TIPAGEM DE DADOS.

SDC™

Sejam todos bem vindos a mais um episódio, aqui na sociedade do código. Como vocês estão? Tudo bem?

Na aula de hoje falaremos sobre tipagem de dados. Ao final desta aula você saberá porque algumas linguagens utilizam a tipagem e quais são seus benefícios, malefícios e como o C# trabalha com a tipagem de dados.

As linguagens de programação podem possuir tipagem forte, tipagem fraca, tipagem dinâmica ou estática

Para esta aula vamos nos ater mais a tipagem utilizada no C#, que é a tipagem forte.

Uma tipagem forte significa que devemos declarar o tipo da variável, que indica o valor que ela deverá armazenar.

Por exemplo:

Esse código aqui inicializa e declara as variáveis e seus respectivos valores.

```
string basicString = "Hello, world";

int basicInt = 192;

float basicFloat = 13.1f;

char basicChar = 'B';

bool basicBool = true;
```

Então se o C# é uma linguagem fortemente tipada, e você deve declarar qual tipo de valor as variáveis recebem, então quais são os tipos de dados do C#? Vamos ver agora em sequência.

Data Type	Tamanho	Descrição
int	4 bytes	-2,147,483,648 … 2,147,483,647
long	8 bytes	S-9,223,372,036,854,775,808 … 9,223,372,036,854,775,807
float	4 bytes	Números fracionados 6 a 7 dígitos decimais
double	8 bytes	Números fracionados 15 dígitos decimais
bool	1 bit	Valores true ou false
char	2 bytes	Caractere único, uma letra, inserido em aspas simples
string	2 bytes por caractere	Sequência de caracteres, inserido em aspas duplas

Essa tabela aqui mostra os principais tipos de dados e os valores que elas ocupam em memória.

Existem muitos outros tipos de dados, mas você raramente vai utilizá-los.

De qualquer forma, vou deixar uma referência a um artigo da Microsoft que fala mais a fundo sobre cada um dos tipos de dados.

Recapitulando, podemos ver na tabela o tamanho ocupado em memória por cada tipo de dado

Isso acontece, porque quando você cria uma variável de um certo tipo, o .NET sabe previamente quanto precisará alocar de memória para aquele tipo específico de variável.

Dessa forma você pode se atentar para qual tipo de dado utilizar ao programar, pensando em um maior aproveitamento de memória e rapidez de um sistema.

Hoje em dia essa diferença não é tão gritante, mas antigamente saber o tipo certo para ser utilizado na criação de um sistema poderia fazer uma diferença e tanto.

Agora vamos discutir mais sobre a descrição de cada tipo especificado pela tabela, começando com os tipos inteiros.

Os tipos inteiros, como o nome diz, armazenam números inteiros.

Se você lembrar da sua quarta série, vai lembrar que seu professor de matemática falava sobre números naturais, números inteiros, números reais.

Mas se você não lembrar - e não vou te julgar por isso - os números inteiros são aqueles que compreendem os números negativos e positivos, sem considerar os decimais, ou seja, aqueles que contêm “números depois do ponto”, ou uma fração de um valor.

Os valores do tipo inteiro, em C#, são int e long, e seu uso depende da quantidade numérica que você precisa.

Um ponto interessante para comentar aqui, é que esse Alias ou palavra-chave, int e long, são apenas conversões para os tipos reais que o C# trabalha. A palavra chave “int” no C# é na verdade o tipo .NET System.Int32.

E a palavra chave para o tipo long do C# é o System.Int64 do .NET.

É interessante saber qual o tipo equivalente ao .NET para que possamos fazer as importações corretas quando criando nossas aplicações.
Mas não saber o tipo .NET não é um impeditivo, ta? Você vai pegando isso conforme o tempo de experiência.

Seguindo para o segundo tipo de dado, o ponto flutuante, ele representa números que contém a parte fracional, ou seja, números após o ponto ou valores fracionados.

e seus tipos são float e double

E quando nós devemos utilizar float, e quando devemos utilizar o double? Isso depende do que precisamos como valor número.

o float tem uma precisão de 6 a 7 dígitos, enquanto o double possui uma precisão numérica de 15 dígitos.

Então se você estiver com dúvida entre qual tipo numérico utilizar, o double é mais seguro e garante mais dígitos à sua disposição.

Continuando com nossa exploração pelos tipos de dados no C#, temos os booleanos.

Esse tipo é muito utilizado na programação, porque ele define apenas dois valores, true e false, que são utilizados nas verificações condicionais.

Então quando você possuir uma estrutura lógica if, que veremos mais à frente, você estará efetuando a verificação condicional de um tipo de dado booleano.

O tipo de dado char é utilizado para armazenar um caractere único, que devem estar inseridos em aspas simples.

Já as strings podem armazenar uma sequência de caracteres, um texto, e devem estar inseridas em aspas duplas.

Você pode perceber que criamos variáveis com a nomenclatura que definimos, e o C# nos deixa livre para fazer isso, até um certo ponto.

O que nós não podemos fazer, é criar variáveis com nomes reservados pelo .NET

São palavras chave que estão reservadas pelo sistema, e criar variáveis novas com a mesma sintaxe pode ocasionar em erros no programa.

Palavras do tipo string, int, bool são palavras utilizadas pelo sistema, então devemos evitar usá-las, beleza?

Até agora o que nós vimos aqui faz parte de um grupo de dados do C# denominado Tipos de valor, mas ele não é o unico que existe.

Temos também os tipos referência.

A diferença entre esses dois tipos é o seguinte.

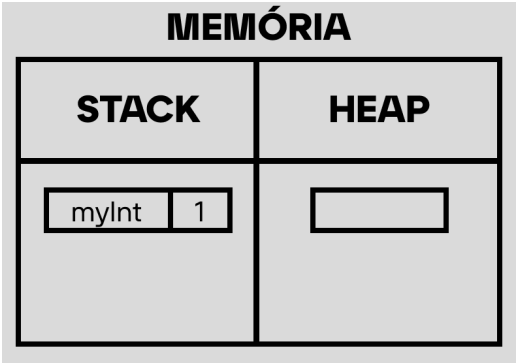
Os tipos de valor contém os dados dentro do próprio local de alocação de memória, enquanto os tipos referência conterão um ponteiro com uma referência para o local de memória que contém o dado real.

Como assim?

Pra te explicar sobre isso precisamos falar sobre memória heap e stack

A memória heap e stack existem na porção alocada pelo computador para executar o sistema, ambas dentro da memória RAM

Quando você cria uma variável do tipo int, que pertence ao tipo de dados de valor, é criado um local na memória stack com o nome que você deu à variável, e seu valor.

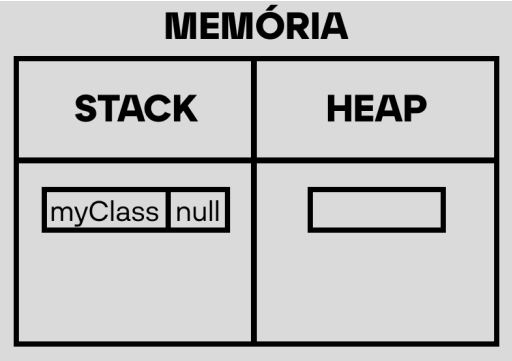


Já nas variáveis de tipo referência é criado um local na memória contendo uma referência para a realização da identificação pela heap.

Dessa forma, quando a gente instância a variável, a heap vai apontar para o valor criado na stack.

MyClass myCreatedClass;

myCreatedClass.Name;



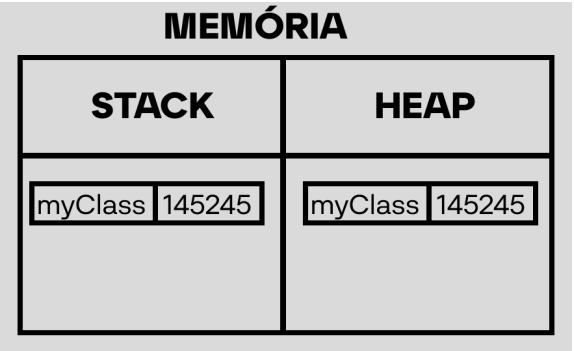
O Código acima nos dará um erro, porque quando criamos a variável ela não contém um endereço definido para seu local na Stack.

Dessa forma, nosso sistema não pode acessá-la por falta de referência.

Agora, se digitarmos

MyClass myCreatedClass = new MyClass();

Estamos criando uma instância da classe, e o sistema então cria o endereço da variável pelo qual a referência na memória heap deve procurar. Dessa forma, encontrando o valor através de uma referência para a Stack.



Vou deixar um artigo completo da própria microsoft explicando bem a fundo sobre como funcionam os tipos de valores no C# e é bem importante que você leia.

É uma leitura técnica então pode ser que você não entenda logo na primeira vez que ler, mas isso faz parte do processo de aprendizado.

Eu peço que você leia mesmo tudo que eu coloco de material pras nossas aulas, pq eles vao incrementar ainda mais seu aprendizado.

E eu não encaro esses materiais como conteúdo complementar. Eles são conteúdos obrigatórios.

Eles estão aqui, pra exercer um papel fundamental no entendimento mais profundo. Então nunca deixei de ler o que eu coloco por aqui, beleza?

Qualquer dúvida é só deixar nos comentários, que vamos procurar responder tudo que você precisa pra entender melhor.

Até a próxima aula. Fui!

OBRIGADO.

SDC™