Esto es CS50x

OpenCourseWare

Donar (https://cs50.harvard.edu/donate) (https://community.alumni.harvard.edu/give/59206872)

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.quora.com/profile/David-J-Malan) (https://www.reddit.com/user/davidjmalan) (https://twitter.com/davidjmalan)

Crédito

Una tarjeta de crédito (o débito), por supuesto, es una tarjeta de plástico con la que puede pagar bienes y servicios. En esa tarjeta hay impreso un número que también se almacena en una base de datos en algún lugar, de modo que cuando su tarjeta se usa para comprar algo, el acreedor sabe a quién facturar. Hay muchas personas con tarjetas de crédito en este mundo, por lo que esos números son bastante largos: American Express usa números de 15 dígitos, MasterCard usa números de 16 dígitos y Visa usa números de 13 y 16 dígitos. Y esos son números decimales (0 a 9), no binarios, lo que significa, por ejemplo, que American Express podría imprimir hasta 10 ^ 15 = 1,000,000,000,000,000 tarjetas únicas. (Eso es, um, un billón).

En realidad, eso es un poco exagerado, porque los números de las tarjetas de crédito en realidad tienen cierta estructura. Todos los números de American Express comienzan con 34 o 37; la mayoría de los números de MasterCard comienzan con 51, 52, 53, 54 o 55 (también tienen otros números iniciales potenciales de los que no nos preocuparemos por este problema); y todos los números de Visa comienzan con 4. Pero los números de tarjetas de crédito también tienen una "suma de verificación" incorporada, una relación matemática entre al menos un número y otros. Esa suma de comprobación permite a las computadoras (o humanos a los que les gustan las matemáticas) detectar errores tipográficos (por ejemplo, transposiciones), si no números fraudulentos, sin tener que consultar una base de datos, lo que puede ser lento. Por supuesto, un matemático deshonesto ciertamente podría crear un número falso que, no obstante, respete la restricción matemática,

Algoritmo de Luhn

Entonces, ¿cuál es la fórmula secreta? Bueno, la mayoría de las tarjetas utilizan un algoritmo inventado por Hans Peter Luhn de IBM. De acuerdo con el algoritmo de Luhn, puede determinar si un número de tarjeta de crédito es (sintácticamente) válido de la siguiente manera:

1. Multiplica cada dos dígitos por 2, comenzando con el penúltimo dígito del número y luego suma los dígitos de esos productos.
2. Sume la suma a la suma de los dígitos que no se multiplicaron por 2.
3. Si el último dígito del total es 0 (o, dicho de manera más formal, si el módulo total 10 es congruente con 0), ¡el número es válido!

Eso es un poco confuso, así que probemos un ejemplo con la visa de David: 4003600000000014.

1. En aras de la discusión, primero subrayemos cada dos dígitos, comenzando con el penúltimo dígito del número:

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

   Bien, multipliquemos cada uno de los dígitos subrayados por 2:

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

   Eso nos da:

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

   Ahora agreguemos los dígitos de esos productos (es decir, no los productos en sí) juntos:

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

2. Ahora agreguemos esa suma (13) a la suma de los dígitos que no se multiplicaron por 2 (comenzando desde el final):

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

3. Sí, el último dígito de esa suma (20) es un 0, ¡así que la tarjeta de David es legítima!

Por lo tanto, validar los números de tarjetas de crédito no es difícil, pero se vuelve un poco tedioso a mano. Escribamos un programa.

Detalles de implementacion

En un archivo llamado `credit.c` en un `~/pset1/credit/` directorio, escriba un programa que solicite al usuario un número de tarjeta de crédito y luego informe (a través de `printf`) si es un número de tarjeta American Express, MasterCard o Visa válido, según las definiciones del formato de cada una en este documento. Para que podamos automatizar algunas pruebas de su código, le pedimos que la última línea de salida de su programa sea `AMEX\n` o `MASTERCARD\n` o `VISA\n` o `INVALID\n`, nada más, nada menos. Para simplificar, puede suponer que la entrada del usuario será completamente numérica (es decir, sin guiones, como podría estar impreso en una tarjeta real). ¡Pero no asuma que la entrada del usuario cabe en un `int`! Es mejor usarlo `get_long` de la biblioteca de CS50 para obtener las opiniones de los usuarios. (¿Por qué?)

Considere el siguiente representante de cómo debe comportarse su propio programa cuando se le pasa un número de tarjeta de crédito válido (sin guiones).

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

Ahora, él `get_long` mismo rechazará los guiones (y más) de todos modos:

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

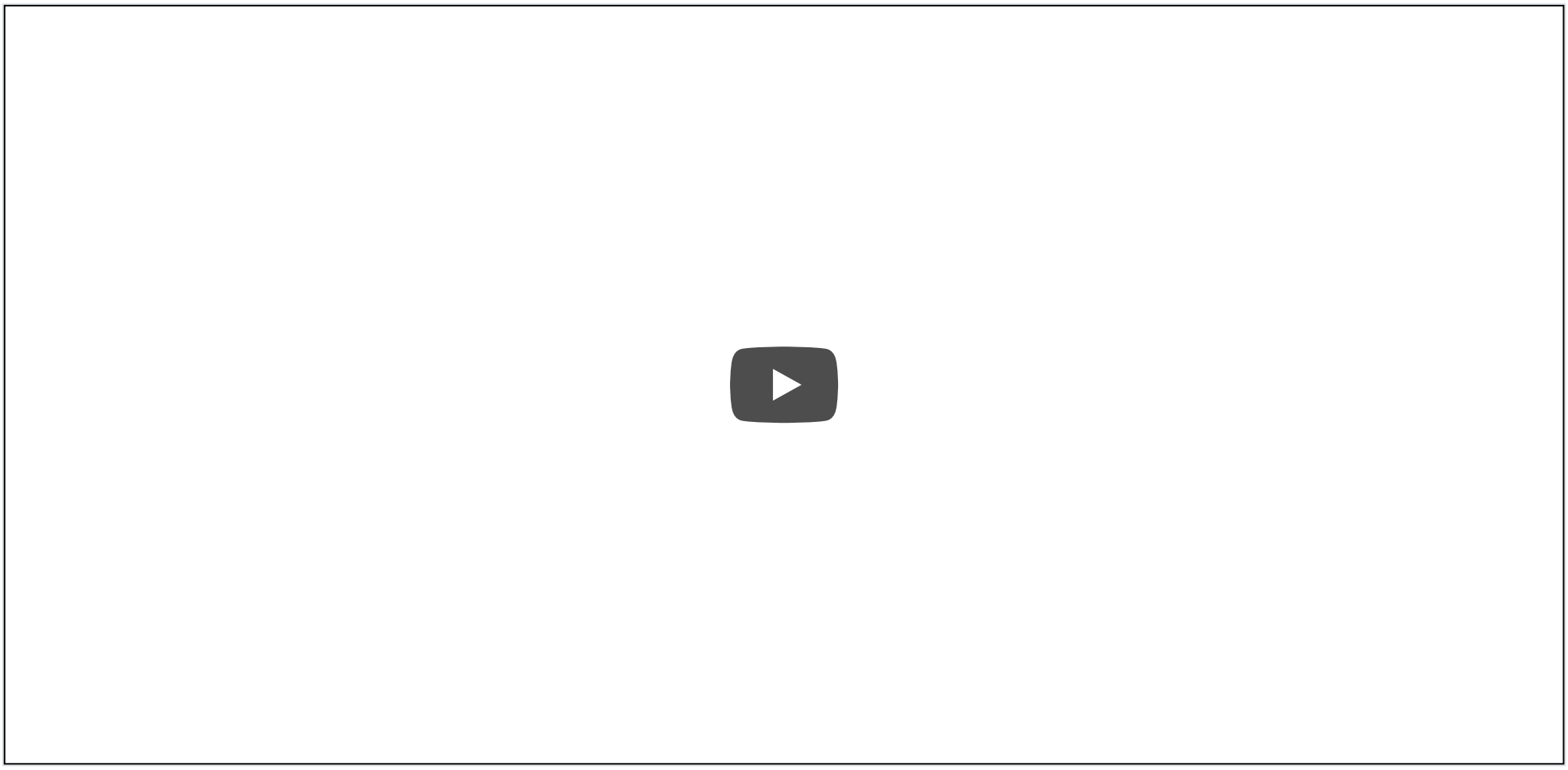
Pero depende de usted capturar las entradas que no sean números de tarjetas de crédito (por ejemplo, un número de teléfono), incluso si son numéricos:

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

Pruebe su programa con un montón de entradas, válidas y no válidas. (¡Por supuesto que lo haremos!) A continuación, se muestran algunos números de tarjeta (https://developer.paypal.com/docs/classic/payflow/payflow-pro/payflow-pro-testing/#credit-card-numbers-for-testing) que PayPal recomienda para realizar pruebas.

Si su programa se comporta incorrectamente en algunas entradas (o no se compila en absoluto), ¡es hora de depurarlo!

Tutorial



Cómo probar su código

También puede ejecutar lo siguiente para evaluar la exactitud de su código usando `check50`. ¡Pero asegúrese de compilarlo y probarlo usted mismo también!

```
check50 cs50/problems/2021/x/credit
```

Ejecute lo siguiente para evaluar el estilo de su código usando `style50`.

```
style50 credit.c
```

Cómo enviar

Ejecute lo siguiente, iniciando sesión con su nombre de usuario y contraseña de GitHub cuando se le solicite. Por seguridad, verá asteriscos (`*`) en lugar de los caracteres reales en su contraseña.

```
submit50 cs50/problems/2021/x/credit
```