LAB: “Mario (*less comfortable*)”

Abrantes Araújo Silva Filho

Revisão: 2023-05-04

Resumo

Este laboratório corresponde ao “Mario (*less comfortable*)” da disciplina **Harvard CS50**, em sua versão integral, traduzido para o português.

Sumário

1 Introdução

Este laboratório corresponde ao “Mario (*less comfortable*)” da disciplina **Harvard CS50**, e deve ser feito pelos alunos que ainda não estão se sentindo confortáveis com a programação na Linguagem C.

O objetivo deste laboratório não é que você se torne um especialista em C mas, sim, que você comece a aprender os conceitos fundamentais da computação e da programação. A tradução para o português foi feita com base na versão de 2023 do laboratório, conforme o laboratório original[1].

[1] https://cs50.harvard.edu/x/2023/psets/1/mario/less/

2 Mundo 1-1

Ao chegar próximo do final do "Mundo 1-1" no jogo **Super Mario Brothers**, para Nintendo, o personagem Mario deve subir em uma pirâmide de blocos alinhados à direita, como na Figura 1 abaixo:

Figura 1: Pirâmide de blocos

Fonte: Harvard CS50

Seu trabalho, neste laboratório, é recriar essa pirâmide utilizando a linguagem C, em uma representação textual, utilizando cerquilhas (#) para substituir os blocos da pirâmide, como ilustrado abaixo (como cada cerquilha tem a altura um pouco maior do que a largura, nossa pirâmide também será um pouco mais alta do que larga):

```
       #
      ##
     ###
    ####
   #####
  ######
 #######
########
```

Criaremos um programa chamado `mario_less`, que será o programa responsável por imprimir a pirâmide. Além disso também permitiremos que o usuário decida a altura da pirâmide, solicitando que o usuário informe um número inteiro positivo entre 1 e 8 (os extremos estão incluídos).

Aqui está um exemplo de como o programa deverá funcionar se o usuário informar o número 8 para a altura da pirâmide:

```
$ ./mario_less
Altura: 8
        #
       ##
      ###
     ####
    #####
   ######
  #######
########
```

Aqui está um exemplo de como o programa deverá funcionar se o usuário quiser uma pirâmide com altura 4:

```
$ ./mario_less
Altura: 4
   #
  ##
 ###
####
```

Aqui está um exemplo de como o programa deverá funcionar se o usuário quiser uma pirâmide com altura 2:

```
$ ./mario_less
Altura: 2
 #
##
```

Aqui está um exemplo de como o programa deverá funcionar se o usuário quiser uma pirâmide com altura 1:

```
$ ./mario_less
Altura: 1
#
```

Se o usuário não informar um inteiro positivo entre 1 e 8 o programa deve solicitar novamente a altura até que um número válido seja informado:

```
$ ./mario_less
Altura: -1
Altura: 0
Altura: 42
Altura: 50
Altura: 4
    #
   ##
  ###
 ####
```

3 Passo a passo

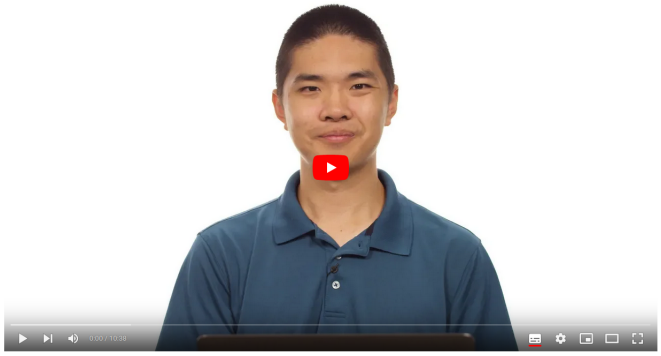
Como começar a resolver esse problema? Bem, vamos abordá-lo um passo de cada vez. Lembre-se que a **decomposição** de um problema em partes menores que podem ser solucionadas de modo mais fácil é uma habilidade importante do **pensamento computacional** que você precisa desenvolver.

Nesta seção vamos mostrar o caminho das pedras para que você possa, por conta própria, construir seu programa!

3.1 Vídeo com dicas

Você deve começar assistindo ao vídeo de dicas preparado por **Brian Yu**, da equipe da disciplina Harvard CS50. O vídeo está disponível no YouTube[2] (apenas em inglês, com legendas também em inglês):

Figura 2: CS50 Walkthroughs: Mario (less)



Fonte: Harvard CS50

[2] https://www.youtube.com/watch?v=NAs4FIWkJ4s

3.2 Pseudocódigo

Agora que você assistiu no YouTube ao vídeo com as dicas, o próximo passo é escrever algum algoritmo em pseudocódigo que implemente a solução para a programação da pirâmide, mesmo que você não saiba (ainda) como escrever essa solução na linguagem C.

Lembre-se: não há uma maneira correta de escrever pseudocódigo, escreva frases claras em português que indiquem como resolver o problema. Pense em como decompor o problema em partes menores e como resolver essas partes. O pseudocódigo deve indicar como resolver todas as partes.

Lembre-se também de que há boas chances de que seu pseudocódigo irá usar (ou implicar o uso de) uma ou mais funções, condicionais, expressões Booleanas, loops e/ou variáveis.

Se necessário você pode expressar seu algoritmo utilizando um fluxograma que indica a seqüência de resolução dos problemas decompostos, a seqüência de ações que seu programa deverá realizar.

Não pule este passo!

Muitos alunos tendem a menosprezar a etapa de desenvolvimento de uma solução inicial em pseudocódigo e/ou em fluxogramas, e partem diretamente para a digitação do código do programa. Essa é uma das **piores coisas** que você pode fazer! Não caia na tentação de pular esta etapa, a melhor coisa que você pode fazer agora é tirar a mão do teclado!

Construir uma solução em pseudocódigo e/ou fluxograma te **obriga a pensar**, a **decompor** o problema, a **reconhecer padrões**, a criar **abstrações** e criar um ou mais **algoritmos** para a solução, ou seja: treina e desenvolve sua capacidade de **pensamento computacional**.

Tire a mão do teclado! Use a cabeça agora, o código vem depois!

Aqui está um pequeno exemplo de como criar um pseudocódigo para este problema (este não é o pseudocódigo correto, é apenas mais um dos possíveis):

1. Pergunte a altura da pirâmide para o usuário
2. Se a altura estiver fora da faixa permitida ou não for um número válido, volte para o passo anterior
3. Quando o usuário informar uma altura válida, faça uma iteração de 1 até a altura informada:
   (a) Na iteração i, imprima i cerquilhas (#) e então uma nova linha (\n)

3.3 Solicitando a altura

Com o pseudocódigo pronto[3], vamos escrever apenas o código C que solicita a altura da pirâmide para o usuário (e continua solicitando até que o usuário informe um valor inteiro válido).

Abra o arquivo `mario_less.c` que você recebeu e insira um código em C que solicita ao usuário a altura da pirâmide e armazena essa altura em uma variável (o programa deve ficar repetindo a solicitação até que o usuário informe um número válido entre 1 e 8, inclusive). Quando o usuário conseguir informar um número válido seu programa deve apenas imprimir o valor da variável que armazenou a altura informada pelo usuário, confirmando para você mesmo que a entrada do usuário foi armazenada com sucesso em uma variável, e que você sabe como fazer para imprimir o valor inteiro armazenado em uma variável. Veja o exemplo abaixo:

```
$ ./mario_less
Altura: -1
Altura: 0
Altura: 42
Altura: 50
Altura: 4
Armazenado: 4
```

Algumas dicas para ajudá-lo nessa primeira tarefa:

- Lembre-se de que você deve usar um compilador para gerar o código de máquina (executável) de seu programa;
- Você pode imprimir o valor de um `int` armazenado usando a função `printf` com o código de formato `%d` ou `%i`;
- Você pode solicitar ao usuário um valor inteiro usando a função `get_int`, que está declarada no `cs50.h`; e
- Lembre-se de que para solicitar um inteiro positivo você pode utilizar um loop `do while`.

3.4 Construindo uma pirâmide oposta

Agora que seu programa está aceitando corretamente os valores de altura fornecidas pelo usuário, devemos dar mais um passo em direção à solução do problema.

Ocorre que é mais fácil construir uma pirâmide oposta, ou seja, alinhada à esquerda, conforme o exemplo abaixo:

[3] Se você não fez o pseudocódigo ou o fluxograma de solução, não continue! Volte e pense agora. O código vem depois.

```
#
##
###
####
#####
######
#######
########
```

Vamos construir essa pirâmide oposta, alinhada à esquerda, primeiro. Depois que conseguirmos, vamos dar um jeito de alinhar à direita.

Modifique seu código fonte de tal forma que ele imprima uma pirâmide alinhada à esquerda, de acordo com a altura informada pelo usuário. Algumas dicas:

- Lembre-se de que a cerquilha (#) é um caractere como outro qualquer e, portanto, pode ser impressa com a função `printf`;
- Lembre-se de que C possui um loop `for` através do qual você pode iterar algum número de vezes. Talvez, em cada iteração i, você possa imprimir esse mesmo número de cerquilhas;
- Você pode **aninhar**[4] loops, iterando com uma variável (por exemplo, i) no loop externo, e com outra variável (por exemplo, j) no loop interno. Por exemplo, o código abaixo mostra como imprimir um quadrado de lado n (claro que não queremos construir um quadrado!):

```
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        printf("%s", "#")
    }
    printf("%s", "\n");
}
```

3.5 Alinhar a pirâmide à direita

Agora vamos alinhar a pirâmide à direita, "empurrando" as cerquilhas para a direita através do uso de pontos, como mostrado a seguir:

[4] Colocar um loop dentro de outro.

```
.......#
......##
.....###
....####
...#####
..######
.#######
########
```

Altere o código em `mario_less.c` para ele imprima uma pirâmide alinhada à direita, usando pontos, de acordo com a altura informada pelo usuário.

Dica: perceba que o número de pontos necessários em cada linha é o "oposto" do número de cerquilhas. Para uma pirâmide de altura 8, como no exemplo acima, a primeira linha tem 1 cerquilha e 7 pontos. A última linha tem 8 cerquilhas e 0 pontos. Tente descobrir uma fórmula aritmética que conseguiria imprimir a quantidade necessária de pontos em cada linha.

3.6 Removendo os pontos

Tudo o que resta a ser feito é remover os pontos e substituí-los por algum caractere adequado (espaços, talvez?). Altere o código em `mario_less.c` e finalize seu programa!

3.7 Como testar seu código?

Uma parte importante é o teste de seu código final. Ele funciona conforme as especificações quando o usuário digitar alturas válidas? E se o usuário digitar alturas inválidas, como -1, 0, 9, letras, palavras, caracteres especiais? E se o usuário não digitar nada, só pressionar a tecla "Enter"?

Lembre-se também de que seu código deve seguir todas as normas de estilo de programação C da disciplina Harvard CS50: Harvard CS50 C Style Guide[5].

[5] https://cs50.readthedocs.io/style/c/