**GARVIL JAIN 2022UCP1302 LAB3**

#include <stdlib.h>

#include <time.h>

#include <omp.h>

#include<stdio.h>

void generateRandomNumbers(int\* numbers,int n) {

#pragma omp parallel

{

//unsigned int seed = omp\_get\_thread\_num();

//#pragma omp for

for (size\_t i = 0; i < n; ++i) {

numbers[i] = rand() % 100 + 1;

}

}

}

int findMaxValue(int\* numbers,int n) {

int maxValue = 0;

#pragma omp parallel

{

#pragma omp barrier //sync point

#pragma omp single

{

for (int i=0;i<n;i++) {

if (numbers[i] > maxValue) {

maxValue = numbers[i];

}

}

}

}

return maxValue;

}

void threadCounters(int numThreads) {

int counters[numThreads];

#pragma omp parallel

{

int threadID = omp\_get\_thread\_num();

for (int i = 0; i < 10; ++i) {

counters[threadID] += threadID;

}

}

for (int i = 0; i < numThreads; ++i) {

printf( "Counter for thread %d is \n", counters[i]);

}

}

int main() {

const size\_t arraySize = 100;

int numbers[arraySize];

generateRandomNumbers(numbers,arraySize);

printf("Generated numbers:\n");

for(int i=0;i<arraySize;i++){

printf("%d ", numbers[i]);

}

numbers[50]=100;

printf("\n");

int maxValue = findMaxValue(numbers,arraySize);

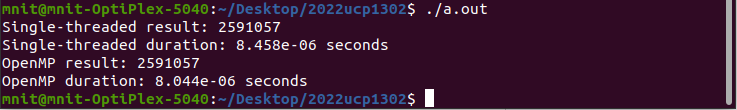
printf("Maximum value: %d \n", maxValue );

int numThreads = omp\_get\_max\_threads();

threadCounters(numThreads);

return 0;

}



#include <stdlib.h>

#include <time.h>

#include <omp.h>

#include<stdio.h>

void generateRandomNumbers(int\* numbers,int n) {

#pragma omp parallel

{

//unsigned int seed = omp\_get\_thread\_num();

//#pragma omp for

for (size\_t i = 0; i < n; ++i) {

numbers[i] = rand() % 100 + 1;

}

}

}

int findMaxValue(int\* numbers,int n) {

int maxValue = 0;

#pragma omp parallel

{

#pragma omp barrier //sync point

#pragma omp single

{

for (int i=0;i<n;i++) {

if (numbers[i] > maxValue) {

maxValue = numbers[i];

}

}

}

}

return maxValue;

}

void threadCounters(int numThreads) {

int counters[numThreads];

#pragma omp parallel

{

int threadID = omp\_get\_thread\_num();

for (int i = 0; i < 10; ++i) {

counters[threadID] += threadID;

}

}

for (int i = 0; i < numThreads; ++i) {

printf( "Counter for thread %d is \n", counters[i]);

}

}

int main() {

const size\_t arraySize = 100;

int numbers[arraySize];

generateRandomNumbers(numbers,arraySize);

printf("Generated numbers:\n");

for(int i=0;i<arraySize;i++){

printf("%d ", numbers[i]);

}

numbers[50]=100;

printf("\n");

int maxValue = findMaxValue(numbers,arraySize);

printf("Maximum value: %d \n", maxValue );

int numThreads = omp\_get\_max\_threads();

threadCounters(numThreads);

return 0;

}

