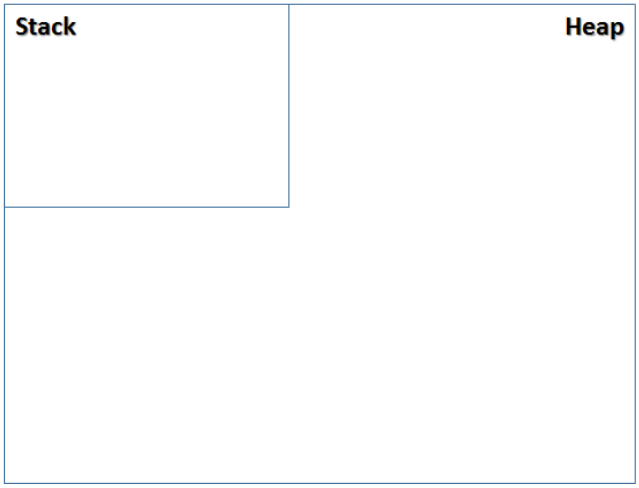
Memória, Tipo Referência e Tipo Valor

Resumo

- Basicamente o compilador divide a memória da nossa aplicação em duas partes: **Stack** e **Heap**.

- **Representação da memória em aplicações .NET**:



Proproção de memória Heap x Stack de uma app .NET

- **Stack**: porção de memória pequena onde os *value-types* e os ponteiros ficam;

- **Value-Type**: são tipos leves (como os tipos primitivos) que ficam armazenados diretamente na memória *stack*. Os valores das variáveis ficam armazenados juntamente com as próprias variáveis, sendo o acesso ao seu conteúdo feito de maneira direta;

- **Heap**: porção maior de memória onde os *reference-types* ficam de fato alocados... Para se fazer o acesso a eles, precisamos de um ponteiro na *stack* que indique a posição de memória na *heap* onde o objeto está de fato alocado.

- **Reference-Type**: tipos pesados (objetos criados a partir de classes, etc.) que ficam armazenados na *heap*. Para não sacrificar a performance, é criada uma referência (ponteiro) na *stack* que aponta para qual posição de memória o objeto está armazenado na heap. O acesso é feito via essa referência na *stack*. Sendo assim, o acesso ao conteúdo é indireto, dependendo dessa referência;

Os **Value Types** ficam alocados na Stack, enquanto os **Reference Types** ficam alocados na Heap.

Reference Types

No C# os Reference Types são:

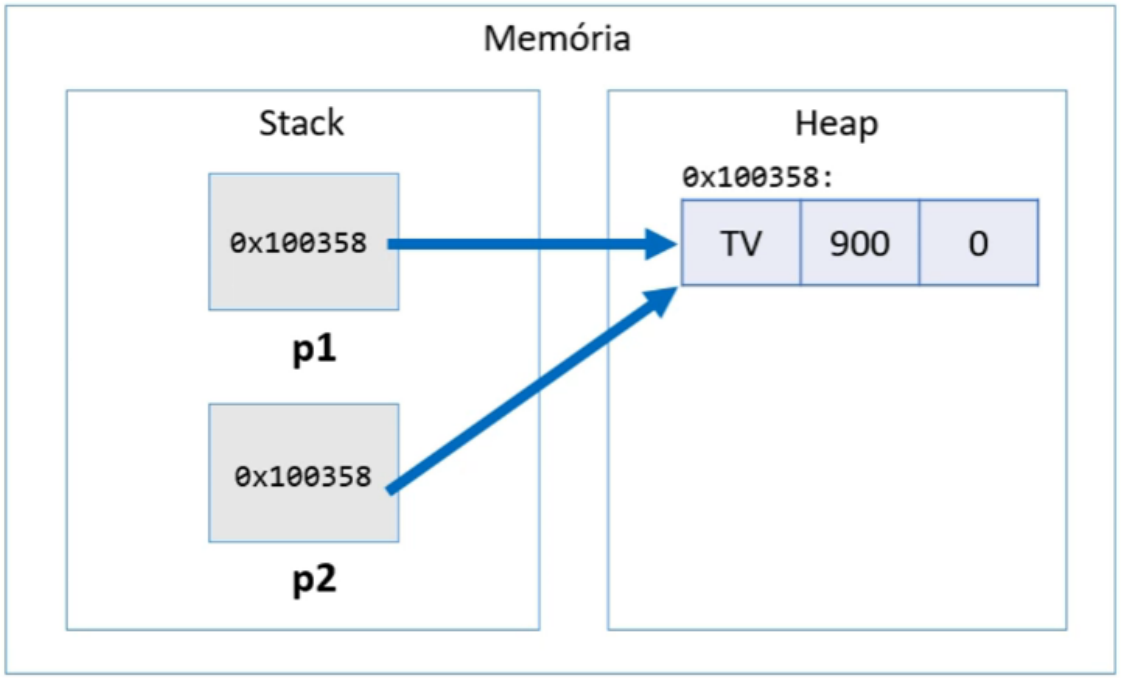
```
Classes -> class

Objetos -> class = new();

Interfaces -> interface

Strings -> string

Arrays -> new[]

Delegates -> delegate
```

Algumas características deles são:

- Não armazenam os dados em si, e sim uma referencia de endereço de memoria de onde o dado está alocado.
- Não são dados independentes. (se você atribuir ele a outra variável, ambas apontam pra mesma referência)

```
// Alocado na heap
class Produto
```

```
{
    public string Nome {get; set;}
    public int Preco{get; set;}
    public int Estoque{get; set;}
}

// Criado na stack que faz uma referência pro endereço de memoria na heap
Produto p1 = new("TV", 900, 0);
Produto p2 = p1;
```



Ponteiro criado na stack, que faz referência pro endereço de memoria existente na heap

Isso acontece pois a Heap é muito grande, então o compilador não acessa diretamente ela quando vai puxar os dados.

É criado uma referencia desse valor dentro da Stack (chamado de ponteiro) que trás a posição de memória da heap que aquele dado está alocado.

E por isso esses tipos de valores são chamados de **Reference Type**.

Value Types

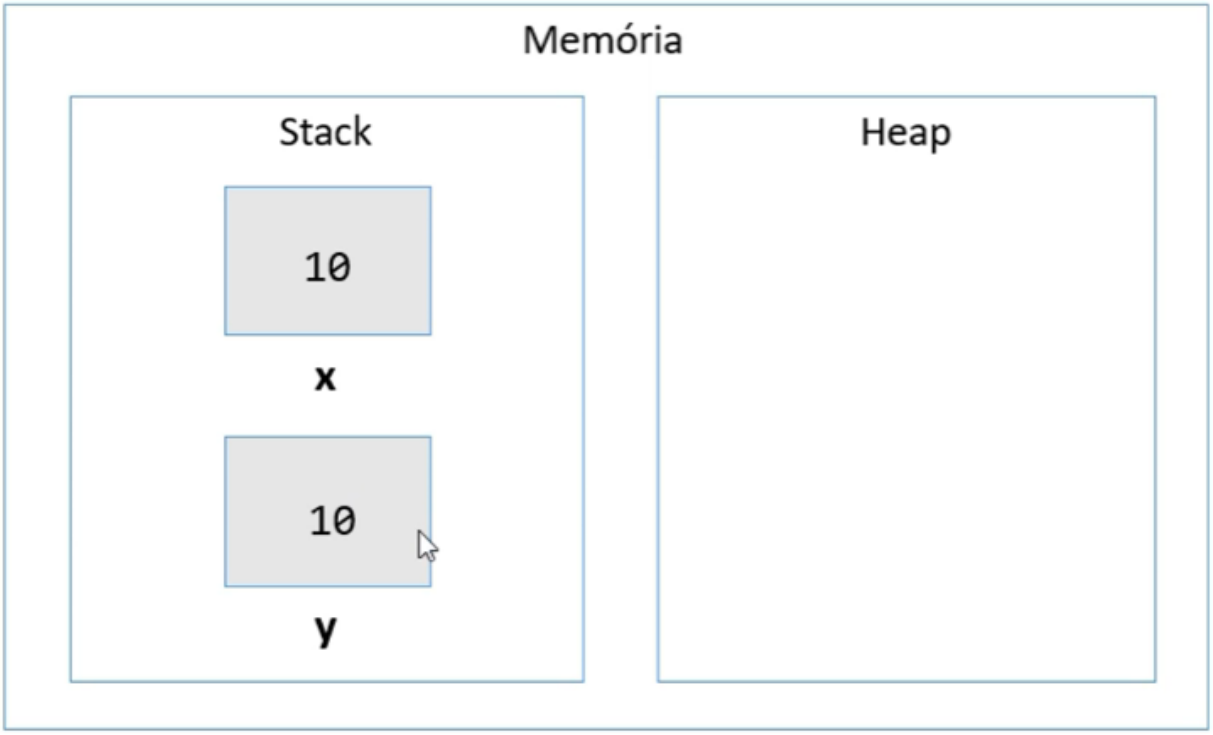
No C# os Value Types são:

```
Tipos Numéricos Integrais ->
sbyte
byte
short
ushort
int
uint
long
ulong

Tipos Numéricos de Ponto Flutuante -> float, double, decimal

Caracteres -> char

Booleanos -> bool

Structs -> struct

Enumeradores -> enum

Datas e Tempo -> DateTime, Timespan
```

Algumas características deles são:

- O espaço em memoria criado, armazena diretamente o dado que foi criado.
- A nossa variável acessa o dado criado diretamente. (não busca em outra parte da memoria)
- Se atribuirmos o valor de uma variável a outra, esse valor é copiado se tornando um novo valor independente.

```
double x = 10; // criado e alocado diretamente na stack
double y = x; // não é uma referência de x, e sim um novo valor independente
```



Mais detalhes sobre esse assunto

Gerenciamento de memória no C#: stack, heap, value-types e reference-types

Opa pessoal, tudo certinho? Neste post, eu vou abordar uma dúvida que aparece com uma certa frequência em nosso suporte: o que são, afinal de contas, as benditas memórias stack e heap? O que são afinal de contas os value-types e os reference-

https://www.treinaweb.com.br/blog/gerenciamento-de-memoria-no-c-stack-heap-value-types-e-reference-types