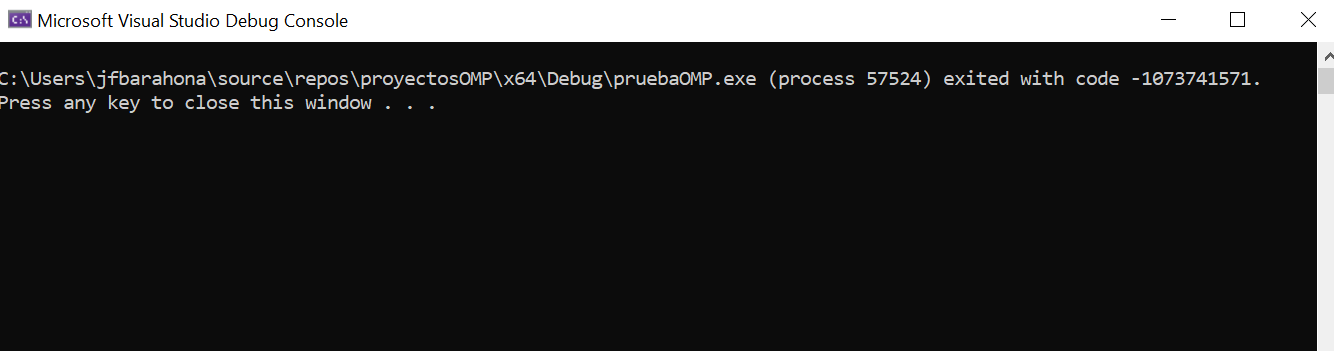
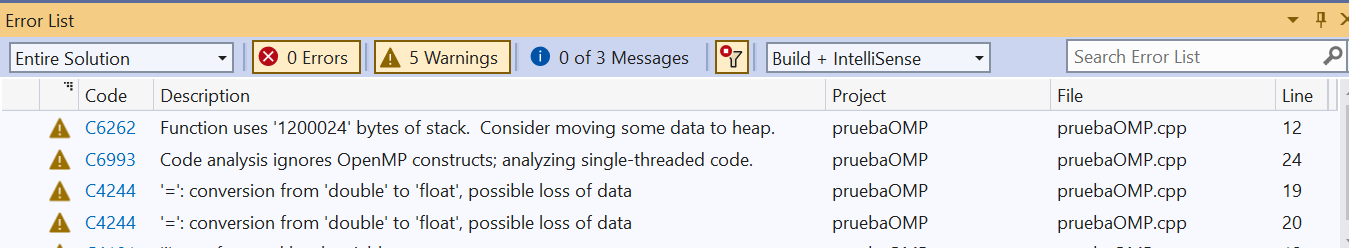


#define N 100000

#define chunk 10000

#define mostrar 1000



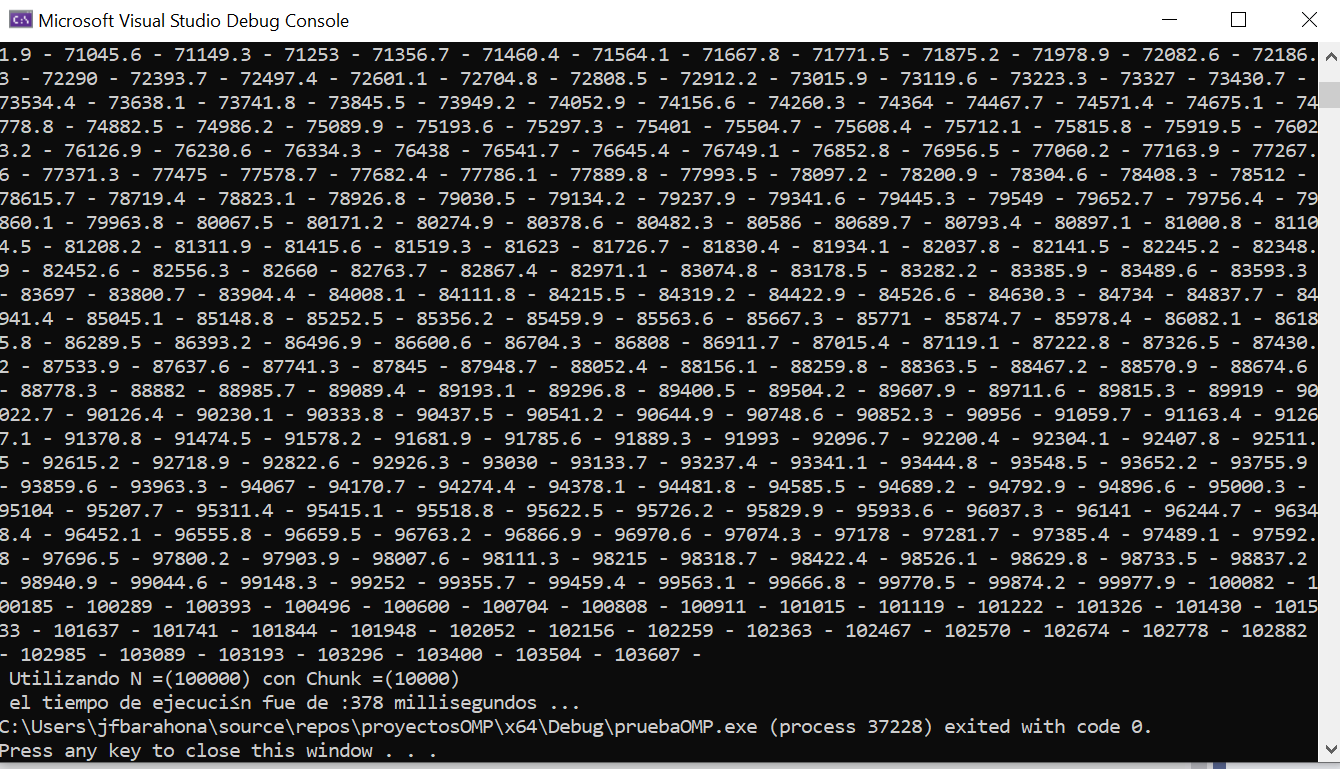


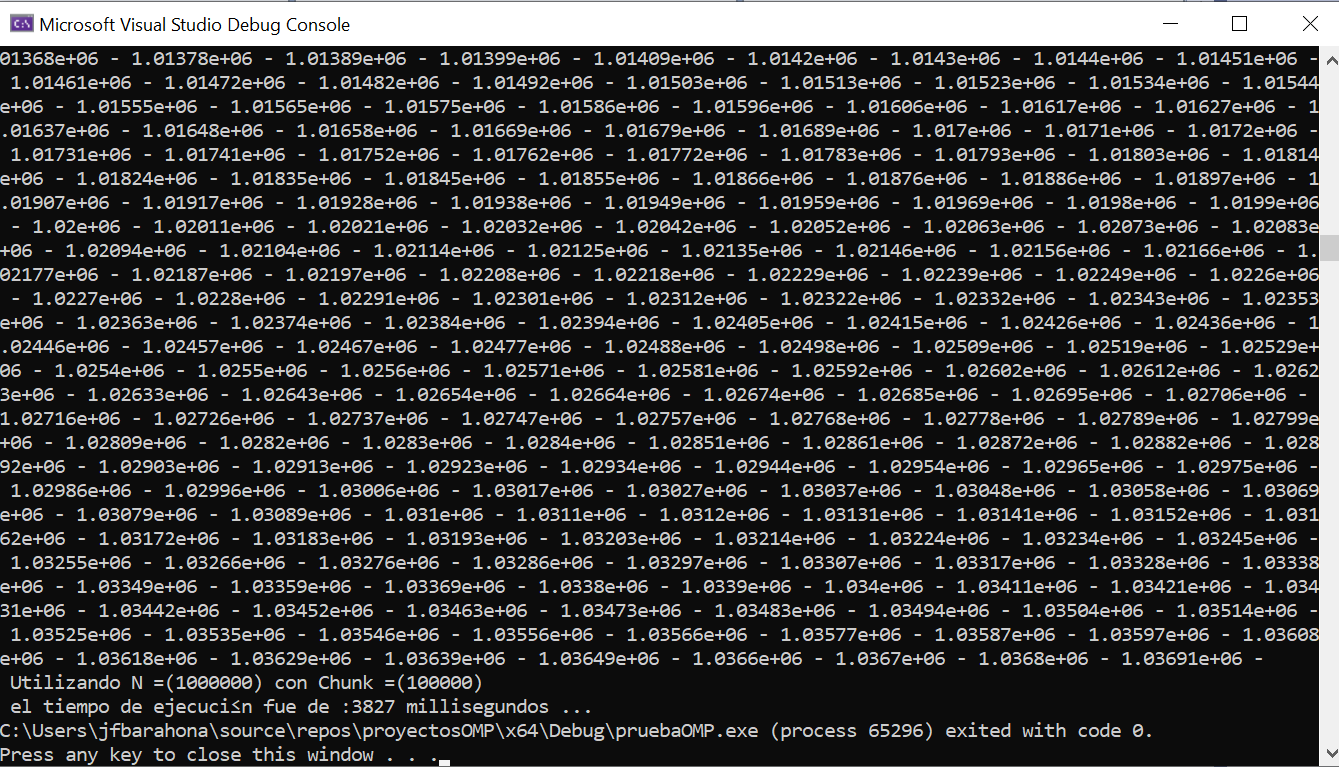
Se modificó

float a [N], b= [N], c [N];

Por

float \*a= new float[N], \*b= new float[N], \*c = new float[N];





Sin Paralelismo

#include <iostream>

#include <chrono>

//#include <omp.h>

#define N 1000000

#define chunk 100000

#define mostrar 10000

void imprimeArreglo(float\*);

int main()

{

"Sumando Arreglos en Parealelo!\n";

float \*a= new float[N], \*b= new float[N], \*c = new float[N];

int i=0;

for (i = 0; i < N; i++)

{

a[i] = (float)i \* 100.0;

b[i] = ((float)i + 3.0) \* 3.7;

}

int pedazos = chunk;

std::chrono::steady\_clock::time\_point begin = std::chrono::steady\_clock::now();

// #pragma omp parallel for \

// shared(a,b,c,pedazos) private(i)\

// schedule(static,pedazos)

for (i = 0; i < N; i++)

{

c[i] = a[i] + b[i];

}

std::cout << "Imprimiendo los primeros (" << mostrar << ") valores del arreglo a: \n";

imprimeArreglo(a);

std::cout << "Imprimiendo los primeros (" << mostrar << ") valores del arreglo b: \n";

imprimeArreglo(b);

std::cout << "Imprimiendo los primeros (" << mostrar << ") valores del arreglo c: \n";

imprimeArreglo(c);

std::chrono::steady\_clock::time\_point end = std::chrono::steady\_clock::now();

std::cout << " Utilizando N =(" << N << ") con Chunk =(" << chunk << ")\n el tiempo de ejecución fue de :" <<

std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin).count() << " millisegundos ...";

}

void imprimeArreglo(float\* d)

{

for (int x = 0; x < mostrar; x++)

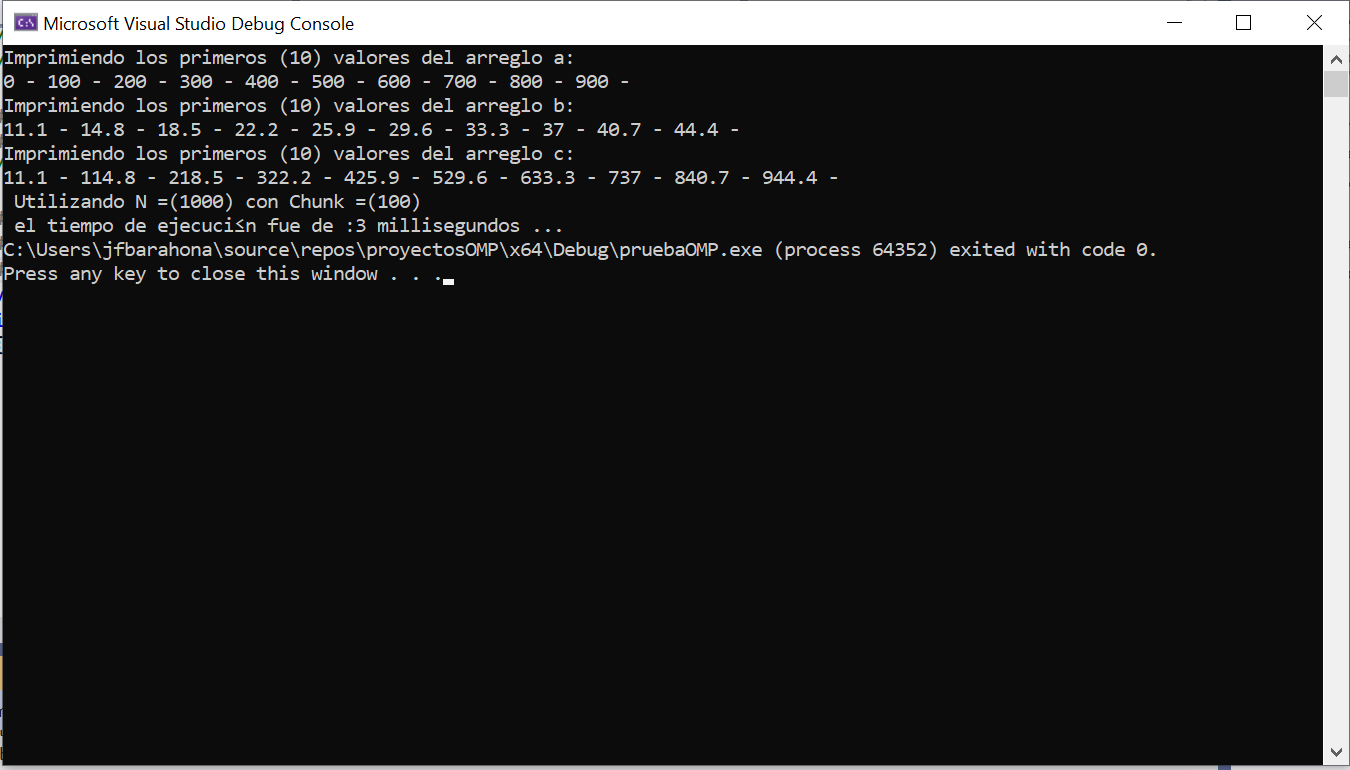
{

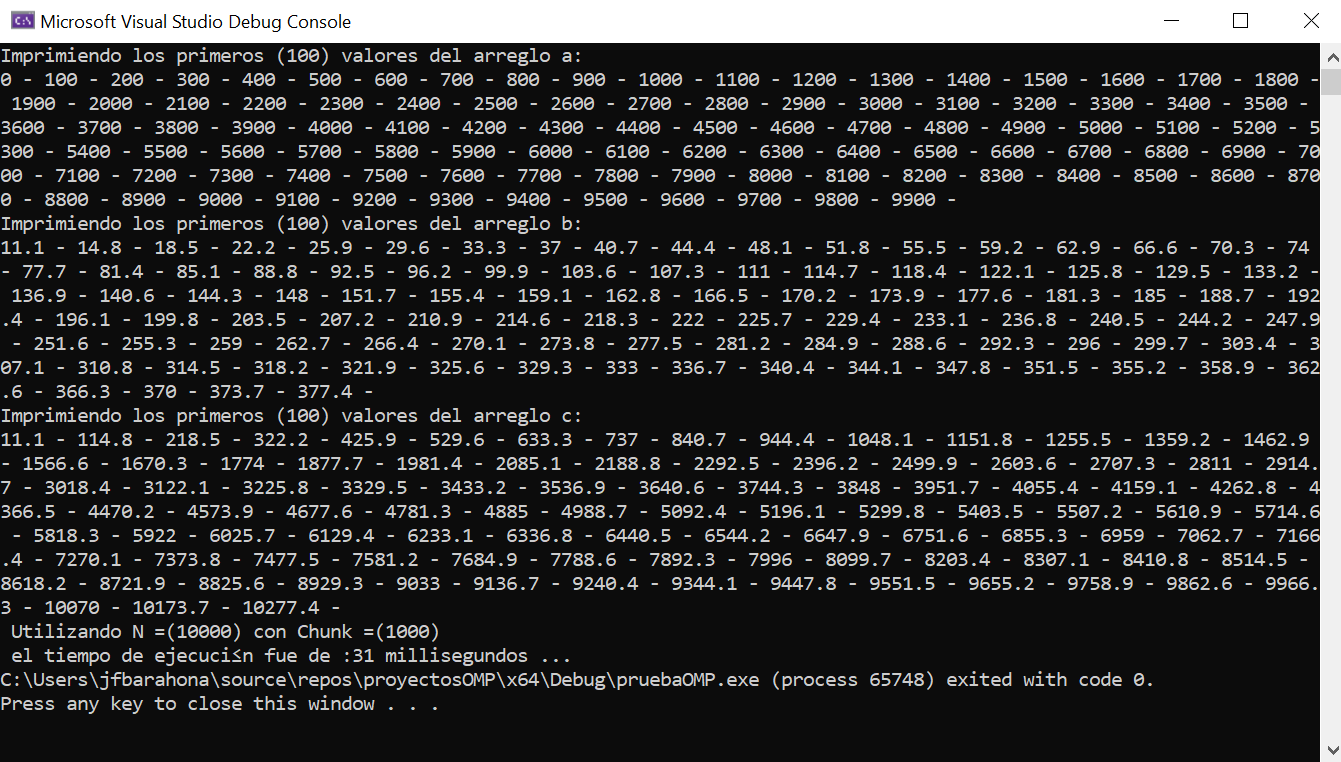
std::cout << d[x] << " - ";

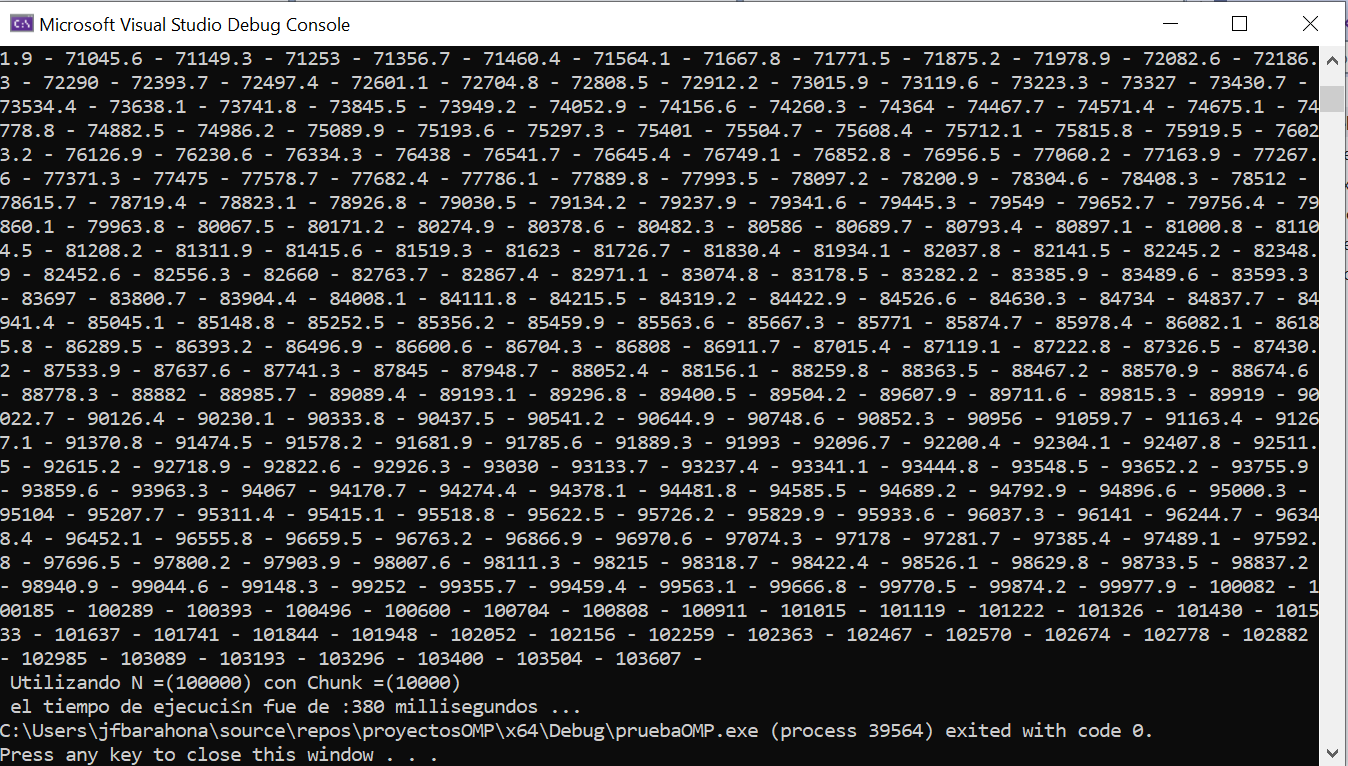
}

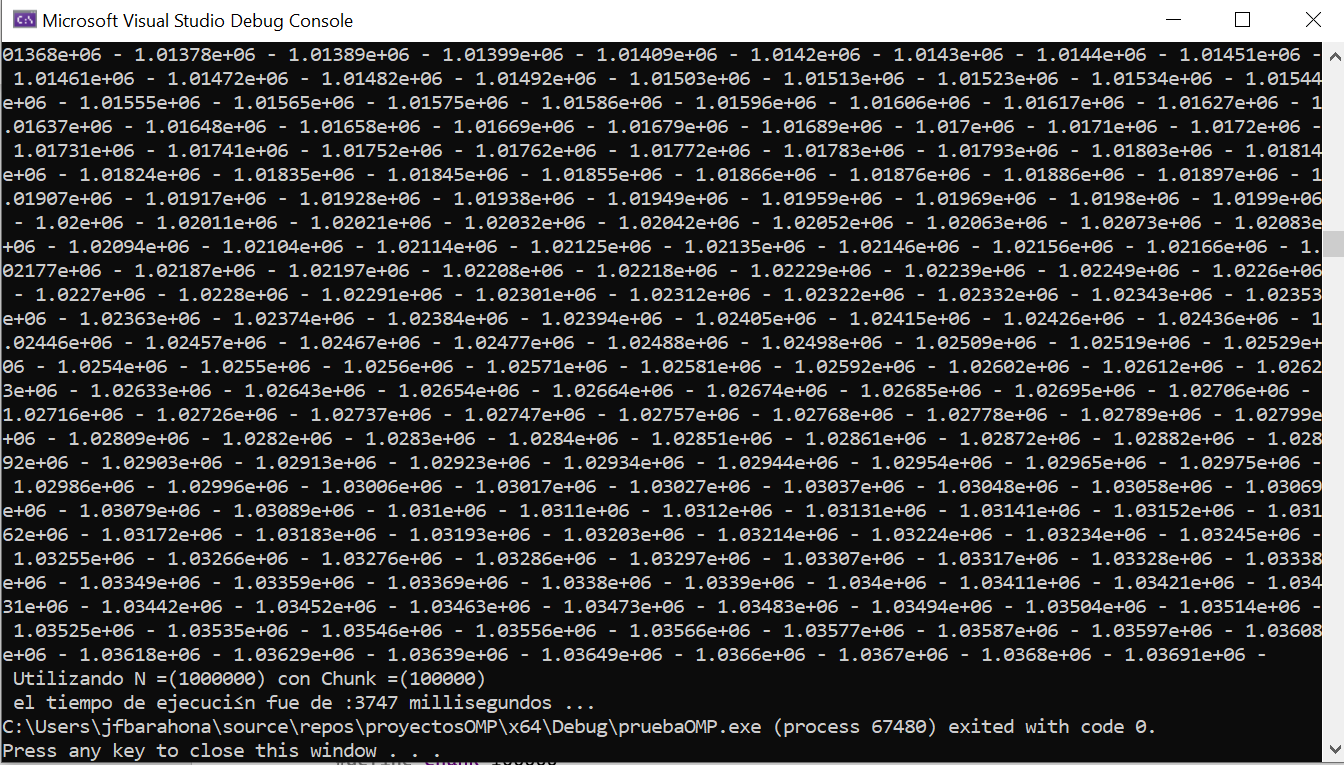
std::cout << std::endl;

}









Con Paralelismo en Heap

