An Introduction to Programming through C++

Abhiram G. Ranade

Introduction

Outline

- Introduction to computers and programming
- Some simple programs
- Remarks on programming
- Spirit of the course

Computers are everywhere!

- Cars, phones, laptops, game consoles, cameras, televisions contain a computer
- Computers used to:
  - Book train/plane/bus tickets
  - Search the internet
  - Predict the weather
  - …
- Goal of the course: Learn how to make computers do things such as the above

What is a computer?

A computer is a giant electrical circuit that can do the following:

- Receive data from the external world
  - data = numbers,
  - images, sounds can be represented using numbers and fed to a computer
- Perform calculations on the data it receives
- Send the results back to the external world

What calculations are performed?

- Determined by a program loaded in the computer

Programs

- Program = a precise description of the calculations we want the computer to perform
- By feeding different programs to a computer you can make it do different calculations.
- This course tells you how to construct (“write”) programs.
- Special notation is to be used to write programs: “Programming Language”

The C++ programming language

- Designed by Bjarne Stroustrup, 1980s.
- Evolved out of the C programming language.
- C++ is a powerful, complex language.
- We will not study all of it.
- What we study will still be more convenient and safer than C.
- We will lay the foundation for learning advanced features later.

The programming environment

Initial weeks: C++ augmented with Simplecpp

Simplecpp is a C++ library developed in IITB

- Provides facilities convenient to learners
  - Graphics programming – more fun!
  - Easy to understand “repeat” statement
  - “main_program” keyword
- Download from www.cse.iitb.ac.in/~ranade/simplecpp
  - Available as Linux/Mac OS library or as IDE for windows and Linux

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 this lecture sequence: Chapter 1

Prerequisites

- Science and math of standard XII
- No knowledge of computers expected
- Enthusiasm!

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 “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.

Drawing pictures seems too much fun?

“You master picture drawing, you master programming!”

The first program

```
#include <simplecpp>
main_program{
  turtleSim();
  forward(100);  right(90);
  forward(100);  right(90);
  forward(100);  right(90);
  forward(100);
  wait(5);
}
```

- “Use simplecpp facilities”
- Main program begins
- Start turtle simulator
  - Creates window + turtle at center, 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.
- `wait(t)` :
  - Do nothing for t seconds.
- `}` : End of main program

How to run this program

- Install simplecpp on your computer,
  - See instructions at www.cse.iitb.ac.in/~ranade/simplecpp
- Type in the program into a file/IDE. Call it square.cpp
- “Compile” it:
  - If you installed library on unix run: `s++ square.cpp`
  - If you installed code blocks IDE: use compile button
- Execute it:
  - On unix, run: `./a.out`
  - On code blocks: use run button

Demo

Exercises

- Write a program that draws a smaller square.
- Write a program that draws an equilateral triangle.
  - Remember that the external angles of a polygon add up to 360 degrees.
  - Also remember that all the external angles of an equilateral triangle are equal.

What we discussed so far

- General information about the course
- Installing simplecpp
- A program to draw a square

A better way to draw a square

```
#include <simplecpp>
main_program{
   turtleSim();
   repeat(4){
     forward(10);
     right(90);
   }
   wait(10);
}
```

Repeat Statement

- Form

```
repeat (x) { yyy }
```

- Causes the statements yyy inside { } to be executed x times.
- The statements yyy are called the body.
- Each execution of yyy is called an iteration.

How to draw a polygon

```
#include <simplecpp>
main_program{
  turtleSim();
  cout << “How many sides?”;
  int nsides;
  cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
  }
  wait(10);
}
```

`cout << msg;`

- Print message `msg` on the screen.

`int nsides;`

- Reserve a cell in memory in which I will store some integer value, and call that cell `nsides`.
- You choose the name as you wish, almost!
- `int` : abbreviation of “integer”

`cin >> nsides;` :

- Read a value from the keyboard and put it in the cell `nsides`.

`repeat(nsides)` : repeat as many times as content of `nsides`

`360.0/nsides` : result of dividing 360 by content of `nsides`.

Demo

More 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.

- `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 book.

Remarks

- You can use commands without worrying about how exactly they do their work.
  - `sqrt(17.35)` : will get calculated somehow.
  - `forward(100)` : Don’t worry how the turtle is moved on the screen

Exercises

- Draw a square of side length 100 as before. On top of this draw a square obtained by joining the midpoints of the sides of the first square.
  - Hint: Use Pythagoras's theorem to determine the length of the side of the inner square.
- Draw a dashed line, say a dash of 10 pixels, a gap of 10 pixels, so on 10 times.

What we discussed

- Repeat statements cause repeated executions of some statements.
- `cin` and `cout` statements can be used to read from keyboard and to type messages to the screen.
- C++ has commands to compute mathematical functions as well as lift the pen up and down.

General remarks about C++ programs

- Program = sequence of statements/commands.

`main_program{`... written here ...`}`

- Statement/command: terminated by “;”
- Commands are executed from top to bottom, left to right.
- Arguments: additional data needed by command to do its work.
  - `forward`: how much forward?
  - `right`: what angle?
  - () if no arguments, e.g. `turtleSim`()

Language syntax

- Syntax = grammatical rules indicating how commands must be written.
- Syntax of programming languages is very strict, e.g.
  - “right(90);” cannot be written as “right 90;”.
  - “penUp()” cannot be written as “penup()” or “penUp”, i.e. without parentheses.
  - We will later learn other kinds of statements which will have their own syntax which must be adhered to.
- Lot of flexibility is still allowed, e.g.
  - Wherever a number is acceptable, often an ”expression” such as 360/n is acceptable
  - repeat statement is allowed wherever other statements are allowed, e.g. we can have a repeat inside another repeat.

Comments

- A program will be executed on a computer, but it will also be read by people.
- Sometimes readers may not understand why the program is written the way it is written.
- To help such human readers, you can place “comments” in your program.
  - Anything placed between /* and */ is a comment
  - Anything between // and end of line is a comment
  - A comment is meant only for human readers and is ignored by the computer during execution.

A program with comments

```
/* Author: Abhiram Ranade
Program to draw polygon */
#include <simplecpp>
main_program{
  turtleSim();
  cout << “How many sides?”;
  int nsides; cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);  // Exterior angle of an n sided polygon is 360/n
  }
  wait(10);
}
```

Indentation

```
#include <simplecpp>
main_program{
  turtleSim();
  cout << “How many sides?”;
  int nsides; cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
  }
  wait(10);
}
```

- You will notice that at the beginning of some lines some space is inserted.
- This space is called indentation.
- The indentation allows you to quickly see which statements constitute the body of the repeat, which statements are part of the main program.
- The general rule: if X is ”inside” Y, then put two additional spaces before every line of X.
- Also note how the { and } are placed.
- Indentation is very helpful for human readers.
- Indentation is ignored during execution.

Repeat within repeat

```
repeat(4){
   repeat(3){
      forward(10); penUp();
      forward(10); penDown();
   }
   right(90);
}
```

How nested repeat statements execute

- `repeat(x){ yyy }` = execute `yyy` `x` times
- If `yyy` contains `repeat (w) {zzz}`, then the `zzz` is executed `w` times in each execution of `yyy`.
- And so on if there are repeats inside `zzz`

What will the following program do?

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

Answer

- The program will print

abccdbccdbccdbccdbccd

Some commonly used terminology

- “Control is at statement w”: Computer is currently executing statement w.
- “Control flow”: The order in which statements get executed.
  - Execution starts at top and goes down. Retraced if there is a repeat statement.
- Variable: region of memory designated for storing some value you need
  - Example: `nsides` which we saw earlier.
  - Named so because the value stored in the region can vary
  - How to change the value: later.

What we discussed

- General form of a program
- Notions of syntax, terms such as control flow.
- Repeat statements can be nested inside other repeat statements.

Why picture drawing

- The calculations/actions needed to solve any problem will contain patterns
  - For example, a sequence of calculations may be repeated
- Key programming activity: The pattern in the calculations must be mirrored by appropriate statements in program.
  - If some calculation is to be repeated 5 times, use `repeat(5){}` rather than writing out the statement 5 times
- Interesting pictures also contain patterns.
- To draw interesting pictures you need to use repeat statements competently
- By drawing interesting pictures you get some practice at representing patterns in your programs.

Other reasons why we focus on picture drawing

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

Spirit of the course

- Learn C++ statements/concepts.
  - We have already covered a lot of ground, even if it doesn't seem so.
- Understand patterns in the calculations that you want to do
  - Very important in all programming, not just drawing.
- Goal: if you can solve a problem by hand, possibly taking an enormous amount of time, by the end of the book, you should be able to write a program for it.
- Learn new ways of solving problems!

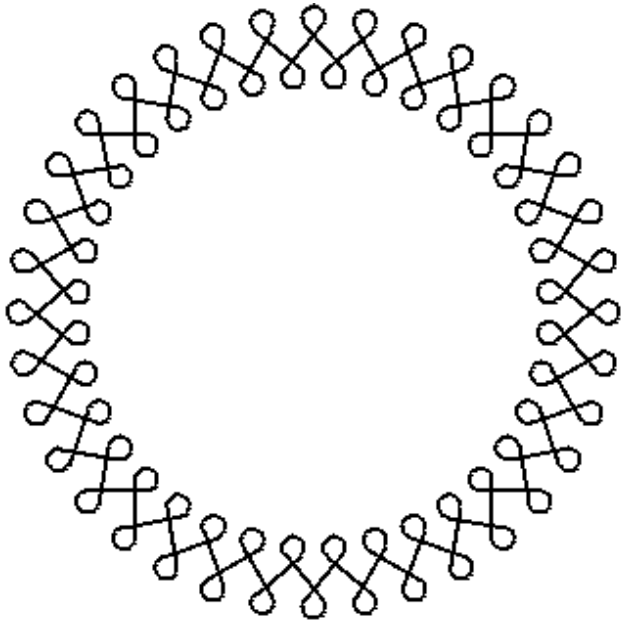
Spirit of the course 2

- 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.

Exercises

- Draw a 5 pointed star.
  - Hint: If the turtle starts at a certain point with a certain orientation and returns to the same point with the same orientation, it must have turned totally through a multiple of 360 degrees
- Draw a 7 pointed star. How many different 7 pointed stars can you have?
- Draw 7 identical circles, with 6 touching the central circle. Circle = polygon of large no of sides, say 360.
- Draw 4x4 array of tiles slightly separated from each other.

Hard Exercise: Plate border design



- You know enough to write a program to do this!
- Combine together ideas from previous exercises.