Aula 1: Algoritmos - variáveis, constantes e tipos de dados; comandos: entrada e saída, atribuição; operadores aritméticos.

Variáveis e Constantes

Para guardar dados no computador, um programa precisa definir:

1. **onde** os dados serão armazenados
2. **que tipo** de dado será armazenado
3. **que valor** será armazenado.

Para isso, utilizamos, em nossos programas, as Variáveis e as Constantes.

Variáveis:

As variáveis precisam ter:

1. um nome,
2. um tipo,
3. um valor.

Para o nome de uma variável, é necessário seguir algumas regras. Os caracteres válidos são:

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
_
```

IMPORTANTE:

- não usar dígito numérico no início do nome da variável
- não usar espaço em branco

Operações sobre variáveis:

1. declaração
2. inicialização
3. atribuição de valores

Podemos atribuir a uma variável um valor constante, uma expressão ou até mesmo uma outra variável. Pode ser ainda uma expressão misturando constantes e variáveis.
Exemplos:

```
soma = 10;
soma = 4 + 6;
soma = total_parcial + 5;
soma = valor_anterior + valor_atual;
```

Incremento e Decremento de variáveis
Uma expressão em uma operação de atribuição poderá conter a própria variável que receberá o resultado da expressão.

Exemplo, a atribuição

```
x = x + 1;
```

é válida.
Como isso acontece?

A expressão é calculada primeiro e seu resultado final é armazenado na variável. Então, se tivermos:

```
x = 10;
x = x + 1;
```

no último comando, `x` recebe o valor da expressão `x + 10`, que para ser calculada precisa conhecer o valor atualmente armazenado em `x`. A última atribuição feita a `x` foi o valor `10`. Então a expressão é calculada como `10 + 1`, resultando em `11`, que é atribuído a `x`.

Comandos do tipo `<variável> = <variável> + <número ou expressão>;` é conhecido como comando de *incremento* da variável `<variável>`.
Exemplos:

```
x = x + 1;        → incremento de 1 unidade da variável x
z = z + 5;        → incremento de 5 unidades da variável z
```

Analogamente, comandos do tipo `<variável> = <variável> - <número ou expressão>` é conhecido como comando de *decremento* da variável `<variável>`
Exemplos:

```
total = total - 2;     → decremento de 2 unidades da variável total
y = y - 5;             → decremento de 5 unidades da variável y
```

OBS: Outros operadores podem ser utilizados em comandos de incremento/decremento.
Exemplos:

`x = x * 2;` → resulta no dobro do valor previamente armazenado em `x`
`y = y / 3;` → resulta em um terço do valor previamente armazenado em `y`

IMPORTANTE:
Uma variável armazena apenas um valor por vez! Isso significa que a cada nova atribuição de valor a uma mesma variável, os valores anteriormente atribuídos a ela são perdidos.
Exemplo: a sequência de comandos

```
a = 10;
a = 20;
```

resulta no armazenamento do valor 20 na variável `a`. O valor 10 anteriormente atribuído a ela se perde.

Constantes:

Como o nome sugere, as constantes têm valor fixo e inalterável. Alguns exemplos são os valores numéricos, os caracteres e os textos.
Podemos declarar uma constante, de modo a dar um nome significativo a um valor a ser usado no programa. Por exemplo, ao invés de utilizar o valor 3.14159265359 em um programa, podemos declarar uma constante de nome “pi” e armazenar nela esse valor. Assim, sempre que quisermos usar o valor de pi, podemos escrever o nome da constante declarada no programa.

NOTE: as constantes podem ser declaradas como as variáveis, mas tem a vantagem de ter seus valores protegidos contra alterações indesejadas (jamais serão alterados em seu programa).
No Processing, a declaração de constantes pode ser feita da seguinte forma:

```
final <tipo> <nome_da_constante>;
```

Exemplo:

```
final float coeficiente_multiplicador;
```

OBS: O Processing possui algumas constantes úteis prontas para uso. `PI` é uma delas. :)

Tipos de dados:

Caracterizam a natureza dos valores armazenados.

Destacamos, por ora, os seguintes tipos:

```
int
long
float
double
char
boolean
```

Os tipos determinam o espaço reservado em memória para armazenar valores. Por esta razão, se quisermos armazenar, por exemplo, um valor real em uma variável do tipo inteiro, o Processing não realizará a operação, exibindo uma mensagem de erro.

No entanto, podemos, em alguns casos, converter dados de um tipo para outro!

Conversões de tipos:

Automáticas: em atribuições e em expressões.

Exemplos:
`float valor = 10;` → resulta em 10.0
`float divisao = 11/2.0;` → resulta em 5.5

Conversão manual:
Usada quando se deseja mudar o tipo de uma variável ou o resultado de uma função em uma expressão.

Exemplo:
```
float preco = 5.4;
print((int)preco);
```
→ vai imprimir apenas a parte inteira (5).

Entrada e Saída de dados

Entrada:

Dedicaremos uma discussão mais detalhada sobre entrada de dados no Processing quando estivermos abordando aspectos de interatividade em nossos programas. Por ora, nossa entrada de dados será por meio de comandos de atribuição de valores a variáveis declaradas.

Saída:

Comandos para impressão no console: `print()` e `println()`

Usamos o `print()` quando desejamos imprimir algo no console sem realizar salto de linha após a impressão.
Para impressões que requerem quebra de linha, usamos o comando `println()`.

OBS: tanto o `print()` quanto o `println()` aceitam mais de uma entrada. Em casos de impressão de mais de uma informação, podemos separar tais informações usando vírgulas ou o sinal de adição (+).
Exemplos:

`print("Olá");` → exibe a mensagem "Olá" na tela do console.
`print(x);` → exibe o valor armazenado na variável x.
`print("x = ", x);` → que equivale a escrever: `print("x = " + x);` e exibirá na tela a mensagem "x = ", seguido do valor armazenado em x.

```
print("O valor total calculado foi de" , total, "reais");
```
→ que equivale a escrever:
```
print("O valor total calculado foi de " + total + " reais");
```

Os comandos `print()` e `println()` exibem, ainda, o resultado de expressões.
Exemplos:
```
print("O produto dos valores de a e b resulta em: ", a*b);
println(" A expressão 50 % 3 resulta em: ", 50%3);
```

Operadores Aritméticos

+ → soma
- → subtração
* → multiplicação
/ → divisão
% → módulo.

O operador módulo produz como resultado o resto da divisão inteira entre dois números.
Exemplos:
```
float resultado;
resultado = 5 % 2;    → resultado recebe 1
resultado = 10 % 4;   → resultado recebe 2
resultado = 6 % 2;    → resultado recebe 0
```

OBS: é importante relembrarmos regras de precedência de operadores que aprendemos na matemática. Em uma expressão aritmética, os operadores de multiplicação e divisão têm precedência sobre os operadores de soma e subtração!
OBS2: Para mudar a precedência de operadores, podemos utilizar parênteses.
Exemplos:
```
float resultado;
resultado = 6 / 2 * (1 + 2);          → resultado recebe 9.0
resultado = 3 - 2 + 7 * 5;            → resultado recebe 36.0
resultado = 3 + (2 * (8 + 2)) * 2;    → resultado recebe 43.0
```

Algumas funções

`abs( )` → resulta no valor absoluto de um número ou expressão.
Exemplos:
```
abs(-1)          → resulta em 1
abs(5 - 40)      → resulta em 35
```

`sq( )` → resulta no quadrado de um número ou expressão.
Exemplos:
`sq(4)` → resulta em 16.0
`sq(4 + 6)` → resulta em 100.0

`pow( )` → resulta em uma potência de dois números
Exemplos:
`pow(2, 3)` → resulta em 2 elevado a 3. Logo, resulta em 8.0
`pow(5, 4)` → resulta em 5 elevado a 4. Logo, resulta em 625.0

`sqrt( )` → resulta na raiz quadrada de um número ou expressão
Exemplo:
`sqrt(9)` → resulta em 3.0
`sqrt(20 + 10/2)` → resulta em 5.0

Exercícios:

1. Uma vez declarada a variável

```
float x;
```

Indique, nos itens abaixo, os valores nela armazenados:
   a. x = 3 + 4 * 5;
   b. x = 120 / 4 * 5 - 5;
   c. x = 120 / 4 * (5 - 5);

2. Uma vez declarada a variável

```
int y;
```

Indique, nos itens abaixo, os valores nela armazenados:
   a. y = 120 / 4 * 5;
   b. y = 5 / 2;
   c. y = 66 % 4;

3. Indique o valor da variável x ao final da execução do código abaixo:

```
int x;
x = 15/4;
x = 6;
x = x + 1;
```

4. Indique os valores armazenados nas variáveis do código abaixo:

```
float x = 5.0/2;
float y = 5/2;
float z = 5/2.0;
int b = 5/2;
```

5. Escreva um programa que exiba um valor inteiro armazenado em uma variável criada por você. Em seguida, seu programa deverá exibir o resultado do incremento dessa variável.

6. Escreva um programa que exiba um número e em seguida exiba o seu antecessor e o seu sucessor.

7. Escreva um programa que crie uma variável inteira com valor negativo, exiba esse valor e em seguida exiba o seu valor absoluto.

8. Escreva um programa que crie uma variável real e exiba as seguintes informações:
    a. O valor armazenado na variável.
    b. O valor do seu quadrado.
    c. O valor do número elevado ao cubo
    d. O valor da raiz quadrada desse número.

9. Escreva um programa que contenha as variáveis A e B e troque os valores dessas variáveis. Exemplo de saída:

    valor armazenado em A : 4
    valor armazenado em B: 3
    após a troca, temos:
    valor armazenado em A : 3
    valor armazenado em B : 4

10. Algumas mães não gostam de ver seus filhos ganhando idade. Assim, elas fazem de tudo para traduzir a idade de um filho em meses, ao invés de anos. Escreva um programa que crie uma variável `idade` e armazene nela um valor em anos. Seu programa deverá exibir essa mesma idade, mas em meses.

11. Escreva um programa que contenha variáveis para armazenar as coordenadas x e y de dois pontos no espaço 2D. Seu programa deverá calcular e exibir a distância entre esses dois pontos. Dica: use as funções `sqrt()` e `sq()` para montar a expressão do cálculo da distância.