

Linguagens de Programação 1

```
printf("\aAula #%d\n", (int) 6.7 * 2 - 5);
```

Message of the Day

"If you can't explain it simply, you don't understand it well enough."

— Albert Einstein

Albert Einstein (1879–1955)

 Criador da Teoria da Relatividade (Especial e Geral)

Prémio Nobel da Física (1921) pelo efeito fotoelétrico

 Fundador da física moderna

Contribuições à mecânica quântica e cosmologia

Defensor da simplicidade e clareza na ciência

Nascido na Alemanha (1879), naturalizado suíço e mais tarde americano.

Ativista pela paz, direitos civis e liberdade intelectual

Famoso por frases inspiradoras e pensamento humanista

Conteúdo

Aritmética de Apontadores

Vectores e apontadores

Aritmética de Apontadores

```
int a = 5;
int b = 10;

int *aptr = &a;
```

Endereço	Conteúdo	Identificador
256	5	a
260	10	b
264	256	aptr

Qual o valor de **aptr** se fizermos:

```
aptr++;
```

Uma vez que **aptr** é do tipo apontador para inteiro...

O operador **++** avança 4 bytes → aponta para o próximo inteiro

Endereço	Conteúdo	Identificador
256	5	a
260	10	b
264	260	aptr

? Qual o novo valor de `aptr`?

Intuitivamente: $256 + 1 = 257$ ❌

Mas 257 é um byte do meio de uma variável 🤯

Com `aptr++`, o ponteiro vai para `260`

🧠 Porque `int` ocupa 4 bytes!

12 **34** Aritmética e Tipo de Dados

O número de bytes depende do tipo:

`char` ➡ 1 byte

`int` ➡ 4 bytes

`float` ➡ 4 bytes

`double` ➡ 8 bytes

12 34 Exemplo com apontador para char

```
char letra = 'a';
char *p;
p = &letra;

p++;
```

Como `p` é `char*`, o `p++` avança 1 byte apenas ➡

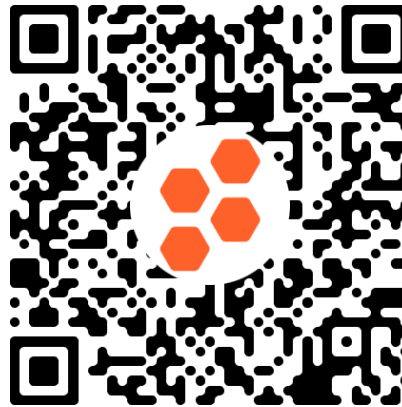
Aponta para o próximo `char` na memória

ab
cd

Endereço	Conteúdo	Identificador
500	'a'	letra
501	500	p

Endereço	Conteúdo	Identificador
500	'a'	letra
501	501	p

? Quizz - Aritmética de Apontadores #1



No campo nome devem colocar o número de aluno 2XXXXXXX.

? Quiz: Aritmética de Apontadores #1

```
#include <stdio.h>

int main(void) {
    char word[] = "BNFOUS";
    char *p1 = &word[1];

    *p1 += 1;
    p1++;
    *p1 += 1;

    puts(word);
    return 0;
}
```

O que é impresso? 🤔

Análise do Quiz

`p1 = &word[1]`  aponta para `'N'`

`*p1 += 1`  `'N'` vira `'O'`

`p1++`  aponta para `'F'`

`*p1 += 1`  `'F'` vira `'G'`

 Resultado: **BOGOUS**

+ Incremento, — Decremento

```
int *ptr = &x[0];  
ptr += 10; // aponta para x[10]  
  
ptr = &x[10];  
ptr -= 10; // volta a x[0]
```


`ptr += 10` ➡ avança `10 * sizeof(int)` bytes


`ptr -= 10` ➡ recua `10 * sizeof(int)` bytes

porque `ptr` é um ponteiro para inteiro. Se fosse um ponteiro para `char` avançaria/recuaria `10 * sizeof(char)`

— Diferença entre Apontadores

```
int strlen(char s[]) {  
    char *ptr = &s[0];  
  
    while (*ptr != '\0')  
        ptr++;  
  
    return (int)(ptr - &s[0]);  
}
```

Permite saber quantos elementos existem entre dois endereços 

Apenas válido entre apontadores do mesmo tipo 

 Exemplo: `strlen`

`ptr = &s[0]`

`ptr` vai andando até `\0`

A diferença é o comprimento da string 

Se `ptr` chegou a 259 e `&s[0]` era 256:

 `259 - 256 = 3`

Comparar Apontadores

Pode-se comparar apontadores do mesmo tipo:

`==` , `!=` , `>` , `<` , `>=` , `<=`

```
while (ptr >= ptr2)

if (ptr1 != ptr2)

if (ptr1 != NULL)
```

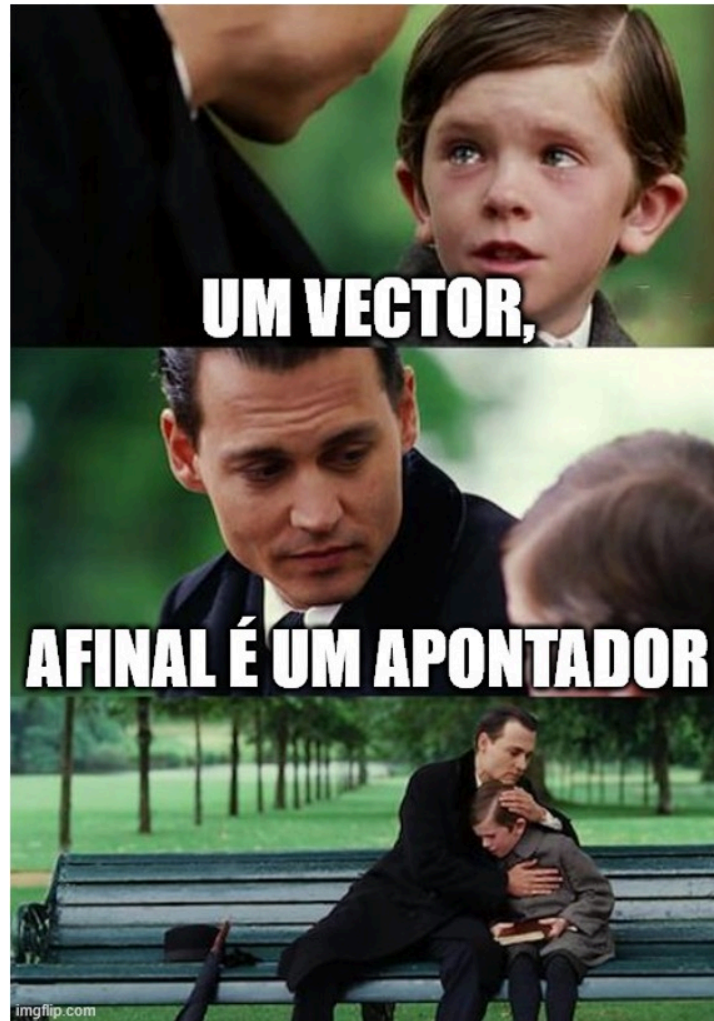
Operações com apontadores

Operação	Exemplo	Observações
Atribuição	<code>ptr = &x</code>	podemos atribuir um valor (endereço) a um apontador. Se quisermos que aponte para nada podemos atribuir-lhe o valor da constante <code>NULL</code>
Incremento	<code>ptr = ptr + 2</code>	Incremento de <code>2*sizeof(tipo)</code>
Decremento	<code>ptr = ptr - 10</code>	Decremento de <code>10*sizeof(tipo)</code>
Apontado por	<code>*ptr</code>	O operador desreferência permite obter o valor existente na posição cujo endereço está armazenado em ptr

Operações com apontadores

Operação	Exemplo	Observações
Endereço de	<code>&ptr</code>	Tal como qualquer outra variável, um apontador ocupa espaço em memória. Desta forma podemos saber qual o seu endereço.
Diferença	<code>ptr1 - ptr2</code>	Permite-nos saber qual o número de elementos entre <code>ptr1</code> e <code>ptr2</code>
Comparação	<code>ptr1 > ptr2</code>	Permite-nos verificar, por exemplo, qual a ordem de dois elementos num vector através dos seus endereços.

Apontadores & Vetores



Apontadores & Vetores

Um vetor é um **apontador** para a primeira posição de memória 🧭 onde está gravada a informação

```
char texto[128] = "OLA";  
scanf("%s", texto);
```

Não precisamos de `&` antes da variável `texto` no `scanf` 😊. Porque `texto` já é um endereço de memória (ou seja, é uma apontador).

Endereço	Conteúdo	Identificador
	500	texto
...	...	
500	'O'	texto[0]
501	'L'	texto[1]
502	'A'	texto[2]
503	'\0'	texto[3]
504	<lixo>	texto[4]
505	<lixo>	texto[5]
...	...	
527	<lixo>	texto[127]

Acesso via Apontador

```
int a[10] = {3,5,8,3,1,0,-1,-4,5,14};

int *p;

p = a;
```

p aponta para a[0]

*(p+1) ➡ a[1]

p[3] ➡ igual a *(p+3)

Endereço	Conteúdo	Identificador
	100	a
...	...	
100	3	a[0]
104	5	a[1]
108	8	a[2]
...
136	14	a[9]
140	100	p
...	...	

Que formas temos de aceder ao 'C'?

```
char s[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
char *ptr = s;
```

s[2]

*(s+2)

ptr[2]

*(ptr+2)

Todas dão acesso ao caractere 'C' !

Igualdades Importantes

```
v == &v[0]
```

```
v[n] == *(v + n)
```

💡 Muito útil para entender como vetores funcionam com apontadores!

Apontadores são variáveis com endereço

```
ptr[0] == *ptr
```

* acede ao valor apontado

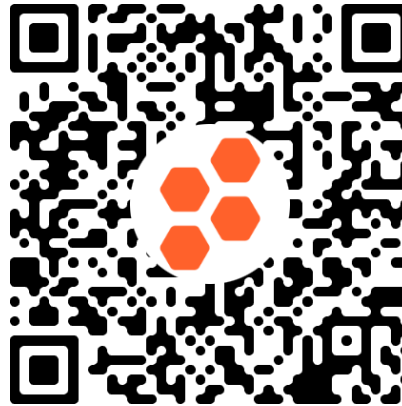
Vetores são passados **por referência**

✗ O que não se pode fazer com vetores

```
int a[10];  
int *p = NULL;  
  
p = a;  
  
a = p;    // ERRO ✗  
  
p++;      // OK ✓  
  
a++;      // ERRO ✗
```

Vetores não podem ser atribuídos nem incrementados!

? Quizz - Apontadores e Vectores



No campo nome devem colocar o número de aluno 2XXXXXXX.



Exercício

```
int conjunto[10]={1,2,3,4,5,6,7,8,9,10};  
int *p;  
p = conjunto;
```

Indique o valor de:

*p

*(p+1)

*(p+2)

p[3]

*conjunto

Exercício

```
int conjunto[10]={1,2,3,4,5,6,7,8,9,10};  
int *p;  
p = conjunto;
```

Indique o valor de:

*p ➡ 1

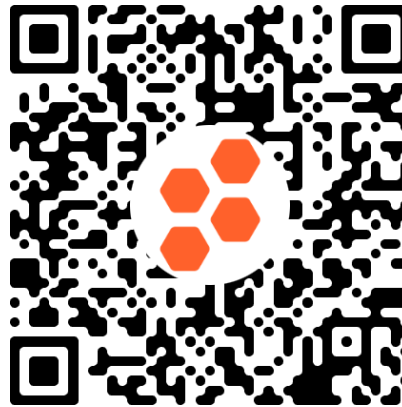
*(p+1) ➡ 2

*(p+2) ➡ 3

p[3] ➡ 4

*conjunto ➡ 1


? Quizz - Apontadores Verdadeiro ou Falso



No campo nome devem colocar o número de aluno 2XXXXXXX.

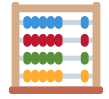
Verdadeiro ou Falso

Não é possível ler o endereço de um apontador ✗

`ptr[0] == *ptr` é verdadeiro 

Vetores são passados por valor ✗

`*` serve para ler o endereço ✗ (serve para **desreferenciar**)



Vetores de Apontadores

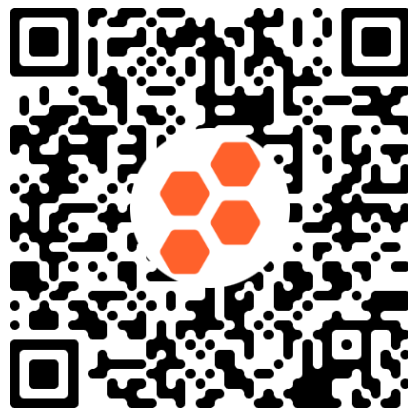
```
char *meses[12];  
char jan[] = "janeiro";  
meses[0] = jan;
```

meses aponta para meses[0]

meses[0] é igual a jan e aponta para jan[0]

Endereço	Conteúdo	Identificador
...	100	meses
100	200	meses[0]
108	300	meses[1]
116	400	meses[2]
...
...	200	jan
200	j	jan[0]
201	a	jan[1]
202	n	jan[2]
...

? Quizz - Super Quizz



No campo nome devem colocar o **número de aluno 2XXXXXXX**.

```
int main() {
    char *rank[3];
    char *easy = "padawan";
    char *medium = "jedi";
    char *hard = "jedi master";

    rank[0] = easy;
    rank[1] = medium;
    rank[2] = hard;

    char *my_rank = rank[1];
    printf("I am %s\n", my_rank);
    my_rank++;
    printf("I am %s\n", my_rank);
    my_rank = *(rank + 2);
    printf("I am %s\n", my_rank);
    return 0;
}
```

Output do Super Quiz

```
I am jedi
```

```
I am edi
```

```
I am jedi master
```

 Entendes o comportamento de apontadores para strings agora?

Apontadores, Vetores e Funções

```
float funcao(int vector[DIMENSAO]);  
float funcao(int vector[]);  
float funcao(int *vector);
```

Formas equivalentes 

A dimensão não faz parte da assinatura !

Útil passar também o **tamanho** do vetor

```
float funcao(int *vector, int tamanho);
```

Exemplo 1

```
int negativas(float *notas, int tam) {
    int i, neg;
    for (i = 0, neg = 0; i < tam; i++) {
        if (*(notas + i) < 9.5) // equivalente a `notas[i]`
            neg++;
    }
    return neg;
}

int main(void) {
    float notas[10] = {1, 2, 0, 7.5, 7, 7, 2, 4, 16, 8};
    int n_neg = negativas(notas, 10);
    printf("numero de negativas: %d\n", n_neg);
    return 0;
}
```

Exemplo 2

Função que converte uma string para letras capitais

```
char * strToUpper(char * s) {
    char *p = s;
    while (*p) // equivalente a: *p != '\0' porque o caractere '\0' tem o código ascii 0
    {
        *p = toupper(*p);
        p++;
    }
    return s;
}
```

```
int main(void) {
    char m[] = "use the force luke";
    puts(strToUpper(m));

    return 0;
}
```


? Q&A

 Dúvidas?