Profa. Michele Fúlvia Angelo

PGCA 028 – Tópicos Especiais em Tecnologia Computacional I – Introdução à Programação de Computadores
Aula 8

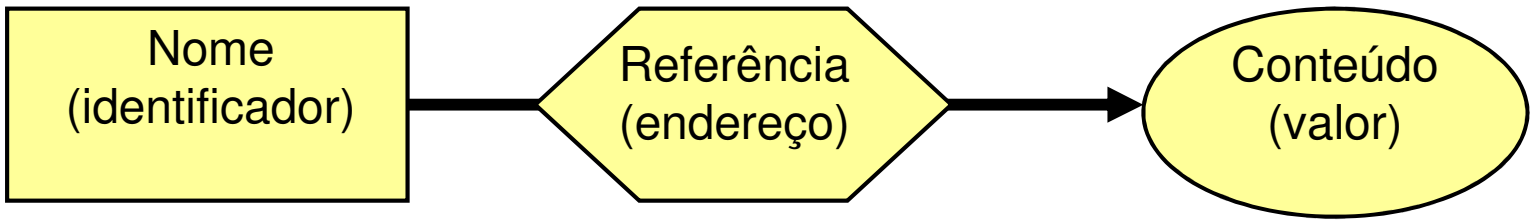
Universidade Estadual de Feira de Santana

Sumário

- Ponteiros
- Exercícios

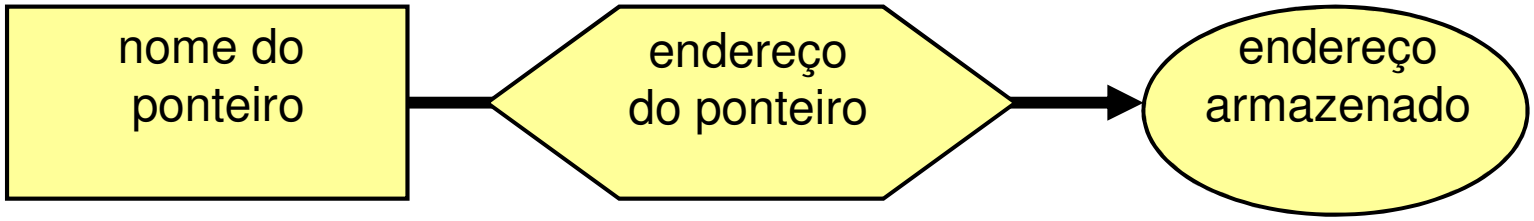
Variáveis

- Estrutura para armazenamento de dados
  - Tipos de dados do conteúdo é especificado
- Possui um nome para identificá-la (identificador)
- Representam uma região da memória
  - Abstração de locais de memória, referenciados por endereços

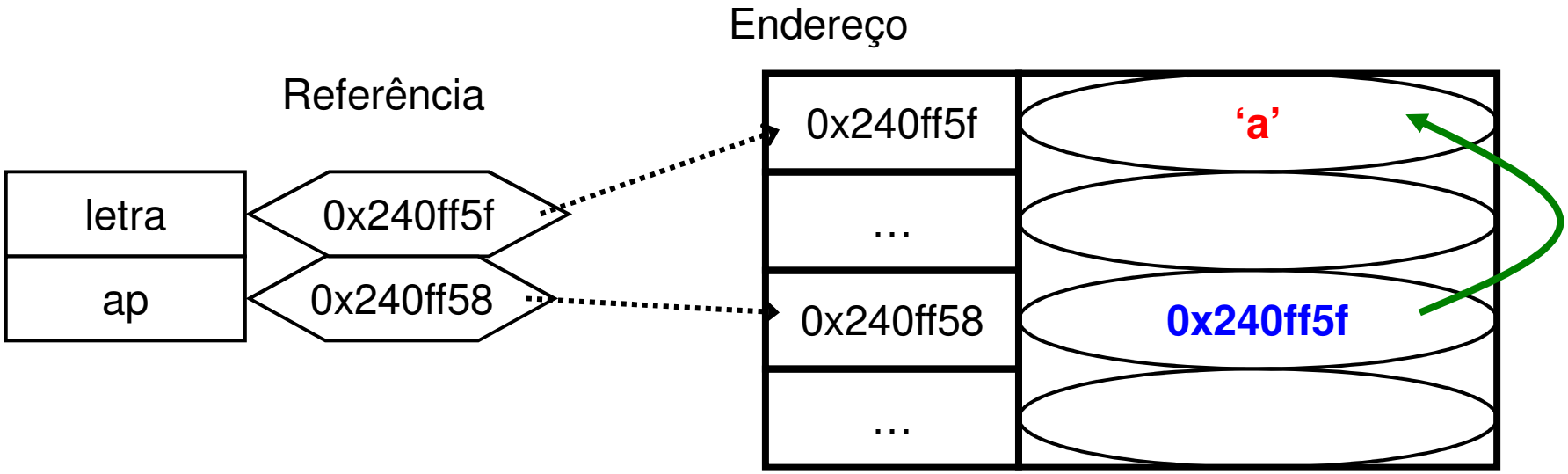


O que são Ponteiros?

- Tipo especial de variável
- Armazena um endereço de memória
- Ele aponta para uma posição na memória
- Utilizados para construir estruturas de dados não alocadas em um bloco contínuo de memória (ex. listas encadeadas)
- Endereço do ponteiro *vs.* Conteúdo do ponteiro



Ponteiros



Como *ap* possui o endereço da variável *letra*,
dizemos que *ap* **aponta** para o conteúdo da variável *letra*,
sendo possível ler e alterar o conteúdo de *letra* via o apontador *ap*.

Ponteiros

- Os ponteiros armazenam o endereço de uma informação e não a própria informação em si.
- Ponteiros acessam indiretamente um dado na memória de outra variável, em dois passos:
  - Primeiro, consulta-se a memória na posição da variável ponteiro.
  - Neste endereço, o valor armazenado é um endereço que é usado para dar a posição de uma segunda consulta na memória, que coincide com a posição de uma variável.

Por que usar Ponteiros?

- Para construir estruturas de dados não alocadas em um bloco contínuo de memória (**ex. listas encadeadas**).

- O espaço de memória utilizado por um programa pode ser reservado (e liberado) dinamicamente, via-ponteiros, diferentemente do que temos feito até o momento.

Declaração de Ponteiros

- **Declaração:**
  tipo *indentificador;

Exemplos:

```
int *apnum;
char *apc;
```

- Utilização
  - Operador referência **&** (endereço de)
  - Operador de referência ***** (conteúdo referenciado por)

```
int *ptr  =>  declaração do ponteiro ptr (apontando para null);
ptr=&k  =>  atribuindo valor a ptr (o endereço de k);
ptr       =>  endereço da variável k;
*ptr      =>  conteúdo da variável k;
&ptr      =>  endereço de ptr;
```

Exemplos de Ponteiros

```
#include <stdio.h>
int main () {

char letra = 'a';
/* Declaração de ponteiro para uma variável do tipo caracter */
char *p;

/* Atribuição de um endereço */
  p = &letra;

/* Exibe o conteúdo da variável letra */
  printf (“%c\t”, *p);

/* Altera o conteúdo da variável letra */
  *p = ‘b’;

  printf (“%c\n”, letra);
```

Exemplos de Ponteiros

```
#include <stdio.h>
int main()
{

    int i = 10, *pi;
    float f = 1.0, *pf;

    pi = &i;

    printf ("i = %d\n", *pi);

    pf = &f;

    printf ("f = %f\n", *pf);

    *pi = *pi + 1;
    *pf = *pf * 10;

     printf ("i = %d\n", *pi);
     printf ("f = %f\n",  *pf);
}
```

Saída do Programa:

```
i = 10
f = 1.0
i = 11
f = 10.0
```

Inicialização de Ponteiros

- **Inicialização**
  Se quisermos indicar que um ponteiro não aponta para uma variável, podemos atribuir a ele um "valor nulo":

```
p = NULL;
```

- Essa informação pode ser útil em expressões condicionais:

```
if (p == NULL) {
        …
}
```

Ponteiros

- **Vetor de Ponteiros**

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

int main()
{
      int vet[] = {4, 5, 6};
      int j, *ptr;
      ptr=vet;
      for(j=0;j<3;j++)
        printf("%d ", (*ptr++)+1);
      getch();
}
```

- **Qual é a saída ?**