Linguagem C: Variáveis do Tipo Ponteiro

Prof. Paulo R. S. L. Coelho
paulo@facom.ufu.br

Faculdade de Computação
Universidade Federal de Uberlândia

Organização

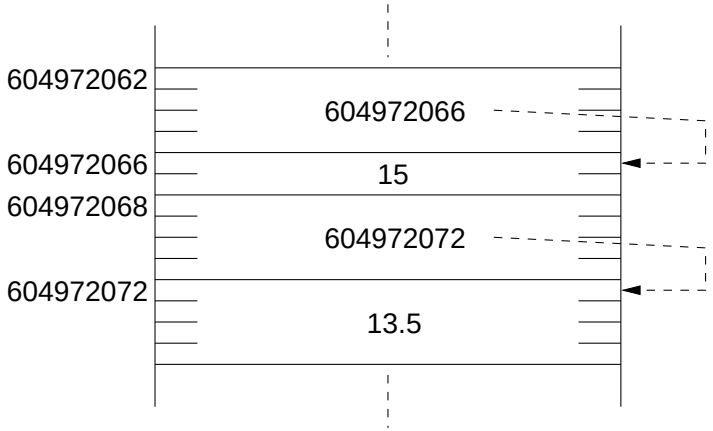
Organização

Declaração e Atribuição

- Variáveis vistas até agora são usadas para guardar e manipular valores de determinados tipos (real, inteiro, caractere).
- Variáveis do tipo **ponteiro** são destinadas a guardar endereços de memória.
- Por exemplo:

```
float a = 13.5, *p;
int b = 15, *q;
p = &a; q = &b;
printf("a=%g; b=%d;\n", a, b);
printf("p=&a=%ld; q=&b=%ld;", p, q);
```

`a` e `b` são variáveis do tipo `float` e `int`, enquanto `p` e `q` são **ponteiros** para o tipo `float` e `int`.



Mapa de memória do código

Declaração e Atribuição

- Considerando as declarações anteriores:
  `p = 1234;` não é permitido, mas
  `p = (float *) 1234;` é permitido
- O fator `(float *)` converte o inteiro 1234 para um valor do tipo ponteiro para `float`, isto é, um endereço destinado a guardar valores do tipo `float`.
- O endereço 0 (zero) é considerado de forma especial em C, de tal forma que existe uma constante especial para ele, a palavra `NULL`.
- Assim, as seguintes atribuições são equivalentes:
  `p = 0;` e `p = NULL;` e indicam que p aponta para *lugar nenhum*.

Acesso ao Ponteiro

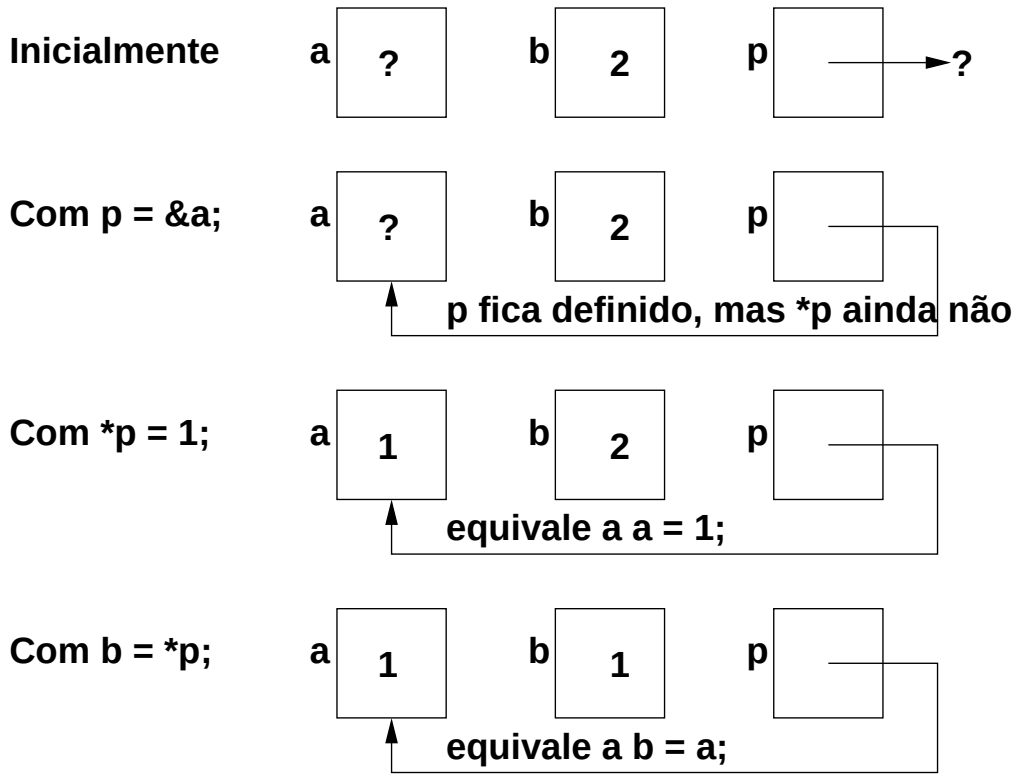
- O asterisco permite referenciar, num comando qualquer, o local apontado pelo ponteiro.
- Assim, se `p` é um ponteiro, `*p` é o local apontado por `p`. Quando `p = &a;` é executado, `a` e `p` referenciam o mesmo local de memória.
- Um ponteiro também pode ser inicializado da seguinte forma:

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

Exemplo 1

- Considere o seguinte fragmento de código:

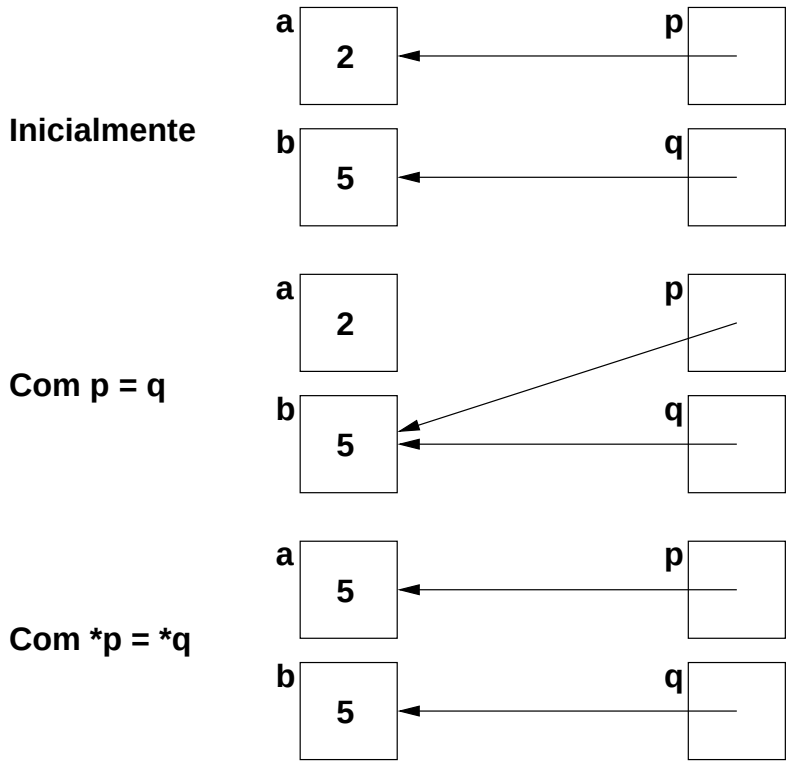
```
int a, b = 2, *p;
p = &a; *p = 1; b = *p;
```



Exemplo 2

- Seja a seguinte declaração e considerando as duas atribuições separadamente:

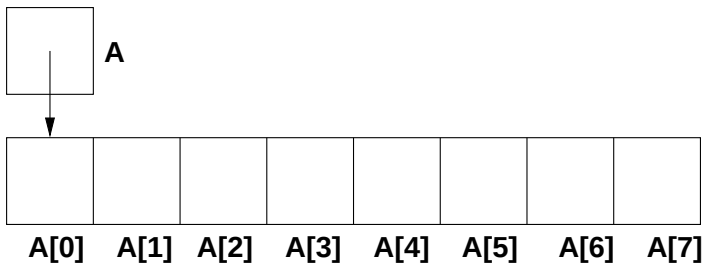
```
int a = 2, b = 5, *p = &a, *q = &b;
p = q;     // atribuicao 1 ou
*p = * q; // atribuicao 2
```



Organização

Relação entre Ponteiros e Variáveis Indexadas

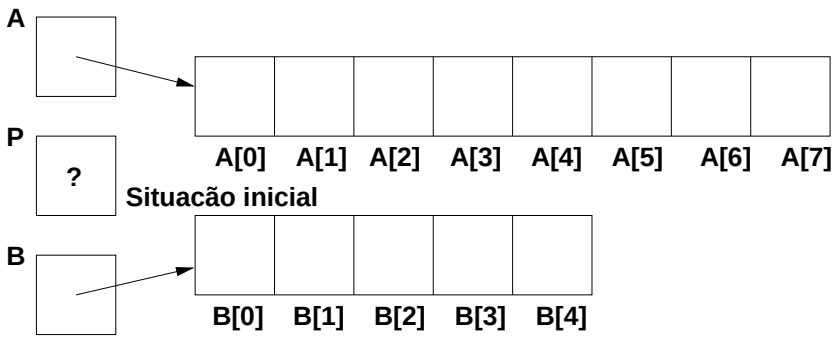
- Na linguagem C, há forte relação entre variáveis indexadas e ponteiros.
- Isto começa pelo fato de o nome da variável indexada ocupar um local de memória contendo o endereço do primeiro elemento.
- Desta forma, este nome é um **ponteiro** para este elemento.
- Este ponteiro, entretanto, aponta para um local fixo e não pode ser alterado durante o programa;
- Considere a seguinte declaração: `int A[8];`
  - A variável `A` pode ser interpretada graficamente como:

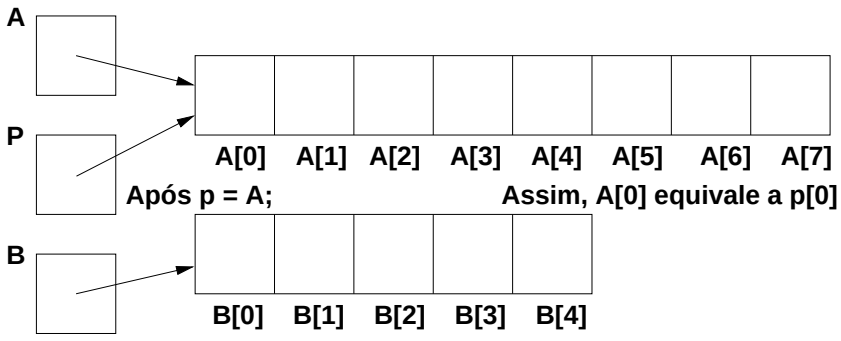


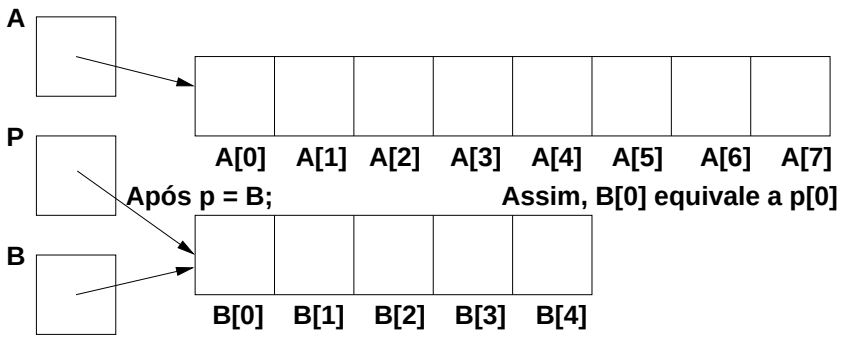
Exemplo

- Considere o seguinte fragmento de código:

```
int A[8], B[5], *p;
p = A; p = B;
```







Aritmética de Endereços

- A linguagem C faz aritmética de ponteiros ou endereços.
- Expressões podem conter um endereço ou um ponteiro a ser somado ou subtraido com uma expressão aritmética inteira.
- Os resultados dessas operações são endereços.
- Por exemplo, seja `p` um ponteiro para inteiro e `i` uma variável inteira, a expressão `p + i` é o endereço do i-ésimo inteiro após aquele apontado por `p`.
- Assim, para o exemplo anterior, temos: `*(p+i)` `equivale a p[i]` e `*(A+i) equivale a A[i]`.
- O valor a ser somado com `p` ou `A` deve ser `i*sizeof(i)`

Organização

Exercícios I

1. Considere o trecho de código a seguir e indique qual será sua saída.

```
float a = 20.8, *p, b = 15.7, *q;
p = &a; q = &b;
printf("a = %g; b = %g;\n", a, b);
printf("p = &a = %ld; q = &b = %ld;\n", p, q);
printf("&p = %ld; &q = %ld;\n", &p, &q);
printf("a = %g; *p = %g;\n", a, *p);
printf("b = %g; *q = %g;\n", b, *q);
*q = *p + 2;
printf("a = %g; *p = %g;\n", a, *p);
printf("b = %g; *q = %g;\n", b, *q);
p = q;
printf("a = %g; *p = %g;\n", a, *p);
printf("b = %g; *q = %g;\n", b, *q);
```

Considere ainda que o endereço de `a` é 1234, de `b` é 1238, de `p` é 1300 e de `q` é 1304.

Exercícios II

2. Considere o trecho de código a seguir e indique qual será sua saída.

```
double *p, *q, *r, A[5];
p = A; q = p+1; r = p+2;
printf("p = %ld; &A[0] = %ld;\n", p, &A[0]);
printf("q = %ld; &A[1] = %ld;\n", q, &A[1]);
printf("r = %ld; &A[2] = %ld;\n", r, &A[2]);
printf("q-p = %ld; r-p = %ld;\n", q-p, r-p);
printf("ender(q) - ender(p) = %ld; ender(r) - ender(p) = %ld;\n",
       (long)q - (long)p, (long)r - (long)p);
```

   Considere ainda que um variável do tipo double ocupa 8 bytes e que o endereço inicial de A é 1234600.

3. Considerando o trecho anterior, indique duas maneiras diferentes de calcular a soma dos elementos do vetor `A` utilizando a variável `p`;