

Vectors are normally passed by value, as opposed to arrays.

However vectors occupy a large space in memory and thus should NOT be passed by value, which makes a copy of it.

Instead pass it by reference:

The function declaration will look like:  
void MyFun(vector<int>&);

Function call looks like:

vector<int> v;

MyFun(v);

/////////////////////////////////////////////////////

//C++ For C Programmers

//Finding PI by Monte Carlo Integration

//Use of random

//Ira Pohl April 2016

#include<iostream>

#include<random>

#include<vector>

#include<tuple> //C++11

#include<algorithm>

#include<ctime> //use time(nullptr) for seed

#include<cmath> //use sqrt

using namespace std;

typedef tuple<double, double> point;

ostream& operator<<(ostream& out, const point& pt)

{

out << "( " << get<0>(pt) << ", ";

out << get<1>(pt) << ") ";

return out;

}

default\_random\_engine e(time(nullptr)); //seed in constructor

point random\_point()

{

uniform\_real\_distribution<double> u(0, 1);

point temp;

get<0>(temp) = u(e);

get<1>(temp) = u(e);

return temp;

}

double mc\_count(double f(double), vector<point>::iterator first , vector <point>::iterator last)

{

int trials = 0; int below = 0;

for(first; first != last; ++first){

++trials;

if (f(get<0>(\*first)) > get<1>(\*first))

++below;

}

return( 1.0\* below/trials);

}

int main()

{

int data\_size;

cout << " How Many Points? " << endl;

cin >> data\_size;

vector<point> data(data\_size);

for(auto& element: data)

element = random\_point();

cout << " PI by MC integration is " <<

4.0 \* mc\_count([](double x){ return (sqrt( 1 - x \* x));}, data.begin(), data.end());

cout <<endl;

}

The above code illustrates a number of points – including features found in C++11.

It uses the new <random> library that improves on rand() that was discussed in the video. It is expected that the use of rand() will be phased out.

See : http://www.cplusplus.com/reference/random/ for details.

We use a lambda expression as an argument. We could also have used an old style named function (You should try this). The function computes the circle of radius 1 centered at (0,0) in the plane.

The Monte Carlo method generates points at random in the unit square in the upper right quadrant. The area of that square is of course 1.0. Now we can ratio of points that are below that function. That ratio should approximate PI/4.

We use the C++11 <tuple> library (can consider it a generalization of pair<> from the <utility> library) to define a point.

See : http://www.cplusplus.com/reference/tuple/ for details.