Semester Project

Credit Card Validator

Subject:	Programming Fundamentals
Submitted By:	Zohaib Ahmad

Abstract

Credit card validator is a terminal-based program. It validates the input credit card number using the Luhn algorithm. This program also shows the company name the credit card belongs to.

Contents

1. Introduction

In this era of technology, people are purchasing online daily. Nowadays, the lifestyle of the people is different. People feel uncomfortable and time-consuming going to crowded markets. So, E-Shopping is a boon as it saves a lot of time. Online shopping is a process whereby consumers directly buy goods, services, etc. from a seller without an intermediary service over the Internet. Shoppers can visit web stores from the comfort of their house and shop by sitting in front of the computer. Online stores are usually available 24 hours a day and many consumers have internet access both at work and at home.

To buy online, the buyer must give his credit card credential. So, how do the online websites check whether the input credit card number is valid or not?

It uses the Luhn algorithm to calculate whether the input number is correct or not.

My semester project is based on this. It is a terminal-based program to check whether the **input number is valid or not.**

1.1 Scope

So, what this program can and can’t do. This program only checks whether the input credit card number is correct and based on the Luhn algorithm. Now with this newfound knowledge, keep in mind you still won't be able to randomly generate genuine workable credit card numbers. The Luhn theory only allows you to generate mathematically compliant credit card test numbers, not hack workable ones. Besides, without valid expiration dates, and valid CVV2, CVC2, or CID numbers (the special security codes printed on the back or front of credit cards as additional authentication measures), you still wouldn't be able to legitimately use your self generated numbers to run credit transactions anyway. In simple words, this program only checks the credit card number on the base of the Luhn algorithm, is it mistyped or not.

2. Literature review

2.1 Luhn algorithm

The Luhn algorithm, also known as the modulus 10 or mod 10 algorithm, is a simple checksum formula used to validate a variety of identification numbers, such as credit card numbers, IMEI numbers, Canadian Social Insurance Numbers. The LUHN formula was created in the late 1960s by a group of mathematicians. Shortly thereafter, credit card companies adopted it. Because the algorithm is in the public domain, it can be used by anyone. Most credit cards and many government identification numbers use the algorithm as a simple method of distinguishing valid numbers from mistyped or otherwise incorrect numbers. It was designed to protect against accidental errors, not malicious attacks.

The industry standard for checking credit card numbers for validity is known in colloquial terms as the 'MOD 10 Check', or more formally as Luhn's Algorithm. It has no cryptographic validity, but it is a useful rule-of-thumb check that can be used to validate that a card number is correct and can be used in its opposite form to generate account (and credit card) numbers.

2.2 Algorithm Overview

The algorithm can be broken down into 4 stages. All stages are carried out using simple mathematics and rules.

2.2.1 Stage 1: sum of integers present at odd places starting from the rightmost

Let's first get a number which we are testing for validity and split it out into its component digits, in a format we can easily understand. The number we will use is a Visa sample card number:

		◆		◆		◆		◆		◆		◆		◆		◆
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Credit Card Number	4	0	3	5	3	0	0	5	3	9	8	0	4	0	8	3
the sum of integers present at odd places starting from the right		***0***		***5***		***0***		***5***		***9***		***0***		***0***		***3***
Sum_1 = 22																

2.2.2 Stage 2: Number *2 for every number present at even places starting from rightmost

The first step is to apply Number*2 to every 2nd digit starting from the right. This is true regardless of the length of the number.

	◆		◆		◆		◆		◆		◆		◆		◆	
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Credit Card Number	4	0	3	5	3	0	0	5	3	9	8	0	4	0	8	3
sum of integers present at even places starting from the right	4		3		3		0		3		8		4		8	
Number*2	***8***		***6***		***6***		***0***		***6***		***16***		***8***		***16***	

2.2.3 Stage 3: If (number*2)>9 then sum the digits else carry the digits

	◆		◆		◆		◆		◆		◆		◆		◆	
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Credit Card Number	4	0	3	5	3	0	0	5	3	9	8	0	4	0	8	3
sum of integers present at even places starting from the right	4		3		3		0		3		8		4		8	
Number*2	8		6		6		0		6		16		8		16	
If (number*2)>9 then sum the digits else carry the digits	***8***		***6***		***6***		***0***		***6***		***7***		***8***		***7***	

Sum_2 = 48

2.2.4 Stage 4: Sum and if Sum%10 == 0 then True else False

In this step, we will take the total sum of integers present at odd and even places. If their mod 10 is equal to 0 then the card number is correct otherwise wrong.

Sum_1 + Sum_2 = 70

70 % 10 == 0

So, in case of this number, it will be correct.

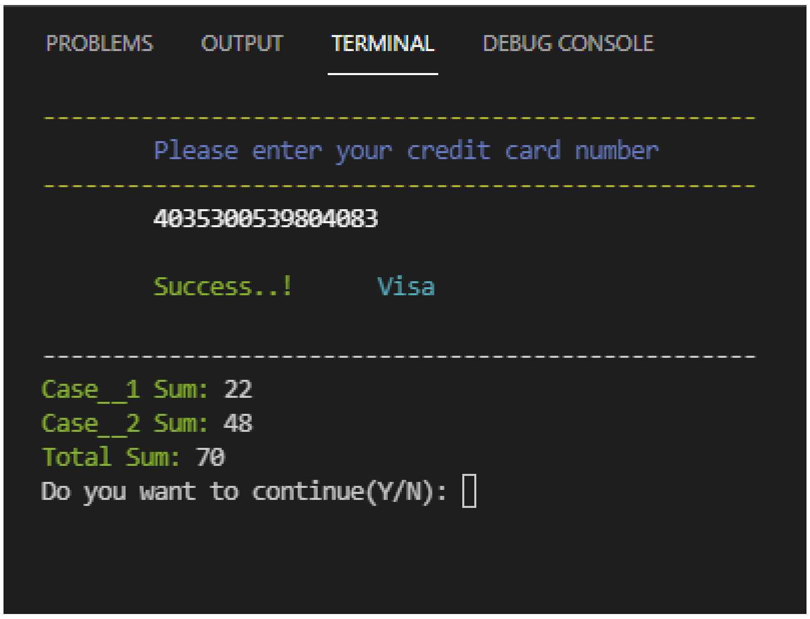
3. Methodology

```
# include <iostream>
# include <cmath>
# include <windows.h>
# include <string>
using namespace std;
void show(string x,int a, int b);
void color(int y);
using namespace std;

int main()
{
    int sum1=0, sum2=0, sum;
    int count ;
    long long divisor = 10;
    long long CreditCardNumber;
    char continue1;

   // Takes the input...
   do{
        do{
            system("cls");
            show("\tPlease enter your credit card number\t", 6, 1);
            color(15);
            cout<<"\t";cin>>CreditCardNumber;
            color(7);
            cout<<endl;
            }while (CreditCardNumber<=0);


        //Case_1...
        //Calculates the sum of integers on odd places
        long long cardnumber = CreditCardNumber;
        while(cardnumber > 0)
        {
            int lastdigit = cardnumber % 10;
            sum1 = sum1 + lastdigit;
            cardnumber = cardnumber / 100;
        }



        //Case_2....
        //Multiply integer present at even places by 2 and then calculates their sum..
        long long cardnumber1 = CreditCardNumber / 10;
        while(cardnumber1 > 0)
        {
            long long lastdigit = cardnumber1 % 10;
            int multiplyer = lastdigit * 2;
            sum2 = sum2 + (multiplyer % 10 + multiplyer / 10);
            cardnumber1 = cardnumber1 / 100;
        }


        // Sum of Case_1 and Case_2
        sum = sum1 + sum2;


        //Finds the numbers present at the start....
        long long cardnumber2 = CreditCardNumber;
        while(cardnumber2 > 0)
        {
            cardnumber2 = cardnumber2 / 10;
            count++;
        }

        long long cardnumber3 = CreditCardNumber;

        int first_one= cardnumber3 / pow(10, count-1);
        int first_two= cardnumber3 / pow(10, count-2);
        int first_three= cardnumber3 / pow(10, count-3);
        int first_four= cardnumber3 / pow(10, count-4);
        int first_six= cardnumber3 / pow(10, count-6);

        //Checks different conditions for different credit Cards..
        if(sum%10 == 0)
        {
            show("\tSuccess..!", 2,7);
            if(first_one ==4 && (count ==13 || count ==16))
            {
                show("\tVisa", 3, 7);
            }
            else if((first_two ==34 || first_two ==37) && count ==15)
            {
                show("\tAmerican Express",3,7);
            }
            else if(first_two ==60 || first_four ==6521 || first_four ==6522  && (count ==16))
            {
                show("\tCredit Card: RuPay",3,7);
            }
            else if((first_two>50 && first_two<56) && count ==16)
            {
                show("\tMasterCard",3,7);
            }
            else if(first_two ==31 && count ==19)
            {
                show("\tChina T-Union",3,7);
            }
            else if(first_two ==62 && (count>15 && count<20))
            {

                  show("\tChina UnionPay",3,7);
              }
              else if(first_two ==36 && (count>13 && count<20))
              {
                  show("\tDiners Club International",3,7);
              }
            else if((first_two ==38 || first_two ==39 || first_four ==3095 || (first_three>299 && first_three<306)) && (count>15 && count<20))
            {
                show("\tDiners Club International",3,7);
            }
            else if((first_two ==64 || first_two ==65 || first_four ==6011 || (first_six>622125 && first_six<622926) ||
            (first_six>623999 && first_six<627000) || (first_six>628199 && first_six<628900)) && (count>15 && count<20))
            {
                show("\tDiscover Card",3,7);
            }
            else if(first_three ==636 && (count>15 && count<20))
            {
                show("\tInterPayment",3,7);
            }
            else if((first_three>636 && first_three<640) && (count ==16))
            {
                show("\tInstaPayment",3,7);
            }
            else if(first_four ==5610 || (first_six>560220 && first_six<560226) && (count ==16))
            {
                show("\tBankcard",3,7);
            }
            else if((first_four ==6759 || first_six ==676770 || first_six ==676774) && (count>11 && count<20))
            {
                show("\tMaestro (UK)",3,7);
            }
            else if((first_two ==50 ||(first_two>55 && first_two<70)) && (count>11 && count<20))
            {
                show("\tMaestro",3,7);
            }
            else if((first_six>979199 && first_six<979290) && (count ==16))
            {
                show("\tTroy",3,7);
            }
            else if((first_four>3527 && first_four<3590 ) && (count>15 && count<20))
            {
                show("\tJCB",3,7);
            }
            else
            {
                show("Invalid",4,7);
            }
        }
        else
        {
            show("Invalid",4,7);
        }

        if(sum%10==0)
        {
            show("Case__1 Sum: ", 2,7);
            cout<<sum1<<endl;
            show("Case__2 Sum: ", 2,7);
            cout<<sum2<<endl;
            show("Total Sum: ", 2,7);
            cout<<sum<<endl;
        }
        else
        {
            show("Case__1 Sum: ", 10,7);
            cout<<sum1<<endl;
            show("Case__2 Sum: ", 10,7);
            cout<<sum2<<endl;
            show("Total Sum: ", 4,7);
            color(4);
            cout<<sum<<endl;
            color(7);
        }
        cout<<"Do you want to continue(Y/N): ";
        cin>>continue1;

   }while (continue1=='y' || continue1=='Y');


    return 0;

}

// Function to show colors
void color(int color)
{
 SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), color);
}

//Function to show strings with colors
void show(string x, int a, int b)
{
    if(x=="\tPlease enter your credit card number\t")
    {
        color(a);
        cout<<"---------------------------------------------------"<<endl;
        color(b);
        cout<<x<<endl;
        color(a);
        cout<<"---------------------------------------------------"<<endl;
        color(7);
    }
    else if(x=="\tSuccess..!" || x=="Case__1 Sum: " || x=="Case__2 Sum: "  || x=="Total Sum: ")
    {
        if(x=="Case__1 Sum: ")
        {
            cout<<endl;
            cout<<"---------------------------------------------------"<<endl;
            color(a);
        }
        else
        color(a);
        cout<<x;
        color(7);
    }
    else
    {
        color(a);
        cout<<x<<endl;
        color(7);

    }
}
```

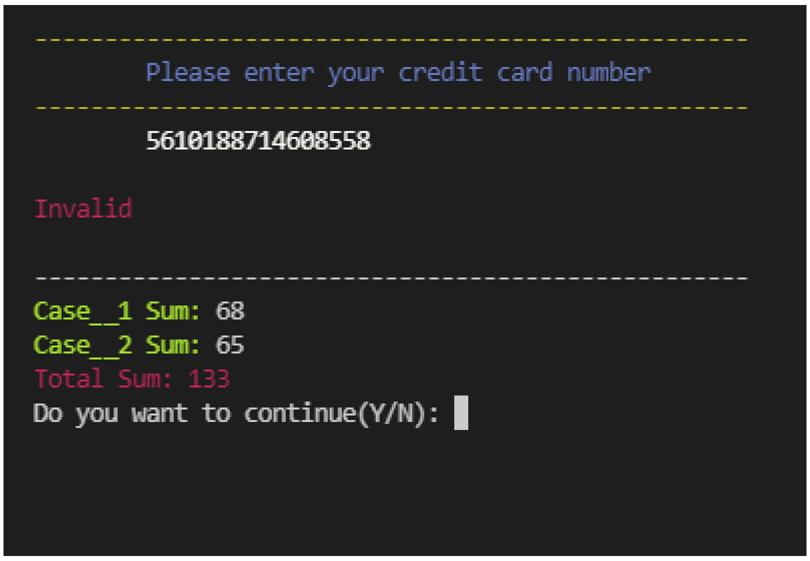
4. Future Work

In the future, we can add a little feature in which the program will tell us a little bit about the credit card if the credit card number is correct.

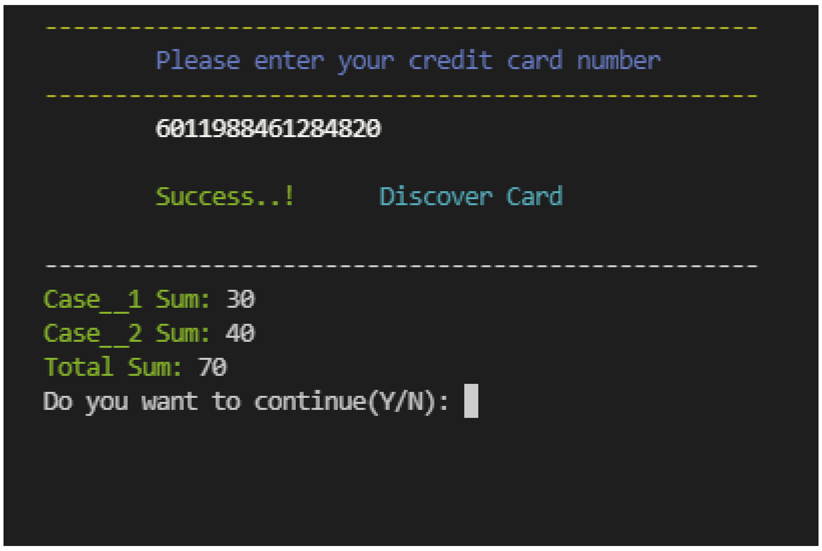
Sample Output



```
---------------------------------------------------
        Please enter your credit card number
---------------------------------------------------
        4035300539804083

        Success..!      Visa

---------------------------------------------------
Case__1 Sum: 22
Case__2 Sum: 48
Total Sum: 70
Do you want to continue(Y/N):
```



```
---------------------------------------------------
        Please enter your credit card number
---------------------------------------------------
        5610188714608558

Invalid

---------------------------------------------------
Case__1 Sum: 68
Case__2 Sum: 65
Total Sum: 133
Do you want to continue(Y/N):
```



```
---------------------------------------------------
        Please enter your credit card number
---------------------------------------------------
        6011988461284820

        Success..!      Discover Card

---------------------------------------------------
Case__1 Sum: 30
Case__2 Sum: 40
Total Sum: 70
Do you want to continue(Y/N): 
```