Desbastar

A área "Controle" contém blocos que controlam se outros blocos colocados dentro deles são executados.

Existem dois tipos de blocos de controle: blocos **se caso contrário** (descritos em uma página separada) e blocos que controlam a frequência com que seus interiores são executados. Os últimos são chamados de loops porque seu interior, também conhecido como corpo ou corpo do loop, é (possivelmente) repetido várias vezes. Cada execução de um loop é chamada de iteração.

Blocos para criar loops

repetir permanentemente

O **bloco de repetição permanente** executa o código em seu corpo até que o programa termine.

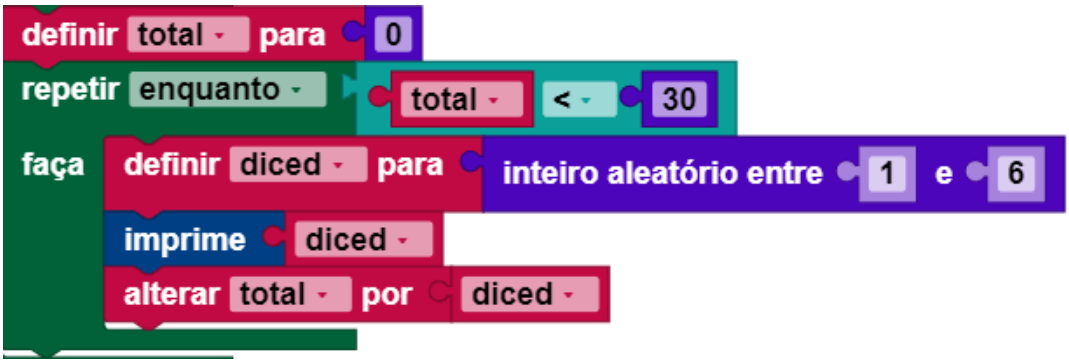
repetição

O bloco **repetição** executa o código em seu corpo quantas vezes forem especificadas. Por exemplo, o bloco a seguir diz "Olá!" Dez vezes:

repetir até

Imagine um jogo em que um jogador lança um dado e soma todos os valores lançados, desde que o total seja inferior a 30. Os blocos a seguir implementam este jogo:

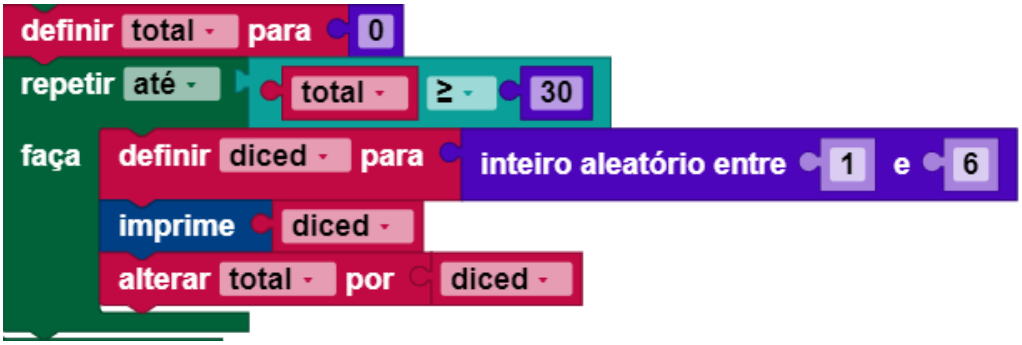
1. Uma variável chamada **total** recebe um valor inicial de 0.
2. O loop começa verificando se o **total** é inferior a 30. Nesse caso, ele passará pelos bloqueios do corpo.
3. Um número aleatório no intervalo de 1 a 6 é gerado (para simular um lançamento de dados) e armazenado em uma variável chamada **lançamento de dados.**
4. O número lançado é dado.
5. A variável **total** é aumentada pelo número de **lançamento de dados**.
6. Quando o final do loop é alcançado, o controle volta para a etapa 2.



Após o término do loop, todos os blocos subsequentes (não representados) são executados. No exemplo, o ciclo do loop termina após um certo número de números aleatórios no intervalo de 1 a 6 terem sido emitidos, e o valor da variável **total** tem a soma desses números, que é pelo menos 30.

repetir até

Loops **repetir até** repetem seu corpo **até** que uma condição seja atendida. loops de **repetição** são semelhantes, exceto que eles repetem seu corpo **até** que uma determinada condição seja atendida. Os blocos a seguir são equivalentes ao exemplo anterior porque o loop continua até que o **total** seja maior ou igual a 30.



contagem de até

O loop de**contagem de até** incrementa o valor de uma variável, começando com um primeiro valor, terminando com um segundo valor e em etapas a partir de um terceiro valor, executando o corpo uma vez para cada valor da variável. Por exemplo, o programa a seguir imprime os números 1, 3 e 5.

Como mostram os dois loops a seguir, que geram os números 5, 3 e 1, esse primeiro valor pode ser maior que o segundo. O comportamento é o mesmo, independentemente de o valor incremental (terceiro valor) ser positivo ou negativo.

para cada

O bloco **para cada** é semelhante ao loop de **contagem de até**, exceto que, em vez de usar a variável de loop em uma ordem numérica, ele usa os valores de uma lista em sequência. O programa a seguir gera cada elemento da lista "alfa", "beta" e "gama":

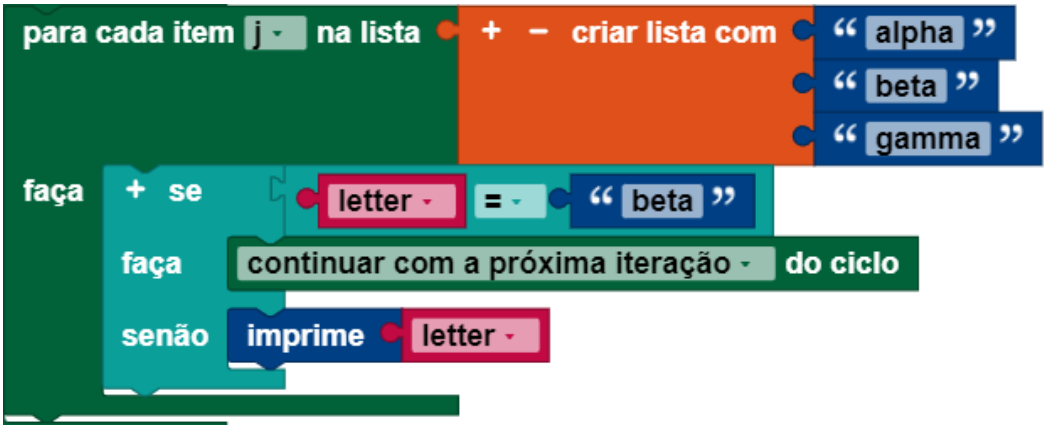
Blocos de cancelar loop

A maioria dos loops é executada até que a condição de término (para **blocos de repetição**) seja satisfeita ou até que todos os valores da variável do loop sejam aceitos (para loops **contar com** e **para cada**). Dois blocos raramente usados, mas ocasionalmente úteis, oferecem opções adicionais para controlar o comportamento do loop. Eles podem ser usados com qualquer tipo de loop, embora os exemplos a seguir mostrem seu uso com o loop **para cada**.

continuar com a próxima interação

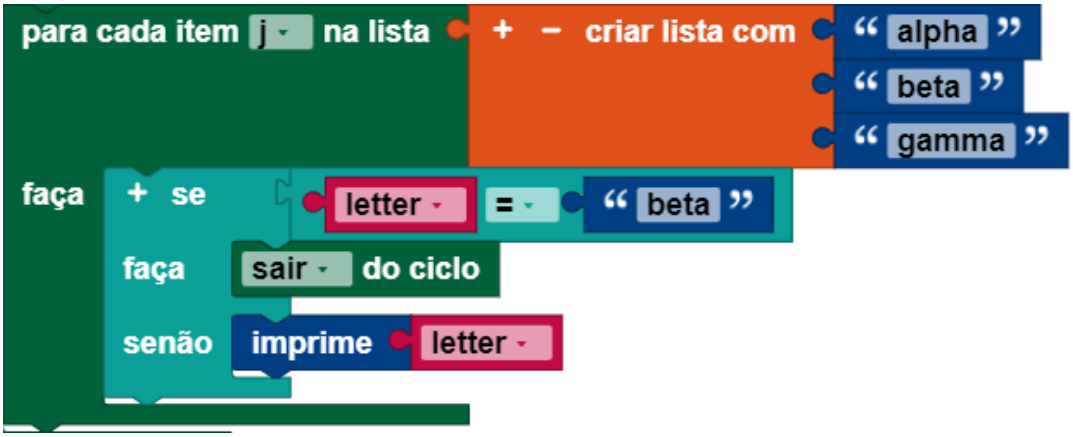
continuar com a próxima iteração faz com que os blocos restantes no corpo do loop sejam pulados e a próxima iteração do loop comece.

O programa a seguir imprime "alfa" na primeira iteração do loop. Na segunda iteração, o bloco **continuar com a próxima iteração** é executado, sendo a saída de "beta" ignorada. A última iteração imprime "gama".



Cancelamento de loop

O bloco de **cancelamento de loop** permite uma saída antecipada de um loop. O programa a seguir gera "alfa" na primeira iteração e interrompe o loop na segunda iteração se a variável do loop for igual a "beta". O terceiro ponto da lista nunca é alcançado.



Revision #4
Created 17 November 2021 21:07:37 by Admin
Updated 18 February 2022 14:27:41 by Admin