Este é o CS50

Introdução à Ciência da Computação (CS50)

OpenCourseWare

Doar (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Crédito

Problema a resolver

Um cartão de crédito (ou débito), claro, é um cartão de plástico com o qual você pode pagar por bens e serviços. Impresso nesse cartão está um número que também é armazenado em algum banco de dados, para que, quando seu cartão for usado para comprar algo, o credor saiba a

quem cobrar. Há muitas pessoas com cartões de crédito no mundo, então esses números são bem longos: a American Express usa números de 15 dígitos, a MasterCard usa números de 16 dígitos e a Visa usa números de 13 e 16 dígitos. E esses são números decimais (de 0 a 9), não binários, o que significa, por exemplo, que a American Express poderia imprimir até 10^15 = 1.000.000.000.000.000 cartões únicos! (Isso é, hum, um quatrilhão.)

Na verdade, isso é um pouco exagerado, porque os números de cartão de crédito têm, de fato, uma estrutura. Todos os números American Express começam com 34 ou 37; a maioria dos números MasterCard começa com 51, 52, 53, 54 ou 55 (eles também têm outros números iniciais possíveis, com os quais não nos preocuparemos neste caso); e todos os números Visa começam com 4. Mas os números de cartão de crédito também têm um "checksum" embutido, uma relação matemática entre pelo menos um número e outros. Esse checksum permite que computadores (ou humanos que gostam de matemática) detectem erros de digitação (por exemplo, transposições), se não números fraudulentos, sem precisar consultar um banco de dados, o que pode ser lento. Claro, um matemático desonesto certamente poderia criar um número falso que, mesmo assim, respeitasse a restrição matemática, então uma consulta ao banco de dados ainda é necessária para verificações mais rigorosas.

Em um arquivo chamado `<nome_do_arquivo>` localizado `credit.c` em uma pasta chamada `<nome_da_pasta>` `credit`, implemente um programa em C que verifique a validade de um determinado número de cartão de crédito.

Algoritmo de Luhn

Qual é a fórmula secreta? Bem, a maioria dos cartões usa um algoritmo inventado por Hans Peter Luhn, da IBM. De acordo com o algoritmo de Luhn, é possível determinar se um número de cartão de crédito é (sintaticamente) válido da seguinte forma:

1. Multiplique cada segundo dígito por 2, começando pelo penúltimo dígito do número, e depois some os dígitos desses produtos.
2. Adicione a soma à soma dos dígitos que não foram multiplicados por 2.
3. Se o último dígito do total for 0 (ou, mais formalmente, se o total módulo 10 for congruente a 0), o número é válido!

Isso é meio confuso, então vamos tentar um exemplo com o cartão Visa do David: 4003600000000014.

1. Para fins de discussão, vamos sublinhar um dígito sim, um dígito não, começando pelo penúltimo dígito do número:

   4 0 0 3 6 0 0 0 0 0 0 0 0 1 4

   Certo, vamos multiplicar cada um dos dígitos sublinhados por 2:

   1•2 + 0•2 + 0•2 + 0•2 + 0•2 + 6•2 + 0•2 + 4•2

   Isso nos dá:

2 + 0 + 0 + 0 + 0 + 12 + 0 + 8

Agora vamos somar os dígitos desses produtos (ou seja, não os produtos em si):

2 + 0 + 0 + 0 + 0 + 1 + 2 + 0 + 8 = 13

2. Agora, vamos adicionar essa soma (13) à soma dos dígitos que não foram multiplicados por 2 (começando do final):

   13 + 4 + 0 + 0 + 0 + 0 + 0 + 3 + 0 = 20

3. Sim, o último dígito dessa soma (20) é 0, então o cartão de David é legítimo!

Validar números de cartão de crédito não é difícil, mas pode ser um pouco tedioso se feito manualmente. Vamos escrever um programa.

Detalhes da implementação

No arquivo localizado `credit.c` no `credit` diretório, escreva um programa que solicite ao usuário um número de cartão de crédito e, em seguida, informe (via ` `printf` get_card_number()`) se o número é válido (American Express, MasterCard ou Visa), de acordo com as definições de formato de cada um aqui apresentadas. Para que possamos automatizar alguns testes do seu código, pedimos que a última linha de saída do seu programa seja `0x00`, `0x00`, `0x00` ou `AMEX\n` ` 0x00` , nada mais, nada menos. Para simplificar, você pode assumir que a entrada do usuário será inteiramente numérica (ou seja, sem hífens, como seria impresso em um cartão real) e que não terá zeros à esquerda. Mas não assuma que a entrada do usuário caberá em um `int` ! É melhor usar a função `get_card_number()` da biblioteca CS50 para obter a entrada do usuário. (Por quê?) `MASTERCARD\n` `VISA\n` `INVALID\n` `int` `get_long`

Considere o exemplo abaixo, que representa como seu próprio programa deve se comportar ao receber um número de cartão de crédito válido (sem hífens).

```
$ ./credit
Number: 4003600000000014
VISA
```

Agora, `get_long` o próprio programa irá rejeitar hífenes (e outros) de qualquer maneira:

```
$ ./credit
Number: 4003-6000-0000-0014
Number: foo
Number: 4003600000000014
VISA
```

Mas cabe a você identificar entradas que não sejam números de cartão de crédito (por exemplo, um número de telefone), mesmo que sejam numéricas:

```
$ ./credit
Number: 6176292929
INVALID
```

Teste seu programa com uma grande variedade de entradas, tanto válidas quanto inválidas. (Nós certamente faremos isso!) Aqui estão alguns números de cartão (https://developer.paypal.com/api/nvp-soap/payflow/integration-guide/test-transactions/#standard-test-cards) que o PayPal recomenda para testes.

Se o seu programa se comportar incorretamente com algumas entradas (ou não compilar de todo), é hora de depurar!

Passo a passo

Como testar seu código

Você também pode executar o comando abaixo para avaliar a correção do seu código `check50`. Mas certifique-se de compilá-lo e testá-lo você mesmo!

Correção

No seu terminal, execute o comando abaixo para verificar se o seu trabalho está correto.

```
check50 cs50/problems/2025/x/credit
```

Estilo

Execute o comando abaixo para avaliar o estilo do seu código usando `style50`.

```
style50 credit.c
```

Como enviar

No seu terminal, execute o comando abaixo para submeter seu trabalho.

```
submit50 cs50/problems/2025/x/credit
```