main\_.cpp

//

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

//

//  main.cpp

//

//  Owner: Yancey Reid

//  Program: AddAndMultiply

//  Date Created: 2/19/17

//

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

//

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

//

#include <iostream>

#include "Constants\_.h"

#include "Number\_.h"

using namespace std;

bool ValidateAmount(string);

bool ValidateNumber(string);

void PrintNumbers(Number[], int, int, char);

Number AddNumbers(Number[], int);

Number MultiplyFirstTwoNumbers(Number[], Number[]);

//

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

//

//  Driver for program. Requests numbers, adds the numbers, multiplies the

//  first two numbers, and prints the intermediate steps with answer.

//

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

//

int main()

{

    unsigned int number\_amount;

    unsigned int product\_amount = 2;

    string holder;

    Number sum, product;

    Number numbers[NUMBERS\_MAX];

    Number product\_additions[DIGITS\_MAX];

    cout << " - This program adds " << NUMBERS\_MIN << " through " << NUMBERS\_MAX << " very large

numbers." << endl;

    cout << " - It multiplies the first two numbers showing the intermediate steps." << endl;

    cout << " - Numbers must have between " << DIGITS\_MIN << " and " << DIGITS\_MAX << " digits." << endl;

    cout << endl << endl;

    cout << "Enter amount of numbers: ";

    getline(cin, holder);

    cout << endl;

    while (ValidateAmount(holder))

    {

        cout << endl;

        holder.clear();

        cout << "Invalid amount, please re-enter: ";

        getline(cin, holder);

    }

    number\_amount = stoi(holder);

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

    {

        cout << "Enter number #" << i + 1 << ": ";

        getline(cin, holder);

        while (ValidateNumber(holder))

        {

            cout << endl;

            holder.clear();

            cout << "Invalid number, please re-enter: ";

            getline(cin, holder);

        }

        numbers[i].SetNumber(holder);

        cout << endl;

    }

    cout << endl;

    cout << "The sum of: \n" << endl;

    sum = AddNumbers(numbers, number\_amount);

    sum.FormatNumber();

    PrintNumbers(numbers, number\_amount, sum.formatted\_number.length(), '+');

    cout << string(SPACE\_OFFSET, ' ') << sum.formatted\_number << endl;

    cout << endl;

    cout << "The product of: \n" << endl;

    product = MultiplyFirstTwoNumbers(numbers, product\_additions);

    product.FormatNumber();

    PrintNumbers(numbers, product\_amount, product.formatted\_number.length(), '\*');

    PrintNumbers(product\_additions, numbers[1].number\_length, product.formatted\_number.length(), '+');

    if (numbers[0].formatted\_number == "0" || numbers[1].formatted\_number == "0")

        cout << string(SPACE\_OFFSET + product.formatted\_number.length() - 1, ' ') << 0;

    else

        cout << string(SPACE\_OFFSET, ' ') << product.formatted\_number << endl;

    cout << endl << endl;;

    return 0;

}

//

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

//

//  Validates that input is a number and is between the numbers min and max.

//

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

//

bool ValidateAmount(string num\_str)

{

    bool error = (num\_str.size() == 0);

    for (int i = 0; i < num\_str.size() && !error; i++)

    {

        error = (num\_str[i] < '0' || num\_str[i] > '9');

    }

    if (!error)

        error = (stoi(num\_str) < NUMBERS\_MIN || stoi(num\_str) > NUMBERS\_MAX);

    return error;

}

//

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

//

//  Validates that input is a number and is between the digits min and max.

//

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

//

bool ValidateNumber(string num\_str)

{

    bool error = (num\_str.size() < DIGITS\_MIN || num\_str.size() > DIGITS\_MAX);

    for (int i = 0; i < num\_str.size() && !error; i++)

    {

        error = (num\_str[i] < '0' || num\_str[i] > '9');

    }

    return error;

}

//

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

//

//  Prints a number array, a given symbol, and a line.

//

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

//

void PrintNumbers(Number numbers[], int amount\_of\_numbers, int format\_length, char symbol)

{

    for (int i = 0; i < amount\_of\_numbers; i++)

    {

        if (i != amount\_of\_numbers - 1)

            cout << string(SPACE\_OFFSET, ' ');

        else

            cout << symbol << ")" << string(SPACE\_OFFSET - 2, ' ');

        numbers[i].FormatNumber();

        cout << string(format\_length - numbers[i].formatted\_number.length(), ' ');

        cout << numbers[i].formatted\_number;

        cout << endl;

    }

    cout << string(format\_length + SPACE\_OFFSET, '-') << endl;

}

//

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

//

//  Adds all the numbers in a Number array.

//

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

//

Number AddNumbers(Number numbers[], int number\_amount)

{

    Number sum = numbers[0];

    for (int i = 1; i < number\_amount; i++)

    {

        sum.AddNumber(numbers[i]);

    }

    return sum;

}

//

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

//

//  Multiplies the first two numbers in a Number array.

//

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

//

Number MultiplyFirstTwoNumbers(Number numbers\_to\_multiply[], Number numbers\_to\_add[])

{

    int remainder, temp\_sum;

    for (int i = 0; i < numbers\_to\_multiply[1].number\_length; i++)

    {

        numbers\_to\_add[i].number\_length += i;

        remainder = 0;

        for (int k = 0; k < numbers\_to\_multiply[0].number\_length; k++)

        {

            temp\_sum = numbers\_to\_multiply[1].number\_array[i] \* numbers\_to\_multiply[0].number\_array[k];

            temp\_sum += remainder;

            remainder = 0;

            if (temp\_sum > 9)

            {

                numbers\_to\_add[i].number\_array[k + i] = temp\_sum % 10;

                remainder = temp\_sum / 10;

            }

            else

            {

                numbers\_to\_add[i].number\_array[k + i] = temp\_sum;

            }

            numbers\_to\_add[i].number\_length++;

        }

        if (remainder != 0)

        {

            numbers\_to\_add[i].number\_array[numbers\_to\_multiply[0].number\_length + i] = remainder;

            numbers\_to\_add[i].number\_length++;

        }

    }

    return AddNumbers(numbers\_to\_add, numbers\_to\_multiply[1].number\_length);

}