CS 101: Computer Programming and Utilization

July-Nov 2016

Prof. Bernard L Menezes
(cs101@cse.iitb.ac.in)

Lecture 1: Introduction

About These Slides

- Based on Chapter 1 of the book
  An Introduction to Programming Through C++
  by Abhiram Ranade (Tata McGraw Hill, 2014)

- Original slides by Abhiram Ranade
  – First update by Varsha Apte
  – Second update by Uday Khedker

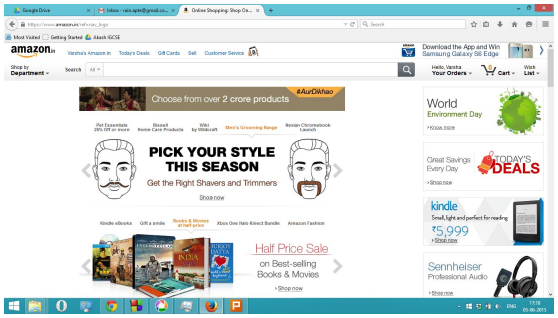
Computers and You

- So far almost all of you have used a computer

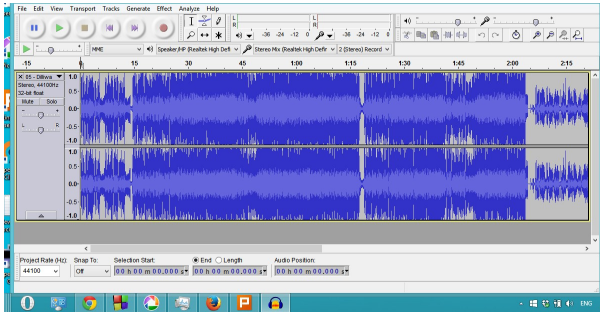
  - Desktop
  - Laptop
  - Smartphone

    (is a computer)

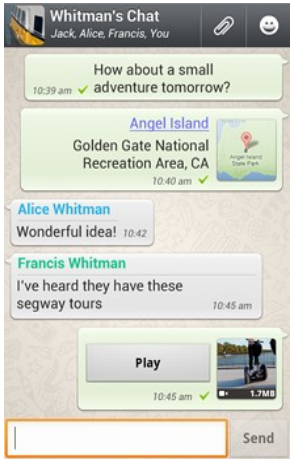
"Applications" or "Apps"



Web Browser



Audio editor



Whatsapp

Games

Application are *programs*

A Computer Program

- A program is a sequence of instructions that a computer can *execute*
  - A *programmer* creates this sequence of instructions - i.e. *writes the program*
    - (or applications or apps)
- So far you have been a *user* of these programs
- In this course you will learn how to become a *programmer*

We will not learn Android App programming. They are just examples of programing. But you will learn enough basic principles of programming to enable you to learn to program in different environment relatively easily.

So Let's Start Programming!

First program in a simplified version of the C++ language called *simplecpp*

Our First Program

- Use a Turtle Simulator* contained in simplecpp
- We drive a turtle on the screen!
- To drive the turtle you write a C++ program
- Turtle has a pen, so it draws as it moves

Basic goal: draw interesting, intricate pictures

*From Logo: A language invented for teaching programming by Seymour Pappert et al. (1967)

Getting Started

- Open a file for editing
- Write turtleSim() in the main_program
- Compile-and-execute
- A green turtle should be seen facing east

```
#include<simplecpp>
  main_program {
  turtleSim();
  }
```

How to Run This Program

- We will use Prutor (Programming Tutor) System

  https://cs101.cse.iitb.ac.in

  We use the same system in the class

- We will learn a more general and more powerful approach of running programs from command line, later in the course

Some Instructions That TurtleSimulator Can Execute

- penUp()
  - Will not draw while moving
- penDown()
  - Will draw while moving
- forward (x): Move forward x pixels

  x:*Distance*
  - E.g. forward(50) moves the turtle forward 50 pixels
- right (x): turn right by x degrees
- left(x): turn left by x degrees

  x:*Angle*

Our First Task

- With these instructions, make the turtle move in such a way that we will draw a square of side length 200

- Note: by default, in the beginning,
  the turtle faces towards east, and
  the pen is down

A Program to Draw A Square

- Instructions:
  - –forward(x)
  - –right (x)
  - –left(x)
  - –penUp()
  - –penDown()
  - –wait(x)

```
#include<simplecpp>

 main_program {

      turtleSim();
      forward(200);right(90);
      forward(200);right(90);
      forward(200);right(90);
      forward(200);
    }
```

Explanation

```
#include <simplecpp>


main_program {
    turtleSim();
    forward(200); right(90);
    forward(200); right (90);

    forward(200);  right(90);
    forward(200);
}
```

the program will use the simplecpp package.

Your commands within these braces {...}package.

Start the turtle simulator (open a window)

Move forward 200 units

Turn right 90 degrees

Program exits

The C++ Programming Language

- Designed by Bjarne Stroustrup, 1980s
- Derived from the C programming language
- Substantial evolution (still continues)
- Early part of our course: C++ augmented with a package called simplecpp

  (designed by Abhiram Ranade)

  More fun and easier to use than bare C++ Built-in graphics

General Ideas

```
#include<simplecpp>
  main_program{
  turtleSim();
        forward(200); right(90);
        forward(200); right(90);
        forward(200); right(90);
        forward(200);
}
```

Commands or statements terminated by semicolon ";"

This sequence of commands in C++ is the program

Some commands need additional information called *arguments*

- 90 is the argument to the command right
- 200 is the argument to the command forward

General Ideas (contd)

```
#include<simplecpp>
 main_program{
     turtleSim();
     forward(200);
     right(90);
     forward(200);
     right(90);
     forward(200);
     right(90);
     forward(200);
     }
```

Commands are generally executed from top to bottom, left to right.
(we can override this default)

General Ideas (contd)

- *Compiling* a program:

  Translating it into a form that your computer can understand

- The result of compilation: An *executable* file

- This is done internally by Prutor

  (By invoking a C++ compiler)

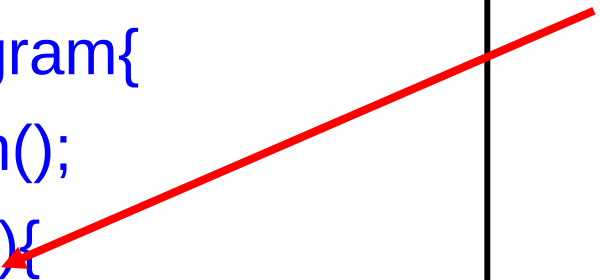
How to Draw An Octagon?

```
main_program{
    turtleSim();
    forward(100);  right(45);
    forward(100);  right(45);
    forward(100);  right(45);
    forward(100);  right(45);
    forward(100);  right(45);
    forward(100);  right(45);
    forward(100);  right(45);
    forward(100);  right(45);
    forward(100);  right(45);
  }
```

- Commands seem quite repetitive?
- There's a better way!

A Better Way

```
#include <simplecpp>
main_program{
   turtleSim();
   repeat(8){
        forward(100);
                right(45);
   }
}
```



repeat statement:

```
repeat (n) {
        some commands
 }
```

is a command that can be compiled by simplecpp

The instructions within {...} are repeated *n* times

Each round of execution is called an *iteration*

How to Draw a Polygon

- We have removed repeated occurrences of a command
- Can we generalize it further to draw a polygon of *any number of sides??*
- Yes! By using *variables*!

```
#include <simplecpp>
main_program{
    turtleSim();
    cout << “No. of sides?”;
    int noofsides;
    cin >> noofsides;
    repeat(noofsides){
        forward(10);
            right(360.0/noofsides);
    }
}
```

Explanation

```
#include <simplecpp>
main_program{
    turtleSim();
    cout << "No. of sides?";
    int noofsides;
    cin >> noofsides;
    repeat(noofsides) {
        forward(200);
        right(360.0/noofsides);
    }
}
```

Print the sentence within the quotes on the screen (required in command line, not in Prutor)

Tell the computer: Reserve space in your memory where I can store an integer (int). I will refer to it by the name noofsides

Read the number that the user types and store it into the space in memory named noofsides

Use the integer stored in the space in memory which is named noofsides

Divide the number 360 by the number stored in the space named noofsides and pass the result as an argument to this command

More Commands/Functions

- sqrt(x) : square root of x
- Trigonometric functions,
  - x is in degrees: sin(x), cos(x), tan(x)
  - x is in radians sine(x), cosine(x), tangent(x)
- Also for arcsine, arccosine, arctangent etc.

Remarks

You can use commands without worrying about how exactly do they do their work

- sqrt(17.35) : will get calculated somehow
- forward(100) : may require calculation which will happen (what calculation?)

Repeat Statement Within Another Repeat Statement

```
repeat(4){
    repeat(3){
        forward(200); penUp();
        forward(200); penDown();
    }
    right(90);
}
```

Nested Repeat Statements

- Basic rule:
  repeat(n){ yyy }
  means
  Statements yyy to be executed x times

- If yyy contains repeat (m) {zzz},
  - Then the zzz is executed m times in each iteration of outer repeat
  - Thus zzz will get executed n X m times

What will the program fragment on previous slide do?

Nested Repeat Statements

```
repeat(4){
    repeat(3){
      forward(200); penUp();
      forward(200); penDown();
    }
    right(90);
}
```

It will draw a square with dashed lines

What Does the Following Program Do?

```
#include <simplecpp>
main_program{
        cout << “a”;
        repeat(5){
                cout << “b”;
                repeat(2){  cout << “c”; }
                cout << “d”;
        }
}
```

Answer

The program prints

abccdbccdbccdbccdbccd

Remarks: Some Terms

- Control is at statement w

  The computer is currently executing statement w

- Control flow

  The order in which statements get executed.

  −Execution starts at top and goes down (Sequence)

  −Retraced if there is a repeat statement (Iteration)

  −Later we will see selective executuion (Selection)

- Variable: used for storing data

  −Computer memory: blackboard

  −Variable: Space on the board in which a value can be written

  −Variables have names, e.g. noofsides. We can use the name to refer to the value written in the variable.

Why Picture Drawing?

- Picture drawing requires calculation

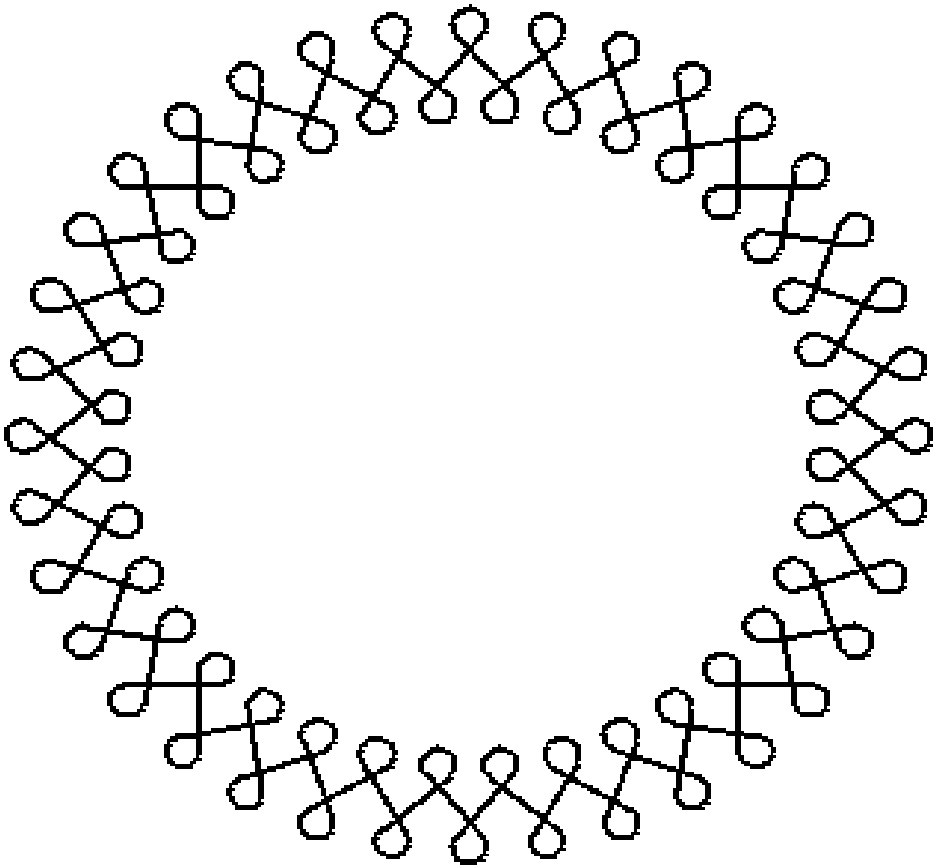
  e.g. 360.0/noofsides

- “Draw a triangle of sides with lengths 3, 4, 5 units”

  You will need to do trigonometric calculations to find out the angles between the sides

- More interesting calculations will be needed to draw more interesting drawings

A pattern with 36 repetitions. You know enough to write a program to do this! Try it.



Why Picture Drawing (contd)

- Interesting pictures contain patterns
- Most interesting calculations of any kind (not necessarily picture drawing) also contain patterns
- The pattern in the calculations must be mirrored by patterns in program
- Example: if a certain sequence of computations needs to be repeated, then do not repeat it textually, but put it in a repeat statement

More Reasons

- Graphical input and output is very convenient and useful.
- “A picture is worth a thousand words.”
- Data Visualization: upcoming area of CS
- Drawing is fun!

The Spirit of The Course

- Learn C++ statements/concepts

  We have covered a lot of ground in this lecture, even if it doesn't seem so
- Learn how to express problems you want to solve using C++.
- Goal: if you can solve a problem by hand, possibly taking an enormous amount of time, by the end of the course, you should be able to write a program for it
- Learn new ways of solving problems!

The Spirit of The Course

- Do not be afraid of using the computer
- “What if I write xyz in my program instead of pqr?”

  Just do so and find out
- Be adventurous.
- Exercise your knowledge by writing programs – that is the real test