Introdução à Programação C# - Entendendo Variáveis e Tipos de Dados

O Que São Variáveis?

Ao construir um programa, é fundamental trabalhar com informações na memória do computador. Essas informações podem ser o nome de um jogador, seu nível, o mapa em que está jogando, ou os itens que possui. Para que o programa possa manipular esses dados, eles precisam ser armazenados na memória RAM do computador.

É aqui que entram as **variáveis**. Uma variável é, essencialmente, um espaço que você reserva na memória do computador para armazenar alguma informação. Imagine que você está pedindo ao computador para "guardar um lugar" para algo que você precisará depois.

Características importantes de uma variável:

Ela pode guardar um tipo específico de informação, como um número, um texto, ou um valor verdadeiro/falso.

Para criar uma variável, você precisa definir duas coisas:

- **Qual é o tipo de informação** que ela vai guardar (é um número? Um texto? Um booleano?).
- **Um nome (identificador)** para a variável, para que você possa acessá-la e utilizá-la posteriormente no seu programa. Esse nome permite que você saiba exatamente onde na memória está a informação que você precisa, especialmente quando você tem gigabytes de memória.

Ter essas informações mapeadas na memória é essencial para que seu programa funcione. Por exemplo, se você quiser exibir o nome do personagem na tela, você sabe exatamente qual variável usar para obter essa informação. Se quiser aumentar o nível de um personagem, você sabe qual variável alterar.

Tipos de Dados Mais Comuns em C#

Existem diversos tipos de dados na linguagem C#, mas alguns são mais frequentemente usados, especialmente para iniciantes. Os quatro tipos mais comuns apresentados são:

string

Usado para armazenar **textos**.

- **Exemplo:** string nomePersonagem;
- **Atribuição de valor:** Valores string são sempre colocados entre aspas duplas. Por exemplo: nomePersonagem = "Mustache".

int

Usado para armazenar **números inteiros** (sem casas decimais).

- **Exemplo:** int nivel;
- **Atribuição de valor:** nivel = 1;
- **Limite:** Uma variável int (int32) pode armazenar números entre aproximadamente -2 bilhões e +2 bilhões. Para a maioria dos casos, int é mais do que suficiente.

float

Usado para armazenar **números com casas decimais**.

- **Exemplo:** float altura;
- **Atribuição de valor:** Use ponto (.) como separador decimal e adicione um f ou F no final do número para indicar que é um float. Por exemplo: altura = 1.8f;.

bool (ou boolean)

Usado para armazenar valores **verdadeiro (true) ou falso (false)**. É ideal para representar estados como "está vivo" ou "está logado".

- **Exemplo:** bool estaVivo;
- **Atribuição de valor:** estaVivo = true; ou estaVivo = false;.

Atribuindo e Modificando Valores em Variáveis

Após criar uma variável (reservar o espaço na memória), você pode atribuir um valor a ela usando o operador de igualdade (=).

> **Sintaxe para criação e atribuição inicial:** TipoDaVariavel nomeDaVariavel = ValorInicial;

Exemplos de Atribuição

```
string nomePersonagem = "Mustache";
int nivel = 1;
float altura = 1.8f;
bool estaVivo = true;
```

Você pode mudar o valor de uma variável a qualquer momento durante a execução do programa, simplesmente atribuindo um novo valor a ela. Isso permite que seu programa acompanhe e reflita as mudanças no estado do jogo ou sistema.

Convenções de Nomenclatura para Variáveis

Embora não seja obrigatório, seguir convenções de nomenclatura torna o código mais legível e padronizado, tanto para você quanto para outros desenvolvedores.

Boas práticas sugeridas:

- **CamelCase:** Comece o nome da variável com letra minúscula e, se o nome for composto por múltiplas palavras, comece cada palavra subsequente com letra maiúscula (ex: nomePersonagem, estaVivo).
- **Evite Acentos e Caracteres Especiais:** Não é comum usar acentos, "til" ou "cedilha" em nomes de variáveis em C#. A linguagem e suas palavras-chave geralmente derivam do inglês, e a padronização segue essa base.

Outros Tipos de Dados e Otimização de Memória

Além dos tipos comuns, C# oferece outros tipos de dados para necessidades mais específicas.

Números Decimais

double e **decimal**. Eles podem ser mais precisos ou guardar uma maior quantidade de casas decimais do que float, mas geralmente ocupam mais espaço na memória.

Números Inteiros

- **long** pode armazenar números inteiros maiores que int, ocupando mais memória.
- **byte** pode armazenar números inteiros muito menores (de 0 a 255), ocupando menos memória.

Considerações sobre o tamanho da variável

Cada tipo de variável que você cria tem um tamanho específico na memória. Por exemplo, um int geralmente ocupa 32 bits (4 bytes). A escolha do tipo de variável depende muito da informação que você deseja armazenar e do seu tamanho. Se você precisar guardar um número muito grande, um int pode não ser suficiente.

> **Otimização para Iniciantes:**
>
> No início da sua jornada como desenvolvedor, focar em economizar espaço na memória pode desviar muito do seu foco no desenvolvimento da lógica do programa. Para pequenas aplicações e em computadores modernos com gigabytes de memória RAM, a diferença no uso de memória de uma única variável (como um int que ocupa 4 bytes) é praticamente imperceptível e não causará problemas de performance. Concentre-se nos tipos comuns (string, int, float, bool) e, com o tempo, conforme sua carreira evolui, você poderá se aprofundar nas otimizações mais extremas.

Interagindo com o Usuário: Entrada e Saída de Dados

Um programa interativo muitas vezes precisa exibir informações para o usuário e receber informações dele.

Exibindo e Lendo Dados no Console

Exibindo Mensagens no Console (Console.WriteLine)

Para escrever mensagens na tela (no console), você usa o método **Console.WriteLine()**.

- **Sintaxe:** Console.WriteLine("Seu texto aqui");
- Se for um texto, ele deve estar entre aspas duplas.
- Você também pode escrever números diretamente (ex: **Console.WriteLine(1);**).

Lendo Informações do Usuário no Console (Console.ReadLine)

Para ler o que o usuário digita no console, você usa o método **Console.ReadLine()**.

- **Comportamento**: Quando este comando é executado, o programa pausa e espera o usuário digitar uma linha de caracteres e pressionar Enter.
- **Retorno**: O método **Console.ReadLine()** retorna o texto digitado pelo usuário como um valor do tipo string.

Armazenando a Entrada do Usuário em uma Variável

Você pode pegar o texto retornado por **Console.ReadLine()** e armazená-lo diretamente em uma variável:

```
// Pede o nome ao usuário
Console.WriteLine("Qual será o nome do seu personagem?");
// Lê o que o usuário digita e guarda na variável
string nomePersonagem = Console.ReadLine();
```

Nesse exemplo, a variável nomePersonagem será criada e, ao mesmo tempo, receberá o texto que o usuário digitar.

Exibindo o Conteúdo de uma Variável

Para mostrar o valor armazenado em uma variável junto com um texto, você pode usar duas formas principais com **Console.WriteLine()**:

Concatenação de Strings com +

Você junta o texto fixo com o conteúdo da variável usando o operador +.

```
// Se nomePersonagem for "Mustache", a saída será "Olá, Mustache"
Console.WriteLine("Olá, " + nomePersonagem);
```

Interpolação de Strings com $ e {}

Esta forma é mais moderna e legível. Você coloca um cifrão (**$**) antes da string entre aspas e insere o nome da variável entre chaves **{}** dentro do texto. O C# processará o conteúdo da variável e o inserirá no texto.

```
// Se nomePersonagem for "Mustache", a saída será "Olá, Mustache"
Console.WriteLine($"Olá, {nomePersonagem}");
```

Ambas as formas alcançam o mesmo resultado; a escolha depende da preferência do programador.

Conclusão

Neste material, exploramos o conceito fundamental de variáveis em programação C#, entendendo que são espaços reservados na memória para armazenar informações. Vimos os tipos de dados mais comuns (**string**, **int**, **float**, **bool**), como atribuir valores a eles, e as convenções de nomenclatura. Além disso, aprendemos a interagir com o usuário, exibindo mensagens e lendo entradas através do console, e como manipular essas informações em variáveis. Esses conceitos são a base para o desenvolvimento de programas mais complexos e interativos.

Conceitos Fundamentais Aprendidos

- Variáveis como espaços na memória
- Tipos de dados básicos (**string**, **int**, **float**, **bool**)
- Atribuição e modificação de valores
- Convenções de nomenclatura
- Interação com o usuário via console