**Exercício Credit**

**Pseudocódigo:**

1. **Solicitar que o usuário digite o número do cartão:**
   1. **O número não pode ser menor do que zero;**
2. **Verificar cada dígito do cartão, se o mesmo estiver em posição ímpar, soma ele ao total, caso o número esteja em posição par, soma os dígitos do resultado da multiplicação deles por 2;**
3. **Verificar se o último dígito da soma é igual a 0:**
   1. **Se for diferente de 0, retorne INVALID;**
   2. **Se for igual a 0, continue;**
4. **Contar o número de dígitos que o usuário forneceu;**
5. **Armazenar os dois primeiros dígitos do cartão;**
6. **Verificar o tipo de cartão digitado:**
   1. **Se nº dígitos for = 16 e os dois primeiros dígitos estiver entre 51 e 55, retorne MASTERCARD;**
   2. **Se nº dígitos for = 15 e os dois primeiros dígitos for 34 ou 37, retorne AMEX;**
   3. **Se nº dígitos for = 13 ou = 16 e o primeiro dígito for = 4, retorne VISA;**
   4. **Caso contrário retorne INVALID;**

**Explicação do pseudocódigo**

1. **Solicitar que o usuário digite o número do cartão:**

Para solicitar o número infinitamente até que o usuário digite um número maior que 0, é necessário um loop com condição, para isso temos a função do while, que ao menos executa 1 vez a pergunta e caso o número seja inválido, repita a pergunta.

**Do**

**{**

**“Código para a pergunta”;**

**}**

**While (“variável número” < 0);**

Porém para solicitar um número, temos que saber onde iremos salvar a informação e qual o tipo da informação salva. Neste caso será um número, então temos duas opções, “int” ou “long”, como salvaremos um número com muitos dígitos, o ideal é usar “long”, pois armazena o dobro de dígitos que a variável “int”.

Obs: Temos que criar a variável antes do loop, pois precisamos que a resposta seja usada depois do mesmo.

**Long “variável número”;**

E dentro do loop usaremos a função que solicitará o número do cartão e salvará na “variável número”, essa função é a get\_long.

**“variável número” = get\_long(“Cartão: “);**

1. **Verificar cada dígito do cartão, se o mesmo estiver em posição ímpar, soma ele ao total, caso o número esteja em posição par, soma os dígitos do resultado da multiplicação deles por 2;**

Aqui vamos adicionar um nº de cartão fictício para conseguirmos entender o cálculo.  
Digamos que o usuário digitou **4003600000000014** e armazenamos o número na “variável número”.

Antes do cálculo eu recomendo salvar o número em uma outra variável para manter o número digitado seguro.

**Long cc = “variável número”;**

Agora vamos para o cálculo, vamos para os números em posição ímpar, para isso temos que lembrar que iremos ler o número de traz para frente, sendo os número ímpares os de vermelho.

**4003600000000014**

**Obs: estamos trabalhando com o tipo de variável long, ou seja, ela é do tipo inteiro, não tem casas decimais. Essa informação é MUITO importante para entender o processo.**

Como separamos o último dígito do cartão? Usamos o símbolo matemático “%” e armazenamos em uma variável “long” ou “int”, no caso vamos usar “int” pois só iremos armazenar nela 1 dígito por vez.

Mas o que significa o “%” no cálculo? Vamos para o exemplo:

**Int “dígito” = cc % 10;**

Isso significa que iremos dividir o número do cartão por 10 e pegar o resto da divisão, ou seja:

O número depois da vírgula é o resto da conta, esse número será salvo na variável “dígito”, então:

**cc % 10 = 4;**

Ok, conseguimos pegar o ultimo dígito, salvaremos ele dentro de uma variável soma e iremos para o próximo número vermelho, mas como fazer isso?

Sabemos que a variável “long” só salva números inteiros, logo ela descarta o valor depois da vírgula, então se dividirmos o valor por 10, passamos ao próximo digito. Ex:

Então se salvarmos de volta na variável ela irá armazenar , logo:

**cc = cc / 10** faz com que o número armazenado seja **.**

Porém fomos para a casa preta (par) e queremos ir para a vermelha(ímpar), para isso podemos repetir esse processo novamente, ou para simplificar podemos dividir o número por 100, assim a virgula andara 2 casas para a esquerda ao invés de 1, deixando a gente no local desejado para repetir o processo.

**cc = cc / 100** faz com que o número armazenado seja

Nos deixando na exata casa para repetir o processo.  
Fazemos isso até que o número do cartão seja menor que 0, ou seja, tenhamos dividido ele por inteiro, ou seja precisamos de um loop!

**Do**

**{**

**“dígito” = cc % 10;**

**“soma” = “soma” + “digito”;**

**cc = cc / 100;**

**}**

**While (cc > 0);**

Assim pegamos todos os dígitos ímpares do cartão, o par é basicamente a mesma coisa, porém temos que pegar o valor do digito e multiplicar por 2.

Então antes de começar, vamos voltar a colocar o número do cartão na variável cc, como nós fomos dividindo e salvando dentro da mesma, ao final do loop está armazenado nela um número menor do que 0, então:

**cc = “variável número”;** (por isso salvamos o número em uma nova variável, pois iremos dividir ela várias vezes, caso contrário teríamos que solicitar o número novamente ao usuário.)

Então o número armazenado na variável cc é ... o ultimo digito é o da posição ímpar... precisamos trabalhar com os dígitos na posição par, como faremos isso? Simples, antes do loop divide o número do cartão por 10, assim o ultimo digito será o preto.

**cc = cc / 10;** nos retornando

Com isso podemos refazer o loop:

**Do**

**{**

**“dígito” = cc % 10;**

**“soma” = “soma” + “digito”;**

**cc = cc / 100;**

**}**

**While (cc > 0);**

Porém antes de somar o valor do dígito temos que multiplica-lo por 2.

**Obs: alguns dígitos quando multiplicados por 2, dão um resultado de 2 dígitos, nesse caso temos que somar o valor de cada dígito e não o valor dele todo.**

**Ex:** 9 \* 2 = 18, então não somaremos 18 ao total e sim 1 + 8.

Para isso temos que fazer com que o código entenda que tem que pegar cada um dos valores. Existem duas formas de fazer isso, você pode criar uma condição onde ele verifica se o digito é maior que 5, e assim ele fazer a conta % e / e somar ao total e se não for só soma o valor ao total... o que daria certo, mas daria muito trabalho, ou você simplesmente faz a conta % e / em todos os dígitos, independente se a multiplicação tem ou não o resultado com 2 dígitos.

Então vamos alterar um pouco o loop

**Do**

**{**

**“dígito” = (cc % 10) \*2;** // pegamos o valor do digito e multiplicamos por dois e salvamos na variável, como solicitado.

**“soma” = “soma” + (“digito” % 10) + (“digito” / 10);** // % 10 pega o ultimo digito e /10 pega o primeiro digito, caso o número só tenha 1 digito, o / 10 dará 0 como resultando, não alterando na soma;

**cc = cc / 100;** // pula para o próximo número para refazer o loop.

**}**

**While (cc > 0);**

Foi longo, mas resolvemos o tópico 2. Ufa!

1. **Verificar se o último dígito da soma é igual a 0:**

Bom, já somamos todos os dígitos e já sabemos como pegar o ultimo valor de uma variável, então aqui fica fácil. Como precisamos verificar uma condição em verdadeiro ou falso, vamos usar um if e para saber o ultimo digito vamos usar um %.

**If (“soma” % 10 == 0)**

**{**

**“Código se verdadeiro”**

**}**

**Else**

**{**

**“código se falso”;**

**}**

Se for diferente de 0, já deve retornar ao usuário que o cartão é INVALID, se for = 0 continuamos para o próximo passo.

1. **Contar o número de dígitos que o usuário forneceu;**

Então deu verdadeiro a condição anterior, dentro dessa continuaremos a verificar novas condições e para algumas, iremos precisar saber quantos números o usuário digitou.

Como faremos isso?

Bom, agora nós já sabemos como andar por cada digito do cartão, dividindo o mesmo por 10, ou 100, ou 1000, etc. Então vamos fazer um loop, onde ficaremos dividindo o número do cartão por 10 até que ele seja menor que 0, assim saberemos exatamente quantos dígitos o usuário nos forneceu.

**Obs: Lembra de retornar o número do cartão completo antes do loop.**

**cc = “variável número”;**

Vamos criar uma variável contador que nos ajudara a saber quantas vezes foi feito o loop, como queremos usar esse valor depois, vamos criar ela fora do loop.

**Int ndigitos = 0;**

E vamos para o loop.

**Do**

**{**

**cc = cc / 10;**

**ndigitos++;**

**}**

**While (cc > 0);**

Pronto já sabemos quantos dígitos temos.

1. **Armazenar os dois primeiros dígitos do cartão;**

Agora que temos o número de dígitos fica fácil saber quantas vezes precisamos fazer a iteração para chegar aos dois primeiros dígitos, ndigitos – 2.

Então novamente retornaremos o numero inteiro do cartão a variável cc e o dividiremos por 10, ate que nos reste somente os 2 primeiros digitos.

**cc = “variável número”;**

**for (int i = 0; i < ndigitos – 2; i++)**

**{**

**cc = cc / 10;**

**}**

Agora temos o valor dos 2 primeiros dígitos do cartão e podemos ir para as verificações

1. **Verificar o tipo de cartão digitado:**
   1. **Se nº dígitos for = 16 e os dois primeiros dígitos estiver entre 51 e 55, retorne MASTERCARD;**

Aqui nada mais é que alguns if’s.

**If (ndigitos == 16 && cc >= 51 && cc <55)**

**{**

**Retorne MASTERCARD;**

**}**

* 1. **Se nº dígitos for = 15 e os dois primeiros dígitos for 34 ou 37, retorne AMEX;**

**If (ndigitos == 15 && (cc == 34 || cc == 37))**

**{**

**Retorne AMEX;**

**}**

* 1. **Se nº dígitos for = 13 ou = 16 e o primeiro dígito for = 4, retorne VISA;**

Aqui só tem 1 detalhe diferente, temos que verificar somente o primeiro número, então teremos que dividir cc +1 vez

**If ((ndigitos == 13 || ndigitos == 16) && cc / 10 = 4)**

**{**

**Retorne VISA;**

**}**

* 1. **Caso contrário retorne INVALID;**

Caso a soma dos digitos der um numero que termine em 0 mas não esteja em nenhuma dessas condições retorne INVALID.

**Tentei ser o mais detalhado possível, espero que ajude a entender a lógica da questão.**

**Caso ainda tenha dúvidas em algum ponto, deixarei o meu código na próxima folha.**

#include <stdio.h>

#include <cs50.h>

#include <math.h>

long crednumb(void);

int main(void)

{

    // "cnum" salva o número inteiro para não precisar pedir novamente, "cc" salva o número para poder ser alterado

    long cnum = crednumb(), cc = cnum;

    // "n" é o dígito solicitado, as "somas" salvam as somas dos digitos solicitados, "ndigitos" salva a quantidade de digitos do cartão

    int n = 0, soma = 0, ndigitos = 0;

    do // Pega o valor do último dígito e soma na variável "somaimpar", após divide o número por 100 para ir ao próximo número

    {

        n = cc % 10;

        soma += n;

        cc = cc / 100;

    }

    while (cc > 0);

    cc = cnum / 10;

    do // Pega o valor dos dígitos pares de trás para a frente, multiplica por 2 e após a multiplicação pega o primeiro e ultimo dígito do resultado e soma

    {

        n = (cc % 10) \* 2;

        soma += (n % 10) + (n / 10);

        cc = cc / 100;

    }

    while (cc > 0);

    cc = cnum;

    if (soma % 10 == 0)

    {

        do // Conta a quantidade de dígitos no cartão

        {

            cc = cc / 10;

            ndigitos++;

        }

        while (cc > 0);

        cc = cnum;

        for (long i = 0; i < ndigitos - 2; i++) // Reduz a quantidade de caracteres para 2 dígitos

        {

            cc = cc / 10;

        }

        // Verificação para saber se o cartão é master

        if (cc > 50 && cc < 56 && ndigitos == 16)

        {

            printf("MASTERCARD\n");

        }

        // Verificação para saber se o cartão é American Express

        else if ((cc == 34 || cc == 37) && ndigitos == 15)

        {

            printf("AMEX\n");

        }

        // Verificação para saber se o cartão é Visa

        else if (cc / 10 == 4 && (ndigitos == 13 || ndigitos == 16))

        {

            printf("VISA\n");

        }

        else

        {

            printf("INVALID\n");

        }

    }

    else

    {

        printf("INVALID\n");

    }

}

// Solicita o número do cartão até que seja fornecido um número positivo

long crednumb(void)

{

    long numb;

    do

    {

        numb = get\_long("Digite o número do cartão: ");

    }

    while (numb <= 0);

    return numb;

}