

CCP130

Desenvolvimento de Algoritmos

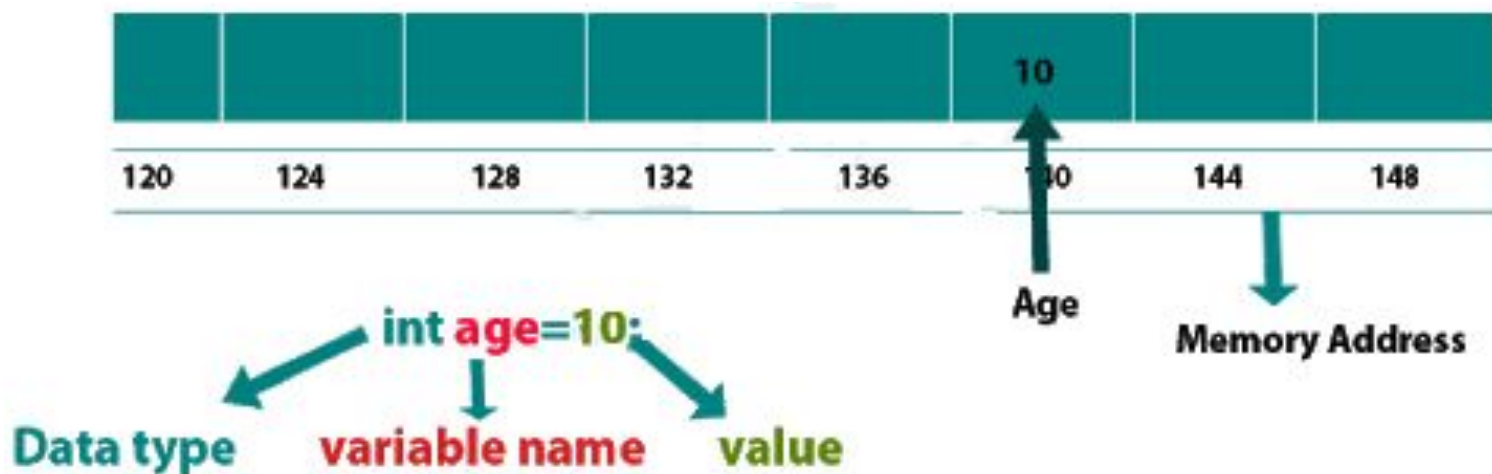
Prof. Danilo H. Perico



Ponteiros

Variáveis

- As **variáveis** declaradas e inicializadas nos programas são armazenadas na memória **RAM**.
- Exemplo: `int age = 10;`





Variáveis

Tem como eu saber em qual posição
da memória minha variável está?!



Variáveis

Sim!!!

Por meio do
operador de endereço: &

Variáveis

```
1  #include <stdio.h>
2
3  int main(void) {
4      int age = 10;
5      printf("%p", &age);
6  }
```

%p = especificador
de conversão para
endereço

```
> ./main
0x7ffd9aeb0e5c>
>
>
```



Variáveis

- O endereço em que uma variável é armazenada é normalmente escolhido pelo computador
- Este **endereço** é dado por um **número** *(quase sempre exibido em hexadecimal)*



Variáveis

É possível **GUARDAR** um **endereço de uma**
variável em outra variável ?



Variáveis

Sim!!!



Ponteiros

- Pode-se criar uma **variável especial** para **armazenar um endereço**:
 - **O ponteiro !**

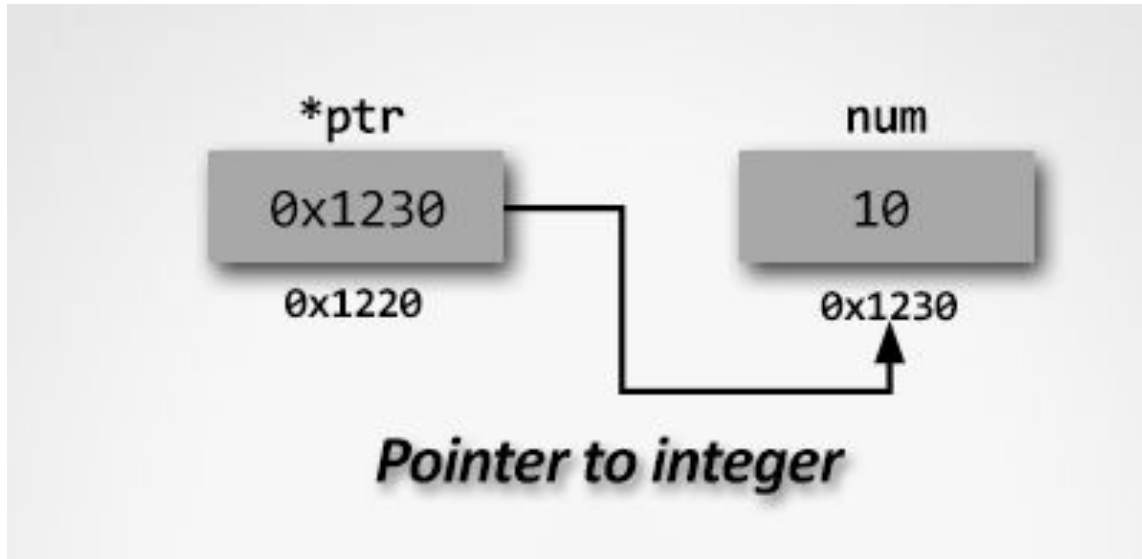


Ponteiros

- O **ponteiro** contém, como valor armazenado, um **endereço de memória**

Ponteiros

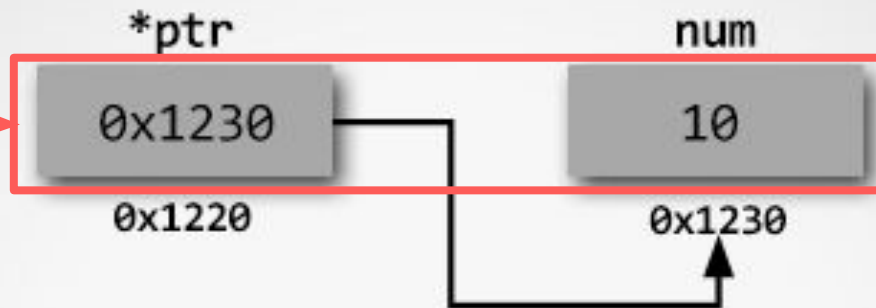
- Dada uma certa variável **num**, um ponteiro **ptr** contém o endereço de **num**



Ponteiros

- Dada uma certa variável **num**, um ponteiro **ptr** contém o endereço de **num**

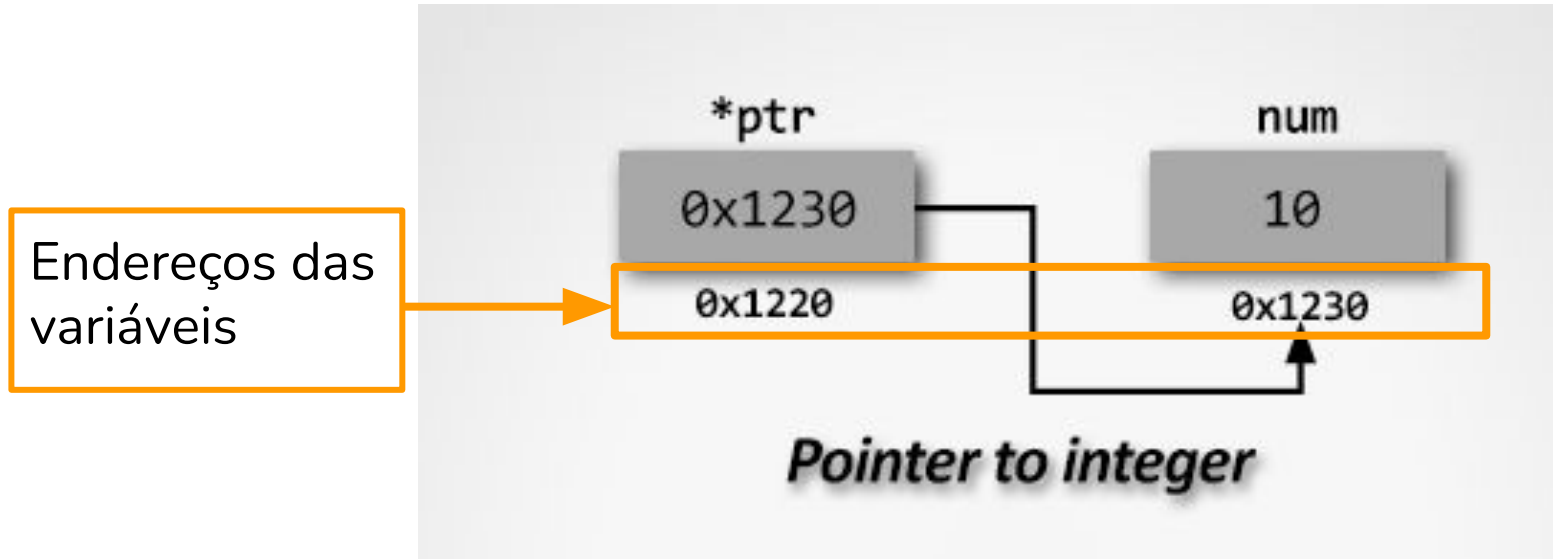
Valores armazenados nas variáveis



Pointer to integer

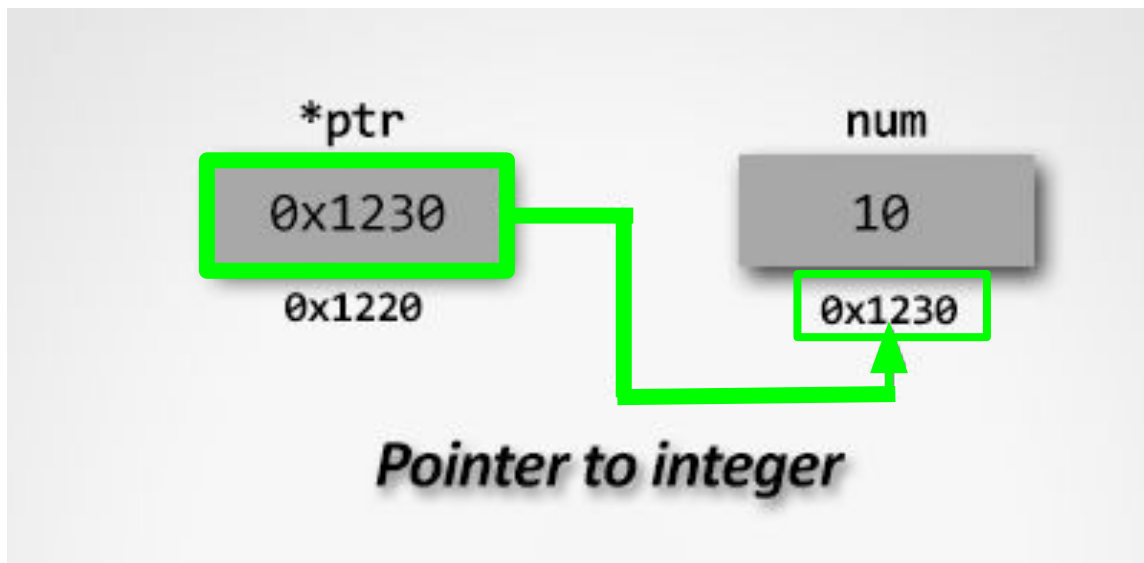
Ponteiros

- Dada uma certa variável **num**, um ponteiro **ptr** contém o endereço de **num**



Ponteiros

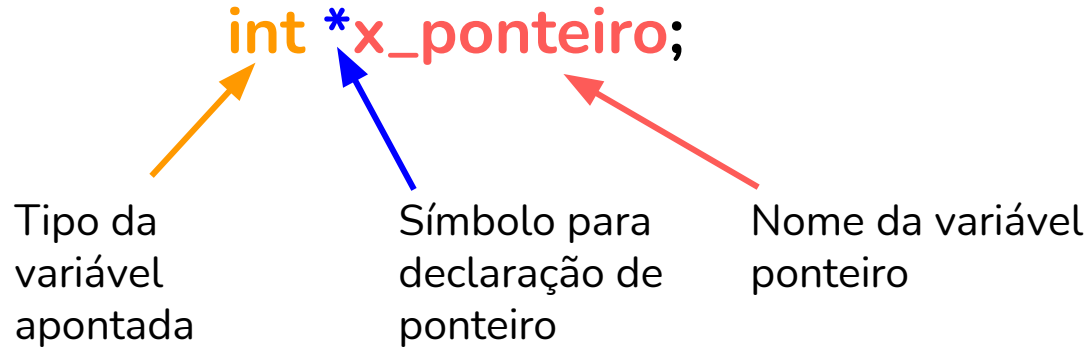
- Assim, *ptr aponta para num*



Ponteiros

Em linguagem C

- Declaração de uma variável ponteiro: **x_ponteiro**
 - **é preciso especificar o tipo da variável “apontada”**





Cuidado!

Um ponteiro deve sempre apontar para um local válido antes de ser utilizado!

Ponteiros

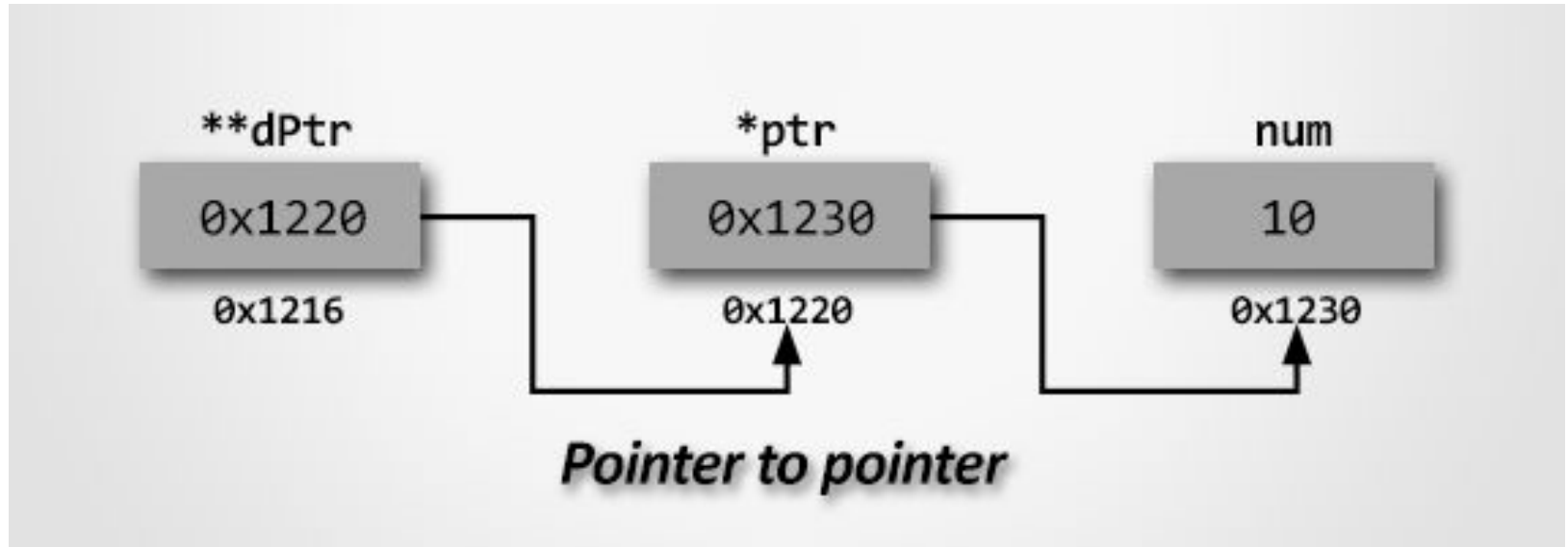
Exemplo em linguagem C : Ponteiro de inteiros

```
1  #include <stdio.h>
2
3  int main(void) {
4      int a = 10; // declarando e inicializando "a"
5      int *aPtr; // declarando a variável ponteiro "aPtr"
6      aPtr = &a; // atribuindo o endereço de "a" para o ponteiro "aPtr"
7
8      printf("conteudo de a %d", a);
9      printf("\nconteudo de aPtr %p", aPtr);
10     printf("\nendereco de a %p", &a);
11     printf("\nendereco de aPtr %p", &aPtr);
12     printf("\nvalor indirecao de aPtr %d", *aPtr);
13     return 0;
14 }
```

```
conteudo de a 10
conteudo de aPtr 0061FF1C
endereco de a 0061FF1C
endereco de aPtr 0061FF18
valor indirecao de aPtr 10
```

Ponteiros

Ponteiro de ponteiro



Ponteiros

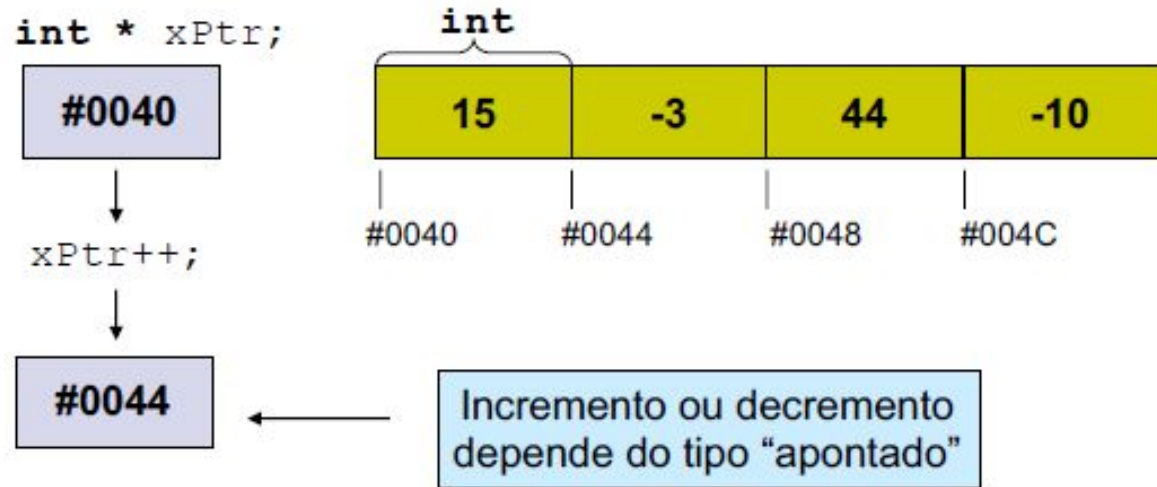
Exemplo em linguagem C : Ponteiro de ponteiro

```
1  #include <stdio.h>
2
3  int main(void) {
4      double x = 10.00;
5      double *xPtr;
6      xPtr = &x;
7      double **ptrXPTr;
8      ptrXPTr = &xPtr;
9      printf("conteudo de x %f", x);
10     printf("\nconteudo de xPtr %p", xPtr);
11     printf("\nendereço de x %p", &x);
12     printf("\nendereço de xPtr %p", &xPtr);
13     printf("\nconteudo de ptrXPTr %p", ptrXPTr);
14     printf("\nendereço de ptrXPTr %p", &ptrXPTr);
15     printf("\nindireção de ptrXPTr %p", *ptrXPTr);
16     printf("\nindireção do conteúdo de ptrXPTr %f", **ptrXPTr);
17     printf("\nindireção de xPtr %f ", *xPtr);
18 }
```

```
conteudo de x 10.000000
conteudo de xPtr 0061FF18
endereço de x 0061FF18
endereço de xPtr 0061FF14
conteudo de ptrXPTr 0061FF14
endereço de ptrXPTr 0061FF10
indireção de ptrXPTr 0061FF18
indireção do conteúdo de ptrXPTr 10.000000
indireção de xPtr 10.000000
```

Ponteiros - Aritmética

- Aritmética de ponteiros
 - Um ponteiro pode ser incrementado (++) ou decrementado (--)

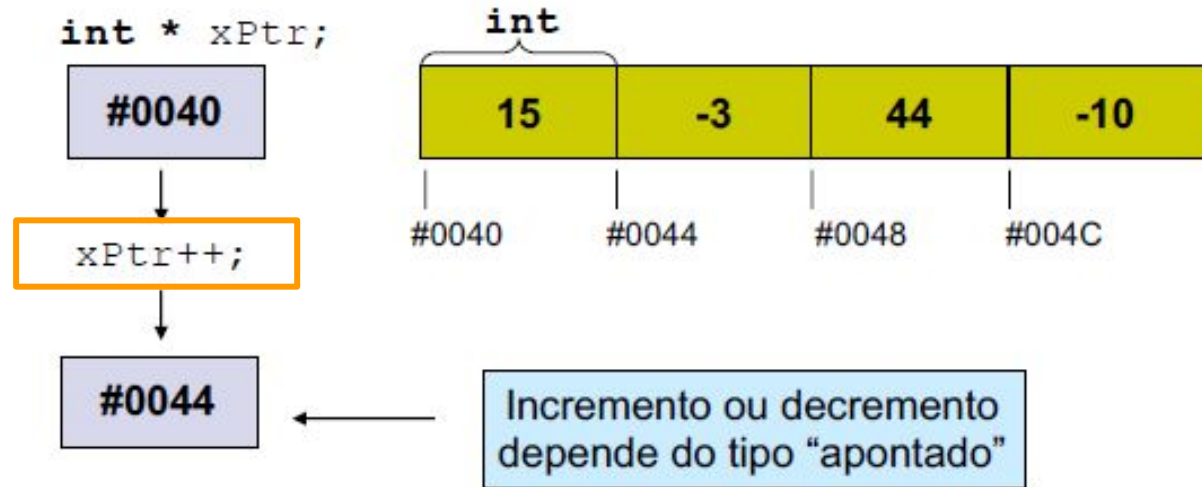


- Relembrando:
Tamanho das
variáveis

Tipo	Tamanho	Valores Possíveis
bool	1 byte	true e false
byte	1 byte	0 a 255
sbyte	1 byte	-128 a 127
short	2 bytes	-32768 a 32767
ushort	2 bytes	0 a 65535
int	4 bytes	-2147483648 a 2147483647
uint	4 bytes	0 to 4294967295
long	8 bytes	−9223372036854775808L to 9223372036854775807L
ulong	8 bytes	0 a 18446744073709551615
float	4 bytes	Números até 10 elevado a 38. Exemplo: 10.0f, 12.5f
double	8 bytes	Números até 10 elevado a 308. Exemplo: 10.0, 12.33
decimal	16 bytes	números com até 28 casas decimais. Exemplo 10.991m, 33.333m
char	1 byte	Caracteres delimitados por aspas simples. Exemplo: 'a', 'ç', 'o'

Ponteiros - Aritmética

- Aritmética de ponteiros
 - Um ponteiro pode ser incrementado (++) ou decrementado (--)



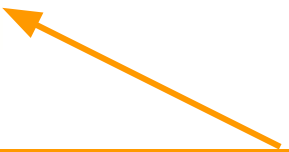
Exemplo

```
3  int main(void) {  
4      int x, y, *p;  
5      y = 1;  
6      p = &y;  
7      x = *p;  
8      x = 4;  
9      (*p)++;  
10     --x;  
11     (*p) += x;  
12 }
```

x	y	p

Exemplo

```
3  int main(void) {  
4      int x, y, *p = NULL;  
5      y = 1;  
6      p = &y;  
7      x = *p;  
8      x = 4;  
9      (*p)++;  
10     --x;  
11     (*p) += x;  
12 }
```



É boa prática inicializar o ponteiro com NULL - desta forma ele começa apontando para nulo! É melhor!

Ponteiros e Vetores

- Na verdade, todo **vetor é um ponteiro**!

```
1  #include <stdio.h>
2
3  int main(void){
4      int v[10];
5      printf("%p \n", v);
6      printf("%p \n", &v[0]);
7      printf("%p \n", &v[1]);
8      printf("%p \n", &v[2]);
9      return 0;
10 }
```

Mesmos endereços

0061FEF8

0061FEF8

Ponteiros e Vetores

- Ao declarar um vetor, uma área sequencial da memória é alocada:

```
int main(void){  
    int v[10];  
    return 0;  
}
```

valores	2	3	2	1	6	4	3	4	7	4
end.	1000h	1004h	1008h	100Ch	1010h	1014h	1018h	101Ch	1020h	1024h

Ponteiros e Vetores

- Ao declarar um vetor, uma área sequencial da memória é alocada:

```
int main(void){  
    int v[10];  
    printf("%d \n", v[0]);  
    printf("%p \n", v);  
    return 0;  
}
```

Imprime 2

Imprime 1000

valores	2	3	2	1	6	4	3	4	7	4
end.	1000h	1004h	1008h	100Ch	1010h	1014h	1018h	101Ch	1020h	1024h

Aritmética de Ponteiros - Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int vetor[5];
5      int *pVetor;
6      vetor[0] = 10;
7      vetor[1] = 20;
8      vetor[2] = 30;
9      vetor[3] = 40;
10     vetor[4] = 50;
11     pVetor = vetor;
12     printf("%p\n", pVetor);
13     pVetor++;
14     printf("%p\n", pVetor);
15     return 0;
16 }
```

```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
❯ clang-7 -pthread -lm -o main main.c
❯ ./main
0x7ffe0dcc73c0
0x7ffe0dcc73c4
❯
```

Cuidado!



`p++;`

`(*p)++;`

`*(++p);`

Os 3 incrementos são diferentes!

Cuidado!



p++; *// incrementa o ponteiro, ou seja o endereço*

(*p)++; *// incrementa o conteúdo apontado por p*

***(++p);** *// incrementa primeiro o ponteiro; depois acessa o
valor da nova posição*

Exemplo



```
1  #include <stdio.h>
2
3  int main(void){
4      int vetor[5];
5      int *p;
6
7      for(int i = 0; i < 5; i++)
8          vetor[i] = i;
9
10     p = vetor;
11     printf("%d\n", (*p)++);
12     printf("%d\n", (*p));
13     p++;
14     printf("%d\n", (*p)++);
15     printf("%d\n", *(++p));
16     printf("%d\n", *(++p));
17     printf("%d\n", *(++p));
18     printf("%d\n", *(++p)); // endereço fora do vetor
19 }
```


Exemplo - Ponteiros e Vetores

- Vamos observar e entender o que faz o programa ao lado:

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int v[10], *pt, i;
6      pt = v;
7      for (i=0; i<10; ++i)
8          v[i]=0;
9      for (i=0; i<10; i+=2)
10         *(pt+i)=6;
11     for (i=1; i<10; i+=2)
12         *(pt+i)=4;
13     printf("Vetor: [ ");
14     for (i=0; i<10; ++i){
15         printf("%d ",*(pt+i));
16     }
17     printf("]\n");
18 }
```

Exercícios

Exemplos



1. Escreva um programa que declare um **inteiro**, um **real**, um **char**, e **ponteiros** para inteiro, real, e char. Associe as variáveis aos ponteiros (use &). Modifique os valores de cada variável usando os ponteiros. Imprima os valores das variáveis antes e após as modificações
2. Faça um programa **para somar dois números reais utilizando ponteiros**
3. Escreva um programa que contenha duas variáveis inteiras. Leia o valor dessas variáveis pelo teclado. Em seguida, **compare seus endereços** e exiba o conteúdo do **maior endereço**