Universidade de São Paulo – São Carlos
Instituto de Ciências Matemáticas e de Computação

Introdução a Ponteiros e Vetores em C

Material preparado pela profa

Silvana Maria Affonso de Lara e utilizado por outros professores (Rosana Vaccari)

2º semestre de 2010

Roteiro da Aula

- Definição de ponteiros
- Como utilizar ponteiros
- Exemplos de ponteiros
- Definição de Arrays
- Como referenciar arrays
- Como referenciar elementos
- Operações válidas sobre ponteiros

Ponteiros

- um ponteiro é uma variável que contém um endereço
- declaração: "*" indica que a variável é um ponteiro

  tipo_dado *nome_ponteiro;
- Ex:

```
int  x;
int *px;   /* compilador sabe que px é ponteiro */
           /* px é um ponteiro para inteiro */
```

Ponteiros

- o operador “&” quando aplicado sobre uma variável retorna o seu endereço
- Ex:

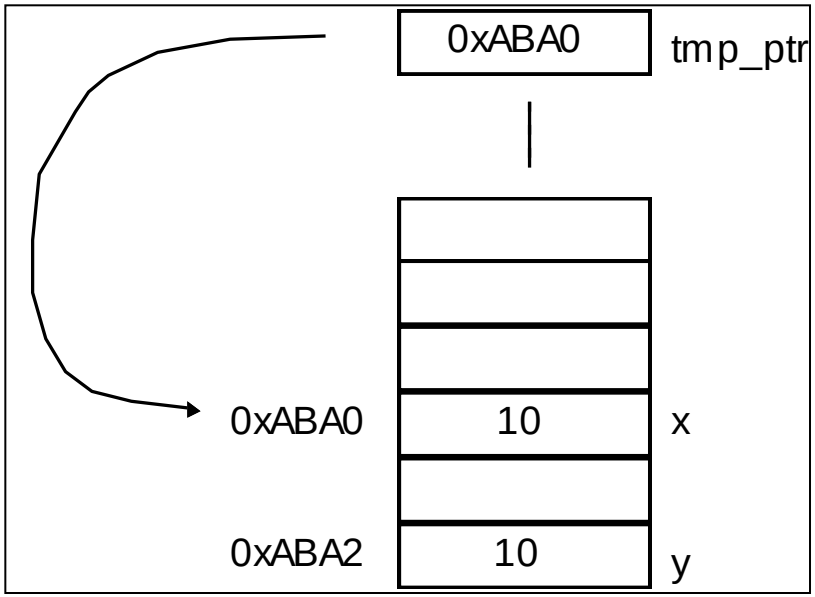
```
int x = 10, *pi;
pi = &x;
printf(“&x: %p pi: %p”, &x, pi);
```

=> &x: 0x03062fd8 pi: 0x03062fd8

Ponteiros

- o operador “*” quando aplicado sobre um ponteiro retorna o dado apontado
- Ex:

```
void main () {
int *tmp_ptr;
int x, y;
  x = 10;
  tmp_ptr = &x;
  y = *tmp_ptr;
   /* (*tmp_ptr) = 10 */
}
```



Ponteiros

- ponteiros são variáveis tipadas:

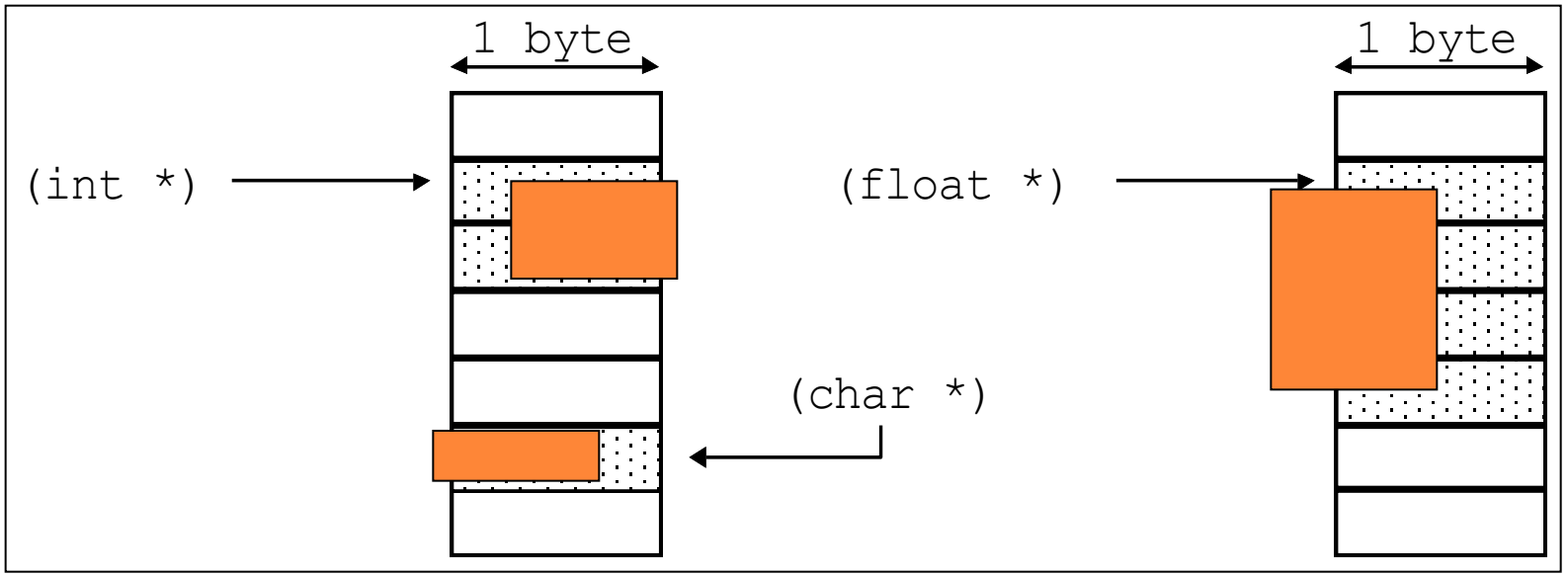
  (int *) ≠ (float *) ≠ (char *)

- Ex:

```
main() {
int *ip, x;
float *fp, z;
  ip = &x; /* OK */
  fp = &z; /* OK */
  ip = &z; /* erro */
  fp = &x; /* erro */
}
```

Ponteiros

- espaço ocupado pelas variáveis



Utilizando Ponteiros

```
void main() {
int x = 10;
int *pi;

   pi = &x;        /* *pi == 10 */
   (*pi)++;        /* *pi == 11 */
   printf("%d", x);
}
```

==> 11

ao alterar *pi estamos alterando o conteúdo de x

Utilizando Ponteiros

```
void main() {
int x = 10;
int *pi, *pj;

   pi = &x;       /* *pi == 10 */
   pj = pi;       /* *pj == 10 */
   (*pi)++;       /* (*pi, *pj, x) == 11 */
   (*pj)++;       /* (*pi, *pj, x) == 12 */
   printf("%d", x); /* ==> 12 */
}
```

Prática 1

- Pratique a declaração e utilização de ponteiros
  - defina e inicialize uma variável inteira
  - defina um ponteiro para inteiro
  - modifique o valor da variável através do ponteiro
  - verifique os novos valores da variável usando *printf*

Arrays

- arrays são agrupamentos de dados adjacentes na memória
- declaração:

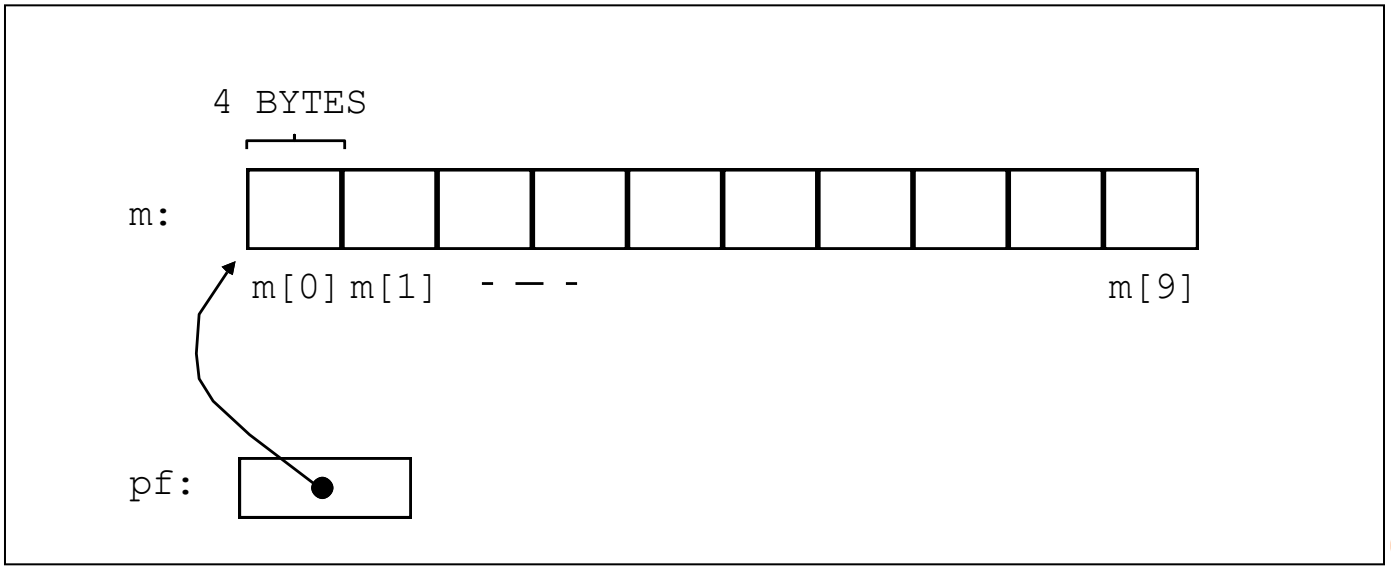
tipo_dado nome_array[<*tamanho*>];

define um arranjo de <*tamanho*> elementos adjacentes na memória do tipo *tipo_dado*

Arrays

- Ex:

```
float m[10], *pf;
pf = m;
```



Referenciando Arrays

- em **float** m[10] *m* é uma constante que endereça o primeiro elemento do array
- portanto, não é possível mudar o valor de *m*
- Ex:

```
float m[10], n[10];
float *pf;

m = n;        /* erro: m é constante ! */
pf = m;       /* ok */
```

Referenciando Elementos

- pode-se referenciar os elementos do array através do seu nome e colchetes:

```
m[5] = 5.5;
if (m[5] == 5.5)
        printf("Exito");
else
        printf("Falha");
```

Referenciando Elementos

- Pode-se referenciar os elementos de um array através de ponteiros:

```
float m[] = { 1.0, 3.0, 5.75, 2.345 };
float *pf;

pf = &m[2];
printf("%f", *pf);      /* ==> 5.75 */
```

Referenciando Elementos

- Pode-se utilizar ponteiros e colchetes:

```
float m[] = { 1.0, 3.0, 5.75, 2.345 };
float *pf;
pf = &m[2];
printf("%f", pf[0]);    /* ==> 5.75 */
```

- Note que o valor entre colchetes é o deslocamento a ser considerado a partir do endereço de referência

pf[n] => indica enésimo elemento a partir de pf

Aritmética de Ponteiros

- É possível fazer operações aritméticas e relacionais entre ponteiros e inteiros
- Soma: ao somar-se um inteiro *n* a um ponteiro, endereçamos *n* elementos a mais (*n* positivo) ou a menos (*n* negativo)

pf[2] equivale a *(pf+2)

*(pf + n) endereça n elementos a frente

*(pf - n) endereça n elementos atrás

pf++ endereça próximo elemento array

pf-- endereça elemento anterior array

EXEMPLO

```
void main ()
{
 int arint[] =  { 1,2,3,4,5,6,7 };
 int size = 7;/* tamanho do array */
 int i, *pi;

 for (pi=arint, i=0; i < size; i++, pi++)
   printf(" %d ", *pi);
}
```

==> 1 2 3 4 5 6 7

EXEMPLO - VARIAÇÃO

```
void main ()
{
 int arint[] =  { 1,2,3,4,5,6,7 };
 int size = 7;/* tamanho do array */
 int i, *pi;

 for (pi=arint, i=0; i < size; i++)
   printf(" %d ", *pi++);
}
```

==> 1 2 3 4 5 6 7

EXEMPLO - VARIAÇÃO

```
void main () {
 int arint[] =  { 1,2,3,4,5,6,7 };
 int size = 7;/* tamanho do array */
 int i, *pi;
   pi = arint;
   printf(" %d ", *pi); pi += 2;
   printf(" %d ", *pi); pi += 2;
   printf(" %d ", *pi); pi += 2;
   printf(" %d ", *pi);
}
```

==> 1 3 5 7

Operações Válidas Sobre Ponteiros

- É valido:
  - *somar* ou *subtrair* um inteiro a um ponteiro (pi ± int)
  - *incrementar* ou *decrementar* ponteiros (pi++, pi--)
  - *subtrair* ponteiros (produz um inteiro) (pf - pi)
  - *comparar* ponteiros (>, >=, <, <=, ==)

- Não é válido:
  - somar ponteiros (pi + pf)
  - multiplicar ou dividir ponteiros (pi*pf, pi/pf)
  - operar ponteiros com *double* ou *float* (pi + 2.0)

Prática 2

- Escreva um programa que imprima um *array* de inteiros na ordem inversa endereçando os elementos com um ponteiro

Universidade de São Paulo – São Carlos
Instituto de Ciências Matemáticas e de Computação

Introdução a Ponteiros e Vetores em C

Material preparado pela profa

Silvana Maria Affonso de Lara e utilizado por outros professores (Rosana Vaccari)

2º semestre de 2010