# Linguagem C

## Ponteiros

Softblue
cursos online

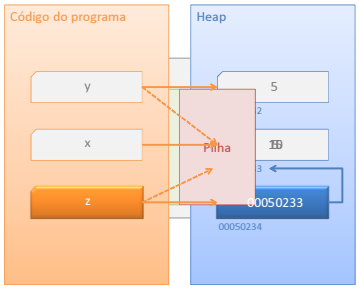
## Operadores

- "E" comercial (&)
  - Permite acessar o endereço de memória de seu operando
- Asterisco (*)
  - Permite acessar o valor armazenado em um endereço de memória fornecido


## Ponteiros

- Apontadores de endereços de memória

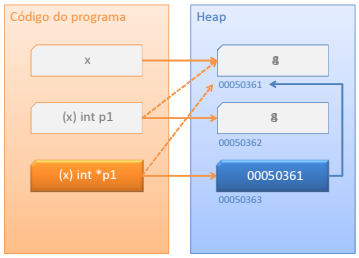
```
int x;
x = 5;
printf("%d", x);          // 5

int y;
y = x;
x = 10;
printf("%d", x);          // 10
printf("%d", y);          // 5

int *z;
z = &x;
x = 15;
printf("%d", x);          // 15
printf("%d", z);          // 00050233
printf("%d", *z);         // 15

printf("%d", &x);         // 00050233
printf("%d", &y);         // 00050232
printf("%d", &z);         // 00050234
```



## Ponteiros em funções

- Facilitam a atualização de variáveis enviadas para funções externas

```
void soma(int p1)
{
    p1 = p1 + p1;
}

void somaPonteiro(int *p1)
{
    *p1 = *p1 + *p1;
}

int x = 4;
soma(x);
printf("%d", x);
// 4

somaPonteiro(&x);
printf("%d", x);
// 8
```



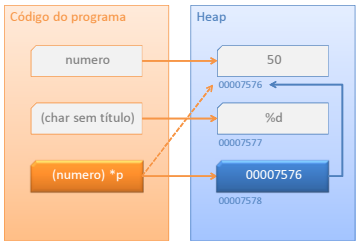

## Ponteiros no scanf

- scanf usa ponteiro para atualizar as variáveis

Código fictício

```
void scanf(forma, *p)
{
    *p = leituraTeclado();
}
```

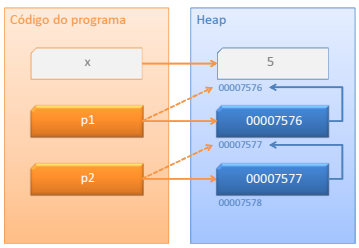
```
int numero;

scanf("%d", &numero);

printf("%d", numero);  // 50
```




## Apontamento múltiplo

- Ponteiro que aponta para outro ponteiro

```
int x = 5;

int *p1 = &x;

int **p2 = &p1;

printf("%d", **p2);
// 5

printf("%d", *p2);
// 00007576

printf("%d", p2);
// 00007577
```

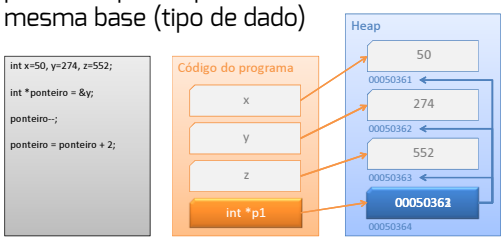


## Características de ponteiros

- Iniciam com valores sujos
  - Podem apontar para áreas inválidas
- Utilização de valor nulo na inicialização
  - int *p = NULL;
- Podem ser comparados em expressões condicionais
  - Comparações numéricas
  - Apontam para mesma área (==)
- Uma string (vetor de char) por si só já é um endereço de memória

## Aritmética de ponteiros

- Operações de adição e subtração
- Ocorrem deslocando os endereços dos ponteiros para o próximo ou anterior de mesma base (tipo de dado)




## Alocação dinâmica de memória

- Biblioteca **stdlib.h**
- Instrução malloc
  - void* malloc(bytes)
- Instrução free
  - free(void*)

```
// Talvez nessa aula mostrar alocando INT, CHAR, DOUBLE apenas

char *minhaStr;            // Declaração
minhaStr = malloc(100);  // Alocação de memória

if(!minhaStr) {            // Validação
      printf("Erro");
}

// Uso
```


Softblue