

VIANNA JUNIOR
INSTITUTO



Algoritmos

Tipos de Dados, Variáveis e
Entrada e Saída

Professor: Camillo Falcão

- **Sistemas de Numeração:** base, notação posicional e conversão de base.
- **Sistemas Computacionais:** processador, memória, linguagens de programação e sistema operacional.
- **Algoritmos:** ação, estado, processo, padrão de comportamento, algoritmo, aspectos estático e dinâmico de um algoritmo e formas de representação de algoritmos.

- Nesta aula, serão construídos programas de computador muito simples.
- Por enquanto, vamos assumir que todo programa tem a seguinte estrutura básica:

```
using System;

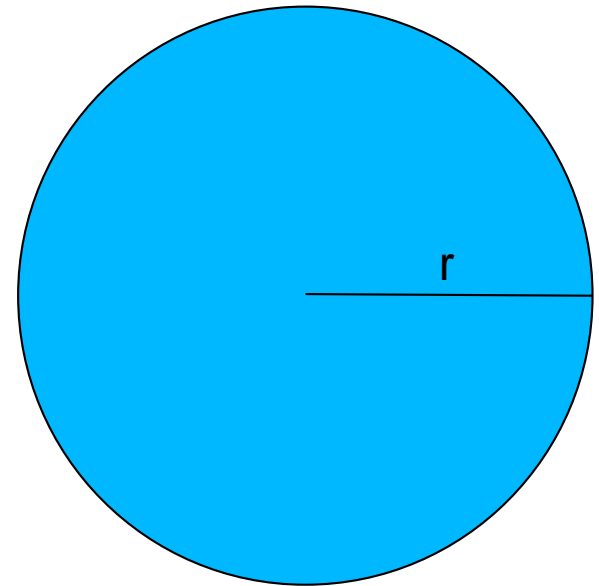
namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {

        }
    }
}
```

(quando estudarmos funções, alguns dos conceitos que definem esta estrutura serão vistos)

- Um programa de computador utiliza diversos dados durante seu processamento.
- Exemplo:
 - Imagine um programa que calcule a área de um círculo.

$$\text{área} = \pi \cdot r^2$$



O programa para calcular a área de um círculo utiliza ao menos os seguintes dados:

- Raio (número real): representa a medida do raio do círculo e seu valor pode **variar** dependendo do tamanho do círculo.
- Pi (número real): representa a **constante** numérica 3,14159... Apresenta sempre o mesmo valor, independente do círculo.
- Área (número real): representa a área de um círculo. Seu valor pode **variar** dependendo do tamanho do círculo.

Assim, um valor, em um programa, pode ser classificado como:

- **Constante:** dado cujo valor se manterá inalterado toda vez que o programa for utilizado.
- **Variável:** dado cujo valor pode ser modificado a cada execução ou, até mesmo, durante a execução do programa.

- Uma constante pode ser representada no texto diretamente pelo seu valor.

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a;
            float b;
            float c;
            a = 0;
            b = 2.5;
            c = 8.7 * b;
        }
    }
}
```


- Uma **variável** armazena um **valor** de determinado **tipo** que pode variar ao longo da execução do programa.
- Para cada variável, é reservado um espaço na memória do computador para armazenar seu valor.

- **Exemplo**: para armazenar um número inteiro, o programa normalmente reserva 4 bytes de memória.

O número binário armazenado nestes 4 bytes representa o valor da variável (neste caso, 10).

	0	1	2	3	4	5	6	7
...	0	1	1	1	0	0	1	0
1712	1	1	0	0	1	1	1	1
1713	0	0	0	0	0	0	0	0
1714	0	0	0	0	0	0	0	0
1715	0	0	0	0	0	0	0	0
1716	0	0	0	0	1	0	1	0
1717	1	1	1	1	1	0	1	1
...	1	0	0	1	0	1	0	0

10

- No texto de um programa, uma variável é representada por um identificador único.

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a;
            float b;
            float c;
            a = 0;
            b = 2.5;
            c = 8.7 * b;
        }
    }
}
```

	0	1	2	3	4	5	6	7
...	0	1	1	1	0	0	1	0
1712	1	1	0	0	1	1	1	1
a {	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	1	0
1717	1	1	1	1	1	0	1	1
...	1	0	0	1	0	1	0	0

11

- O valor da variável pode ser alterado ao longo do programa, mas seu nome permanece o mesmo.

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a;
            float b;
            float c;
            a = 0;
            b = 2.5;
            c = 8.7 * b;
        }
    }
}
```

	0	1	2	3	4	5	6	7
...	0	1	1	1	0	0	1	0
1712	1	1	0	0	1	1	1	1
a	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	1	0
1717	1	1	1	1	1	0	1	1
...	1	0	0	1	0	1	0	0

Identificador da variável:

- Nome único criado pelo programador
- Não pode ser uma palavra reservada da linguagem C#
 - (Exemplos: `int`, `void`, `if`, `return`, ...)
- Pode conter apenas **letras**, **dígitos** e **sublinha**, pode conter acentos (acentos não são recomendados por dificultarem a digitação do código).
- Deve começar com uma **letra** ou com **sublinha** (por padrão utilizam-se letras minúsculas)
- Deve permitir a identificação do valor que representa (ex: `raio`, `area`, etc)

Exemplos

- Válidos:
 - **nome**
 - **x1**
 - **nota_01**
 - **telefone**
 - **salario_base**
 - **nota2aProva**
- Inválidos/não recomendados:
 - **1ano**
 - **salário**
 - **valor-1**
 - **endereço**
 - **salario/hora**
 - **2aProva**

Tipos de dados em C#

- Toda constante e toda variável de um programa tem um tipo de dados associado.
- **int** → utilizado para representar um número inteiro.
 - Exemplo: 1, -5, 1024 ,etc.
- **float** ou **double** → utilizados para representar números reais (ponto flutuante).
 - Exemplo: -1.0, 3.14159, 2.718281
- **bool** → utilizado para representar um valor lógico:
 - Verdadeiro (true) ou falso (false).
- **char** → utilizado para representar um único caractere (letra, dígito, símbolo, ponto, etc).
 - Exemplo: 'a', '5', '@', '!', etc.
- **string** → utilizado para representar uma cadeia de caracteres.
 - Exemplo: "Algoritmo", "Olá mundo!", etc.

- Atenção

- Para valores dos tipos **float** ou **double**, o separador decimal é o ponto.

3.14159
↑

- Constantes do tipo de dados **char** sempre aparece entre aspas simples.

'a'

- Constantes do tipo de dados **string** sempre aparece entre aspas duplas.

"C# é legal!"

Exercício

1) Indique quais das constantes abaixo são do tipo **int**:

- | | | |
|-------------------------------|------------------------------|---------------------------------|
| <input type="checkbox"/> 1000 | <input type="checkbox"/> '8' | <input type="checkbox"/> "-900" |
| <input type="checkbox"/> -456 | <input type="checkbox"/> 34 | <input type="checkbox"/> -1.56 |

2) Indique quais das constantes abaixo são do tipo **float**:

- | | | |
|---------------------------------|----------------------------------|----------------------------------|
| <input type="checkbox"/> -678.0 | <input type="checkbox"/> "0.87" | <input type="checkbox"/> "-9.12" |
| <input type="checkbox"/> -456.0 | <input type="checkbox"/> "Cinco" | <input type="checkbox"/> -1.56 |

3) Indique quais das constantes abaixo são do tipo **char**:

- | | | |
|------------------------------|---------------------------------|------------------------------|
| <input type="checkbox"/> 'z' | <input type="checkbox"/> "onze" | <input type="checkbox"/> d |
| <input type="checkbox"/> 45 | <input type="checkbox"/> '8' | <input type="checkbox"/> 'F' |

4) Indique quais das constantes abaixo são do tipo **bool**:

- | | | |
|---------------------------------------|-----------------------------------|-------------------------------------|
| <input type="checkbox"/> "verdadeiro" | <input type="checkbox"/> " Falso" | <input type="checkbox"/> verdadeiro |
| <input type="checkbox"/> true | <input type="checkbox"/> "true" | <input type="checkbox"/> false |

Declaração de variáveis

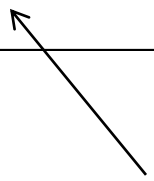
- A declaração de uma variável é o momento em que esta é criada no programa.
- Para criar uma variável, é necessário indicar:
 - o *tipo* da variável
 - o *identificador* da variável

```
int idade;  
float peso;  
float altura;  
char sexo;
```

Declaração de variáveis

- Como todo comando simples em C#, a declaração termina com um ponto e vírgula.

```
int idade;  
float peso;  
float altura;  
char sexo;
```



Declaração de variáveis

Implicações da declaração de variáveis:

- É alocado um espaço na memória que possa conter qualquer valor do tipo especificado.

int → 4 bytes

float → 4 bytes

char → 2 bytes

double → 8 bytes

- O nome da variável é associado ao endereço de memória reservado.

Toda vez que a variável for referenciada, o computador vai trabalhar com o conteúdo deste endereço de memória.

Declaração de variáveis

Observações importantes:

- Durante todo o programa, a variável armazenará apenas valores do tipo especificado na sua declaração.
- Uma variável só pode ser utilizada em um programa após sua declaração. Por isso, as declarações de variáveis são realizadas no início do programa.

Indique as opções com declarações válidas:

- char endereco-residencial;
- int valor1;
- float 1aArea;
- int 21;
- char char;

Operador de atribuição

Um comando de atribuição é a forma mais básica de modificar o valor de uma variável:

```
nomeVar = expressao;
```

nomeVar

- identificador da variável que será modificada
- apenas **uma** variável pode ser modificada por vez
- o nome da variável fica **sempre no lado esquerdo** do operador de atribuição

Operador de atribuição

Um comando de atribuição é a forma mais básica de modificar o valor de uma variável:

```
nomeVar = expressao;
```

operador =

- operador de atribuição
- para não confundir com o operador de comparação, evite ler **var=10**; como “*var é igual a 10*”; normalmente, lê-se “*var **recebe** 10*”

Operador de atribuição

Um comando de atribuição é a forma mais básica de modificar o valor de uma variável:

```
nomeVar = expressao;
```

expressao

- expressão cujo resultado será armazenado na variável
- pode ser composta por um valor constante, uma outra variável ou uma expressão (matemática ou lógica) que utilize constantes e variáveis, etc
- fica **sempre do lado direito** do operador de atribuição

Operador de atribuição

Um comando de atribuição é a forma mais básica de modificar o valor de uma variável:

```
nomeVar = expressao;
```

; (ponto e vírgula)

- como todo comando simples em C#, o comando de atribuição é finalizado com um ponto e vírgula.

Exemplos:

```
raio = 2.5;  
area = 3.14159 * (raio * raio);  
raio2 = raio;  
sexo = 'F';  
delta = (b * b) - 4 * a * c;  
digito = '5';
```

- Em C#, é possível inicializar uma variável em sua declaração:

```
int a = 10;  
char setor = '1';  
float elem1 = 7.0;  
int x = 20;
```

Combinação de variáveis, constantes e operadores que, quando avaliada, resulta em um valor.

–*Expressão aritmética:*

resulta em um número (inteiro ou real).

–*Expressão lógica:*

resulta em VERDADEIRO ou FALSO.

Expressões aritméticas

```
10  
a + 15  
base * altura  
90 / 4.0  
3.065  
189 % divisor  
(x1 - 5) * x2
```

Variáveis:

a, base, altura,
divisor, x1, x2

Constantes:

10, 15, 90, 4.0,
3.065, 189, 5

Operadores:

+, *, /, %, -

Operadores → Para $a = 5$ e $b = 5.0$:

			INTEIROS	REAIS
UNÁRIO	-	sinal negativo	-2 $-a \rightarrow -5$	-2.0 $-b \rightarrow -5.0$
BINÁRIOS	+	adição	$a + 2 \rightarrow 7$	$b + 2.0 \rightarrow 7.0$
	-	subtração	$a - 2 \rightarrow 3$	$b - 2.0 \rightarrow 3.0$
	*	multiplicação	$a * 2 \rightarrow 10$	$b * 2.0 \rightarrow 10.0$
	/	divisão	$a / 2 \rightarrow 2$	$b / 2.0 \rightarrow 2.5$
	%	módulo (resto da divisão)	$a \% 2 \rightarrow 1$	(operação não definida para reais)

- Operações aritméticas mais complexas:

```
Math.Pow (base, 2)  
Math.Sqrt (16)  
Math.Sin (x)  
Math.Cos (x)  
Math.Cos (2 * x)  
Math.Sin (x) * Math.Cos (y)  
Math.Abs (-5)
```

- Veremos mais detalhes sobre funções adiante.

- Envolvem os operadores:

–Relacionais:

igual (`==`), diferente (`!=`),
menor que (`<`), menor ou igual a (`<=`),
maior que (`>`), maior ou igual a (`>=`)

–Lógicos:

negação (`!`), conjunção (`&&`), disjunção (`||`)

- Sempre resultam em VERDADEIRO ou FALSO.
- Serão abordadas mais detalhadamente na introdução de **estruturas condicionais**.

Prioridade para execução de operações em uma expressão:

1. Parênteses (dos mais internos para os mais externos);
2. Expressões aritméticas, seguindo a ordem: funções, multiplicação e divisão, adição e subtração;
3. Expressões lógicas relacionais: $<$, $<=$, $=$, $>$, $>=$ e $!=$;
4. Expressões lógicas, seguindo a ordem: negação, conjunção, disjunção;
5. Da esquerda para a direita, quando houver indeterminações.

Na dúvida, use parênteses em suas expressões.

O procedimento **WriteLine** da classe **Console** escreve um texto no dispositivo de saída padrão do computador (isto é, na tela do computador).

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Olá mundo!");
            Console.ReadKey();
        }
    }
}
```

O procedimento **WriteLine** da classe **Console** também pode ser usado para imprimir o valor de uma variável ou expressão.

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            int x;

            x = 100;

            Console.WriteLine(x);
            Console.ReadKey();
        }
    }
}
```

É possível utilizar o procedimento **WriteLine** da classe `console` para imprimir mais de uma variável ou constante.

- Primeiramente é necessário fornecer um texto contendo as indicações sobre a posição de cada variável ou constante que se deseja imprimir.
- Após isso é necessário informar cada variável que se deseja imprimir. A primeira variável passada será impressa no lugar do texto {0}, a segunda no lugar do {1}, a terceira no lugar do {2} e assim sucessivamente.

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            int x;

            x = 100;

            Console.WriteLine("O dobro de {0} é {1}.", x, 2 * x);
            Console.ReadKey();
        }
    }
}
```

- A função **ReadLine** da classe Console retorna dados informados pelo usuário e esse dado retornado pode ser atribuído a uma variável.

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            string nome;

            Console.Write("Informe o seu nome: ");

            nome = Console.ReadLine();

            Console.WriteLine("Olá {0}!", nome);
            Console.ReadKey();
        }
    }
}
```


- A leitura de dados utilizando a função **ReadLine** da classe **Console** é uma forma alternativa de inicialização de variáveis.
- O retorno dessa função é uma string. Para fazer a leitura de um dado do tipo **int**, **float** ou **double** é necessário converter a string recebida para o tipo de dados desejado. Para isso, utilize:
 - `int NOME_VARIAVEL = Convert.ToInt32(Console.ReadLine());`
 - `float NOME_VARIAVEL = Convert.ToSingle(Console.ReadLine());`
 - `double NOME_VARIAVEL = Convert.ToDouble(Console.ReadLine());`

- Exemplo de leitura de dado e sua respectiva conversão para variável do tipo double:

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            double raio;
            double area;

            Console.Write("Informe o raio do círculo: ");
            raio = Convert.ToDouble(Console.ReadLine());

            area = 3.14159 * raio * raio;

            Console.WriteLine("Área do círculo: {0}", area);
            Console.ReadKey();
        }
    }
}
```

- Textos explicativos ao longo do programa
- Ignorados pelo compilador
- Ajudam outros programadores a entender determinado programa
- Duas alternativas: `//` ou `/* ... */`

```
...  
  
/*  
Programa que calcula raízes de  
uma equacao de 2o grau  
*/  
public static void Main(string[] args)  
{  
    double a, b, c; //coeficientes  
    double r1, r2; //raízes da equação  
  
    ...  
}
```

Estruturas de controle

- As estruturas básicas de controle definem a sequência em que os passos de um algoritmo são realizados ao longo da execução do programa.
- São elas:
 - Sequência simples
 - Condicionais ou alternativas *
 - Repetições *

—* serão vistas nas próximas aulas

Sequência simples:

–*Sintaxe*

– Sequência de comandos sem limite de tamanho. Esta sequência pode incluir declarações de variáveis, atribuições, leitura e impressão de dados.

–*Semântica*

– A execução é iniciada no primeiro comando da estrutura, continua executando os comandos na ordem em que aparecem, de cima para baixo, e sai da estrutura.

```
Comando1 ;  
Comando2 ;  
...  
ComandoN ;
```

Até agora, foram estudados os seguintes recursos visando especificamente a construção de algoritmos:

- Conceito de Constante, Variável e Tipos de Dados.
- Tipos de Dados Básicos.
- Declaração de Variáveis.
- Expressões (Aritméticas e Lógicas).
- Sintaxe e Semântica dos Comandos de Atribuição e de Entrada e Saída.
- Seqüência Simples (primeiro dos três tipos de estruturas de controle).

Já é possível criar alguns algoritmos simples.

1. Faça um programa que lê uma temperatura em graus Celsius e apresenta-a convertida em graus Fahrenheit. A fórmula de conversão: $F \leftarrow (9 * C + 160) / 5$
2. Faça um programa que lê um valor de salário mínimo e o salário de um funcionário. O programa deve calcular e imprimir quantos salários mínimos esse funcionário ganha.
3. Faça um programa que leia os valores dos lados e altura de um triângulo, calcule e imprima seu perímetro e área.

4. Faça um programa que leia um número inteiro e imprima o seu antecessor e sucessor.
5. Construa um programa que aplique um desconto de 25% sobre o preço de um produto recebido como entrada e imprima o valor resultante.
6. Construa um programa para ler um intervalo de tempo em segundos, converter para horas, minutos e segundos e imprimir o resultado.

Tipos de Dados, Variáveis e Entrada e Saída em C#

Aula de Exercícios

- Variáveis, Tipos de dados
- Declaração de variáveis
- Impressão
- Formatação da impressão
- Operadores
- Entrada de dados

- Uma variável representa um espaço na memória do computador para armazenar um determinado tipo de dado.
- Em C#, todas variáveis devem ser explicitamente declaradas, isto é, devemos especificar:
 - Tipo de dado
 - Nome (ou identificador)

Identificadores

- São nomes usados para se fazer referência a variáveis, funções, rótulos e vários outros objetos definidos pelo usuário.
- Exemplos:
 - a, b, peso, i, contaCorrente, saldo, x1, x2.....
- **Obs:** A linguagem C# é ***case sensitive***, ou seja, as letras maiúsculas diferem das minúsculas.

Nomes de Variáveis

O nome de uma variável (identificador) é formado por um ou mais caracteres, sendo que:

- o primeiro caractere é uma letra ou sublinha (_);
- os outros caracteres podem ser letras, algarismos ou sublinha (_).

Exemplos:

- a) Nomes válidos: **L a de op1 V9a Lista_Notas a_2 p56 A1 _2A**
- b) Nomes Inválidos: **x+ t.6 43x &ah \$dolar Lista-Notas n! %p**

Em C#, letras maiúsculas e minúsculas são diferentes. Os seguintes nomes são distintos:

- **PESO**
- **Peso**
- **peso**
- **peSo**

Uma variável não pode ter o mesmo nome de uma palavra-chave de C#, como por exemplo:

- **main, cout, int, float, char, short, return, case, void**

Tipos de Dados

Quando você declara um **identificador** dá a ele um tipo. Um tipo de objeto de dados determina como valores de dados são representados, que valores pode expressar, e que tipo de operações você pode executar com estes valores.

Tipo	Espaço que ocupa na memória	Faixa
<code>char</code>	1 byte	<i>-128 a 127</i> (incluindo letras e símbolos)
<code>int</code>	4 bytes	-2147483648 a 2147483647
<code>float</code>	4 bytes	<i>3.4E-38 a 3.4E+38</i> (6 casas de precisão)
<code>double</code>	8 bytes	<i>1.7E-308 a 1.7E+308</i> (15 casas de precisão)

Declaração de variáveis

- Para armazenar valores na memória é preciso reservar o espaço correspondente ao tipo de dado.
- A declaração de uma variável reserva espaço para armazenar um dado do tipo da variável e associa o nome da variável a este espaço.
- Na linguagem C# podemos inicializar as variáveis com valores na declaração.

Declaração de Variáveis

- As variáveis só podem armazenar informações ou dados sempre de um mesmo tipo (inteiro, real, caractere, char, etc).
- Na linguagem C#, a declaração de variáveis obedece a seguinte sintaxe:

<tipo> <nome_var>;

ou

<tipo> <nome_var1>, <nome_var2>, ,<nome_varn>;

- Toda e qualquer variável deve ser declarada e sua declaração deve ser feita antes de sua utilização no programa.

Exemplos:

```
int a, b, c;  
char letra, d, e;  
float f1;  
float f2, f3;
```

Marque as declarações válidas:

```
( ) int a,  
( ) char c;  
( ) int a,b,a;  
( ) float f1,f2,f3,4f;  
( ) int meu_nro;  
( ) float leitura_sensor;
```

Declaração de variáveis - Exemplo

```
int a;  
int b;  
float x;  
char c;  
a=25;  
b=50;  
x=3.14;  
c='M';
```

```
int n1, n2;  
n1=0;  
n2=0;  
int val; val=2.5; // armazena 2  
int y=5, z=15;  
float e=2.718;  
char s='f';
```

Declaração de variáveis - Exemplo

- Erro comum: uso de variáveis com valor não inicializado.
- Declarar variável sem explicitamente inicializar seu valor.
- Valor indefinido

```
// Erro !!!  
// Variável y não foi  
// inicializada e  
// contém "lixo" de  
// memória
```

```
float x, y, z;  
x = 1.0;  
  
z = x + y;
```

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            int x;

            x = 100;

            Console.WriteLine("O dobro de {0} é {1}.", x, 2 * x);
            Console.ReadKey();
        }
    }
}
```

Impressão de Códigos Especiais

Código	Ação
<code>\n</code>	leva o cursor para a próxima linha
<code>\t</code>	executa uma tabulação
<code>\b</code>	executa um retrocesso
<code>\f</code>	leva o cursor para a próxima página
<code>\a</code>	emite um sinal sonoro (<i>beep</i>)
<code>\"</code>	exibe o caractere "
<code>\\</code>	exibe o caractere \
<code>%%</code>	exibe o caractere %

```
#include <stdio.h>
int main()
{
    printf("\t\t\t\t\t");
    printf("\t\t\t\t\t");
    printf("\t\t\t\t\t");
    return 0;
}
```

Fixando as Casas Decimais

- Por padrão, o compilador C# exibe os números de *ponto flutuante* com seis casas decimais.
- Para alterar este número podemos acrescentar **:Nx** ao código de formatação da saída, sendo x o número de casas decimais pretendido. Ao exibir um número com uma quantidade de decimais inferior à necessária para sua representação, esse número será arredondado.

```
using System;
namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("{0:N1} ", 3.14159); //Imprime 3,1
            Console.WriteLine("{0:N2} ", 3.14159); //Imprime 3,14
            Console.WriteLine("{0:N3} ", 3.14159); //Imprime 3,142
            Console.WriteLine("{0:N4} ", 3.14159); //Imprime 3,1416
            Console.ReadKey();
        }
    }
}
```

Alinhamento de Saída

- O programa pode fixar a coluna da tela a partir da qual o conteúdo de uma variável, ou o valor de uma constante será exibido.
- Isto é obtido acrescentando-se um inteiro **m** ao código de formatação. Neste caso, *m* indicará o número de colunas que serão utilizadas para exibição do conteúdo.

```
using System;
namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("{0,3},{1,3}", 10, 100); //Imprime 10,100
            Console.WriteLine("{0,4},{1,4}", 10, 100); //Imprime 10, 100
            Console.ReadKey();
        }
    }
}
```

1. Fazer um programa para imprimir na primeira linha o dia, na segunda linha o mês e na terceira o ano de seu nascimento.
2. Imprimir o valor 2.346728 com 1, 2, 3 e 5 casas decimais.
3. Fazer um programa para ler o valor de um produto (em reais) e o valor de um desconto (em reais). Imprima o valor do produto após a aplicação do desconto. Dica: declare duas variáveis: `valorProduto` e `valorDesconto`, sendo ambas do tipo `double`. Exemplo de leitura da variável `valorProduto`:

```
valorProduto = Convert.ToDouble(Console.ReadLine())
```


Operador de Atribuição

O operador de atribuição em C# é o sinal de igual (=)

Sintaxe:

<variavel> = <expressão>;

Exemplos:

```
int a,b,c,d;  
  
a=5;  
c=7;  
b = a;  
d = a + b + c;
```

ou

```
int a=5;  
int c=7;  
int b,d;  
b = a;  
d = a + b + c;
```

Operadores Aritméticos

Exemplos:

```
a = a + b;  
a += 4;           // similar a      a = a + 4;  
a = b / 2;  
a = 4 * 2 + 3;    // qual o valor final de a?  
b = 2 * 3 - 2 * 2; // qual o valor final de b?  
a++;              // similar a      a = a + 1;  
b--;              // similar a      b = b - 1;
```

Conversão de tipo

- Conversões automáticas de valores na avaliação de uma expressão.

```
int a;  
float b, c, d;  
a = 4.5;           // conversao implicita  
b = a / 2.0;       // conversao implicita  
c = 1/2 + b;       // conversao implicita  
d = 1.0/2.0 + b;
```

- Saída: a=4, b=2.0, c=2.0, d=2.5

Leitura de dados e conversão explícita

- Exemplo de leitura de dado e sua respectiva conversão para variável do tipo double:

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            double raio;
            double area;

            Console.Write("Informe o raio do círculo: ");
            raio = Convert.ToDouble(Console.ReadLine());

            area = 3.14159 * raio * raio;

            Console.WriteLine("Área do círculo: {0}", area);
            Console.ReadKey();
        }
    }
}
```

Exercícios

1. Escreva um programa para efetuar as quatro operações matemáticas básicas (adição, subtração, produto e divisão) sobre dois valores informados.
2. Elaborar um programa que calcule o índice de massa corporal (IMC) de um usuário qualquer, sabendo-se que o IMC é determinado pela divisão da massa do indivíduo (em quilogramas) pelo quadrado de sua altura (em metros).
3. Elaborar um programa que calcule e apresente o volume de uma caixa retangular, por meio da fórmula:
$$\text{volume} = \text{comprimento} * \text{largura} * \text{altura}.$$