

Algoritmos II – prof. Daniel Oliveira

Recuperando Algoritmos I

Objetivos

- Revisar conceitos abordados na disciplina anterior
- Abordar conceitos vistos com a linguagem C#

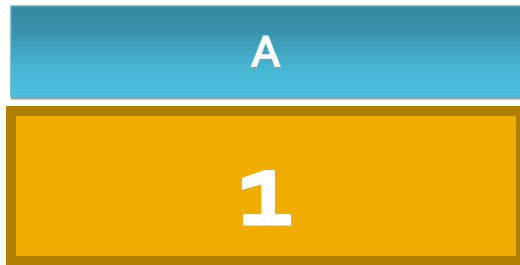
Agenda

- Variáveis e listas
- Expressões
- Estruturas de controle do tipo condicional
- Estruturas de controle do tipo repetição
- Estruturas de repetição e condicionais aninhadas

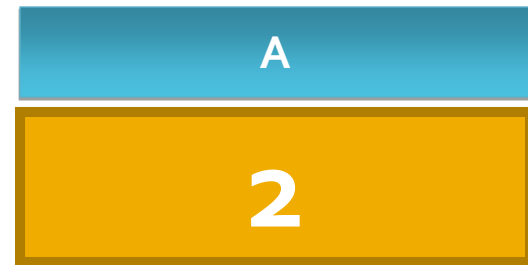
Variáveis e listas

- Variáveis....

A = 1



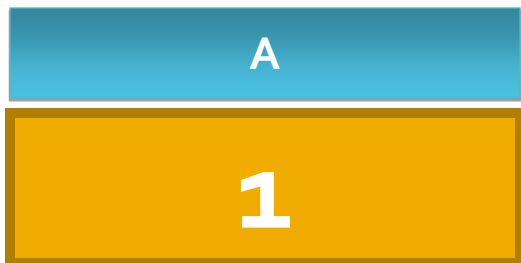
A = A + 1



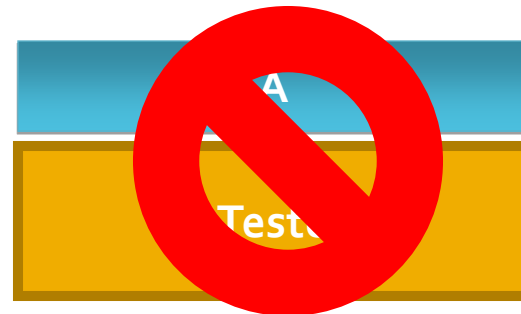
Variáveis e listas

- Em C# é necessário informar qual o tipo de dado será armazenado na variável
- Uma vez informado o tipo, não será possível armazenar na variável qualquer valor diferente do tipo informado.

A = 1



A = "TESTE"



Variáveis e listas

- As informações armazenadas em um tipo podem incluir o seguinte:
 - O espaço de armazenamento que requer uma variável do tipo.
 - Os valores máximo e mínimos que ela pode representar.
 - Os membros (métodos, campos, eventos e assim por diante) que ele contém.
 - O tipo base que ela herda.
 - O local de memória onde as variáveis serão alocadas em tempo de execução.
 - Os tipos de operações que são permitidos.

Variáveis e listas

- O compilador usa informações do tipo para verificar se todas as operações que são executadas em seu código são seguras.

```
int a = 5;  
int b = a + 2; //OK
```

```
bool test = true;
```

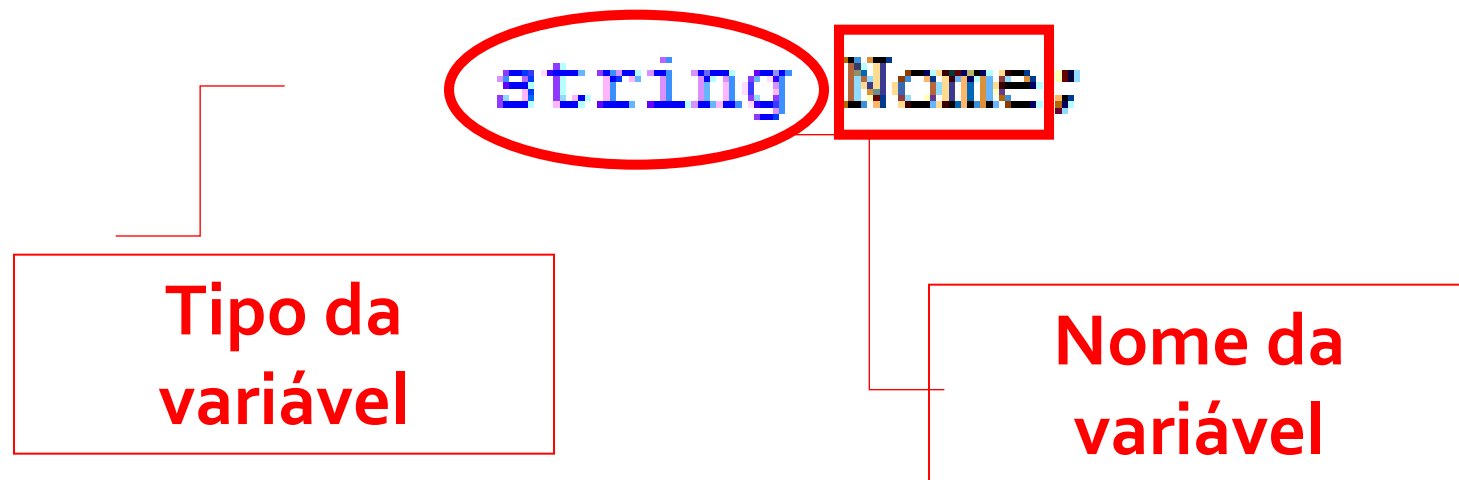
```
*
```

```
// Error. Operator '+' cannot be applied to operands of type 'int' and 'bool'.
```

```
int c = a + test;
```

Variáveis e listas

- Como declarar uma variável em C#



Variáveis e listas

- Ao se declarar um variável pode-se inicializar a mesma:

```
int A;  
A = 0;
```



```
int A = 0;
```

Variáveis e listas

- Exemplo

```
int Variavel1;
```

```
Variavel1 = 1;
```

```
Console.WriteLine("O valor da variável é:{0}", Variavel1);
```

Variáveis e listas

- Em alguns casos é possível a transformação de tipo para outro.
- Por exemplo: Uma string contendo um valor inteiro poderá ser convertida para inteiro.

```
string Variavel1 = "10";  
int varInteiro;  
varInteiro = int.Parse(Variavel1);  
Console.WriteLine("O valor da variável é:{0}", varInteiro);
```

Variáveis e listas

```
int A;
int B;
int C;

string valor;

Console.Write("Informe valor A:");
valor = Console.ReadLine();
A = int.Parse(valor);

Console.Write("Informe valor B:");
valor = Console.ReadLine();
B = int.Parse(valor);

C = A + B;
Console.WriteLine("A + B = {0}", C);
```

Variáveis e Listas

- Principais tipos

Tipo de C#	Descrição
bool	Tipo Booleano (true ou false)
Byte	Representa 1 byte (8 bits) – 0 a 256
char	Um caractere
decimal	Tipo decimal
double	Tipo de dupla precisão
float	Ponto flutuante
int	Inteiro de 32 bits
long	Inteiro de 64 bits
object	Tipo genérico, ancestral dos demais tipos
short	Inteiro de 16 bits
string	Seqüência de caracteres

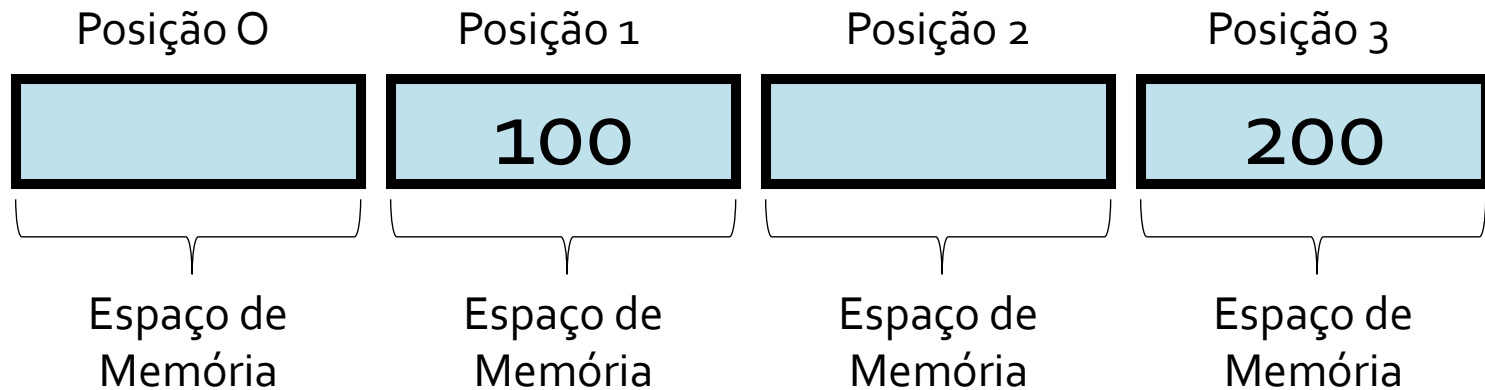
Variáveis e Listas

```
double TemperaturaF;  
double TemperaturaC;
```

```
Console.Write("Informe a temperatura em Fahrenheit:");  
TemperaturaF = double.Parse(Console.ReadLine());  
TemperaturaC = (TemperaturaF-32.0) * (5.0/9.0);  
Console.WriteLine("A temperatura{0} F em Celsius são {1} C",  
                  TemperaturaF, TemperaturaC);
```

Variáveis e Listas

- Listas, Arrays, Vetores....



Variáveis e Listas

- Arrays ou vetores são elementos dimensionados de um determinado tipo de dados
- Cada elemento de uma array é acessado através de um índice (posição) sempre a partir do zero.
- Uma array poderá ser unidimensional ou multidimensional

Variáveis e Listas

- Na declaração de um *array* (vetor) selecionamos o tipo de dado seguido de colchetes ([]). Após, coloca-se a variável.

```
string[] Nomes;
```

```
//array declarado, mas não criado ou  
inicializado
```

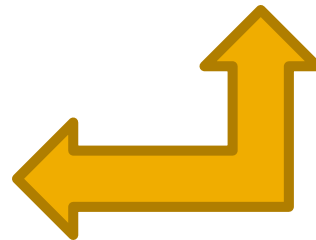
```
Nomes = new string[5];
```

```
//array criado com 5 elementos
```

Variáveis e Listas

```
string[] Nomes;  
Nomes = new string[5];  
  
Nomes[0] = "Pedro";  
Nomes[1] = "Daniel";  
Nomes[2] = "Karla";  
Nomes[3] = "Maria";  
Nomes[4] = "João";
```

Memória	
Nomes	
0	Pedro
1	Daniel
2	Karla
3	Maria
4	João



Variáveis e Listas

- Lendo uma posição da array

```
int Posicao;  
Console.Write("Digite um valor 0 a 4:");  
Posicao = int.Parse(Console.ReadLine());  
Console.WriteLine("Nome na posição {0}:{1}", Posicao, Nomes[Posicao]);
```

Variáveis e Listas

- Os vetores podem ser inicializados durante a declaração.

```
double[] ar1 = new double[4]{1.1, 2.2, 3.3, 4.4};  
double ar1 = new double[]{1.1, 2.2, 3.3, 4.4};  
double ar1 = {1.1, 2.2, 3.3, 4.4};
```

Variáveis e Listas

```
string[] Nomes = new string[]{"Pedro", "Daniel", "Karla", "Maria", "João"};
int Posicao;
Console.Write("Digite um valor 0 a 4:");
Posicao = int.Parse(Console.ReadLine());
Console.WriteLine("Nome na posição {0}:{1}", Posicao, Nomes[Posicao]);
```

Variáveis e Listas

- Atividade

- 1) Desenvolver uma rotina que receba três nomes e os exiba em seguida.
- 1) Desenvolver uma rotina que tenha dois vetores (com 3 posições cada): um de string e um de double. O usuário deverá informar um nome e o salário (cada um armazenado no respectivo vetor). E, o programa deverá exibir, ao final, a listagem no formato: <NOME> - <SALÁRIO>

Constantes

- O que são constantes?



Constantes

- Em C# usa-se a palavra reservada ***const*** para se declarar uma constante

```
const int x = 0;  
public const double gravitationalConstant = 6.673e-11;  
private const string productName = "Visual C#";
```

Constantes

```
static void Main(string[] args)
{
    const int ValorConstante = 1;
    Console.WriteLine("O valor da constante é:{0}",ValorConstante);
}
```

Operadores

- Operadores são símbolos que realizam alguma ação em uma expressão.
- C# oferece vários operadores diferentes

Operadores

- Operador (+)
- Quando aplicado a tipos numéricos realiza a operação de adição
- Quando aplicado a string, realiza a concatenação (união) de duas strings

Operadores

```
using System;
class MainClass
{
    static void Main()
    {
        Console.WriteLine(+5);
        Console.WriteLine(5 + 5);
        Console.WriteLine(5 + .5);
        Console.WriteLine("5" + "5");
        Console.WriteLine(5.0 + "5");
    }
}
```

Operadores

- Operador (-)
- Realiza uma operação de subtração em tipos numéricos ou inversão de sinal

```
using System;
class MainClass
{
    static void Main()
    {
        int a = 5;
        Console.WriteLine(-a);
        Console.WriteLine(a - 1);
        Console.WriteLine(a - .5);
    }
}
```

Operadores

- Operador (++)
- Realiza o incremento de 1 de um tipo numérico
- Pode ser pós-fixado (x++);
- ou pré-fixado (++x)

Operadores

```
using System;
class MainClass
{
    static void Main()
    {
        double x;
        x = 1.5;
        Console.WriteLine(++x);
        x = 1.5;
        Console.WriteLine(x++);
        Console.WriteLine(x);
    }
}
```

Operadores

- Operador (--)
- Realiza o decrementa de 1 de um tipo numérico
- Pode ser pós-fixado (x--);
- ou pré-fixado (--x)

Operadores

```
using System;
class MainClass
{
    static void Main()
    {
        double x;
        x = 1.5;
        Console.WriteLine(--x);
        x = 1.5;
        Console.WriteLine(x--);
        Console.WriteLine(x);
    }
}
```

Operadores

- Operador (*)
- Realiza operação de multiplicação em tipos numéricos

```
using System;
class MainClass
{
    static void Main()
    {
        Console.WriteLine(5 * 2);
        Console.WriteLine(-.5 * .2);
        Console.WriteLine(-.5m * .2m);
    }
}
```

Operadores

- Operador (/)
- Realiza a divisão de dois números
- Como ficaria a divisão de dois inteiros ?

```
using System;
class MainClass
{
    static void Main()
    {
        Console.WriteLine(5/2);
        Console.WriteLine(5 % 2);

        Console.WriteLine(5 / 2.1);
        Console.WriteLine(5.1 / 2);
        Console.WriteLine(-5 / 2);
    }
}
```

Operadores

- Operador (%)
- Cálculo o resto de uma divisão

```
using System;
class MainClass
{
    static void Main()
    {
        Console.WriteLine(5 % 2);    // int
        Console.WriteLine(-5 % 2);   // int
        Console.WriteLine(5.0 % 2.2); // double
        Console.WriteLine(5.0m % 2.2m); // decimal
        Console.WriteLine(-5.2 % 2.0); // double
    }
}
```

Operadores

- Precedência
- Como na matemática operação de multiplicação e divisão tem precedência sobre as operações de adição e subtração
- Utiliza-se os parênteses () para indicar a precedência.

Exemplo

```
static void Main(string[] args)
{
    Console.WriteLine(5 * 6 + 1);
    Console.WriteLine(5 * (6 + 1));
}
```

Atividade

- Desenvolva um programa que receba um valor de temperatura em graus Celsius e retorne o valor em graus Fahrenheit. Sabendo que a relação de transformação é:

$$F = C \cdot \frac{9}{5} + 32$$

Estruturas de controle

- Condicional simples



Estruturas de controle

Operadores condicionais

$a == b$	a tem o mesmo valor de b
$a > b$	a é maior do que b
$a >= b$	a é maior ou igual a b
$a < b$	a é menor do que b
$a <= b$	a é menor ou igual a b
$a != b$	a não é igual a b

Estruturas de controle

Declaração if

```
if (<Condição booleana>
{
}
}
```

Estruturas de controle

Declaração if

```
int a;  
int b;  
Console.Write("Informe um número:");  
a = Convert.ToInt16(Console.ReadLine());  
Console.Write("Informe outro número:");  
b = Convert.ToInt16(Console.ReadLine());  
  
if (a > b)  
{  
    Console.WriteLine("O número {0} é maior do que {1}.", a, b);  
}
```

Estruturas de controle

Declaração if...else

```
if (<Condição booleana>
{

}
else
{

}
}
```

Estruturas de controle

```
int a;  
int b;  
Console.Write("Informe um número:");  
a = Convert.ToInt16(Console.ReadLine());  
Console.Write("Informe outro número:");  
b = Convert.ToInt16(Console.ReadLine());  
  
if (a > b)  
{  
    Console.WriteLine("O número {0} é maior do que {1}.", a, b);  
}  
else  
{  
    Console.WriteLine("O número {0} é menor do que {1}.", a, b);  
}
```

Estruturas de controle

Declaração if...else...if

```
if (<Condição booleana>)
```

```
{
```

```
}
```

```
else
```

```
if (<Condição booleana>)
```

```
{
```

```
}
```

Estruturas de controle

```
int a;
int b;
Console.Write("Informe um número:");
a = Convert.ToInt16(Console.ReadLine());
Console.Write("Informe outro número:");
b = Convert.ToInt16(Console.ReadLine());

if (a > b)
{
    Console.WriteLine("O número {0} é maior do que {1}.", a, b);
}
else
{
    if (a < b)
    {
        Console.WriteLine("O número {0} é menor do que {1}.", a, b);
    }
    else
    {
        Console.WriteLine("O número {0} é igual a {1}.", a, b);
    }
}
```

Estruturas de Repetição

Loop FOR

```
static void Main(string[] args)
{
    for (int i = 0; i < 10; i++)
    {
        Console.WriteLine(i);
    }
}
```

Variável do Loop

Teste condicional
do Loop

Variação de valor da
variável do Loop

Estruturas de Repetição

```
static void Main(string[] args)
{
    for (int i = 0; i < 10; i++)
    {
        Console.WriteLine(i);
    }
}
```

Estruturas de Repetição

- ***Loop while*** – realiza as instruções até que a condição de validação seja falsa

```
while(Expressão Booleana  
{  
// ...Comandos para ser executados pelo loop  
}
```

Estruturas de Repetição

```
int n = 1;
while (n < 6)
{
    Console.WriteLine("O valor atual de n é {0}", n);
    n++;
}
```

Estruturas de Repetição

- ***Loop Do..While*** – Realiza uma instrução pelo menos uma vez até a condição de verificação seja falsa.

```
do
{
    // ...Instrução do Loop...
} while (Condição Booleana);
```

Estruturas de Repetição

```
static void Main(string[] args)
{
    int x = 0;
    do
    {
        Console.WriteLine(x);
        x++;
    } while (x < 5);
}
```

Estruturas de repetição

- Declaração *break*
- Instrução para parar um processamento de um loop
- A execução do programa continua na primeira declaração depois do loop

Estrutura de repetição

```
static void Main(string[] args)
{
    Console.WriteLine("Informe um numero entre 1-10 para parar:");
    int num = Convert.ToInt16(Console.ReadLine());

    for (int i = 1; i < 11; i++)
    {
        Console.WriteLine(i);
        if (i == num)
        {
            Console.WriteLine("Parei");
            break;
        }
    }
}
```

Atividade

- O fatorial de um número inteiro positivo é dado por: $n! = n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot 1$. E, o fatorial de zero é igual a 1 por definição. Faça um programa que receba um número inteiro positivo e exiba seu fatorial.