

Ponteiros

- Variáveis:
 - Espaços na memória para guardar informações;
 - Recebem nomes únicos para identificação;
 - Não ligamos para sua localização física em si.

Ponteiros

- Variáveis:
 - As linguagens C e C++ apresentam uma série de situações em que a localização física é importante.
 - Ponteiros são variáveis que indicam a localização física de outras variáveis.

Ponteiros

- Memória do computador:
 - 'Células' de *bytes* sucessivos → Posições únicas.



Ponteiros

- Ponteiros - declaração
 - *tipo *nome_do_ponteiro;*
 - Exemplo: *char *pont;* é um ponteiro que pode apontar para variáveis do tipo char.

Ponteiros

- Ponteiros - operadores & e *
 - $\&var1$ = endereço de variável *var1*
 - $*var2$ = valor da variável **APONTADA** pela variável *var2*

Ponteiros

- Ponteiros

```
#include <stdio.h>

void main( )
{
    char a;
    char *b;

    a = 23;
    b = &a;
    printf("Antes, a = %d\n",a);
    *b = 33;
    printf("Depois, a = %d\n",a);
}
```

Ponteiros

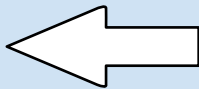
- Ponteiros

```
#include <stdio.h>
```

```
void main( )  
{
```

```
    char a;  
    char *b;
```

```
    a = 23;
```



a recebe o valor 23

```
    b = &a;
```

```
    printf("Antes, a = %d\n",a);
```

```
    *b = 33;
```

```
    printf("Depois, a = %d\n",a);
```

```
}
```

Ponteiros

- Ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```

```
    char a;
```

```
    char *b;
```

```
    a = 23;
```

```
    b = &a;
```

```
    printf("Antes, a = %d", a);
```

```
    *b = 33;
```

```
    printf("Depois, a = %d\n", a);
```

```
}
```

Assumindo que *a* está na posição 1567,
b recebe o valor 1567

Ponteiros

- Ponteiros

```
#include <stdio.h>
```

```
void main( )  
{
```

```
    char a;  
    char *b;
```

```
    a = 23;
```

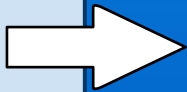
```
    b = &a;
```

```
    printf("Antes, a = %d\n",a);
```

```
    *b = 33;
```

```
    printf("Depois, a = %d\n",a);
```

```
}
```



Antes, a = 23

Ponteiros

- Ponteiros

```
#include <stdio.h>
```

```
void main( )  
{
```

```
    char a;  
    char *b;
```

```
    a = 23;
```

```
    b = &a;
```

```
    printf("Antes, a = %d\n", a);
```

```
    *b = 33;
```

```
    printf("Depois, a = %d\n", a);
```

```
}
```

A variável na posição 1567 recebe o valor 33. Isto é, a variável a, por estar na posição 1567, recebe o valor 33.

Ponteiros

- Ponteiros

```
#include <stdio.h>
```

```
void main( )  
{
```

```
    char a;  
    char *b;
```

```
    a = 23;
```

```
    b = &a;
```

```
    printf("Antes, a = %d\n", a);
```

```
    *b = 33;
```

```
    printf("Depois, a = %d\n", a);
```

```
}
```

Não confundir com o uso do * na declaração do ponteiro. Aqui, o * está somente indicando que *b* é um ponteiro.

A variável na posição 1567 recebe o valor 33. Isto é, a variável *a*, por estar na posição 1567, recebe o valor 33.

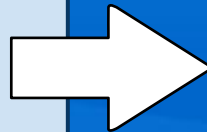
Ponteiros

- Ponteiros

```
#include <stdio.h>

void main( )
{
    char a;
    char *b;

    a = 23;
    b = &a;
    printf("Antes, a = %d\n",a);
    *b = 33;
    printf("Depois, a = %d\n",a);
}
```



Antes, a = 23
Depois, a = 33

Ponteiros

- Ponteiros

```
#include <stdio.h>

void main( )
{
    char a;
    char *b;

    a = 23;
    b = &a;
    printf("Antes, a = %d\n",a);
    *b = 33;
    printf("Depois, a = %d\n",a);
}
```

Devido a essa capacidade de alterar valores diretamente, é preciso indicar o tipo de dado do ponteiro

Antes, a = 23
Depois, a = 33

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```

```
    char a;
```

```
    char *b;
```

```
    a = 23;
```

```
    b = &a;
```

```
    b++;
```

```
}
```

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )  
{
```

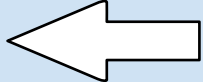
```
    char a;  
    char *b;
```

```
    a = 23;
```

```
    b = &a;
```

```
    b++;
```

```
}
```



Assumindo que *a* está na posição 1567,
b recebe o valor 1567

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```

```
    char a;
```

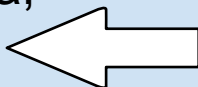
```
    char *b;
```

```
    a = 23;
```

```
    b = &a;
```

```
    b++;
```

```
}
```



O endereço 1567 diz respeito ao *byte* de número 1567. Quando incrementamos *b*, andamos mais um byte na memória (porque *b* é do tipo char, que ocupa um *byte*).

Ou seja, *b* = 1568.

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```

```
    int a;
```

```
    int *b;
```

```
    a = 23;
```

```
    b = &a;
```

```
    b++;
```

```
}
```

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```

```
    int a;
```

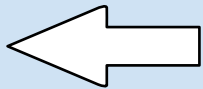
```
    int *b;
```

```
    a = 23;
```

```
    b = &a;
```

```
    b++;
```

```
}
```



Assumindo que *a* está na posição 1567,
b recebe o valor 1567

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )  
{
```

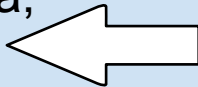
```
    int a;  
    int *b;
```

```
    a = 23;
```

```
    b = &a;
```

```
    b++;
```

```
}
```



Quando incrementamos *b*, andamos mais QUATRO bytes na memória (porque *b* é do tipo int, que ocupa quatro *bytes*).

Ou seja, *b* = 1571.

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>

void main( )
{
    int a[4]; int *b;

    a[0] = 23;
    b = &(a[0]);
    b++;
    *b = 789;
    *(b+1) = 354;
    b = &(a[3]);
    *b = 90;
    printf("a = {%d, %d, %d, %d}",
        a[0], a[1], a[2], a[3]);
}
```

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```

```
    int a[4]; int *b;
```

```
    a[0] = 23;
```

```
    b = &(a[0]);
```

```
    b++;
```

```
    *b = 789;
```

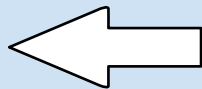
```
    *(b+1) = 354;
```

```
    b = &(a[3]);
```

```
    *b = 90;
```

```
    printf("a = {%d, %d, %d, %d}",  
          a[0], a[1], a[2], a[3]);
```

```
}
```



b recebe o endereço de *a[0]*
b = endereço do primeiro elemento do vetor *a*

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```

```
    int a[4]; int *b;
```

```
    a[0] = 23;
```

```
    b = &(a[0]);
```

```
    b++;
```

```
    *b = 789;
```

```
    *(b+1) = 354;
```

```
    b = &(a[3]);
```

```
    *b = 90;
```

```
    printf("a = {%d, %d, %d, %d}",  
          a[0], a[1], a[2], a[3]);
```

```
}
```

b = endereço de *a*[0] + 4 bytes (tamanho de um int)
==> *b* = endereço de *a*[1]

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```


```
{
```

```
    int a[4]; int *b;
```

```
    a[0] = 23;
```

```
    b = &(a[0]);
```

```
    b++;
```

```
    *b = 789; 
```

```
    *(b+1) = 354;
```

```
    b = &(a[3]);
```

```
    *b = 90;
```

```
    printf("a = {%d, %d, %d, %d}",  
          a[0], a[1], a[2], a[3]);
```

```
}
```

a[1] = 789

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```


```
    int a[4]; int *b;
```

```
    a[0] = 23;
```

```
    b = &(a[0]);
```

```
    b++;
```

```
    *b = 789;
```

```
    *(b+1) = 354; 
```

```
    b = &(a[3]);
```

```
    *b = 90;
```

```
    printf("a = {%d, %d, %d, %d},\n",  
          a[0], a[1], a[2], a[3]);
```

```
}
```

Guarde o valor 354 no endereço
apontado por $b + 4$ bytes

==> Guarde o valor 354 no endereço de $a[1] + 4$ bytes
==> $a[2] = 354$

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```

```
    int a[4]; int *b;
```

```
    a[0] = 23;
```

```
    b = &(a[0]);
```

```
    b++;
```

```
    *b = 789;
```

```
    *(b+1) = 354;
```

```
    b = &(a[3]);
```

```
    *b = 90;
```

```
    printf("a = {%d, %d, %d, %d}",  
          a[0], a[1], a[2], a[3]);
```

```
}
```

b = endereço de *a*[3]

==> *b* = endereço da quarta posição do vetor *a*

Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>
```

```
void main( )
```

```
{
```

```
    int a[4]; int *b;
```

```
    a[0] = 23;
```

```
    b = &(a[0]);
```

```
    b++;
```

```
    *b = 789;
```


```
    *(b+1) = 354;
```

```
    b = &(a[3]);
```

```
    *b = 90;
```

```
    printf("a = {%d, %d, %d, %d},\n",  
          a[0], a[1], a[2], a[3]);
```

```
}
```



a[3] = 90

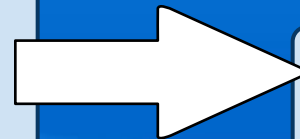
Ponteiros

- Aritmética de ponteiros

```
#include <stdio.h>

void main( )
{
    int a[4]; int *b;

    a[0] = 23;
    b = &(a[0]);
    b++;
    *b = 789;
    *(b+1) = 354;
    b = &(a[3]);
    *b = 90;
    printf("a = {%d, %d, %d, %d}",
        a[0], a[1], a[2], a[3]);
}
```



a = {23, 789, 354, 90}

Ponteiros

- Ponteiros e vetores
 - Quando declaramos, por exemplo, *char str[20];* , estamos indicando que queremos alocar 20 posições de memória do tamanho char. Quando acessamos *str[13]*, estamos acessando a 14^a posição do vetor, ou 13 posições além do início do vetor.

Ponteiros

- Ponteiros e vetores
 - Se criarmos um ponteiro *char *p_str;*, e guardarmos nele o endereço de *str[0]*, então as expressões

str[0] = 0;

**p_str = 0;*

são equivalentes, bem como

str[19] = 76;

**(p_str+19) = 76;*

Ponteiros

- Ponteiros e vetores
 - Na verdade, o identificador *str* guarda o primeiro endereço da memória de 20 chars que foi alocada. Tanto faz usar

p_str = &(str[0]);

ou

p_str = str;

Ponteiros

- Ponteiros e vetores
 - Fazer o contrário ($str = p_str$) não faz sentido. str já é um espaço de 20 *bytes* alocado na memória, não havendo necessidade de fazê-lo apontar para um outro endereço qualquer (p_str).