**CMP305L - Data Structures and Algorithms Lab**

**Lab. Assignment 8 – Advanced Topics in Recursion**

***Exercise 1***

Without accumulator:

#include<iostream>

#include<cmath>

using namespace std;

double PI(int n) {

if (n < 1)return 3.0;

else if (n % 2 == 0) {

//int x = -1;

return((-4.0) / ((2 \* n)\*((2 \* n) + 1)\*((2 \* n) + 2))) + PI(n - 1);

}

return (4.0 / ((2 \* n)\*((2 \* n) + 1)\*((2 \* n) + 2))) + PI(n - 1);

}

void main(){

int n;

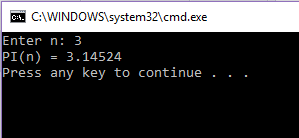
cout << "Enter n: ";

cin >> n;

double pi = PI(n);

cout << "PI(n) = " << pi << endl;

}



With accumulator:

#include<iostream>

#include<cmath>

using namespace std;

double PI(int n, double i) {

if (n < 1)return i;

else if (n % 2 == 0) {

//int x = -1;

return PI(n - 1, i + ((-4.0) / ((2 \* n)\*((2 \* n) + 1)\*((2 \* n) + 2))));

}

return PI(n - 1, i + ((4.0) / ((2 \* n)\*((2 \* n) + 1)\*((2 \* n) + 2))));

}

void main(){

int n;

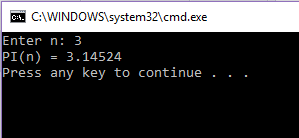
cout << "Enter n: ";

cin >> n;

double pi = PI(n, 3.0);

cout << "PI(n) = " << pi << endl;

}



***Exercise 2***

//https://blogs.msdn.microsoft.com/calvin\_hsia/2013/08/30/call-managed-code-from-your-c-code/

using namespace System;

using namespace System::IO;

void listDirctories(array<String^>^ entries, int count)

{

if (count == entries->GetLength(0)) return;

Console::WriteLine(entries->GetValue(count));

listDirctories(entries, count + 1);

}

void listFiles(array<String^>^ entries, int count)

{

if (count == entries->GetLength(0)) return;

entries[count] = entries[count]->Replace("C:\\", "");

Console::WriteLine(entries[count]);

listFiles(entries, count + 1);

}

void main()

{

// Process the list of files found in the directory.

Console::WriteLine("Print all directories in the directory C:\\ ");

array<String^>^ dirEntries = Directory::GetDirectories("C:\\");

listDirctories(dirEntries, 0);

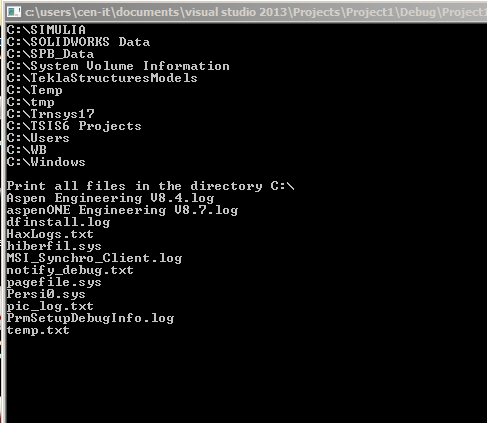
Console::WriteLine();

Console::WriteLine("Print all files in the directory C:\\ ");

array<String^>^ fileEntries = Directory::GetFiles("C:\\");

listFiles(fileEntries, 0);

Console::Read();



}***Exercise 3***

void draw(int n, double x, double y, double size)

{

if (n == 0)return drawH(x, y, size);

else {

double x0 = x - size / 2;

double x1 = x + size / 2;

double y0 = y - size / 2;

double y1 = y + size / 2;

draw(n - 1, x0, y0, size / 2);

draw(n - 1, x0, y1, size / 2);

draw(n - 1, x1, y0, size / 2);

draw(n - 1, x1, y1, size / 2);

}

}

int main(int argc, char \*\*argv)

{

CImg<unsigned char> img0(640, 500, 1, 3, 0), img1;

CImgDisplay disp(img0, "H-Tree");

while (!disp.is\_closed() && !disp.is\_keyESC()) { // Esc to quit

img1 = img0; img = &img1;

int x = 320, y = 250, size = 100;

int n = argc;

//Draw H recursively

drawH(x, y, size);

//Draw H-Tree

draw(n, x, y, size);

img0.resize(disp.display(img1).resize(false).wait(100));

}

return 0;

}

