Ponteiros

- Ponteiro é uma variável que guarda o endereço de memória de outra variável.
- ao declarar um ponteiro, requer a definição de um tipo de dado como float, int, char, struct.
- a variável ponteiro irá "apontar" para aquele endereço de memória que armazena um determinado tipo de valor (int, float...).
- A grande dificuldade relacionada com os ponteiros é saber quando está sendo acessado o seu valor, ou seja, o endereço de memória, e quando está acessando a informação apontada pelo ponteiro (conteúdo).

Sintaxe

- Forma geral: **tipo *nome_ponteiro;**
  - **tipo**: qualquer tipo válido em C.
  - nome_ponteiro: qualquer identificador válido em C.
  - *: símbolo para declaração de ponteiro. Indica que o **Identificador(**nome_ponteiro**)** aponta para uma variável do tipo **tipo**.
- Exemplo:
  - int *p;

Operadores de Ponteiros

- Os operadores de ponteiros são:
  - &
  - *

Operador &

- &: operador unário
- Devolve o endereço de memória de seu operando(ou seja, da variável).
  - Usado para atribuir endereços a ponteiros.
  - Exemplos

```
int *p, tam = 35;
p = &tam;   // p recebe “o endereço de” tam
```

Operador *

- Devolve o valor da variável apontada.
  - Ex.:

```
int *p, i, tam = 35;
p = &tam;
i = *p;      /*   i recebe o conteúdo da variável
                  “no endereço” p   */
```

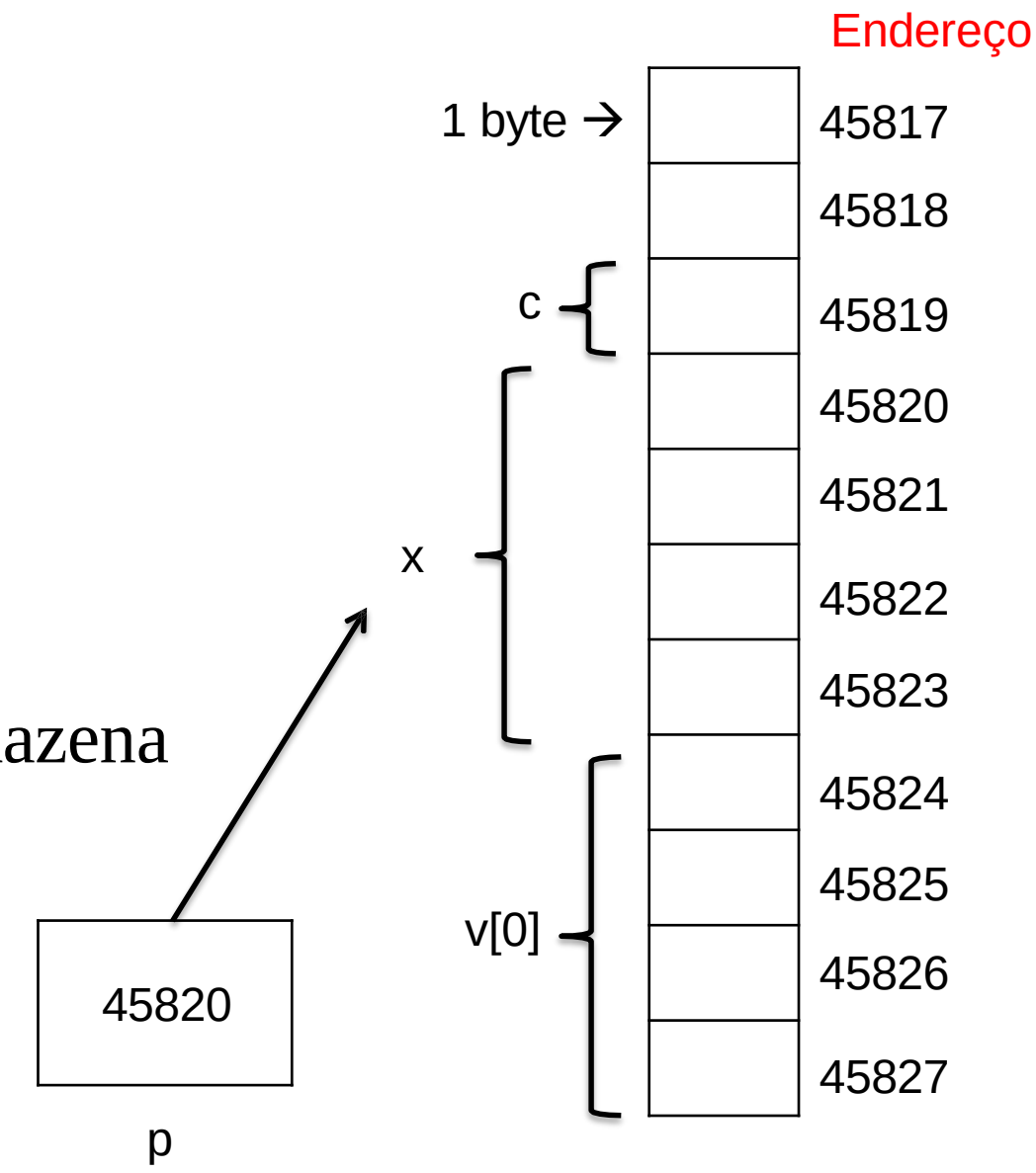
  - O valor de i é 35

Memória

```
char c;
int x;
int v[10];
```

- Um ponteiro armazena endereços

```
int *p = &x;
//p = 45820
```



Exercício

- Seja a seguinte seqüência de instruções em um programa C:

  ```
  int *pti;
  int i = 10;
  pti = &i;
  ```

Qual(ais) afirmativa(s) é **falsa**?

a. pti armazena o endereço de i

b. *pti é igual a 10

c. ao se executar *pti = 20; i passará a ter o valor 20

d. ao se alterar o valor de i, *pti será modificado

e. pti é igual a 10

Ver 9 exemplos de manipulação de ponteiros

Ponteiros para outros tipos

```
char *ptr;
float *ptr;
double *ptr;
void *ptr
```

```
struct pessoa *p1
```

Ponteiro para vetor

Ponteiro para ponteiro

Ponteiros genéricos

- Pode apontar para todos os tipos de dados existentes ou que serão criados

```
void *ptr;
```

Ponteiros genéricos

```
void *pp;
int *p1, p2 = 10;

p1 = &p2;
pp = &p2;       //Endereço de int
pp = &p1;       //Endereço de int *
pp = p1;        // Endereço de int
```

Ponteiros genéricos

- Acesso depende do tipo!

```
void *pp;
int p2 = 10;

pp = &p2;
Cout << *pp ;
```

Ponteiros genéricos

- Acesso depende do tipo!

```
void *pp;
int p2 = 10;

pp = &p2;
```

Ponteiros genéricos

- Aritmética de ponteiros:

```
void *p = 0x9C4; //2500
p++; //2501 -- Sempre soma 1 byte
p = p + 15; //2516
p--;   // 2515 -- Sempre subtrai um byte
```

O programador deve considerar o tipo

Exercícios de Ponteiros

Crie um programa que contenha um vetor de inteiros com tamanho 5. Preencha o vetor, usando o ponteiro (faça o ponteiro percorrer o vetor), com o usuário passando os dados para armazenar no conteúdo deste ponteiro(ou seja no vetor) e após isso, imprima o dobro de cada valor do vetor. Depois imprima o endereço de cada casa do vetor. Utilizar apenas ponteiros.