Variáveis

Em qualquer linguagem de programação você precisará usar variáveis, que tem basicamente o objetivo de armazenar uma informação de forma temporária na memória RAM do computador.

Além desta informação precisamos saber que existem tipos diferentes de variáveis, basicamente quanto ao tipo de dado que podemos armazenar.

Em C# temos dois tipos principais básicos, são os tipos referência e valor.

Variáveis do tipo valor contém diretamente seus dados, já variáveis do tipo referência são os objetos, da programação orientada a objetos e contém somente uma referência a seus dados.

As variáveis do tipo valor são dividias em vários outros tipos, veja a seguir.

Inteiros com sinal

sbyte = 8 bits, entre -128 e 127

short = 16 bits, entre -32.768 e 32.767

int = 32 bits, entre -2.147.483.648 e 2.147.483.647

long: 64 bits, entre -9.223.372.036.854.775.808 e 9.223.372.036.854.775.807

Inteiros sem sinal

byte = 8 bits, entre 0 e 255

ushort = 16 bits, entre 0 e 65.535

uint = 32 bits, entre 0 e 4.294.967.295

ulong = 64 bits, entre 0 e 18.446.744.073.709.551.615

Ponto Flutuante

float = 32 bits, entre $1,5x10^{-45}$ e $3,4x10^{38}$ (7 dígitos de precisão)

double = 64 bits, entre $5,0x10^{-324}$ e $1,7x10^{308}$ (15 dígitos de precisão)

decimal = 128 bits, entre $-7,9x10^{-28}$ e $7,9x10^{28}$ (28 dígitos de precisão)

Char

char = 8 bits, pode representar até 256 caracteres distintos. Somente um caractere por variável.

Boolianos

bool = Armazena os valores true ou false (verdadeiro ou falso), diretamente relacionados com os valores 0 ou 1.

Enum

enum = Enumeradores, basicamente pode receber constantes nomeadas.

As variáveis do tipo referência

string = Basicamente uma variável do tipo string funciona armazenando uma cadeira de caracteres, para armazenar textos, basicamente uma sequência de caracteres do tipo char.

array = Este é um caso especial que iremos estudar a parte, basicamente representa um grupo de variáveis do mesmo tipo. Os arrays podem ser uni ou multidimensionais.

class = Estrutura de dados que pode conter outros dados chamados de propriedades e funções chamadas de métodos, são basicamente a definição dos objetos antes de serem instanciados.

struct = Semelhante ao tipo class, basicamente representa uma estrutura com dados e funções, a diferença para as classes é que a struct são tipos de valor e não precisam ser instanciados, assim, não precisam ser alocados no heap como os objetos, em contrapartida não podem usar herança pelo programador.

interface = Uma interface basicamente é uma referência para uma classe ou struct que a implemente. A interface fornece as regras que obrigatoriamente devem ser usadas por quem a implementar.

delegade = Referências para os métodos com uma lista de parâmetros de tipo e de retorno específicos. Possibilitam o tratamento de métodos como entidades que podem ser atribuídos a variáveis e passadas como parâmetros. Parecidos com os ponteiros de função em C.

null = Valor nulo.

Declaração de variáveis

Declarar uma variável é simples, basicamente você precisa indicar o tipo e o nome da variável, outro aspecto importante é o escopo, que especifica onde a variável será visível. Vamos entender melhor isto neste capítulo.

Vamos declarar três variáveis do tipo int, veja a seguir o código.

```
using System;
class Program{
     static void Main(){
        int num1;
        int num2,num3;
    }
}
```

JavaScript

```
<html>
<script>
       var num1;
       var num2,num3;
</script>
</html>
```

C++

```
#include <iostream>
using namespace std;
int main(){
       int num1;
       int num2,num3;
}
```

PHP

```
<?php
       $num1;
```

```
        $num2,$num3;
?>
```

Note que declaramos de duas formas diferentes, neste caso não existe uma regra sobre qual das formas está correta, na verdade as duas estão, se forem variáveis do mesmo tipo, você poderá declara-las todas na mesma linha, então podemos usar qualquer uma das formas a seguir.

int num1,num2,num3;

int num1;
int num2;
int num3;

int num1,num2;
int num3;

Todas são formas válidas, só depende de como você quer organizar seu código.

Escopo de variáveis

Quanto ao escopo podemos dizer que é o local onde a variável será válida e ou visível ao código. No caso acima as variáveis tem seu escopo definido ao método Main e não poderão ser operadas fora deste método. Isso significa que se houver outro método nesta classe "Program" ele não terá acesso às variáveis num1, num2 e num3.

Agora veja a situação a seguir.

```
using System;
class Program{
    int num1,num2,num3;
    static void Main(){

    }
}
```

Note que as variáveis foram declaras no escopo da classe Program, fora do método Main, isso indica que estas variáveis têm o escopo global à classe Program e podem ser acessadas por qualquer método da classe.

Atribuindo valores às variáveis

Para inserir um valor a uma variável usamos o operador de atribuição = (igual), basta indicar o nome da variável, o operador de atribuição e o valor que deseja armazenar na variável, o valor deve ser do tipo que a variável foi declara é claro.

```
using System;
class Program{
    static void Main(){
        int num1,num2;
        int num3=0;

        num1=10;
        num2=20;
        num3=num1+num2;

        Console.WriteLine(num3);
```

```
    }
}
```

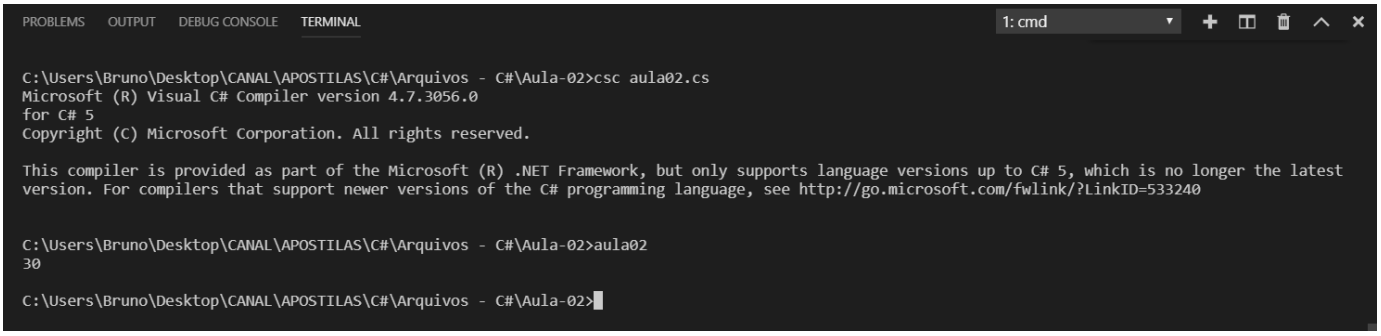
O programa acima mostra o uso do operador de atribuição em três situações diferentes, veja:

1) Primeiro usamos como inicialização da variável num3, onde na declaração da variável já inserimos o valor 0 na variável, assim, no ato da declaração a variável já possui um valor definido e não um “lixo” qualquer.

2) O segundo caso foi usado nas variáveis num1 e num2, onde inserimos os valores respectivos 10 e 20 nestas variáveis.

3) O terceiro caso foi usado para atribui o resultado de uma expressão, que no caso é a soma dos valores das variáveis num1 e num2, sendo assim, a variável num3 passa ter o valor 30.

Veja o resultado do programa que é a impressão do valor da variável num3.



Veja que no terminal, primeiro compilamos o programa com o comando csc aula02.cs e depois simplesmente rodamos o programa digitando o nome do executável gerado.