start by alura

Habilidades trabalhadas nesta aula:

(EF67LP14) Definir o contexto de produção da entrevista (objetivos, o que se pretende conseguir, porque aquele entrevistado etc.), levantar informações sobre o entrevistado e sobre o acontecimento ou tema em questão, preparar o roteiro de perguntas e realizar entrevista oral com envolvidos ou especialistas relacionados com o fato noticiado ou com o tema em pauta, usando roteiro previamente elaborado e formulando outras perguntas a partir das respostas dadas e, quando for o caso, selecionar partes, transcrever e proceder a uma edição escrita do texto, adequando-o a seu contexto de publicação, à construção composicional do gênero e garantindo a relevância das informações mantidas e a continuidade temática.

(EF06CO05) Identificar os recursos ou insumos necessários (entradas) para a resolução de problemas, bem como os resultados esperados (saídas), determinando os respectivos tipos de dados, e estabelecendo a definição de problema como uma relação entre entrada e saída.

(EF06CO09) Apresentar conduta e linguagem apropriadas ao se comunicar em ambiente digital, considerando a ética e o respeito.

Aula 3

Organizando o código

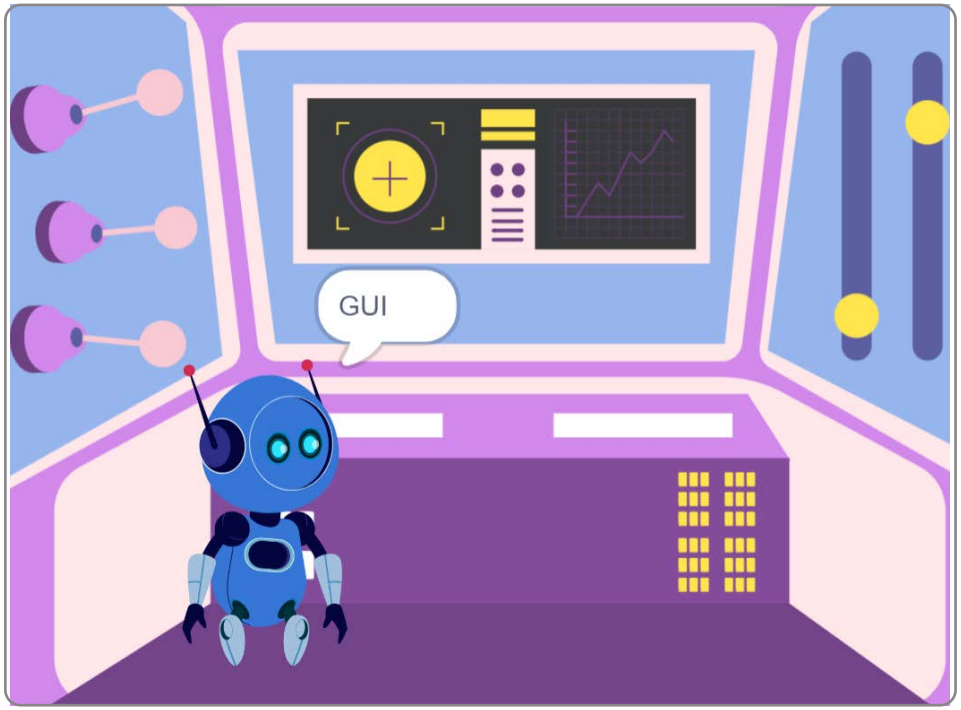
▶ Unidade

Entrada e saída de dados: criando um gênio virtual – Parte 2

O que vamos aprender?

- Implementar uma função personalizada no código inicial.
- Identificar momentos ideais de armazenamento e remoção de dados.
- Modificar os nomes e quantidades das variáveis.

ACOMPANHE O VÍDEO DA AULA

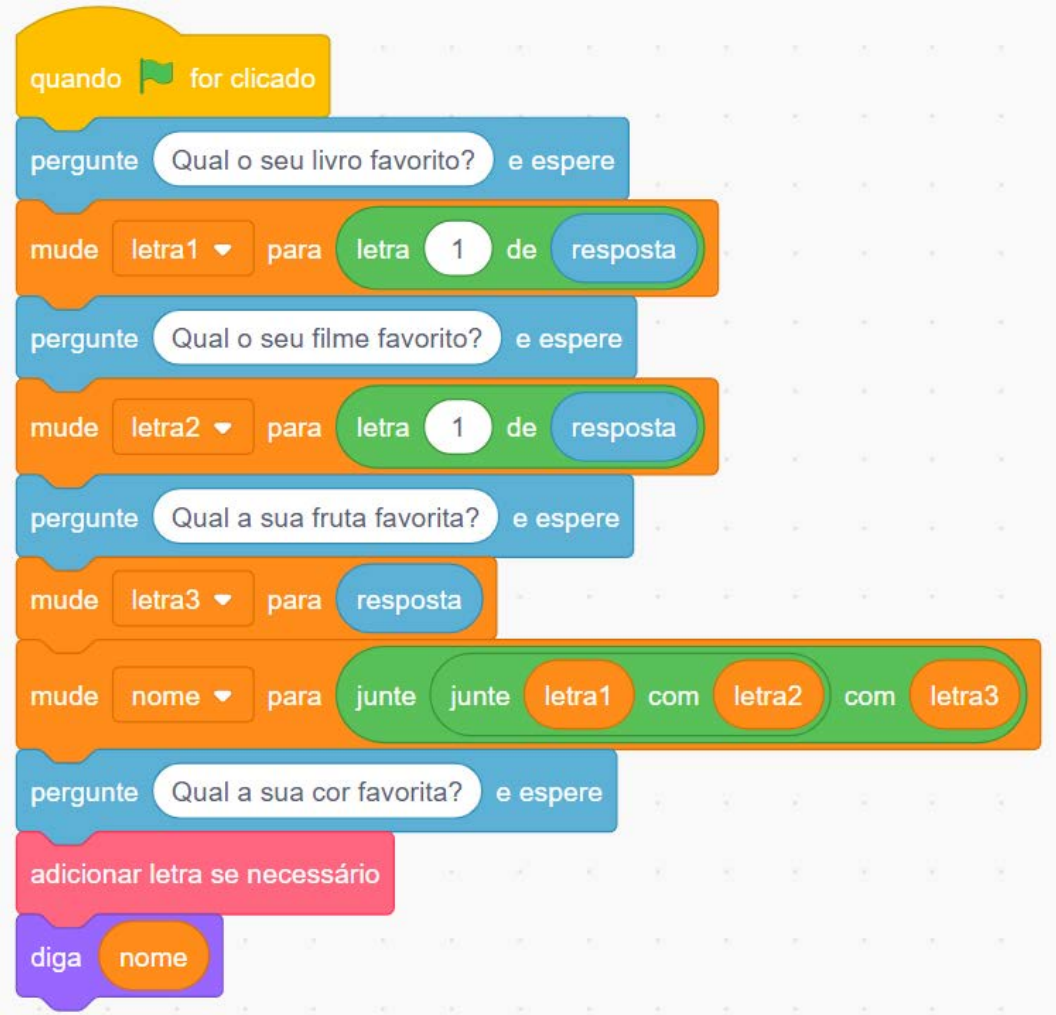


Ajustando condicionais e variáveis

Na aula anterior, otimizamos o código, pois ele estava se tornando cada vez maior. Agora, organizaremos o código de forma que possamos implementar novas funções e identificar momentos de armazenamento ou remoção de dados, bem como as informações das variáveis.

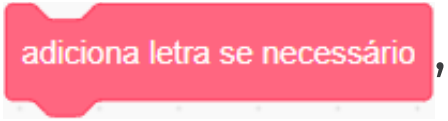
Para iniciar a aula, sugere-se que o professor faça a seguinte pergunta aos estudantes: se você estivesse criando um programa para gerar convites personalizados para uma festa, como organizaria o código para garantir que cada convite esteja correto e sem duplicações?. Espera-se que os estudantes tragam diversas ideias, como a organização do código a partir de uma lista com os nomes dos convidados e o uso de uma função para gerar o texto do convite para cada nome na lista. Assim, garantiríamos que todos os convites sejam criados corretamente e evitaríamos erros e duplicações. Também é possível usar comentários no código para explicar cada parte e facilitar a compreensão.

Observem o código otimizado na aula anterior:

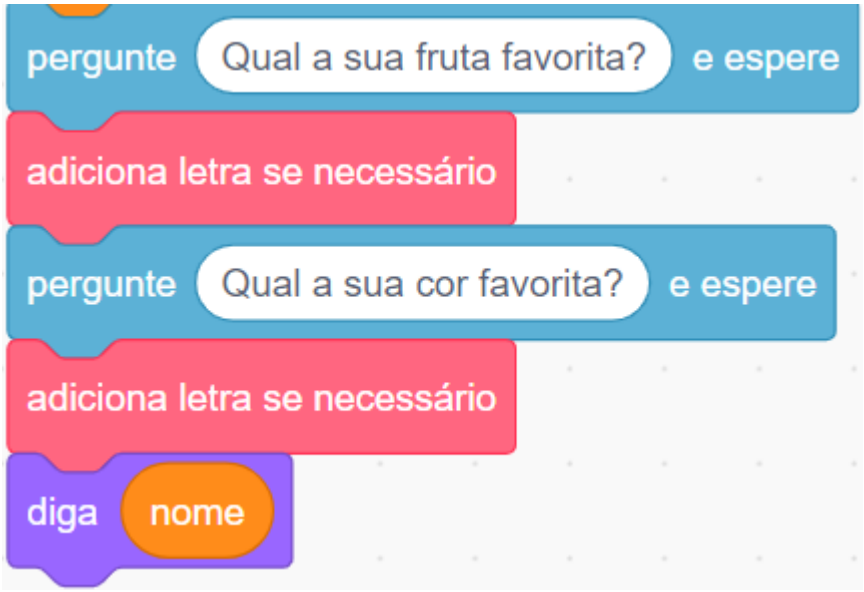


O que deveríamos fazer para adicionar o caractere ponto-final a todas as respostas?

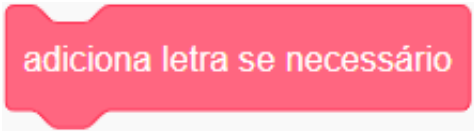
Para isso, removeremos o conjunto de blocos 

e adicionaremos em seu lugar o bloco 

, da seção *Meus Blocos*. Nosso código ficará da seguinte forma:



Feito isso, removeremos os demais conjuntos de blocos 

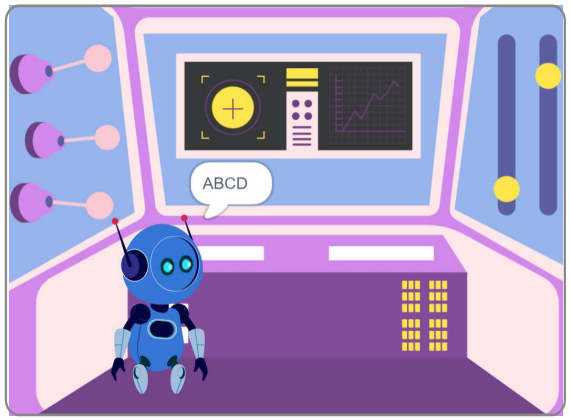
e colocaremos o bloco 

em seu lugar.

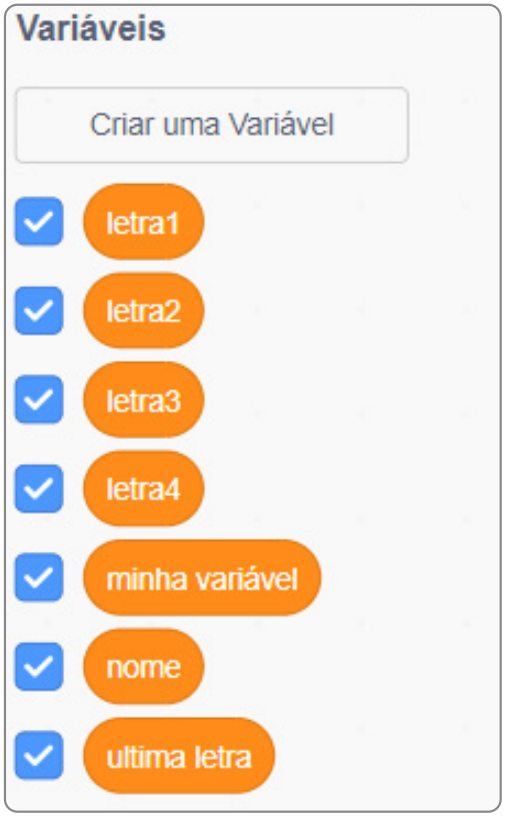
Nosso código ficará como mostrado a seguir:

Professor, leia o código com os estudantes, reforçando a lógica estabelecida de: pergunta, adiciona letra se necessário, pergunta, adiciona letra se necessário, e assim por diante, demonstrando que nosso código está otimizado.

Vamos testar? Clique no ícone bandeira verde e insira as letras A, B, C e D. Cada uma delas pode ser seguida de caracteres aleatórios. Observe:

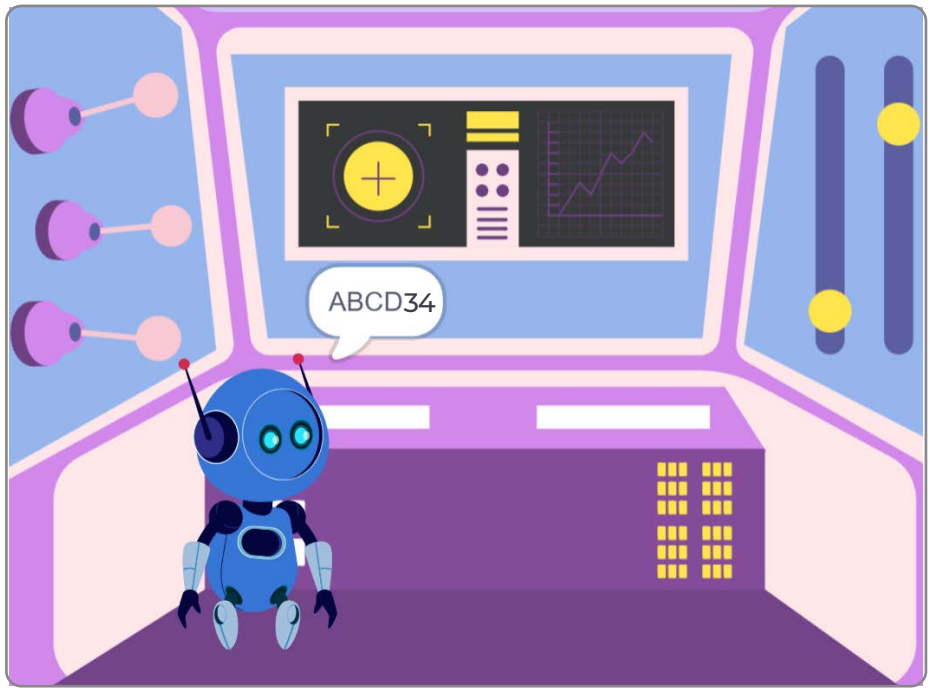


Agora, precisamos verificar o que está sendo armazenado. Assim, na seção *Variáveis*, clicaremos nas caixas de seleção correspondentes a *letra1*, *letra2*, *letra3*, *letra4*, *minha variável*, *nome* e *ultima letra*, tornando todas visíveis.



Professor, pense em um organizador de ferramentas no qual você pode escolher quais ferramentas estão visíveis na mesa para facilitar o trabalho. Na programação, temos algo semelhante chamado variáveis, que armazenam informações importantes para o código. Para ver todas essas variáveis e entender melhor o que está acontecendo, você precisa exibi-las. Isso é feito clicando na seção *Variáveis*, marcando as caixas ao lado de cada nome de variável. Assim, você coloca suas ferramentas na mesa para fácil acesso, podendo acompanhar e organizar melhor o seu código.

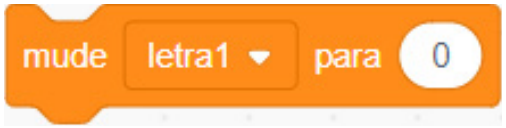
Para testar, iniciaremos cada resposta com os respectivos valores *1*, *2*, *3* e *4* junto de algumas letras, e, nas respostas 3 e 4, vamos inserir um ponto-final. Observe o resultado:



Por que o *ABCD* está no nome? Não digitamos *A*, *B* e *C*, apenas *1*, *2*, *3* e *4*. O 3 e o 4 estão sendo mostrados corretamente, devido a nossa regra de armazenamento, pois adicionamos ponto-final a essas respostas, mas, e o restante?

Relembrando a regra: pegamos a primeira e a última letra. Se a última letra for um ponto-final, adicionamos a primeira letra da resposta; caso contrário, deixamos como está e a primeira letra da resposta não é adicionada.

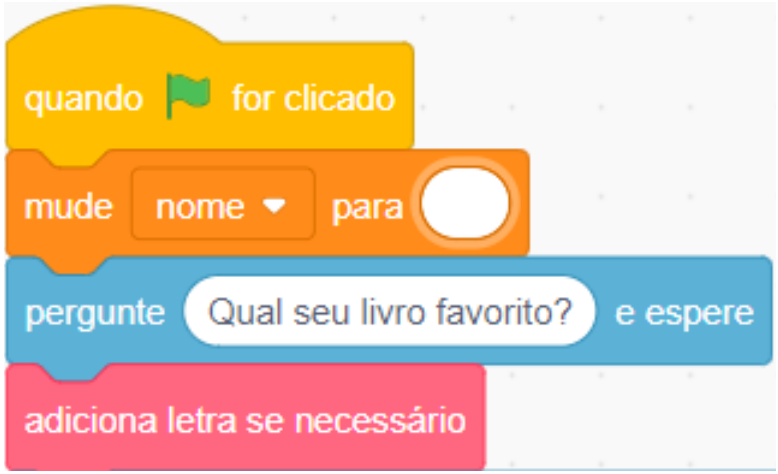
Isso ocorre pois o armazenamento não foi limpo e está exibindo uma informação antiga (lixo de memória). Para corrigir isso, precisamos iniciar o programa com o nome limpo. Dessa forma, quando iniciarmos o programa, precisamos mudar o nome para um texto vazio.

Assim, da seção *Variáveis*, arrastaremos o bloco 

e o colocaremos logo após o bloco 

. Modificaremos o menu suspenso para *nome* e a segunda lacuna deixaremos em branco. Então, teremos:

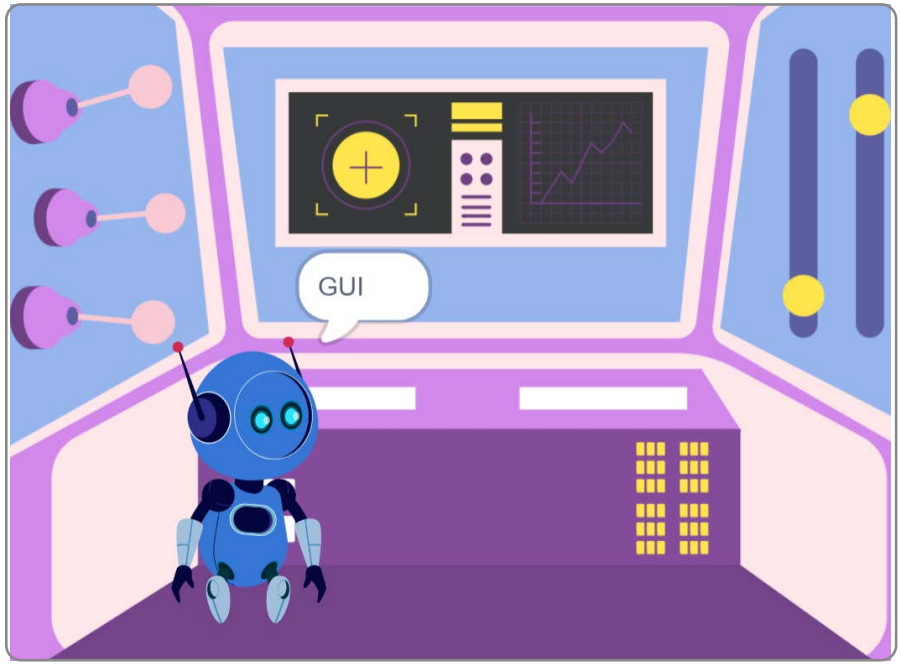
Lixo de memória é como uma gaveta desorganizada: quando você não remove itens antigos e desnecessários, eles se acumulam e dificultam que encontremos o que precisamos. No computador, isso significa que dados antigos e não mais utilizados ocupam espaço e podem deixar o sistema mais lento e menos eficiente. Manter a memória limpa ajuda o computador a funcionar melhor e mais rápido, assim como organizar uma gaveta facilita encontrar o que você precisa.



Finalmente, temos o que queríamos! Reinicie o programa e teste com um nome, como “Guilherme”, por exemplo.

Digitaremos *G* seguido de vários caracteres e ponto-final. Em seguida, *U* com vários caracteres e ponto-final. Por fim, *I* com vários caracteres e ponto-final. Por último, digitaremos qualquer sequência de caracteres, mas sem o ponto-final.

Quando o ponto não é incluído na resposta, a primeira letra dessa resposta não é mostrada. Portanto, nesse exemplo, o valor da variável *nome* será *GUI*.



Recapitulando: dividimos um código extenso em partes menores para facilitar a leitura e evitarmos repetições desnecessárias. Em seguida, encontramos e corrigimos o problema do lixo de memória, garantindo que a variável que armazena o nome começasse vazia a cada execução.

Até breve!

Quebrar o código em partes ajuda a torná-lo mais compreensível e gerenciável, assim como dividir um livro em capítulos facilita a leitura. Quando o código é dividido, é mais fácil identificar e corrigir erros, além de evitar repetições desnecessárias, tornando o processo mais eficiente e rápido.

Desafio

Nesta aula, aprendemos a organizar nosso código. Seu desafio será fazer a leitura de todo o código construído até o momento e verificar se a lógica dele está clara para você e se os nomes das variáveis fazem sentido com aquilo que elas executam dentro do código. Em seguida, descreva em tópicos a sequência de ações realizadas pelo código e compare suas anotações com a de um colega.

Espera-se que os estudantes pratiquem a habilidade de ler e analisar um código construído, verificando sua integridade e melhorando sua capacidade de compreensão daquilo que foi programado.

CLIQUE **AQUI** PARA AVALIAR ESTE MATERIAL