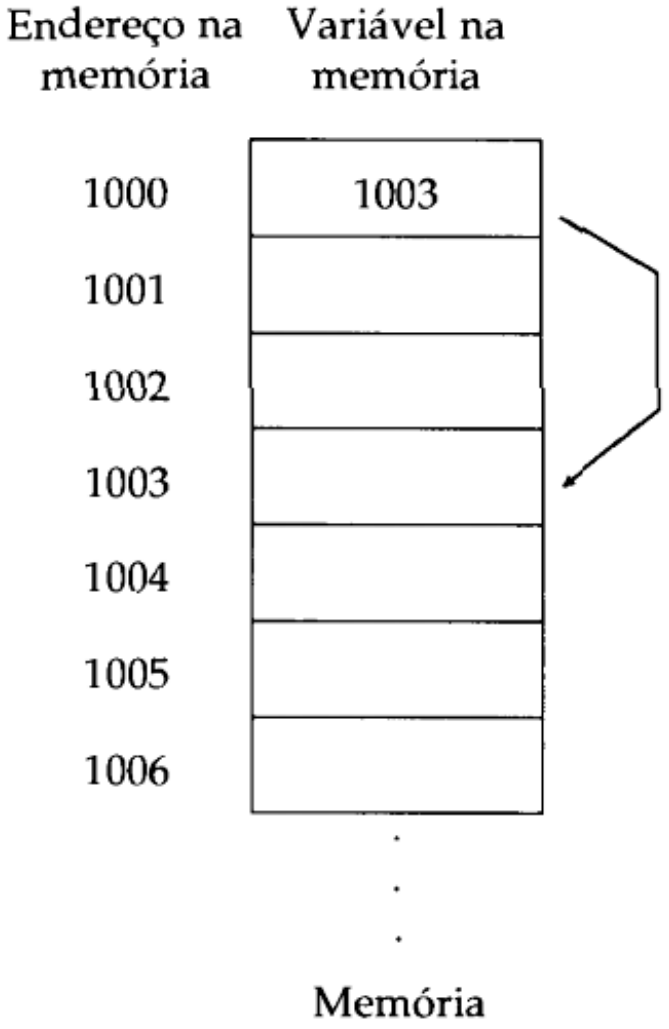
Introdução

- Quando usamos o nome de uma variável em um programa, o compilador compila no código o endereço, para que, quando executado, o processador acesse o conteúdo;
- Isso significa que o compilador só vê o endereço e o programador só vê o conteúdo;
- Nesse âmbito é importante compreender como funciona o endereçamento de memória em um computador



Ponteiros

- É uma **variável** que contém um endereço de memória;
- O conteúdo de uma variável ponteiro é o endereço de memória para o qual está apontando;
- Um ponteiro pode referenciar e "des-referenciar";

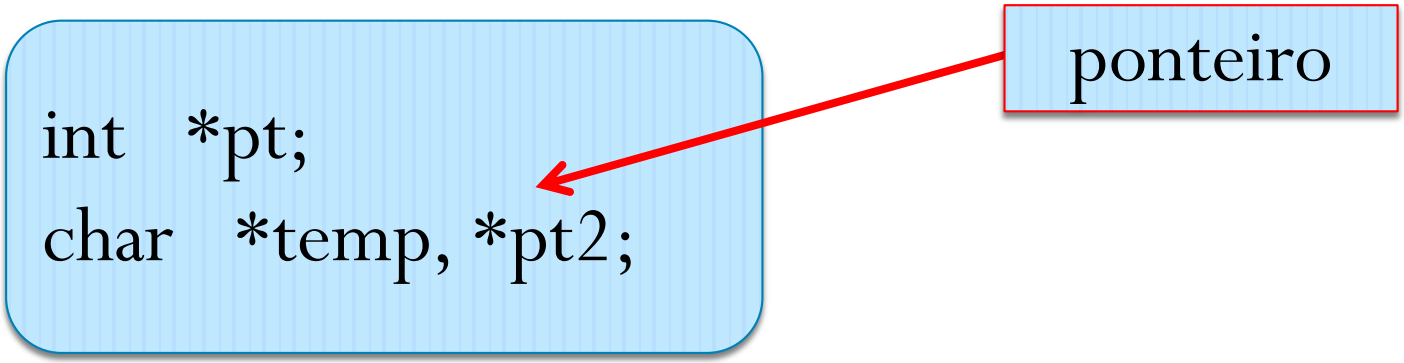


Ponteiros

- Um ponteiro pode apontar para:
  - Uma área com informação
    - Uma variável ou conteúdo de uma variável;
  - Uma rotina (procedimento ou função);
  - Endereço nulo.

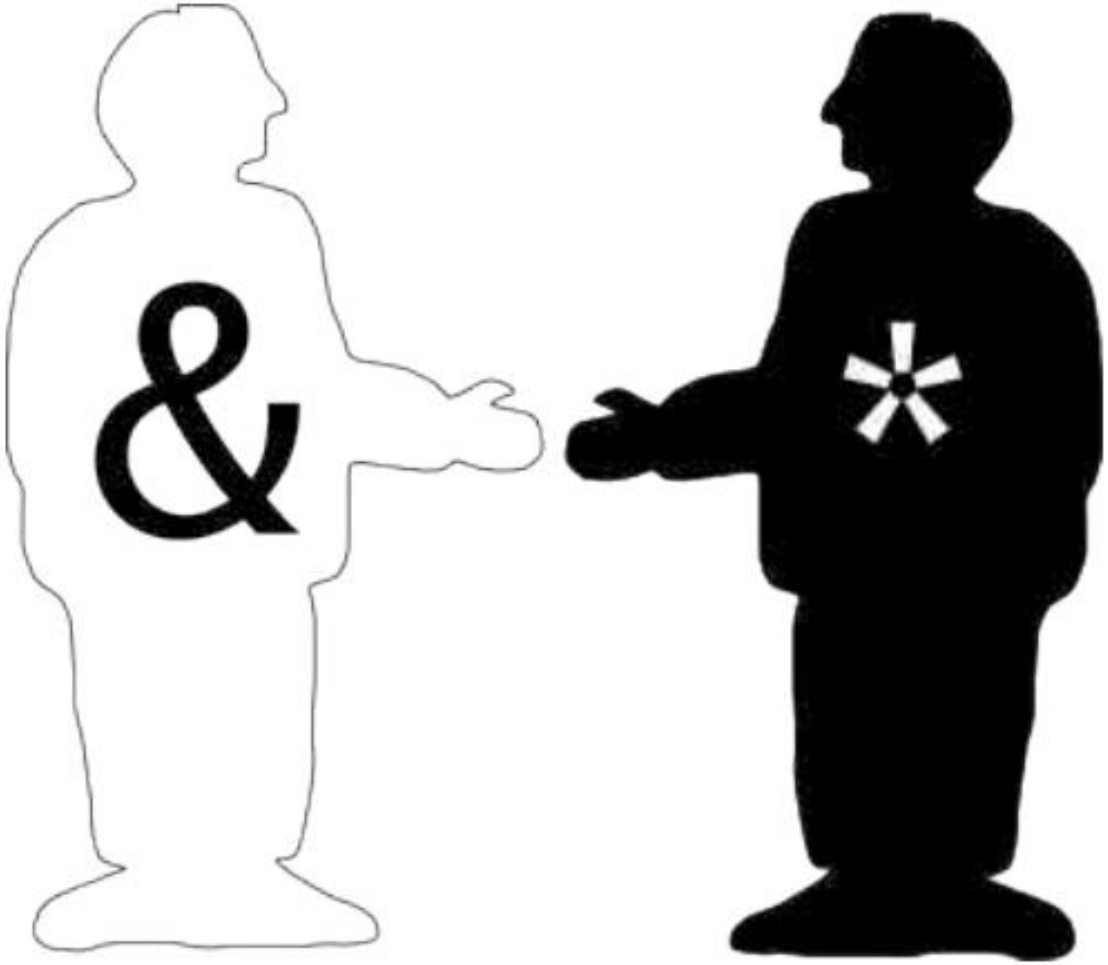
Ponteiros

*tipo_do_ponteiro *nome_da_variável*



- **Tipo**: O tipo de variável que o ponteiro deve apontar

Ponteiros



Ponteiros

- Operadores
  - **&** - Retorna o endereço de memória de seu operando
  - * - Retorna o valor da variável localizada no endereço de seu operando

LINGUAGEM C

```
int count=10;
int *pt;
pt = &count;
```

```
int count=10;
int *pt;
pt = &count;
*pt = 12;
```

Ponteiros

- Operadores

LINGUAGEM C

```
int a;
int *p; //armazenam lixo da memória
```

int a=5;

…	…	…
p		108
a	5	104

p = &a;

…	…	…
p	104	108
a	5	104

***p** = 6;

…	…	…
p	104	108
a	6	104

Ponteiros

- Cuidado !
  - Com o tipo do ponteiro

Sempre inicialize o ponteiro!

```
void main(void)
{
    float x, y;
    int *p;
    p = &x;
    y = *p;
}
```

Ponteiros

- Expressões com ponteiros
  - Atribuições

```
void main(void)
{
    int x;
    int *p1, *p2;
    p1 = &x;
    p2 = p1;
}
```

Exemplo


```c
#include <stdio.h>

void main ()
{
int *piValor; /* ponteiro para inteiro */
int iVariavel = 2712;
piValor = &iVariavel; /* pegando o endereço de memória da variável */

printf ("Endereco: %d\n", piValor);
printf ("Valor : %d\n", *piValor);

*piValor = 2713;
printf ("Valor alterado: %d\n", iVariavel);
printf ("Endereco : %d\n", piValor);
system ("PAUSE");

}
```

Ponteiros

- Vantagens:
  - Aumento de capacidade
  - Necessários em alguns cenários
- Desvantagens:
  - Erros são difíceis de serem encontrados
    - Ponteiros não inicializados
    - Apontadores para locais inadequados

Passando Ponteiros para Funções


```
void somaProduto(int a, int b, int *p, int *q)
{
    *p = a+b;
   *q = a*b;
}
int main (void){
int s, p;
somaProduto(3,5,&s, &p);
return 0;
}
```

Struct


```
struct Livro
{
int codigo;
 int paginas;
};

void exibir_dados(struct Livro livro)
{
 printf("Codigo: %d\nPaginas: %d\n", livro.codigo, livro.paginas);
}
      int main()
      {
      Livro Livro a;
      a.codigo = 342;
       a.paginas = 230;
      exibir_dados(a);
       system("PAUSE");
      return 0;
      }
```

Struct


```
struct Livro
{
int codigo;
 int paginas;
};

void exibir_dados(struct Livro *livro)
{
 printf("Codigo: %d\nPaginas: %d\n", livro->codigo, livro->paginas);
}
      int main()
      {
      Livro Livro a;
      a.codigo = 342;
       a.paginas = 230;
      exibir_dados(&a);
       system("PAUSE");
      return 0;
      }
```

Struct


```
typedef struct estrutura1 {
  int var1;
  float var2;
 } MinhaEstrutura;

  struct estrutura2 {
  int var3;
  MinhaEstrutura var4;

  };
```

Laboratório 05

- Crie as variáveis inteiras x, y e os ponteiros para inteiro p e q. Coloque o valor 2 em x, o valor 8 em y, o endereço de x em p, e o endereço de y em q. Em seguida imprima as seguintes informações:
  - O endereço de x e o valor de x
  - O valor de p e o valor de *p
  - O endereço de y e o valor de y
  - O valor de q e o valor de *q.
  - O endereço de p
  - O endereço de q

Exercício

- Elabore um programa que armazene valores aleatórios em um vetor de inteiros de 5 posições e depois, em outro vetor de ponteiros de inteiros de tamanho 5, coloque os endereços dos valores do vetor de inteiros. Ordene o vetor de inteiros de forma crescente, ficando a primeira posição do vetor de inteiros com o endereço do menor valor até a última posição que conterá o endereço do maior valor.