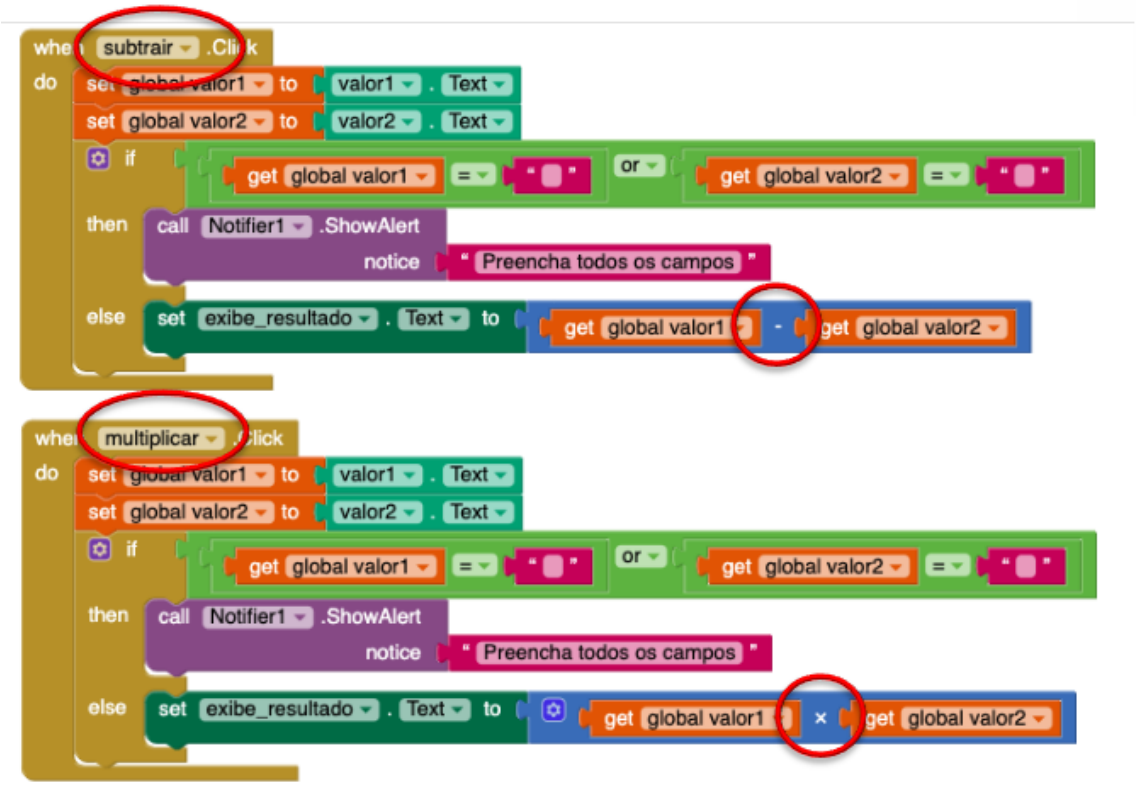
6.Procedimentos

6.1 Conceito no algoritmo

Nos exemplos práticos que vimos até a última aula, podemos perceber que a construção do algoritmo está muito repetitiva. As únicas alterações entre os conjuntos de blocos são os nomes dos botões que são clicados e a operação que é realizada. Vamos ver um trecho da construção de blocos:



Restam ainda as operações de dividir e somar, que são iguais a estas (com exceção da operação de dividir que tem uns blocos a mais).

Muito improdutivo, não acha? Imagine a seguinte situação: a simples necessidade de mudar o texto “Preencha todos os campos” requer que você execute a mesma alteração em 4 lugares diferentes. Isto não é nada prático.

No exemplo acima, praticamente o mesmo trecho de código é executado quando há um “click” sobre cada botão de operação. E se pudéssemos isolar este

trecho de código e dar um nome para ele, assim poderíamos executar o código sempre que precisarmos dele. Pois este é exatamente o conceito de *procedure* (procedimentos), que pode ser de 2 tipos:

- **Com retorno (*return*):** executa um trecho de código e retorna um valor ao terminar. Por exemplo, uma procedure pode ter um código para executar um cálculo e retornar seu resultado.

- **Sem retorno**: apenas executa seu trecho de código.

As *procedures* também podem receber valores para sua execução. Esses valores são *variáveis de entrada* e são chamadas de ***parâmetros***. Vamos reescrever nosso algoritmo usando *procedures* com retorno e parâmetros.

Algoritmo original	Algoritmo com Procedure e Parâmetros
[1] Salário = 1000 [2] Tem_Filhos = FALSE [3] SE Tem_Filhos [4] ENTÃO [5] Imposto = 50 [6] SENÃO [7] Imposto = 100 [8] FIM SE [9] Salário_Final = Salário – Imposto	[1] PROCEDIMENTO **Calcula_Salário** (Salario_in, Filhos_in) [2] SE Filhos_in [3] ENTÃO [4] Imposto = 50 [5] SENÃO [6] Imposto = 100 [7] FIM SE [8] RETORNE (Salario_in - Imposto) [9] FIM_PROCEDIMENTO [10] [11] Salário = 1000 [12] Tem_Filhos = FALSE [13] Salário_Final = **Calcula_Salário** (Salário, Tem_Filhos)

Vamos entender como ficou nosso algoritmo com procedure e parâmetros, em comparação com nosso algoritmo original:

- Primeiro, toda a parte da lógica do programa foi separada e agrupada entre PROCEDIMENTO e FIM_PROCEDIMENTO, entre as linhas 1 e 9.

- O procedimento está aceitando como entrada 2 parâmetros, destacados de amarelo e verde. Repare que a estrutura do nome do procedimento e de seus parâmetros é exatamente a mesma de seu uso. Esta estrutura e ordem são chamadas de ***assinatura*** e precisam ter esta correspondência para seu correto funcionamento.

- As variáveis “Salário” e “Tem_Filhos” são passadas para dentro do procedimento, através de seus ***parâmetros***. Dentro do procedimento os parâmetros podem ter outros nomes, pois não são mais as mesmas variáveis. Para exemplificar isto, chamamos elas de “Salario_in” e “Filhos_in” (“*in*” significa entrada em inglês).

- Na linha 8 executamos o cálculo aritmético e retornamos (*RETURN*) os valores para a linha que “***chamou***” (*CALL*) o procedimento, que foi a linha 13.

- Na linha 13, por fim, é feita a chamada do procedimento respeitando sua assinatura. O procedimento é executado e retorna o valor calculado. Este valor é então atribuído à variável “Salário_Final”.

Certo, agora você deve estar se perguntando: como isto pode ser mais eficiente se está fazendo a mesma coisa, porém com mais linhas que o anterior, que tinham apenas 9?

Se você estava se questionando sobre isso, saiba que fez uma excelente observação. Pois bem, lembra que em nossos exemplos tínhamos 2 funcionários, um com filhos e outro sem? Vamos ver como ficaria o algoritmo agora, considerando os dois funcionários da empresa:

Algoritmo original	Algoritmo com Procedure e Parâmetros
[1] Salário = 1000 [2] Tem_Filhos = FALSE [3] SE Tem_Filhos [4] ENTÃO [5] Imposto = 50 [6] SENÃO [7] Imposto = 100 [8] FIM SE [9] Salário_Final_1 = Salário- Imposto [10] [11] Salário = 1500 [12] Tem_Filhos = TRUE [13] SE Tem_Filhos [14] ENTÃO [15] Imposto = 50 [16] SENÃO [17] Imposto = 100 [18] FIM SE [19] Salário_Final_2 = Salário- Imposto	[1] PROCEDIMENTO Calcula_Salário (Salario_in, Filhos_in) [2] SE Filhos_in [3] ENTÃO [4] Imposto = 50 [5] SENÃO [6] Imposto = 100 [7] FIM SE [8] RETORNE (Salario_in - Imposto) [9] FIM_PROCEDIMENTO [10] [11] Salário_Final_1 = Calcula_Salário (1000, FALSE) [12] [13] Salário_Final_2 = Calcula_Salário (1500, TRUE)

Agora podemos ver a vantagem do uso de ***procedure***, certo? Vamos reparar outras coisas importante que aconteceram neste novo código:

- Primeiro, podemos reparar que as linhas de 3 a 8 são exatamente iguais às linhas de 13 a 18.

- Segundo, e mais importante, vamos reparar que não precisamos usar variáveis quando precisamos passar os valores para a procedure. Podemos **passar seus valores diretamente** como parâmetros.

O uso de ***procedures*** **sem retorno é mais simples**, pois não possuem retorno (*RETURN*) e não precisam ser atribuídas (depois) de uma variável. Seu uso é normal no decorrer do código, vamos ver um exemplo disso:

```
[1] Salário = 1000
[2] Tem_Filhos = FALSE
[3] Executa_Outros_Comandos_Iniciais()
[4] SE Tem_Filhos
[5]     ...
```

Um outro detalhe importante é que os parâmetros não são obrigatórios, como vemos neste exemplo.

Agora, vamos aprender um pouco na prática.