

AN INTRODUCTION TO PROGRAMMING THROUGH C++

with

Manoj Prabhakaran

Lecture 1

Introduction

and drawing some pictures

Based on material developed by Prof. Abhiram G. Ranade

Plan for Today

- About this course
- Getting started by coding up some simple programs



- In the lab, later today: setup your system and start coding!

About CS 101

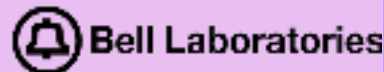
Computer Programming

- Computers can be used to do a lot of things
 - All the different apps in your phone, laptop, ...
 - All the online services you use are provided by large computers (servers) working together (network)
 - Computers embedded inside washing machines, car engines, ...
 - And then there are the "Cyber-Physical Systems": Flying an aeroplane, controlling a robot or a self-driving car, running a nuclear reactor, ...
- But they need to be programmed to do so
- In this course: Introduction to programming
 - No prior knowledge of computers expected
 - Will lay the foundation for all sorts of uses later on

Programming Language

- Programs (instructions for a computer) can be quite complex
- Programming Languages: Provide a human-friendly way to write the programs, while still being very precise
- Programming language used in this course: C++

1970s: **C** Programming Language (Dennis Ritchie)
developed for coding up the UNIX operating system



1980s: **C++** (Bjarne Stroustrup) for "Object Oriented Programming"

Standards: C++98, C++03, **C++11**, C++14, C++17, C++20, C++23, ...



- Note: We will not learn about all the features of C++ in this course.

The programming environment

- Will set up in today's lab!

Initial weeks: C++ augmented with Simplecpp

Simplecpp is a C++ library developed in IITB

- Provides facilities convenient to learners
 - Graphics programming – more fun!
 - Some minor conveniences

Later weeks: Only C++

- We may continue to use Simplecpp graphics

The textbook

An introduction to programming through C++, Abhiram Ranade, McGraw Hill Education, 2014.

- www.cse.iitb.ac.in/~ranade/book.html
- Available in physical and on-line bookstores
- Integrated with use of simplecpp
- Reading for today's lecture: Chapter 1

Course Logistics

- Every week
 - Two lectures
 - One lab
 - Exercises for practice
- Learning Management System: Bodhitree
 - Lab submissions
 - Slides will be available soon after/before lecture
- Evaluation
 - Lab participation (10%)
 - Two hand-written quizzes (7% x 2)
 - Two lab quizzes (13% x 2)
 - Mid-semester exam (20%)
 - Final exam (30%)
- Course website:
<https://www.cse.iitb.ac.in/~cs101/>

Academic Integrity

Do not copy, consult or share in quizzes and exams!

Do not copy in lab submissions. You can consult references, discuss ideas with peers, and ask TAs for guidance/hints. But blindly copying defeats the purpose of the exercises.

All code submitted should be written by yourself

Any suspected cheating will be reported to D-ADAC


Let's Get Started!

Let us write some simple C++ programs

- The programs will draw pictures on the screen!
- Use “Turtle Simulator” contained in simplecpp
 - Based on Logo: A language invented for teaching programming to children by Seymour Pappert et al.
 - We give commands to a turtle to move around.
 - Turtle has a pen, so it draws as it moves.

The first program

A simplecpp
convenience



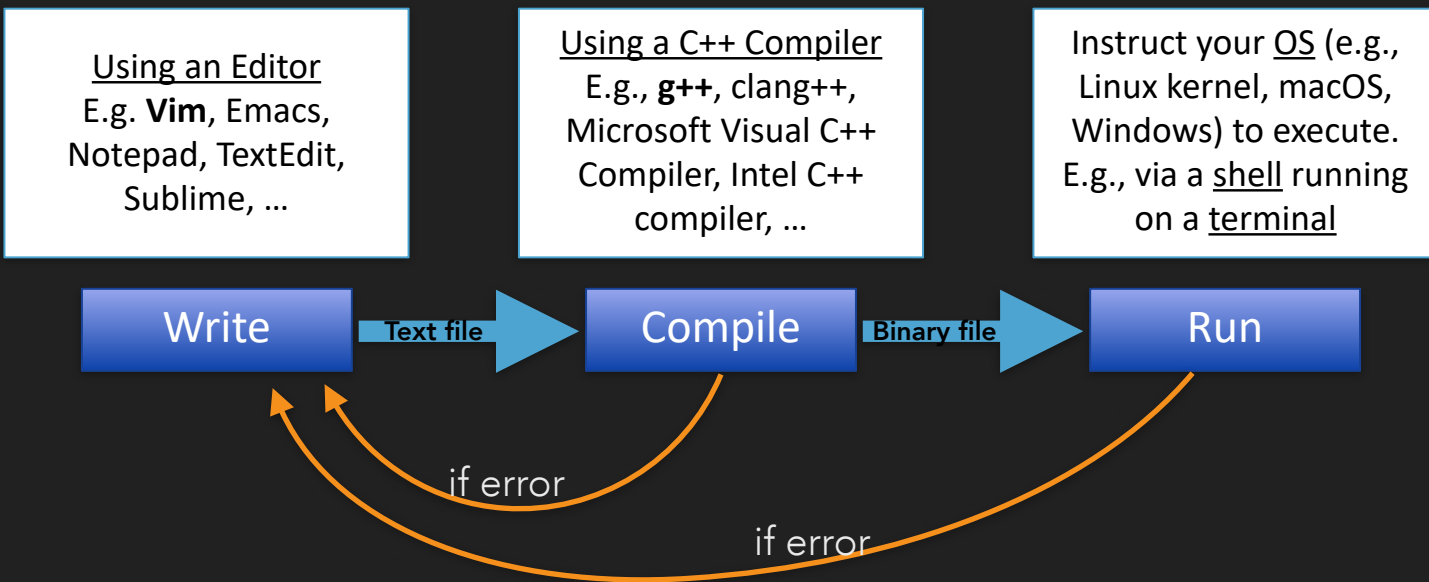
```
#include <simplecpp>

main_program {
    turtleSim();
    forward(100);    right(90);
    forward(100);    right(90);
    forward(100);    right(90);
    forward(100);
    getClick();
}
```

- ▶ “Use simplecpp facilities”
- ▶ Main program, enclosed in { ... }
- ▶ Start turtle simulator
 - ▶ Creates window + turtle at centre, facing right
- ▶ `forward(n)` :
 - ▶ Move the turtle n pixels in the direction it is currently facing.
- ▶ `right(d)` :
 - ▶ Make turtle turn d degrees to the right.
- ▶ `getClick()` :
 - ▶ Wait for a click.

How to run this program

Demo



A better way to draw a square

```
#include <simplecpp>
main_program{
    turtleSim();
    repeat(4){
        forward(100);
        right(90);
    }
    getClick();
}
```

Repeat Statement

A simplecpp
convenience

- **Syntax** (i.e., the form):

```
repeat (n) { body }
```

- Here, *body* consists of one or more statements.
- **Semantics** (i.e., the meaning): Body should be executed n times.
- repeat is an example of a **loop** (will see more loop structures later)
- In a loop, each execution of the body is called an **iteration**

Drawing a polygon

Demo

```
#include <simplecpp>
main_program {
    turtleSim();
    cout << "How many sides?";
    int nsides;
    cin >> nsides;
    repeat(nsides){
        forward(100);
        right(360.0/nsides);
    }
    getClick();
}
```

The *type* of the variable.
int for "integer".

Print to the screen

A *variable* called "nsides"
whose value can be set or
changed later

Read from keyboard into the
variable nsides

Repeat "nsides" times

In each iteration of the body,
draw one side, and turn
enough to start the next side

An arithmetic operation

More simplecpp commands

- `left(A)` : turn left `A` degrees. Equivalent to `right(-A)`
- `penUp()` , `penDown()` : Causes the pen to be raised, lowered
Drawing happens only if the turtle moves while the pen is low.
- `hide(true)` , `hide(false)` : The pen indicator (triangle) is hidden, shown
- `sqrt(x)` : square root of `x`.
- `sine(x)` , `cosine(x)` , `tangent(x)` : `x` should be in degrees.
- `sin(x)` , `cos(x)` , `tan(x)` : `x` should be in radians.
- Also commands for arcsine, arccosine, arctangent... See the book.

Nested Loops

- Consider

```
repeat (n) { body }
```

- Suppose *body* itself has a loop within it:

```
repeat (n) { ... repeat(m) { Inner body } ... }
```

- In this example, *Inner body* will be executed $m \cdot n$ times
- The rest of the outer body will be executed only n times

Drawing the axes

Demo

```
#include <simplecpp>
main_program{
    turtleSim();
    repeat(4) { // in each of four directions
        repeat(5) { // will mark 5 big steps, each 50 units long
            right(90); forward(4); forward(-8); forward(4); right(-90); // a big tick mark
            repeat(9) { // One big step has 10 small steps, and 9 small ticks
                forward(5); // a small step
                right(90); forward(2); forward(-4); forward(2); right(-90); // a small tick mark
            }
            forward(5); // 10th small step.
        }
        forward(-250); // come back to the origin
        right(90); // and turn to the next axis
    }
    hide(true);
    getClick();
}
```

Comment

Indentation

Exercises

- Draw a square of side 100 and inscribe a square inside it



- Hint: Use Pythagoras's theorem to determine the length of the side

- Draw the outer square using a dashed line.



- Can you do it for the inner square also? How will you choose the dash-gap length?

- Draw an n-point star



- Hint: There are two angles of interest. One determines how "pointy" the star is. The other related to the first via n : turtle turns left and right, but when it finishes drawing it'd have turned net 360°