Number\_.cpp

//

//  ==========================================================================

//

//  Number\_.cpp

//

//  Owner: Yancey Reid

//  Program: Digit-by-Digit Arithmetic

//  Date Created: 2/19/17

//

//  Copyright © 2017 Yancey Reid. All rights reserved.

//

//  ==========================================================================

//

#include <stdio.h>

#include <iostream>

#include "Number\_.h"

using namespace std;

//

//  ==========================================================================

//

//  Assigns every element in number\_array to 0.

//

//  ==========================================================================

//

Number::Number()

{

    for (int i = 0; i < DIGITS\_MAX \* 2; i++)

    {

        number\_array[i] = 0;

    }

}

//

//  ==========================================================================

//

//  Assigns the number\_array to the number being passed with the ones place

//  starting at index 0.

//

//  ==========================================================================

//

void Number::SetNumber(string num\_to\_set)

{

    string temp\_num;

    bool found = false;

    if (num\_to\_set[0] == '0')

    {

        for (int i = 0; i < num\_to\_set.length();  i++)

        {

            if (num\_to\_set[i] != '0')

                found = true;

            if (found)

                temp\_num += num\_to\_set[i];

        }

        if (!found)

            temp\_num = "0";

        num\_to\_set = temp\_num;

    }

    number\_length = num\_to\_set.length();

    for (int i = number\_length - 1; i >= 0; i--)

    {

        number\_array[(number\_length - 1) - i] = num\_to\_set[i] - '0';

    }

}

//

//  ==========================================================================

//

//  Formats the string representation of the number\_arr with commas.

//

//  ==========================================================================

//

void Number::FormatNumber()

{

    formatted\_number.erase();

    for (int i = 0; i < number\_length; i++)

    {

        formatted\_number = char(number\_array[i] + '0') + formatted\_number;

        if ((i + 1) % 3 == 0 && i != number\_length - 1)

            formatted\_number = ',' + formatted\_number;

    }

}

//

//  ==========================================================================

//

//  Adds the number to the current instance digit-by-digt.

//

//  ==========================================================================

//

void Number::AddNumber(Number num\_to\_add)

{

    int max\_length;

    int remainder = 0;

    int temp\_sum = 0;

    Number total\_sum;

    if (number\_length >= num\_to\_add.number\_length)

        max\_length = number\_length;

    else

        max\_length = num\_to\_add.number\_length;

    for (int i = 0; i < max\_length; i++)

    {

        temp\_sum = number\_array[i] + num\_to\_add.number\_array[i] + remainder;

        if (temp\_sum < 10)

        {

            total\_sum.number\_array[i] = temp\_sum;

            total\_sum.number\_length++;

            remainder = 0;

        }

        else

        {

            total\_sum.number\_array[i] = temp\_sum % 10;

            total\_sum.number\_length++;

            remainder = temp\_sum / 10;

        }

    }

    if (remainder != 0)

    {

        total\_sum.number\_array[max\_length] = remainder;

        total\_sum.number\_length++;

    }

    \*this = total\_sum;

}