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)

Dinheiro

Problema a resolver

Suponha que você trabalhe em uma loja e um cliente lhe dê R$ 1,00 (100 centavos) por um doce que custa R$ 0,50 (50 centavos). Você precisará pagar o troco, o valor restante após pagar o preço do doce. Ao dar o troco, provavelmente você deseja minimizar a quantidade de moedas que precisa fornecer a cada cliente, para não ficar sem (ou irritar o cliente!). Em um arquivo chamado `__init__.cs` `cash.c` em uma pasta chamada `__init__ `cash`.cs`, implemente um programa em C que imprima a quantidade mínima de moedas necessárias para dar o troco, em centavos, conforme o exemplo abaixo:

```
Change owed: 25
1
```

Mas solicite ao usuário um `int` valor maior que 0, para que o programa funcione com qualquer quantia de troco:

```
Change owed: 70
4
```

Solicite novamente ao usuário, repetidamente, conforme necessário, se a entrada dele não for maior ou igual a 0 (ou se a entrada for nula `int`!).

Demonstração

```
$ make cash
$ ./cash
Change owed: 25
1
$ ./cash
Change owed: 70
4
$ ./cash
Change owed: 113
8
$
```

Recorded with **asciinema**

Algoritmos gananciosos

Felizmente, a ciência da computação proporcionou aos caixas de supermercado maneiras de minimizar a quantidade de moedas a receber: algoritmos gulosos.

De acordo com o Instituto Nacional de Padrões e Tecnologia (NIST), um algoritmo guloso é aquele "que sempre escolhe a melhor solução imediata, ou local, ao buscar uma resposta. Algoritmos gulosos encontram a solução ótima geral, ou global, para alguns problemas de otimização, mas podem encontrar soluções subótimas para algumas instâncias de outros problemas."

O que tudo isso significa? Bem, imagine que um caixa deve troco a um cliente e que sua gaveta contém moedas de 25 centavos (25¢), 10 centavos (10¢), 5 centavos (5¢) e 1 centavo (1¢). O problema a ser resolvido é decidir quais moedas e quantas de cada tipo entregar ao cliente. Pense em um caixa "ganancioso" como aquele que quer aproveitar ao máximo cada moeda que

retira da gaveta. Por exemplo, se um cliente tem 41¢ a receber, a maior primeira (ou seja, a melhor opção imediata) que pode ser dada é 25¢. (Essa opção é "melhor" porque nos aproxima de 0¢ mais rapidamente do que qualquer outra moeda.) Observe que uma quantia desse tamanho reduziria o problema de 41¢ para um problema de 16¢, já que 41 - 25 = 16. Ou seja, o restante é um problema semelhante, porém menor. É evidente que mais uma mordida de 25 centavos seria demais (presumindo que o caixa prefira não perder dinheiro), então nosso caixa ganancioso passaria para uma mordida de 10 centavos, ficando com um problema de 6 centavos. Nesse ponto, a ganância o levaria a uma mordida de 5 centavos seguida de uma de 1 centavo, resolvendo assim o problema. O cliente recebe uma moeda de 25 centavos, uma de 10 centavos, uma de 5 centavos e uma de 1 centavo: quatro moedas no total.

Descobriu-se que essa abordagem gananciosa (ou seja, o algoritmo) não é apenas localmente ótima, mas também globalmente para a moeda americana (e também para a da União Europeia). Isto é, desde que um caixa tenha moedas suficientes de cada tipo, essa abordagem do maior para o menor resultará no menor número possível de moedas. Quantas? Bem, diga-nos você!

Conselho

Clique nos botões abaixo para ler algumas dicas!

▶ **Escreva um código que você sabe que será compilado.**

▶ **Escreva um pseudocódigo antes de escrever mais código.**

▶ **Converta o pseudocódigo em código.**

Como testar

Para este programa, tente testar seu código manualmente. É uma boa prática.

- Se você digitar `-1`, o programa solicita novamente a sua entrada?
- Se você inserir `0`, seu programa produzirá a saída `0`?
- Se você inserir um valor `1`, o seu programa produzirá uma saída `1` (ou seja, um centavo)?
- Se você inserir `4`, seu programa produzirá a saída `4` (ou seja, quatro centavos)?
- Se você inserir `5`, seu programa produzirá a saída `1` (ou seja, um níquel)?
- Se você inserir `24`, seu programa produzirá a saída `6` (ou seja, duas moedas de dez centavos e quatro moedas de um centavo)?
- Se você inserir um valor `25`, o seu programa produzirá a saída `1` (ou seja, "um quarto de dólar")?
- Se você inserir `26`, seu programa produzirá a seguinte saída `2` (ou seja, uma moeda de 25 centavos e uma moeda de 1 centavo)?

- Se você inserir `99`, seu programa produzirá a seguinte saída `9` (ou seja, três moedas de 25 centavos, duas moedas de 10 centavos e quatro moedas de 1 centavo)?

Correção

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

```
check50 cs50/problems/2025/x/cash
```

Estilo

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

```
style50 cash.c
```

Como enviar

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

```
submit50 cs50/problems/2025/x/cash
```