Amarilda Celhaka

Project #1

Part D

(D)

Implementing each Fibonacci with the help of an integer array of size 100

SOURCE CODE

#include "stdafx.h"

#include <iostream>

using namespace std;

int main()

{

//Fibonaci array will hold the fibonacci number, no bigger than 99 digits

int Fibonaci[99];

//Array A will represent the first fibonaci of N-1, first element initialized to 0

int A[99] = { 0 };

//Array B will represent the second fibonaci of N-2, first element of the array initialized

// with zero and that represent the second element of the sequence

int B[99] = { 1 };

int carry = 0, //carry holds carry value when two numbers are added, and some more than 9

quotient = 0, //quotient used to hold the carry value before it is assigned to the carry

subscript, //subscript holds the user input

biggestSubscript = 0, //biggestSubscripts to hold the biggest Fibonacci Number of 99 digits subscript

count = 0; //count will keep track of how many times is entering the if condition

bool biggestFibonaciFound = false,

biggestSubscriptFound = false;

cout << "Enter the subscript of a Fibonaci Number: ";

cin >> subscript;

while (subscript < 0)

{

cout << "ERROR! Subscript must be bigger than 0.";

cout << "Re-enter the subscript: ";

cin >> subscript;

}

//since we are starting our calculations from j = 2, if user input == 1, then Fibonacci Array is equal with F of N-2

if (subscript == 1)

{

for (int i = 98; i >= 0; i--)

{

Fibonaci[i] = B[i];

}

}

//the first for loop, loops a certain amount of times, depending on subscript

//to find the Fibonaci Number

for (int j = 2; j <= subscript; j++)

{

//this for loop loops through the arrays, from position 0 up to 99

for (int i = 0; i < 99; i++)

{

//testing if the sum is bigger than 9

quotient = (A[i] + B[i] + carry) / 10;

//Fibonacci array of postion i, will hold the sum of array A and B, plus the carry

//coming from the previous iteration

Fibonaci[i] = (A[i] + B[i] + carry) % 10;

//Array A gets the value of array B, s

A[i] = B[i];

//Array B gets the value of Fibonaci Array, in order that for the next iteration

//A and B will have the next element of arrays values

B[i] = Fibonaci[i];

carry = quotient;

/\* when in position 98 which is the 99 digits (array starting from 0)

//check if the sum of the two last digits is greater than 9, we need

// one more place in our array, but our array should hold only 99 position

// so we need to stop the iterations and display the value of fibonacci Number\*/

if (i == 98)

{

if( (A[i] + B[i] + carry) > 9)

{

//we found biggest Fibonacci number, so biggestFibonacci is set to true

biggestFibonaciFound = true;

//this count is used to check how many times this process will happen

//and we want to stop once it hits if condition once

count = count + 1;

if (count == 1)

{

//this if condition I am using to stop the loops from iterating,

//by assigning the value of subscript to j, because j at that position is the biggest subscript,

//for the fibonacci number that our array can hold

biggestSubscriptFound = true;

subscript = j ;

biggestSubscript = j ;

}

}

}

}

}

//if both condtions are true, our array went out of bound so I am displaying the biggest Fibonacci Number

if (biggestFibonaciFound == true && biggestSubscriptFound == true)

{

cout << "ERROR: Out of Range "<< endl;

cout << "Biggest Fibonaci Subscript is : " << biggestSubscript << endl ;

cout << "Biggest Fibonaci is ";

//Fibonacci number is in reversed format, since we worked from left to right, and math

//operations are right to left

for (int i = 98; i >= 0; i--)

{

cout << Fibonaci[i];

}

}

//if conditions not met,

//then our subscript did not belong to a Fibonacci number with more than 99 digits

//then, display the Fibonacci number of the subscript given.

else

{

//reversing the array

cout << "Fibonaci Number is: ";

for (int i = 98; i >= 0; i--)

{

cout << Fibonaci[i];

}

}

cout << endl;

system("Pause");

return 0;

}

EXCECUTION





