Homework 2

Programming in C++

Due Sep 16, 2020

1: Compiled versus interpreted languages (10 pts)

In your own words, please summarize the major similarities and differences between compiled and interpreted programming languages, and provide at least three examples (programming languages) of each.

Both compilers and interpreters perform syntactical analysis on the code. If there is a syntactical error in the code, they might both break. The main difference between the two is that a compiled language requires that the entire code be translated into something the machine can understand - through a compiler, while the interpreted language is interpreted line by line on a virtual machine and translated to something the machine can understand.

Compiled languages: C, C++, Go, Fortran, Pascal, a million others:
https://en.wikipedia.org/wiki/Compiled_language
Interpreted languages: Python, PHP, Javascript, a million others:
https://en.wikipedia.org/wiki/Interpreted_language

2: Boolean operators (10 pts)

Below are five separate lines of `C++` code. Each of them evaluates to a boolean logical (*true* or *false*). Please write whether the expression will evaluate to *true* or *false*:

`!false`: **True**

`true || false`: **True**

`true && false`: **False**

`(true || false) && false`: **False**

`(10 == 11 ) && (true || false) && true`: **False**

3: Input / output (30 pts)

Please write a C++ program which performs the following:

1. asks the user to input two values (of type `double`), one at a time
2. computes the product of the two numbers
3. outputs the product to the screen, with at least 6 digits of precision

HINT: Use the `cin` and `cout` commands. We did not do an example in lecture. You must use what you've learned in class and apply it to this new package. If you get stuck, please contact myself or Liam.

The program should compile and execute without errors. This program should be stored in a file called `mult.cpp` and submitted along with the PDF of your assignment solutions in a compressed folder. ALL CODES MUST BE WELL COMMENTED. If your code is not well commented, you will receive at most 50% of the points for this problem.

Code in folder. Students may either do this in their main routine or in a function.

4: A randomized algorithm to approximate π (20 pts)

Complete the class activity from Lecture 5. Assume you are given a domain with $x, y \in [0, 1]$. Generate a random pair of coordinates within this domain, and determine if they fall within a circle with radius $r = 0.5$ centered at $(0.5, 0.5)$. Count the number of times that the point falls within the circle, and compute π as

$$\hat{\pi} = 4\frac{N_{hit}}{N_{tot}}$$

where $\hat{\pi}$ is an approximation of π, N_{hit} is the number of points that lie inside the circle, and N_{tot} is the number of random coordinate pairs drawn, total. Please fill in the following table by reporting your approximation to π for varying values of N_{tot}:

N_{tot}	$\hat{\pi}$
10^2	**This will be somewhere between 3ish and 3.5ish**
10^3	**This will be somewhere between 3.1ish and 3.3ish**
10^4	**This will be somewhere between 3.13ish and 3.16ish**
10^5	**This will be somewhere between 3.137ish and 3.147ish**
10^6	**This will be somewhere between 3.141ish and 3.144ish**

Since we are using random numbers, these results will vary everytime the code is executed. Students do not need to hand in code for this problem.

What do you notice?

As we increase the number of total darts thrown, we are better able to approximate π. This is what we expect to happen, as we throw more and more darts we expect the entire surface to eventually become covered in darts. If the space is fully covered in darts, our equation to approximate π becomes more accurate.

5: Stopping criteria in approximating π (30 pts)

Copy the file `computepi.cpp` into a new file called `user_computepi.cpp`. Redesign the program so that instead of stopping after a prescribed number of iterations, the program exits once the relative absolute error in computing π has diminished to a value, ε, **specified by the user**. In other words, there the for loop is not bounded by N_{tot} (HINT: use a `while` loop instead of a `for` loop). The absolute relative error can be calculated as

$$E = \frac{|\pi - \hat{\pi}|}{\pi},$$

The iterations stop when

$$E < \varepsilon.$$

The program should ask the user for a stopping tolerance (HINT cin). Report the total number of iterations required when running the program with $\varepsilon = 10^{-2}$, 10^{-3}, and 10^{-4}. Use the command `#define`

```
#define MYPI c
```

to define the exact value of π.

N_{tot}	$\hat{\pi}$
10^2	**Varied between 9 and 36**
10^3	**Varied between 14 and 1247**
10^4	**Varied between 191 and 657**

These could really be anything. In the future I will advise to execute 5 times for each and take an average. Grade more on their code structure than these responses. The program should compile and execute without errors. This program should be stored in a file called `user_computepi.cpp` and submitted along with the PDF of your assignment solutions in a compressed folder. ALL CODES MUST BE WELL COMMENTED. If your code is not well commented, you will receive at most 50% of the points for this problem.

Submission:

Please include all files and PDF of your assignment solutions in a compressed folder (i.e. zip).