UNIVERSIDADE VILA VELHA
ESPÍRITO SANTO

Disciplina: Fundamentos da Computação		**Visto:**
Professor: Abrantes Araújo Silva Filho		
Aluno:		
Turma:	**Semestre:**	**Valor:** —
Data:	**Diário** 2: Programação	

DIÁRIO DE APRENDIZAGEM:

- Este **Diário de Aprendizagem** é uma das atividades integrantes da disciplina de **Fundamentos da Computação** do curso de Ciência da Computação, Universidade Vila Velha (UVV).
- A confecção do diário de aprendizagem é atividade **obrigatória** e **altamente recomentada** por três motivos: a) você aprenderá muito mais a matéria se mantiver o diário; b) ao entregar todos os diários ao professor você está cumprindo parte das atividades avaliativas que contam pontos na disciplina (10% da nota); e c) as provas bimestrais discursivas seguirão o formato e conteúdo das perguntas do diário.
- Se você tiver dificuldade em responder alguma questão do diário, estude novamente a matéria. Se você realmente entendeu a matéria, não deveria ter muita dificuldade para responder o diário.
- Responda com caneta ou lápis escuro (2B, 4B, 6B).
- Verifique no calendário de sua turma a **data de entreg**. Após uma rápida avaliação e visto pelo professor ou pelos monitores, seu diário será devolvido.
- O diário não será corrigido pelo professor: cabe a você estudar e dar a resposta correta para todas as questões. Obviamente o professor está à disposição para esclarecimento de dúvidas, e os monitores podem auxiliar caso você tenha dificuldade.
- Manter o diário de aprendizagem atualizado pode ser a diferença entre você aprender a matéria e ser aprovado, ou não aprender a matéria e não ser aprovado.
- Bons estudos!

Fundamentos da Programação

1. Por que é mais importante pensar primeiro no **algoritmo** (através de pseudocódigo, fluxograma ou qualquer outra coisa) e não na **programação** em si?

2. Com exceção de alguns conceitos mais **avançados** e de conceitos genéricos como as **boas práticas** de programação, quais os 5 (cinco) grandes fundamentos da programação (escreva um por linha):

3. Por que utilizaremos a **Scratch** como nossa primeira linguagem de programação, ao invés de alguma outra linguagem mais tradicional como a linguagem **C**?

4. A interface de programação principal da Scratch é o navegador de Internet. Essa interface tem 4 (quatro) grandes áreas. Explique brevemente a função de cada uma dessas áreas:
   (a) Guias de blocos, fantasias e sons (*code*, *costumes*, *sounds*):
   (b) Área de programação (*script area*):
   (c) Área dos personagens (*sprites area*):
   (d) Palco (mundo) dos personagem (*stage area*):

5. Faça um esquema do **sistema de coordenadas** do palco dos personagens.

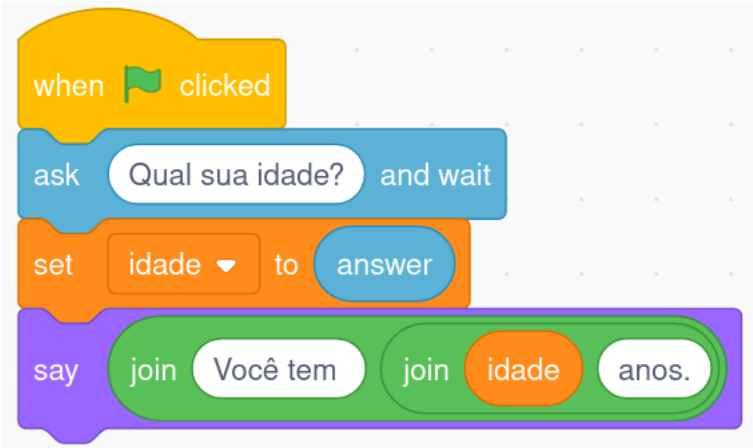
6. O que é uma **função**?

7. O que é um **evento**? Como os eventos são disparados?

8. Seu algoritmo deve obrigatoriamente **escutar** e **responder** à eventos que ocorram em seu programa?

9. Em relação às funções, explique:
   (a) O que é um **argumento**?
   (b) O que é um **parâmetro**?
   (c) O que é o **valor de retorno**?
   (d) O que é um **efeito colateral**?

10. Analise o código abaixo e responda às perguntas a seguir (`idade` é uma variável criada pelo programador).



(a) Explique resumidamente o que este código está fazendo:

(b) Qual é o **argumento** da função `ask`?

(c) Qual é o **retorno** da função `ask`?

(d) A função `ask` está causando algum **efeito colateral**?

(e) Qual o **argumento** (*input*) da função `set`?

(f) Qual o **retorno** da função `set`?

(g) A função `set` está causando alguma **efeito colateral**?

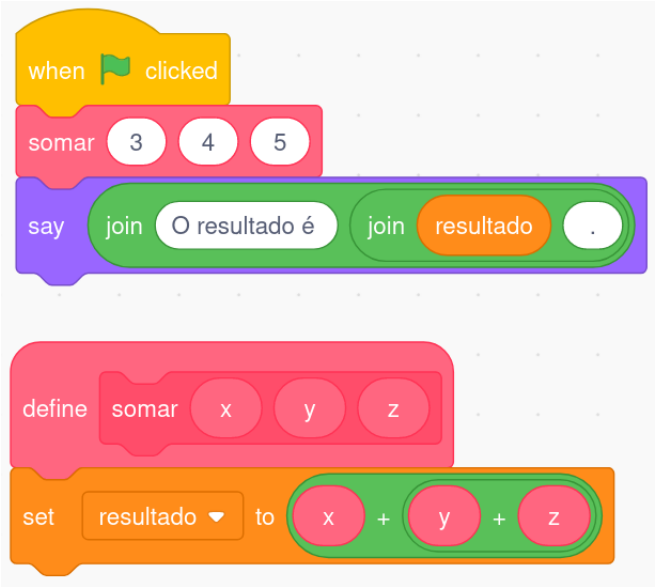
(h) Qual o **retorno** da função `say`?

(i) Qual o **argumento** da função `say`?

(j) A função `say` está causando algum **efeito colateral**?

(k) O argumento da função `say` é, na verdade, uma **combinação de funções**. Explique em detalhes como essa combinação funciona.

11. Analise o código abaixo e responda às perguntas a seguir (`resultado` é uma variável criada pelo programador, e `somar` é uma função criada pelo programador).



(a) Explique resumidamente o que este código está fazendo:

(b) O que representam as letras `x`, `y` e `z`?

(c) O que representam os números 3, 4 e 5?

(d) Em computação, **chamar uma função** corresponde a utilizar essa função no código, passando para ela os argumentos necessários (se existirem parâmetros para receber esses argumentos). A função `somar` está sendo chamada nesse programa? Explique.

(e) Explique, em detalhes, como a função `somar` funciona, desde a chamada da função, a passagem de argumentos para os parâmetros, o uso dos parâmetros dentro da função, o uso das funções de adição (+) e o efeito de colocar a soma na variável `resultado`.

12. O que é um **bug** no código de seu programa?

13. O que é **debugar** um programa?

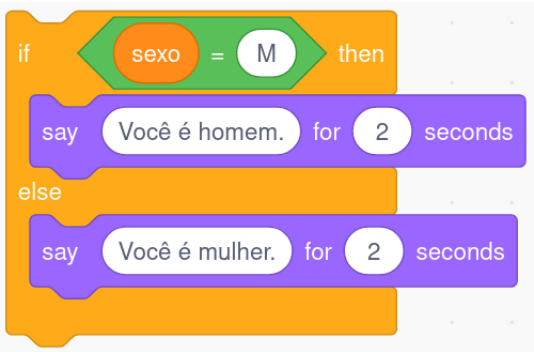
14. O que é a **composição de funções**?

15. Quando criamos um algoritmo para resolver um problema, é mais importante que ele seja **correto** ou que ele seja **bem projetado**? Por quê?

16. O que são **expressões booleanas**?

17. O que são **condicionais**? Por que as condicionais dependem do resultado das expressões booleadas?

18. Explique o que o código abaixo faz.



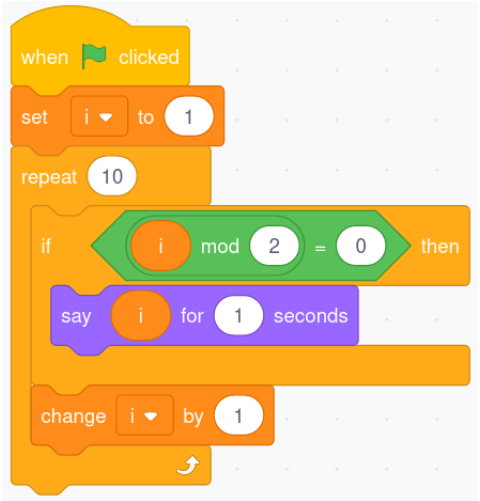
19. Explique o que o código abaixo faz.

20. Você acha que o código da questão anterior está bem projetado? Por quê?

21. O que são os **loops**? Para que servem?

22. No Scratch não há nenhum bloco que nos permita criar uma expressão booleana que utilize o comparador $\leq$ (menor do que ou igual a). Suponha que precisamos analisar a seguinte expressão: "`numero` $\leq$ `23`". Utilizando somente os blocos `and`, `or`, `not`, =, < ou >, construa **duas expressões booleanas** equivalentes à "`numero` $\leq$ `23`".

23. Analise o código abaixo e responda às perguntas a seguir (`i` é uma variável criada pelo programador). Procure na Internet o funcionamento da função `mod` para descobrir o que essa função faz.



(a) Descreva **o que esse programa está fazendo**.

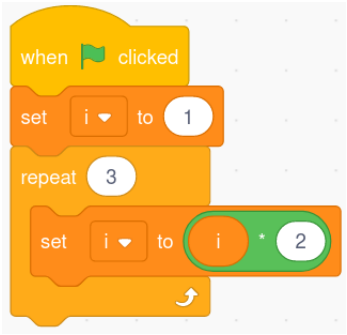
(b) Explique o **funcionamento** da expressão booleana "`(i mod 2) = 0`".

(c) Na chamada da função `say`, a variável `i` é um **argumento** ou é um **parâmetro**? Por quê?

(d) A expressão booleana "`(i mod 2) = 0`" será **verdadeira** ou **falsa**? Por quê?

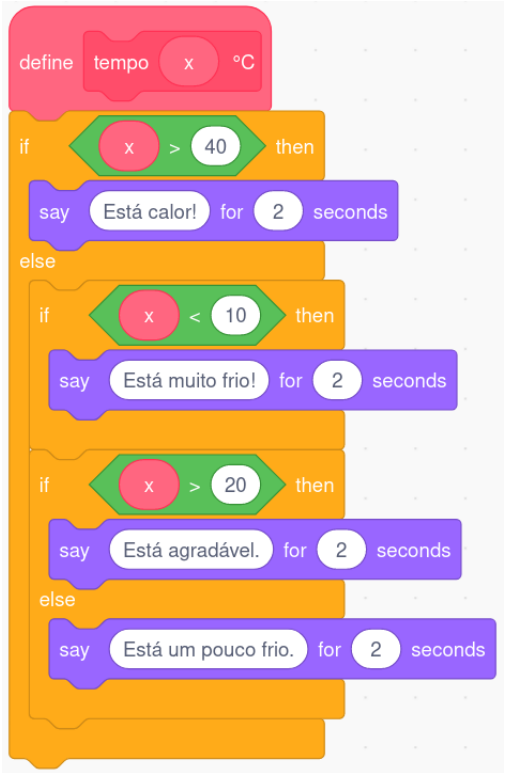
(e) Quais funções no programa acima apresentam **efeito colateral**? Por quê?

24. Analise o código abaixo (`i` é uma variável criada pelo programador):



Qual será o valor da variável `i` quando o loop terminar a execução?

25. O código abaixo cria uma **função**, ou seja, uma abstração para a determinar se o tempo está agradável ou não. Avalie esse código:

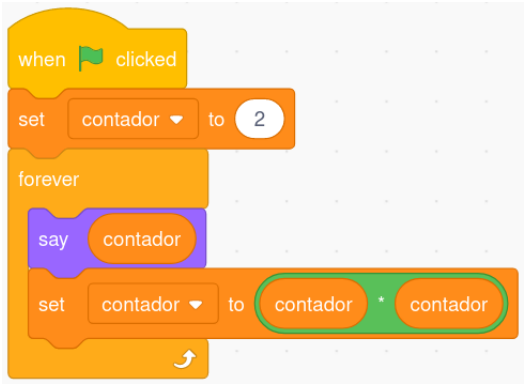


O que esse código informará se a temperatura estiver em 16ºC? Por quê?

26. O que é a **refatoração** de código? Explique com um exemplo.

27. O que são **variáveis**?

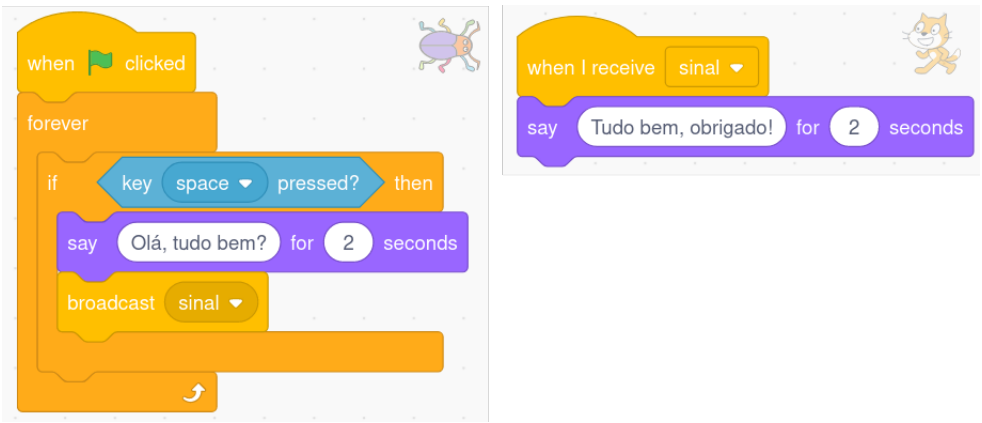
28. Rode o código abaixo em um programa Scratch e aguarde alguns instantes. Você verá que ocorreu um **overflow**. O que é isso? Por que esse programa gerou um *overflow*?



29. Em geral os **loops infinitos** indicam erros em seu algoritmo ou em seu programa. Explique uma situação na qual loops infinitos podem ser úteis.

30. O que é **programação concorrente**?

31. Dois ou mais personagens (*sprites*) podem "conversar" entre si através de **sinais** ou **mensagens** disparados pela ocorrência de algum evento. Analise o trecho de código a seguir que mostra a "conversa" entre dois personagens:



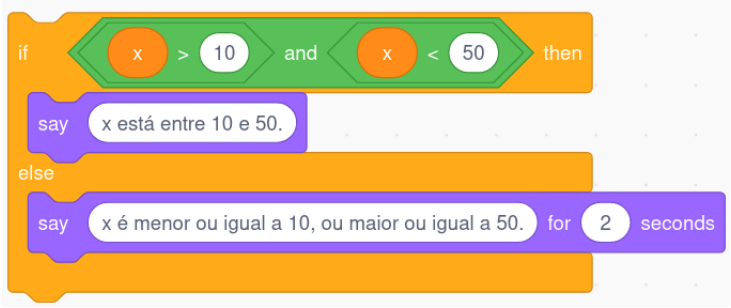
Explique como funciona a conversa entre os personagens através da troca de mensagens disparadas por eventos.

32. Para que servem os blocos avançados do Scratch?

33. Uma expressão booleana pode ser, em geral, de dois tipos:

    - **Relacional**: é uma expressão booleana que utiliza operadores matemáticos relacionais, tais como: $<$, $\leq$, $>$, $\geq$, $=$, $!=$; ou
    - **Lógica**: é uma expressão booleana que utiliza operadores lógicos, tais como: `and`, `or`, `not` e `xor` (existem outros operadores lógicos, mas esses são os mais comumente usados em programação).

    Também existem expresões booleanas que misturam os dois tipos citados acima, por exemplo:



    Interpretar uma expressão booleana relacional é relativamente fácil. O mesmo não pode ser dito de expressões booleanas lógicas, que podem ser combinadas de formas complexas. Pesquise na Internet a **tabela verdade** dos operadores lógicos `and`, `or`, `not` e `xor`, e faça a complementação das tabelas abaixo, indicando em que situações o resultado será verdadeiro (V) ou falso (F) ao compararmos duas coisas, representadas por "A" e "B".

AND

A	B	A AND B
V	V	
V	F	
F	V	
F	F	

OR

A	B	A OR B
V	V	
V	F	
F	V	
F	F	

NOT

A	NOT A
V	
F	

XOR

A	B	A XOR B
V	V	
V	F	
F	V	
F	F	

    Agora responda: para que serve a tabela verdade? Qual a diferença entre o operador relacional `or` e `xor`?