PSET: Substituição (*Substitution*)

Harvard CS50 Staff

2024-04-01

Resumo

Esta atividade corresponde ao PSET *Substitution* (Substituição) 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 "*Substitution*" (Substituição) da disciplina **Harvard CS50**, e deve ser feito por todos os alunos que estão estuando o conteúdo sobre *arrays*.

O objetivo deste exercício é você praticar manipulação de strings, caracteres e arrays. Também aprenderá mais sobre criptografia.

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

2 Substituição (*Substitution*)

Em uma cifra de substituição nós criptografamos (ou seja, ocultamos de forma reversível) uma mensagem substituindo cada letra por outra letra. Para isso, utilizamos uma chave (*key*): neste caso, um mapeamento de cada letra do alfabeto para a letra que deve corresponder quando a criptografamos. Para descriptografar a mensagem, o receptor da mensagem precisaria conhecer a chave, para que ele possa reverter o processo: traduzindo o texto criptografado (geralmente chamado de texto cifrado — *ciphertext*) de volta para a mensagem original (geralmente chamada de texto puro — *plaintext*).

[1] https://cs50.harvard.edu/x/2023/psets/2/substitution/

Uma chave, por exemplo, pode ser a string "NQXPOMAFTRHLZGECYJIUWSKDVB". Essa chave de 26 caracteres significa que "A" (a primeira letra do alfabeto) deve ser convertida em "N" (o primeiro caractere da chave), "B" (a segunda letra do alfabeto) deve ser convertida em "Q" (o segundo caractere da chave), e assim por diante.

Uma mensagem como "HELLO", então, seria criptografada como "FOLLE", substituindo cada uma das letras de acordo com o mapeamento determinado pela chave.

Vamos escrever um programa chamado `substitution` que permite criptografar mensagens usando uma cifra de substituição. Quando o usuário executar o programa ele deve decidir qual será a chave a ser utilizada na criptografia, e informar essa chave via argumento de linha de comando, no momento de executar o programa.

Aqui estão alguns exemplos de como o programa pode funcionar. Por exemplo, se o usuário inserir a chave "YTNSHKVEFXRBAUQZCLWDMIPGJO" e o texto puro "HELLO":

```
$ ./substitution YTNSHKVEFXRBAUQZCLWDMIPGJO
texto_puro: HELLO
texto_cifrado: EHBBQ
```

Note que o mesmo texto puro pode ser cifrado para outros textos, dependendo da chave:

```
$ ./substitution JTREKYAVOGDXPSNCUIZLFBMWHQ
texto_puro: HELLO
texto_cifrado: VKXXN
```

Se o usuário usar a chave "VCHPRZGJNTLSKFBDQWAXEUYMOI" e o texto "Hello, World", veja como o programa deve funcionar:

```
$ ./substitution VCHPRZGJNTLSKFBDQWAXEUYMOI
texto_puro: Hello, World
texto_cifrado: Jrssb, Ybwsp
```

Note que nem a vírgula nem o espaço foram substituídos pela cifra. Substitua apenas caracteres alfabéticos (letras)! Observe também que as letras maiúsculas e minúsculas foram preservadas: letras minúsculas permanecem minúsculas e letras maiúsculas permanecem maiúsculas.

Em relação à chave, note que a chave "VCHPRZGJNTLSKFBDQWAXEUYMOI" é funcionalmente idêntica a uma chave como "vchprzgjntlskfbdqwaxeuymoi" (assim como é idêntica a uma chave como "VcHpRzGjNtLsKfBdQwAxEuYmOi"), ou seja: não importa se os caracteres da chave são maiúsculos ou minúsculos.

Se o usuário não fornecer uma chave válida, o programa deve fornecer uma mensagem de erro e explicar que a chave deve ter 26 letras:

```
$ ./substitution ABC
A chave deve ter 26 letras.
```

Se o usuário fornecer mais argumentos de linha de comando do que o necessário, seu programa deve lembrar ao usuário como ele deve ser usado:

```
$ ./substitution 1 2 3
Uso: ./substitution chave
```

3 Especificação

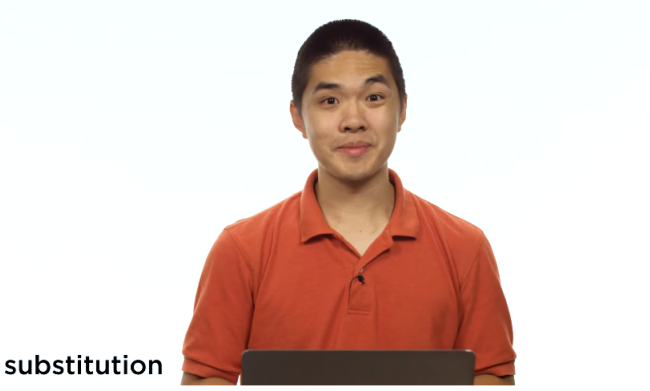
Projete e implemente um programa, chamado `substitution`, que criptografa mensagens de texto puro usando uma cifra de substituição. Atenção para as seguintes especificações:

- Implemente seu programa em um arquivo chamado `substitution.c`;
- Seu programa deve aceitar um único argumento de linha de comando, a chave para a substituição. A chave, por si mesma, deve ser indiferente às letras maiúsculas e minúsculas (assim, qualquer chave mistura de letras maiúsculas e minúsculas na chave não deve afetar o resultado de seu programa);
- Se seu programa for executado sem argumentos de linha de comando, ou com mais de um argumento, o programa deve exibir uma mensagem de erro de sua escolha (com a função `printf`) e retornar o valor `1` (que geralmente significa um erro) pela função `main` imediatamente;
- Se a chave for inválida (não tiver 26 caracteres, tiver algum caractere que não seja uma letra, ou não tiver cada letra exatamente uma única vez), seu programa deve imprimir uma mensagem de erro de sua escolha (com a função `printf`) e retornar o valor `1` pela função `main` imediatamente;
- Seu programa deve mostrar `texto_puro:` (sem uma quebra de linha) para que o usuário digite o texto (string) que ele quer criptografar (use a função `get_string`);
- Depois seu programa deve mostar `texto_cifrado:` (sem uma quebra de linha) seguido pelo texto criptografado correspondente ao texto puro criptografado com a chave que o usuário passou via argumento de linha de comando (cada caractere no texto puro deve ser substituído pelo caractere correspondente da chave). Caracteres não alfabéticos não devem ser alterados;
- Seu programa deve manter a distinção entre letras maiúsculas e minúsculas do texto puro que o usuário digitou;
- Após imprimior o texto cifrado, imprima uma quebra de linha. Termine o programa retornando o valor `0` pela função `main`.

Talvez você ache algumas funções úteis para seu programa na biblioteca `ctype.h`. Consulte a documentação dessa biblioteca em https://manual.cs50.io/.

4 Passo a passo

Se você precisar de ajuda para começar, assista este vídeo:



Fonte: Harvard CS50 Staff[2]

[2] https://www.youtube.com/watch?v=cXAoZAsgxJ4

5 Como testar seu código?

Teste seu código com diversas chaves e diversos textos. Tenha certeza de que o programa está substituindo corretamente as letras, mantendo as diferenças entre as letras minúsculas e maiúsculas. Lembre-se de que caracteres não alfabéticos não devem ser cifrados.

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[3].

Se você quiser, pode usar o `debug50`. Para isso, após você compilar seu código com o `make`, faça isso:

```
debug50 ./substitution CHAVE
```

Note que rodar o programa sem a chave, conforme mostrado a seguir, fará seu programa terminar e pedir a chave para o usuário:

```
debug50 ./substitution
```

6 Como enviar seu código?

Utilize o starter file "`substitution.c`" (mantenha esse padrão de nome), preencha as informações de identificação e envie o arquivo no Autolab, no exercício denominado "Substituição (Substitution)".

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