Estudo de Laços de Repetição

Laços de repetição é uma função que a linguagem de programação usa para contar algo,
basicamente é uma condição que vai fazer que o código se repita determinado número de vezes.

Vamos simplificar: Imagine um Buraco Negro, lá no espaço sugando tudo para dentro do seu centro, se você olhar bem para um buraco negro você vai ver em volta dele existem vários objetos interestelares orbitando, antes de cair no nada eterno, mas a cada volta esses objetos vão chegando perto do ponto central. E eles ficam girando e girando para cada vez mais perto do centro, até que a força gravitacional deles não consegue mais se segurar e são sugados para o centro do buraco negro.

Um loop de repetição funciona assim, enquanto tiver dentro das condições loop continua rodando, até a condição se cumprir.

Para laços de Repetição temos duas funções

While – Enquanto (isso) {faça aquilo}

For – Para (isso) {faça isso}

Vamos começar com Enquanto ( )… While…

WHILE

Veja isso por exemplo:

Enquanto (forca_gravital_objeto > forca_do_buraco_negro) Faça “De mais uma volta”

Vamos passar para Linguagem de Programação: (Lembrando que a linguagem nativa do computador é o inglês)
Também vamos simplificar o nome das variáveis.

While (FG_Obj > FG_BN) { Volta_1 + 1 }

(É a mesmo código de cima não se confunda, apenas foram mudados os nomes para que se encaixe melhor numa programação real)

Vamos traduzir, muita calma: Enquanto a força gravitacional do objeto for mais que a do buraco negro ele não será puxado e continuará rodando infinitamente até que algo aconteça para a força do buraco negro se tornar mais forte.

Laços “while” vão executar sequencialmente e para sempre, enquanto a expressão dentro dos parênteses ( ) for verdadeira. Algum evento deve mudar o estado da variável, ou então o laço “while” não terminará nunca.

Por exemplo, seu computador faz várias repetições para saber se você está conectado a internet.
Se esse teste der resultado positivo, nada acontece.
Porém, se ele testar e ver que a conexão não existe mais, ele manda uma mensagem dizendo que você perdeu a conexão.

Fazer testes você já está careca de saber.
Porém, como faria todos esses testes? Colocaria um if else a cada segundo, durante 1h?
E se o usuário passar 2h conectado? Ou 1 dia?

Outro exemplo.

Um agrônomo precisa que sua plantação seja regada cada 5 minutos durante o dia enquanto a temperatura do solo se manter acima dos 29°C, para isso ele vai usar um módulo no solo, acoplado ao sistema de irrigação, e precisará de um programa instalado nesse módulo que cumpra as condições solicitadas.

Int temperatura

While (temperatura > 29) {“regar plantas” delay (300000);}

Bem simples de resolver né.

Então você agora

1. Uma empresa precisa manter a pressão de gases de uma máquina hidráulica a 156 bar de pressão, e precisa de um programa que mantenha e estabilize essa pressão enquanto a máquina estiver trabalhando com 5 prensas ligadas.

2. Uma empresa de placas solares precisa ajustar o módulo de controle do gerador de energia que deu problema e foi desconfigurado, crie um código para que mantenha o gerador ligado enquanto a placa solar estiver recebendo menos que 144 kilowats de energia.

3. Uma viatura da polícia militar será instalada com uma sirene que funciona enquanto o botão de dentro do veículo for ativado. Crie esse código.

4. Crie um laço de repetição que ESCREVA de 10 a 1 usando somente WHIILE.

5. Crie um laço de repetição que some variável X com a variável Y, e Escreva a soma das variáveis, enquanto X for maior que Y.

6. Crie um programa para dizer quais dos 300 alunos passaram, a média da escola é 6.

Função FOR (PARA)

Atenção: Não confunda, Enquanto ( ) com Se ( ) e nem com Para ( ), Enquanto e Para são laços de repetição e agem diferente, e Se é uma estrutura de condição, um erro muito comum é confundir o uso desses termos.

Agora vamos falar de Para, e novamente meus amigos, muita calma nessa hora, não estou falando de um Estado do Brasil, e muito menos de alguém parado, não é verbo e sim preposição, para ele, para ela, então ok, você entendeu podemos seguir:

Não sei se é só comigo mas ao falar para eu automaticamente quero direcionar para alguém, pensamos automaticamente em um sujeito, em alguém, e voce??

A função For também funciona assim:

Para que eu possa folgar (quarta feira/ já chegou a sexta?/ se não soma mais um dia então) {enquanto isso vamos ir para o trabalho, fazer o que}

Agora vamos transformar para Linguagem de Programação:

For (dia_da_semana = 3, dia_da_semana > 4, dia_da_semana ++) { “mais um dia de trabalho”}

A estrutura do For ficou clara? Não?

For (Início, Teste, Incremento) { repete até chegar no número que está na condição}

E agora?

O que aconteceu?
Vou te explicar:

Estávamos no terceiro dia da semana dia 3, quarta feira, e comparamos para ver se já é o quinto dia, e enquanto não chegamos no dia 5 da semana, que no caso é sexta, continuamos a ir para o trabalho por mais 2 vezes, até que chegou a gloriosa sexta e o código foi interrompido. Se você viu que colocamos > 4 é simplesmente por que o primeiro número maior que 4 é 5.

Sei que não gostou muito desse exemplo rsrsrs nem eu kkk então vamos para outro exemplo.

Vamos produzir um LED, para que ele pisque 6 vezes, exatamente 6 vezes apenas, é bem simples.

Para (led = 1, verifica se já piscou 6 vezes, se não pisca mais uma) { led acende, pausa, led apaga, pausa)

Vamos sofisticar nosso código ainda este feio para nosso computador.

For (led = 1, led > 6, led ++) { (led, HIGH), delay(2000), (led, LOW), delay (2000) }

Simples, acho que você já está ficando craque.
O valor do led começa em 1, o programa verifica se já chegou a piscar 6 vezes se não ele soma mais 1 na variável led (led = 1 + 1) e pisca uma vez. O programa continua a fazer isso 6 vezes até que o valor que está no led se torne maior que 6, é simples se começou do 1, 1 + 6 = 7, ele pisca 6 vezes até que se some 7 dentro da nossa variável led.

Pode ser difícil de entender de primeira, sugiro uma releitura deste exemplo.

Continuando…

Podemos ver uma clara diferença entre o While e entre o For, o While faz repetição sem parar enquanto a condição for verdadeira, e o For repete até que sua condição se torne verdadeira. Os dois podem ser usados para gerar um loop de repetição mas fica o questionamento, você que repetir um número de vezes especifico enquanto a condição é falsa? Então use for, por outro lado, se você quer repetir infinitamente até que a condição se torne falsa, use While, lembrando que você pode caso seja possível usar um WHILE no lugar de FOR e vice e versa.

Um último exemplo antes de seguirmos para prática.

Digamos que estou fazendo um bolo, liguei o forno a 24 graus e preciso verificar se a temperatura do forno já esta boa, e enquanto não está boa preciso esperar aumentar a temperatura do forno, até que chegue a acima de 170 graus. Lembrando que o forno aquece a cada 5 graus.

Você consegue identificar qual é função se é FOR ou WHILE?

Veja que enquanto não chega na temperatura ideal ele continua aumentando, e irá para até que a condição se cumpra, então é um FOR;

FOR (temperatura == 24, temperatura >170, temperatura=temperatura+5) {“Ainda não esta quente o suficiente, espere” temperatura, delay (30000); }

Quantas vezes o usuário vai ver esta mensagem que não esta quente??

Simples, vai aumentar de 5 em 5 em 30 segundos até que chegue acima de 170, começando a partir do 24

170-24 = 146

146 / 5 = 29,2 como tem que passar de 170 somamos mais 1 grau = 30,2

Aproximadamente 30 vezes.

Quer tirar a prova?

30*5 = 150
Mais os 24 que já tinha 150+24=174

O usuário vai ver 30 vezes a mensagem até que a temperatura chegue a 174 graus.

E o tempo? 30 segundos * 30 aumentos = 900 segundos que equivale a 15 minutos.

A programação está muito ligado aos cálculos matemáticos, pois precisamos usar as operações matemáticas para definirmos valores ou repetições.
Agora Vamos para os exemplos de códigos reais de variadas maneiras de contagem.

i. Contador duplo com soma de x e y

```
//imprime os números de 0 a 498 em incremento de 2
int main()
{
 int x,y;
 for(x=0, y=0; x+y < 500; x=x+1, y=y+1)
 printf("%d ",x+y);
system("PAUSE");
 return 0;
}
```
0 2 4 6 8 10 12 ....494 496 498

ii. Maneiras de contar até 10

```
for(i = 1; i<=10; i++)
 printf("%d ", i); ⇒ 1 2 3 4 5 6 7 8 9 10

 for(i = 1; i<=10; i=i+1)
 printf("%d ", i); ⇒ 1 2 3 4 5 6 7 8 9 10

 for(i = 10; i>=1; i--)
```

printf("%d ", i); ⇒ 10 9 8 7 6 5 4 3 2 1

iii. Outras formar de contagem

```
for(i = 1; i<=10; i+=2)
printf("%d ", i); ⇒ 1 3 5 7 9

for(i = 10; i>=1; i=i-3)
printf("%d ", i); ⇒ 10 7 4 1

for(i = -10; i<=10; i=i+5)
printf("%d ", i); ⇒ -10 -5 0 5 10
```

Exercícios de Fixação.

Faça um contador regressivo de 20 segundos, aparecendo a mensagem espere sua vez.

Faça um contador que escreva tic e tac a cada segundo, durante 1 minuto.

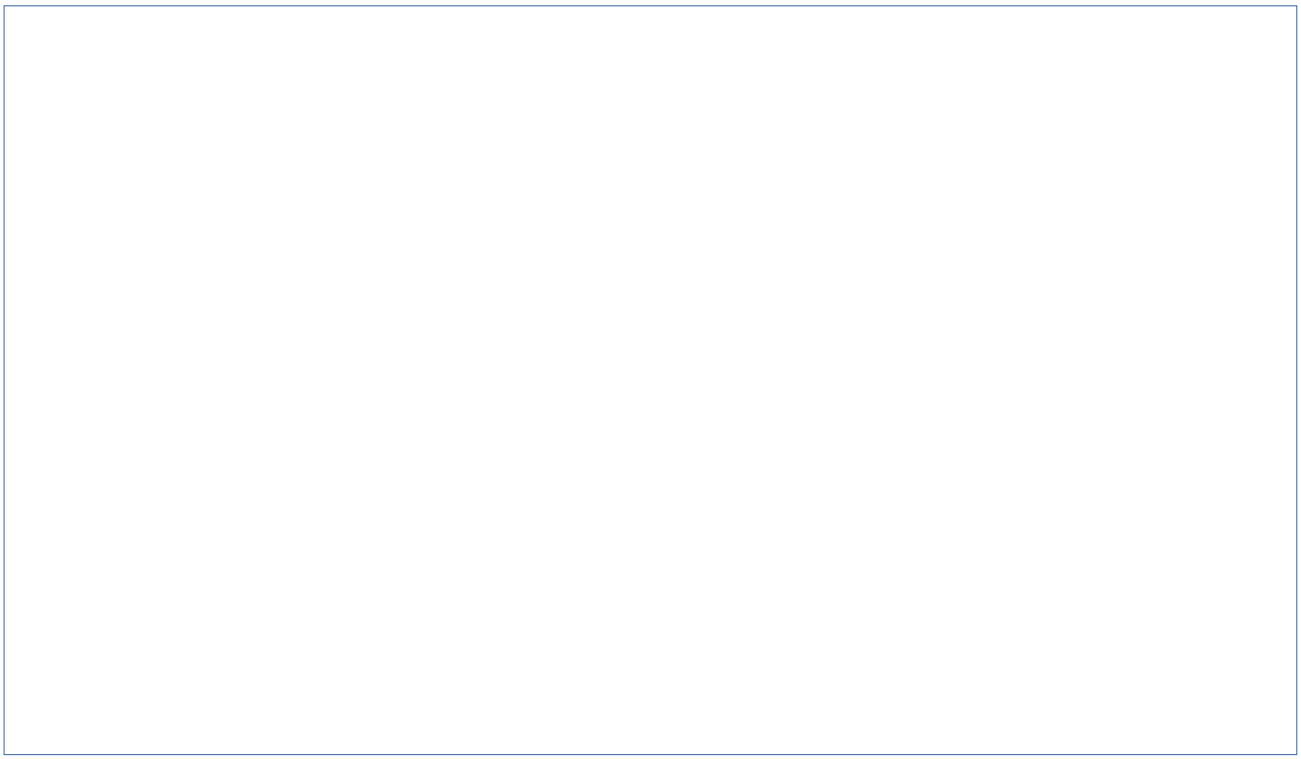
Faça um led acender e apagar 6 vezes a cada 1 segundo

Usando FOR.

Desafios

1 – Um padeiro precisa de um programa que lhe apresente uma mensagem a cada 10 minutos que a leva de pão que ele colocou no forno já está no ponto, e que já pode colocar outra leva de pães, o padeiro separou 10 levas de pães, e acabou de colocar a primeira leva. Desenvolva esse programa para o Padeiro usando apenas uma função FOR, até que acabe as levas de pães.

2 – Um engenheiro precisa calcular o histórico de compras de vigas de uma empreiteira, para isso ele definiu um limite de 15 vigas por inventário, o programa analisará a cada segundo a quantidade, e a cada compra o programa vai atualizar a quantidade e se chegar a 100 vigas vai avisar que o inventário está cheio. Você vai usar FOR e IF para resolver essa questão.



Fim de mais um módulo de programação, lembre-se esse conteúdo só será absorvido se você leu os módulos de programação básica, estruturas de condição, e de variáveis.

Também depois de absorver esse conteúdo você entenderá melhor sobre programação e já poderá idealizar seus primeiros programas, vale lembrar que o conteúdo exposto aqui está simplificado e feito para pessoas do 0, então é válido procurar por outras fontes para complementar seu conhecimento.