PSET: Lâmpadas (*Bulbs*)

Harvard CS50 Staff

2023-09-30

Resumo

Esta atividade corresponde ao PSET *bulbs* (lâmpadas) original da disciplina Harvard CS50, em sua versão integral, traduzido e adaptado para o português pelo prof. Abrantes Araújo Silva Filho.

Sumário

1 Introdução

Este exercício corresponde ao PSET "*bulbs*" (lâmpadas) da disciplina **Harvard CS50**, e deve ser feito pelos alunos que estão estuando o conteúdo sobre *arrays* mas não se sentem muito confortáveis ainda.

O objetivo deste exercício é que você pratique manipulação de strings e arrays, e aprenda a codificar textos em binário, além de entender como Emojis podem ser impressos no terminal através de Unicode.

A tradução e adaptação para o português foram feita com base na versão de 2023 do PSET, conforme o PSET original[1].

2 Lâmpadas (*Bulbs*)

Nas aulas da CS50, no *Sanders Theatre*[2], Universidade de Harvard, você deve ter reparado que uma série de lâmpadas ficam posicionadas bem na beirada do palco e que essas lâmpadas parecem ter alguma espécie de "bug": algumas estão acesas e outras apagadas:

[1] https://cs50.harvard.edu/x/2023/psets/2/bulbs/
[2] https://websites.harvard.edu/memhall/home-2/buildings/sanders-theatre/

Fonte: Harvard CS50

Cada sequência de lâmpadas, no entanto, codifica uma mensagem em **binário**, a linguagem que os computadores "falam". Vamos escrever um programa para fazer mensagens secretas de sua autoria (que talvez até podem ser colocadas no palco)!

3 Implementação

Para escrever o seu programa, você precisa pensar primeiro no conceito de **base de um sistema numérico**.

3.1 Base de um sistema numérico

A base de um sistema numérico corresponde à **quantidade de algarismos** que esse sistema numérico possui para representar todos os números. O sistema numérico mais simples é o de base-1, o sistema **unário**, que contém somente um único algarismo, o 1. Assim, para escrevermos um determinado número N no sistema unário, simplesmente escrevemos N algarismos 1 consecutivos. Por exemplo: para representar o número 4, em unário, devemos escrever $\boxed{1111}$, e o número 12 como $\boxed{111111111111}$. Pense no sistema unário como equivalente a contar com seus dedos ou fazer "pauzinhos" que totalizam o número que você quer representar.

Obviamente o sistema unário não é muito utilizado hoje em dia: pense em como é difícil representar números grandes apenas com "pauzinhos" ou com uma sequência de números 1. Ao invés disso nós costumamos usar o sistema **decimal**, de base-10, que possui dez algarismos para representar os números $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$. No sistema de base-10, cada algarismo é multiplicado por alguma potência de base 10, para representar qualquer número. Por exemplo, o número 123, na verdade, corresponde à:

$$\begin{aligned} 123 &= (1 \times 10^2) + (2 \times 10^1) + (3 \times 10^0) \\ &= (1 \times 100) + (2 \times 10) + (3 \times 1) \\ &= 100 + 20 + 3 \\ &= 123 \end{aligned} \tag{1}$$

Se quisermos expressar os números em outro sistema numérico, com outra base, basta seguir o mesmo raciocínio e apenas trocar a base da potência. Por exemplo, o número 123 em um sistema

de base-4 corresponde, na verdade, ao número 27 em decimal:

$$\begin{aligned} 123_{(4)} &= (1 \times 4^2) + (2 \times 4^1) + (3 \times 4^0) \\ &= (1 \times 16) + (2 \times 4) + (3 \times 1) \\ &= 16 + 8 + 3 \\ &= 27_{(10)} \end{aligned} \tag{2}$$

Generalizando esse pensamento, podemos rapidamente saber o número representado em qualquer base somando o **valor posicional** dos algarismos,

$$\sum_{i=0}^{n-1} a_i \times b^i \tag{3}$$

onde n é a quantidade total de algarismos no número, i é a posição do algarismo no número (começando a contagem em 0, progredindo da direita para a esquerda, e terminando em $n-1$), b é a base do sistema numérico, e a é o algarismo na posição i.

Os computadores utilizam um sistema com base-2, o sistema **binário**, que só tem dois algarismos $(0, 1)$ para representar todos os outros números. Em binário, escrever 123 é um erro pois nesse sistema só temos o 0 e o 1. Note que o processo de descobrir qual número decimal corresponde um certo número binário é o mesmo: some o valor posicional dos algarismos binários, por exemplo: o número binário `10101` corresponde ao decimal 21:

$$\begin{aligned} 10101_{(2)} &= (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ &= (1 \times 16) + (0 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) \\ &= 16 + 0 + 4 + 0 + 1 \\ &= 21_{(10)} \end{aligned} \tag{4}$$

3.2 Codificando uma mensagem

Lâmpadas só podem estar acessas ou apagadas. Em outras palavras, podemos dizer que lâmpadas representam dois **estados** possíveis: ou a lâmpada está acesa ou a lâmpada está apagada, exatamente como os números binários só são o 1 e o 0. Neste PSET iremos encontrar uma maneira de codificar um texto como uma seqüência de números binários.

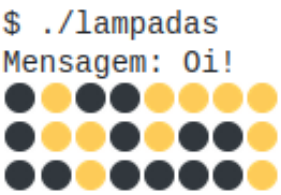
Escreva um programa chamado `lampadas`, que recebe uma mensagem e a converte para um conjunto de lâmpadas que podemos mostrar para uma audiência qualquer. Faça isso em dois passos:

- O primeiro passo consiste em converter o texto em números decimais. Digamos que queremos converter o texto "Oi!". Para nossa sorte já temos uma convenção que nos diz como fazer isso, o ASCII[3]. A letra `O` é representada pelo decimal `79`, a letra `i` é representada por `105`, e o caractere `!` é representado por `33`.
- O próximo passo envolve pegar os números decimais que representam as letras da mensagem e convertê-las em seus equivalentes em números binários que só usam 0 e 1. Para mantermos um padrão consistente, represente cada letra com 8 *bits*: `79` é `01001111`, `105` é `01101001` e `33` é `00100001`.

Para completar o programa temos que interpretar esses números binários como instruções para ligar ou desligar as lâmpadas: 0 é apagada, 1 é acessa. Cada letra deve ser impressa em uma linha. O *starter file* `lampadas.c` já tem uma função pronta, a função `imprimir_lampada`, que recebe um único bit (um 0 ou 1) e imprime um Emoji que representa uma lâmpada acessa ou apagada.

[3] https://asciitable.com/

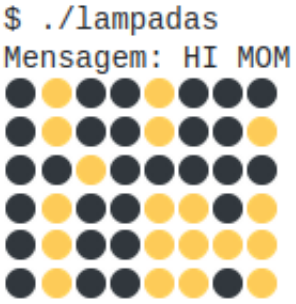
A imagem a seguir representa como o seu programa deverá funcionar:



Para verificar se as lâmpadas acima estão certas, basta ler um lâmpada acessa como `1` e uma lâmpada apagada como `0`. Assim, "Oi!" corresponde à

```
01001111
01101001
00100001
```

que é exatamente o que estamos esperando. Aqui está um outro exemplo:



Note que todos os caracteres serão codificados como instruções para as lâmpadas, incluindo caracteres não alfabéticos, como espaços (`00100000`).

4 Especificações

Projete e implemente o programa `lampadas`, que converte texto em instruções para acender uma série de lâmpadas que podem ser utilizadas no palco do CS50:

- O código fonte deve se chamar `lampadas.c`, e você deve utilizar o *starter file* fornecido;
- Seu programa deve solicitar que o usuário informe uma mensagem de texto a ser codificada utilizando a função `get_string`;
- Seu programa deve converter a `string` informada pelo usuário em uma série de números binários com 8 bits, um para cada caractere da `string`;
- Você deve utilizar a função `imprimir_lampada`, que já está pronta, para imprimir os `0` e `1` como lâmpadas apagadas e acessas (a função faz isso imprimindo Emojis); e
- Cada byte de 8 símbolos devem ser impressos em uma linha separada no output, ou seja, deve haver uma `\n` ao final de cada byte de símbolos.

Só use caracteres ASCII imprimíveis, não estendidos!

Para evitar a complicação de ter que lidar com caracteres ASCII não imprimíveis e caracteres ASCII estendidos (letras acentuadas, etc.), só vamos testar seu código para os caracteres imprimíveis sem acentuação, juntamente com os símbolos de escrita mais comuns. Assuma que seu código deve imprimir corretamente todos os caracteres da tabela ASCII desde o caractere 32, o espaço, até o caractere 126, o "~". Todos os outros caracteres ASCII (abaixo de 32 e acima de 126) não precisam ser levados em conta no seu código. Não testaremos seu código com nenhum caractere nessas faixas extremas.

4.1 Dica para a conversão de decimal em binário

Vamos ver um exemplo de como converter um número decimal em binário. Como poderíamos converter o número 4 em binário? Com o método de divisões sucessivas pela base, 2. A cada divisão, o **resto** será 0 ou 1, o que indicará o estado da lâmpada.

Comece dividindo o número 4 por 2. Temos o seguinte:

$$4/2 = 2 \text{ (quociente 2 com resto 0)}$$

Como o resto dessa primeira divisão foi 0, sabemos que o bit mais à direita será 0 e, assim, a última lâmpada (a da direita) deverá ficar apagada. Nesse momento nosso binário estará assim:

```
0
```

O próximo passo agora é dividir o **quociente** obtido na divisão anterior por 2:

$$2/2 = 1 \text{ (quociente 1 com resto 0)}$$

Como o resto dessa segunda divisão foi 0, podemos afirmar que o segundo bit, da direita para a esquerda, também será 0 e, assim, a lâmpada deverá ficar apagada. Nesse momento nosso binário estará assim:

```
00
```

O próximo passo é dividir o quociente obtido na divisão anterior por 2:

$$1/0 = 0 \text{ (quociente 0 com resto 1)}$$

Como o resto dessa terceira divisão foi 1, sabemos que o terceiro bit, da direita para a esquerda, será 1 e, assim, a lâmpada deverá ficar acessa. Nesse momento nosso binário estará assim:

```
100
```

Agora que já dividimos todo nosso número (alcançamos o quociente 0), não são necessários mais bits para representá-lo. Note que nós descobrimos os bits para representar o número 4 na ordem oposta na qual devemos imprimir: provavelmente você precisará de alguma estrutura que nos permite armazenar esses bits, para que possamos imprimi-los posteriormente. E, obviamente, no código de seu programa, estaremos trabalhando com `char`s de 8 bits, então você pode precisar completar os bits restantes à esquera com 0s.

5 Como testar seu código?

Teste seu programa com várias frases e mensagens, e verifique se as lâmpadas estão sendo impressas como o esperado.

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[4].

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

6 Como enviar seu código?

Utilize o starter file "`lampadas.c`" (mantenha esse padrão de nome), preencha as informações de identificação e envie o arquivo no Autolab, no PSET denominado "Lâmpadas (Bulbs)".