ESTRUTURA DE DADOS

Prof.ª Priscilla Abreu

priscilla.braz@rj.senac.br

Estrutura de dados

Roteiro de Aula

- Objetivo da aula
- Ponteiros

PONTEIROS

Estrutura de dados

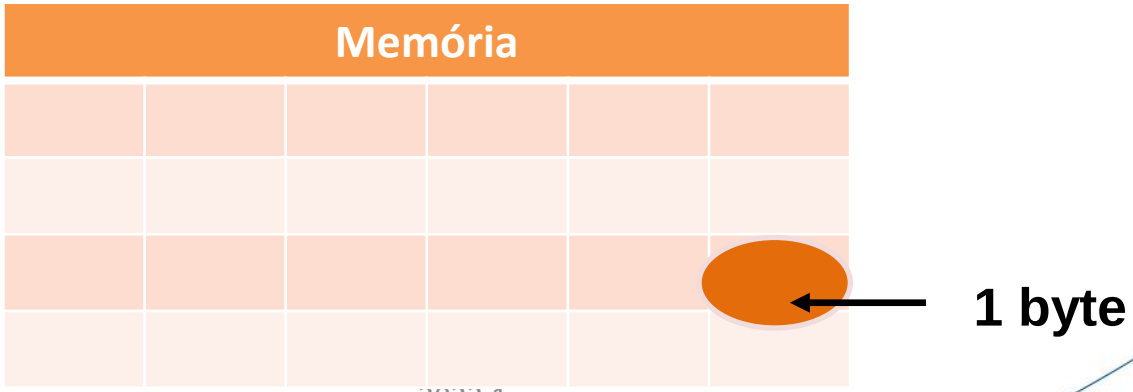
REVISÃO

1. O que são ponteiros?
2. Em que situações os ponteiros costumam ser usados?

Estrutura de dados

ARMAZENANDO INFORMAÇÃO

- Armazenamento de informação -> ocupa espaço;
- Declaração de variável -> alocação de espaço de memória;
- Variável associada a um espaço de memória do computador. Tal espaço de memória é representado por um valor, seu endereço.



Estrutura de dados

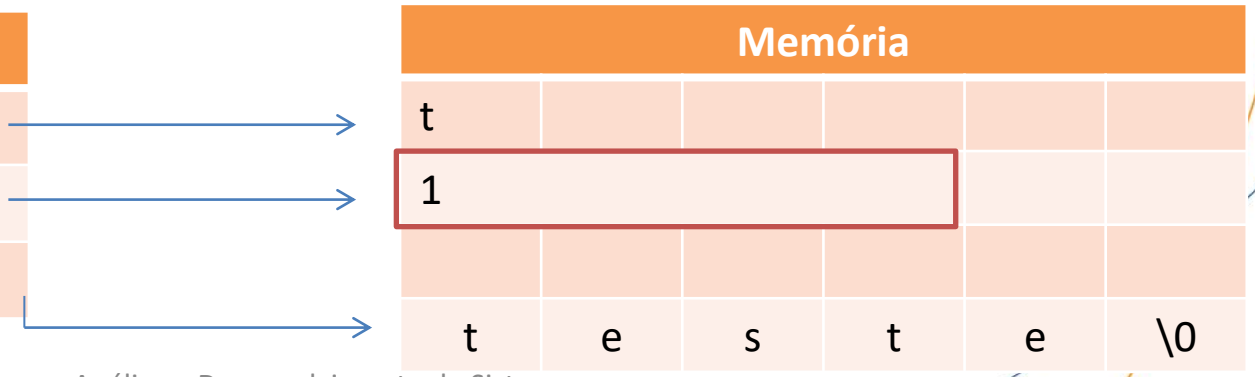
ARMAZENANDO INFORMAÇÃO

Cada tipo de dado declarado em um programa ocupa um tamanho diferente na memória do computador.

Tipo de Dado	Tamanho
char	1 byte
int	4 bytes
float	4 bytes
Double	8 bytes

Uso do comando sizeof() para obter o tamanho que cada tipo de dado ocupa.

Instruções
char x = ‘t’;
int y = 1;
char z[6] = ‘teste’;



PONTEIRO

Estrutura de dados

O QUE É UM PONTEIRO?

- Variável que armazena um endereço de memória.
- É um tipo de variável especial que armazena o endereço de uma segunda variável alocada na memória;
- Tem por função apontar para um endereço de memória determinado, isto é, o endereço que o ponteiro armazena.
- Os ponteiros são muito úteis e bastante utilizados nas situações onde é necessário conhecer o endereço onde está armazenada fisicamente uma variável e não propriamente o seu conteúdo.

Estrutura de dados

DECLARANDO UM PONTEIRO?

- Sintaxe da declaração de ponteiros:

  tipo *nome_ponteiro;

- Declaração:

  int x, * pt_x;

- Variável x do tipo int e ponteiro *pt_x que armazenará o endereço de uma variável do tipo int.

Estrutura de dados

INICIALIZAÇÃO DE PONTEIROS

Inicialização de um ponteiro

- Ponteiros devem ser inicializados antes de serem usados, o que pode ser feito na declaração ou através de uma atribuição.

- Um ponteiro pode ser inicializado com um endereço ou com o valor NULL. O valor NULL é uma constante definida na biblioteca <stdio.h> e significa que o ponteiro não aponta para lugar nenhum.

Estrutura de dados

OPERADORES DE MANIPULAÇÃO

- Manipulação de ponteiros ocorre de duas maneiras:
  - Por meio do endereço de uma variável;
  - Por meio do conteúdo armazenado no endereço apontado pelo ponteiro.
- Operadores:
  - Operador de endereço: &
  - Operador de conteúdo: *

Estrutura de dados

OPERADORES DE MANIPULAÇÃO

Operador de endereço e conteúdo (& e *)

Programa

```
#include <stdio.h>
int main(){
    char x = 't', *p;
    p = &x;
    printf("Conteúdo do ponteiro: %p", p);
    printf("Conteúdo de x é: %c ", x);
    printf("Conteúdo apontado:  %c", *p);
}
```

Busca o endereço armazenado

Busca o conteúdo de x

Busca o conteúdo apontado pelo ponteiro

Memória

x			p
t			2000
2000	2001	2002	2003

Estrutura de dados

ARMAZENANDO INFORMAÇÃO

Endereço de memória	Identificador da variável	Valor armazenado no espaço
1001	x	8
1002		
1003		
1004		
1005		
1006	px	1001

***px = ?**

px = ?

Estrutura de dados

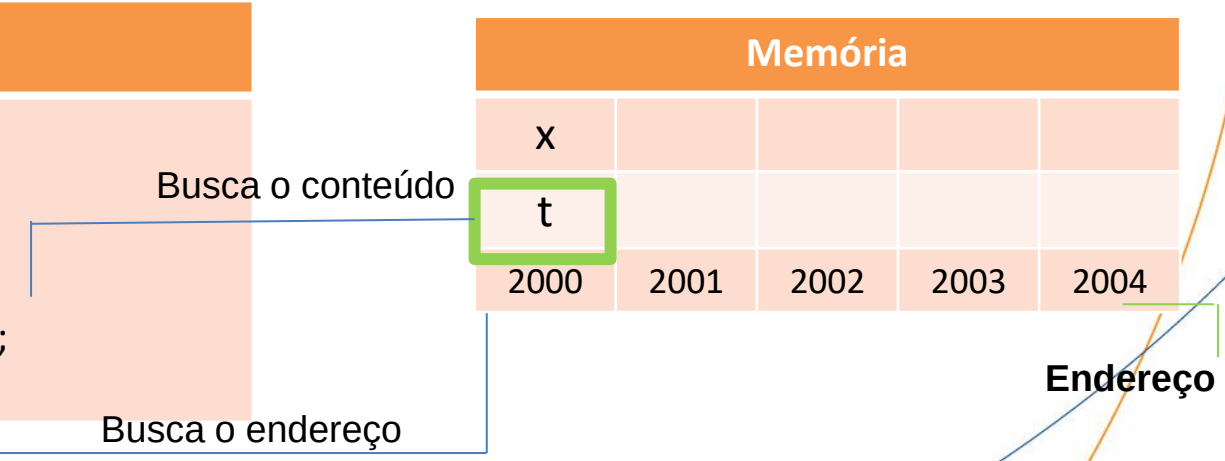
OPERADORES DE MANIPULAÇÃO

Operador de endereço (&)

Indica o endereço de uma variável, que pode ser impresso ou lido com printf e scanf a partir dos operadores de conversão %x ou %X.

Programa

```
#include <stdio.h>
int main(){
    char x = 't';
    printf("Conteúdo de x é: %c ", x);
    printf("Endereço de x é: %x", &x);
}
```



OPERADORES DE MANIPULAÇÃO

Operador de conteúdo (*)

Utilizado na declaração de um ponteiro, mas também toda vez que se deseja saber qual o valor contido no endereço armazenado pelo ponteiro.

Endereço de um ponteiro: endereço físico alocado para essa variável no momento da sua declaração;
Endereço armazenado pelo ponteiro: conteúdo do ponteiro.

Estrutura de dados

OPERADORES DE MANIPULAÇÃO

Operador de conteúdo (*)

Através do operador de conteúdo, também é possível atribuirmos um valor à variável apontada pelo ponteiro.

Exemplo:

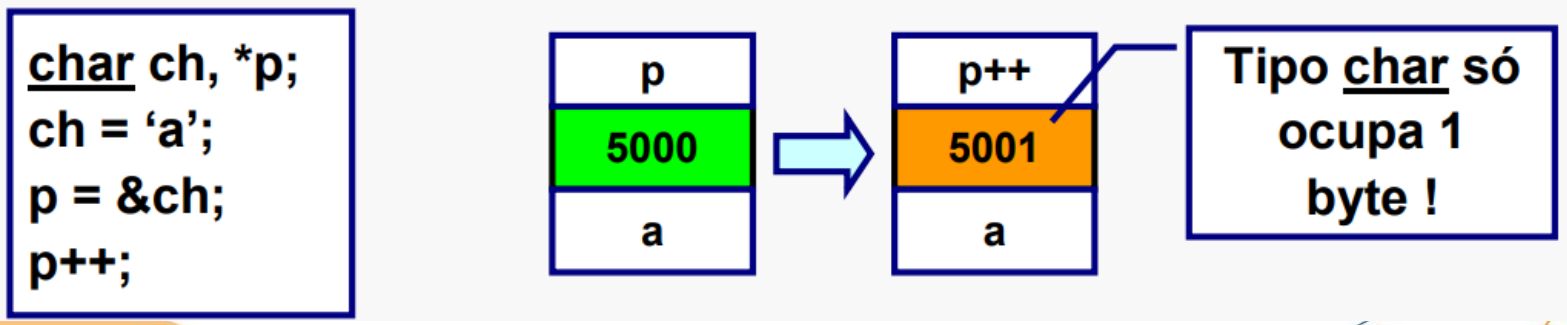
```
int a = 5, *p;
p = &a;
printf("a = %d", a);
*p = 10;
printf("a = %d", a);
```

Estrutura de dados

ARITMÉTICA DE PONTEIROS

É possível realizar operações de soma e subtração nos ponteiros.

Ao somar o valor 1 a um ponteiro, o endereço contido no ponteiro será modificado para o próximo endereço de memória correspondente ao tipo de dado especificado.

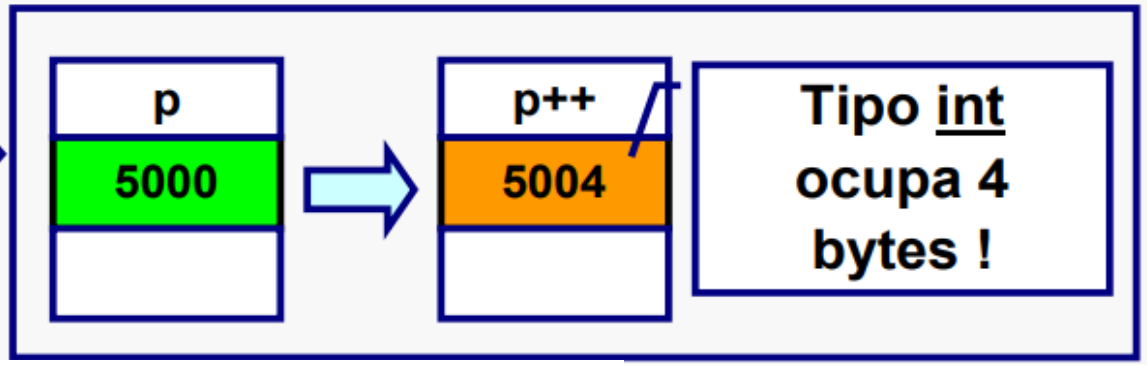


Estrutura de dados

ARITMÉTICA DE PONTEIROS

```
int x, *p;
x = 10;
p = &x;
p++;
```

Memória



Espaço de 4 bytes para armazenar a variável x tipo int.

A operação p++ percorre sizeof(tipo p) bytes !

p				p++			
x	x	x	x	?	?	?	?
5000	5001	5002	5003	5004	5005	5006	5007

Estrutura de dados

COMPARAÇÃO DE PONTEIROS

É possível comparar ponteiros em uma expressão relacional. No entanto, só é possível comparar ponteiros de mesmo tipo.

```
if (px == py) // se px aponta para o mesmo bloco que py ...

if (px > py) // se px aponta para um bloco posterior a py ...

if (px != py) // se px aponta para um bloco diferente de py ...

if (px == NULL) // se px é nulo...
```

Estrutura de dados

PONTEIROS

O programa (trecho de código) abaixo possui erros. Qual(is)? Como deveriam ser?

```
void main() {
        int x, *p;
        x = 100;
        p = &x;
        printf(“Valor de p: %d.\n”, *p);
}
```

Estrutura de dados

PONTEIROS

O programa (trecho de código) abaixo possui erros. Qual(is)? Como deveriam ser?

```
void main() {
        int x, *p;
        x = 100;
        p = x; /* p deveria receber o endereço de x, já que p é um ponteiro (e x não). Ponteiros "armazenam" o endereço para o qual eles apontam! O código correto seria: p = &x; */
        printf("Valor de p: %d.\n", *p);
}
```

Estrutura de dados

PONTEIROS

Qual será a saída deste programa, supondo que i ocupa o endereço 4094 na memória?

```
int main() {
        int i=5, *p;
        p = &i;
        printf("%p – %d  –  %d \n", p, (*p)+2, 3*(*p));
}
```

4094 - 7 - 15

Estrutura de dados

PONTEIROS E VETORES

Na inicialização de um ponteiro com variáveis do tipo vetor, matriz ou string não é preciso usar o operador &, pois eles já são considerados ponteiros constantes.
Na atribuição será repassado o endereço alocado referente à primeira posição da variável.

Exemplo:

```
char nome[34];
char *ponteiro;
ponteiro = nome;
```

Estrutura de dados

PONTEIROS E VETORES

```
#include <stdio.h>
int main (void){
        float v[] = {1.0 , 2.0, 3.0, 4.0, 5.0, 6.0, 7.0};
        int i;
        float *p;
        p = v;
        for (i = 0; i < 7; i++)
                printf ("%.1f ", p[i]);
        printf("\n");
}
```

Estrutura de dados

PONTEIROS E FUNÇÕES

A passagem de parâmetros entre as sub-rotinas se dá de duas formas:

- Passagem por valor;
- Passagem por referência.

Estrutura de dados

PONTEIROS E FUNÇÕES

Passagem por referência:

permite a alteração do valor de uma variável

necessária a passagem do endereço do argumento para a função.

Uso de ponteiros.

DÚVIDAS?