**Mário Fácil – Pseudocódigo**

1. **Perguntar ao usuário qual a altura da pirâmide, sendo que a mesma só pode estar entre 1 e 8, não sendo permitido outros números;**

Para perguntar ao usuário um número inteiro, usamos a função:

**“get\_int(“pergunta”);”**

só que este valor tem que ser armazenado em algum lugar, logo antes da função declaramos a variável na qual irá ser armazenado o valor:

**int “variável” = get\_int(“pergunta”);**

Porém, como não podemos permitir que o número seja menor que 1 ou maior que 8, temos que criar um loop condicionado para que fique perguntando até que o usuário digite um número aceitável.

Existem 3 versões de variáveis de loop, elas são:

**while(“condição”)**

**{**

**“código que quer que se repita”**

**}**

Que basicamente significa enquanto tal condição não for concluída, fique repetindo essa função;

**for ( “tipo da variável” “variável” = “valor”; “condição”; "variável++ ou variável--“)**

**{**

**“código que quer que se repita”**

**}**

Que faz basicamente a mesma coisa só que tem um contador embutido, ou seja, ele irá ficar repetindo até que a condição seja concluída, e a cada vez que ele rodar o código, ele irá somar ou subtrair 1 da variável contador;

e por fim, mas não menos importante temos a variável de repetição

**do**

**{**

**“código desejado que se repita”**

**}**

**While (“condição”);**

Diferente das outras, a “do while” sempre executa pelo menos uma vez, ou seja, no nosso caso, ela irá perguntar ao usuário qual o valor e na condição irá verificar se aquele valor está entre 1 e 8, caso não esteja, ele irá perguntar novamente ao usuário o valor, repetindo até que o mesmo escolha um número entre 1 e 8.

1. **Fazer com que o computador imprima a quantidade de “#” necessárias naquela linha;**
2. **Assim que terminar a linha, tem que passar para a próxima linha, para isso é necessário criar uma variável que controle a altura da pirâmide;**

Os passos 2 e 3 são meio que executados juntos, então vou tentar explicar os dois da melhor forma possível.

Para fazermos com que o computador repita o mesmo passo várias vezes, precisamos de uma ferramenta de loop, como falei antes existem 3, mas como saber qual usaremos agora? Simples, para sabermos em qual linha estamos, precisamos de um contador, logo, usaremos a função “for”.

Agora vamos pensar no que o computador tem que fazer, primeiro ele vai escrever os #? Não, primeiro ele vai te falar em qual linha você está, por que a linha quem te dirá quantos # o computador precisa digitar.

Ex: linha 1 = 1 #

Linha 2 = 2 #

E assim por diante, então inicialmente iremos criar um for para descobrir em qual linha estamos:

**For (int y = 0; y < “variável com o número de linhas”; y++)**

**{**

**“código desejado”**

**}**

Veja que a variável “y” é um contador de quantas vezes esse algoritmo foi repetido, ou seja, a na primeira iteração o valor dele será 0, mas ao final da mesma, ele será 1 e ele irá se repetir enquanto o valor de “y” for menor do que o escolhido pelo usuário.

Após fazer com que o computador conte a altura da escada, temos que fazer com que ele escreva os “#”, mas onde devemos colocar o próximo loop? Abaixo da última chave? Dentro do “for” da altura, onde fica escrito “código desejado”? Temos que colocar dentro do “for”, mas porquê?

Pense comigo, se colocarmos depois do “for” da altura, o computador irá passar por todas as alturas para depois começar a digitar os “#”, mas precisamos que ele digite em cada altura, ou seja, ele tem que estar dentro do “for”, mas como funciona isso?

Ao ler o algoritmo, o computador irá começar a rodar o loop da altura, ou seja, verá que estamos na altura zero, logo ele irá para o “for” do “#”, que verá que estamos nessa linha e digitará o número de “#” necessários, depois o computador voltará para o loop da altura, indo para a linha 1, repetindo o ciclo com o “for” dos “#”.

Ficando assim:

**For (int y = 0; y < “variável com o número de linhas”; y++)**

**{**

**For (int x = 0; x < y; x++)**

**{**

**“código necessário”**

**}**

**}**

Observe que a variável x adicionada neste “for” serve para contar o número de “#” que se deve ser impresso na tela, ou seja, ele irá se repetir até que o “x” seja menor que “y” (que é a altura da pirâmide”

Chegando aqui, só falta adicionar a função de impressão, “printf”.

**Printf(“texto escolhido”, “variável se necessária”);**

Vamos explicar um pouco sobre o printf. Essa função tem como intuito mostrar um texto ao usuário, algo gráfico, pode ser somente um texto ou o valor de uma variável escolhida.

Nesse caso vamos imprimir somente “#”, não temos uma variável, logo, o printf fica assim:

**Printf(“#”);**

Adendo, para que o computador não imprima vários “#” alinhado, temos que dar uma quebra de linha, para fazermos isso, usamos a mesma função printf, contudo o conteúdo dela será “\n”, onde \n é um código que significa quebra de linha.

Eu dei algumas variáveis, alguns exemplos, mas cabe a você colocar tudo nos seus devidos lugares, infelizmente é assim que se aprende.

**Com esse pseudocódigo teríamos o seguinte resultado:  
#**

**##**

**###**

**####**

**#####**

**######**

**#######**

**########**

1. **Inverter a escada;**

Beleza, mas aqui não chegamos ao objetivo do exercício, que é a escada invertida, subindo.

Pensa comigo, se eu escrever “.” antes das “#”, eu consigo fazer com que a escada fique do outro lado correto? Pois o computador digita na seguinte ordem “12345678”, então se os dígitos de 1 a 7 forem “.” e o 8º dígito for a “#”, eu teria a primeira linha da pirâmide perfeitamente, correto?

Seguindo essa linha de raciocínio, e sabendo que podemos escrever um certo número de dígitos usando o loop for, do while ou while, então o que precisamos é de um contador que nos diga quantos “.” precisamos colocar em cada linha.

Concorda que a quantidade de “.” é inversamente proporcional a linha que estamos?

**Ex: linha 1 – precisamos de 1# e 7”.” = 8 dígitos**

**Linha 3 – precisamos de 3# e 5”.” = 8 dígitos**

**Linha 5 – precisamos de 5# e 3 “.” = 8 dígitos**

**Linha 8 – precisamos de 8# e 0”.” = 8 dígitos**

Ou seja, precisamos de uma variável que diminua enquanto a linha aumenta, como podemos fazer isso?

Se pegarmos o valor total “8” e subtrairmos o valor da linha “1”, teríamos 7 dígitos, logo podemos armazenar em uma variável e sempre que mudarmos a linha, irá mudar o número na conta e consequentemente o número de dígitos também irá diminuir:

**Int “variável” = “variável altura” – “y”;**

Aqui eu vou colocar mais um adendo, lembra que a linha começa no 0 e nós não temos a linha 0 só da linha 1 em diante, logo para corrigir esse erro, vamos acrescentar um “-1” à equação, sendo assim o cálculo dará certo.

**Int “variável” = “variável altura” – “y” - 1;**

Ok, agora temos que fazer o loop que irá imprimir os “.”

Usaremos o for, do while ou while?

O for não serve pois já temos a nossa variável contador fora do loop. O do while não serve pois não queremos que o programa execute o loop pelo menos uma vez, queremos que execute a quantidade exata de vezes, logo, só nos resta o while.

**While (“variável da conta” > 0)**

**{**

**Printf(“.”);**

**“variável da conta”--;**

**}**

Em outras palavras, o que estamos dizendo ao computador é que, enquanto o valor de espaços for maior que zero, digite um “.” e subtraia um da variável.

Ex: variável está com o número 7 armazenado nela, então ela irá imprimir um “.” e diminuir o número para 6, em seguida ela irá imprimir “.” e diminuir o número para 5, e irá repetir esse processo enquanto o número for maior que 0, chegando em 0 ele irá parar e passará para o “for” que imprime o “#”.

**Adicionando esse pseudocódigo a mais no antigo, teremos o seguinte resultado:**

**.......#**

**......##**

**.....###**

**....####**

**...#####**

**..######**

**.#######**

**########**

Depois de ter certeza que deu certo, é só alterar o **printf(“.”)** para **printf(“ “)**

**Caso ainda tenha dúvidas, o algoritmo dessa questão está na última página.**

#include <stdio.h>

#include <cs50.h>

int main(void)

{

    int number;

    do // solicita ao usuário um número entre 1 e 8

    {

        number = get\_int("Escreva um número entre 1 e 8: ");

    }

    while (number < 1 || number > 8);

    int control = number;

    // Variável "y" escolhe a linha da pirâmide

    for(int y = 0; y < control; y++)

    {

            // Variável "control\_space" imprime o número de " " necessários para fazer a pirâmide inverter

            int control\_space = control - y - 1;

            while(control\_space > 0)

            {

                printf(" ");

                control\_space--;

            }

            // Variável "x" escolhe quantas vezes tem que repetir o "#" na linha

            for (int x = 0; x <= y; x++)

            {

                printf ("#");

            }

        printf ("\n");

    }

}