Introduction to Programming: Simplecpp

Ajit Rajwade

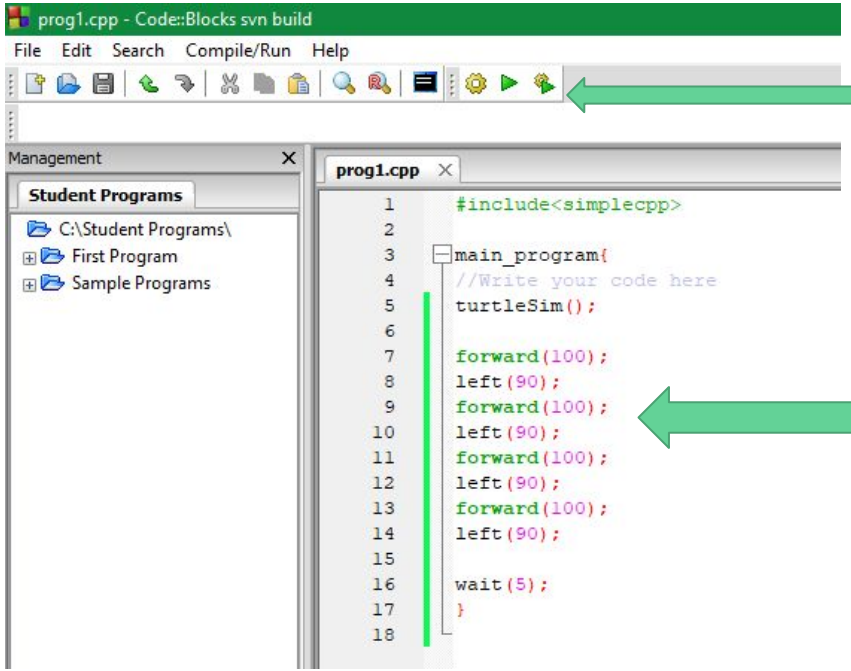
Refer: Chapter 2 of the book by Abhiram Ranade

Let’s dive headlong into programming

- We will work with a version of C++ called simplecpp
- It is associated with the book by Prof. Abhiram Ranade
- Excellent tool to give initial flavours of C++
- Allows for easy graphical programs
- Easier syntax
- Installed on CSE machines
- Simplecpp is for the initial 2-3 weeks of the course
- We will then switch to barebones C++

To install simplecpp on your machine

- See https://www.cse.iitb.ac.in/~ranade/simplecpp/ and follow instructions on it
- The package above includes a graphical interface called codeblocks which looks like this:



To compile and execute your program, click this button

Type your program here

Installation Instructions: Simplecpp

https://drive.google.com/drive/folders/1oSJ1nW2tNlxi18ecq0gqTNwC87xmHwKM?usp=share_link

For Windows, Linux, IOS

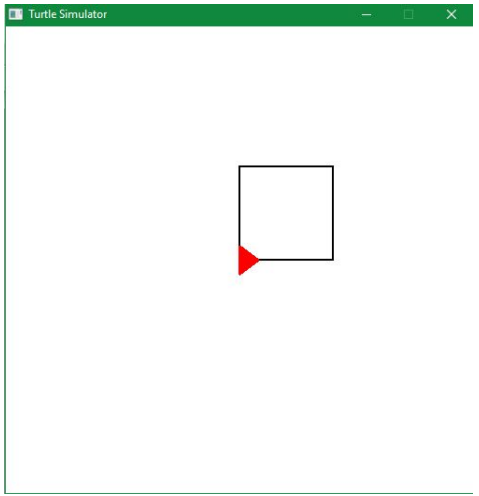
Your first program: to draw a square

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

Most commands end with a semicolon (;).

Compiling and executing

- A computer doesn't understand C++
- The C++ code needs to be translated into a form which a computer can understand.
- This is called **compilation** and is performed by a separate program called the **compiler**.
- The compilation happens when you press that key (see previous slide) - it invokes a command called `s++` which is part of simplecpp
- The `s++` command invokes the well-known GNU C++ compiler called `g++` .
- The compiler will by default produce a file called `a.out` which the computer can **understand and execute**.
- Upon execution, you see a new window opens with a square drawn in it.
- Hurray! This was your first C++ program :-)



Compiling and executing

- On a linux shell, you can execute the program via the command line: `s++ prog1.cpp`
- This produces the file `a.out`
- You execute this program on the command line as `./a.out`
- You can change the name of the executable file to something else you want by: `s++ prog1.cpp -o prog1.out`
- If you are using Windows, the graphical user interface does all this for you.

Understanding the program

```
#include <simplecpp>  // makes use of simplecpp package

main_program{  // tells us that what follows is our main program (there exist non-main programs too)

turtleSim();  // opens a window with a triangle (turtle) at the center pointing East

forward(100);  // moves the turtle forward by number of pixels (dots on the screen!) in parentheses

left(90);  // rotates the turtle to the left by 90 degrees

forward(100); left(90);

forward(100); left(90);

forward(100);

wait(5);  // program waits for 5 seconds (does nothing)

} // main_program ends here
```

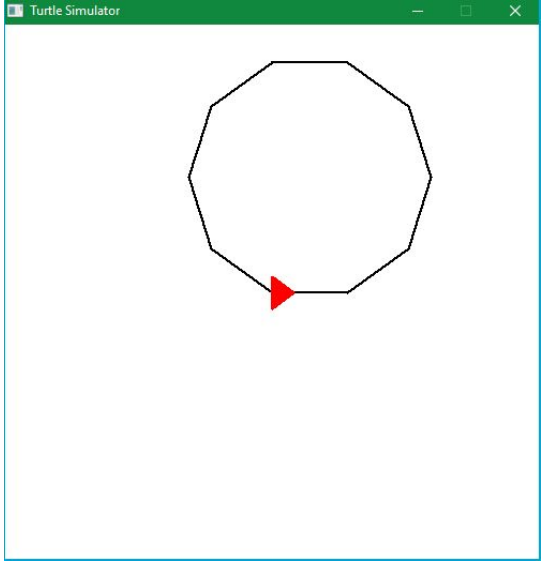
The stuff after // is called a comment. It is a remark written by a programmer to help others understand the purpose of that line. In large programs, it is useful as notes even for the programmer himself/herself.

Notice how English-like C++ is! :-)

Understanding the program more deeply

- We just drew a square.
- You used the forward command four times and the left command three times.
- What if you wanted to draw a regular decagon?
- The decagon has exterior 10 sides and 10 exterior angles
- The exterior angles of a polygon add up to 360 degrees, and hence each exterior angle is 36 degrees
- When you draw a side of the decagon, you need to rotate the turtle by 36 degrees before you draw the next side.
- To draw a decagon with side length 100, you must execute `forward(100); left(36);` ten times.
- Writing this out ten times is boring!

Understanding the program more deeply



- So we use a so-called **repeat loop** which looks like:

```
#include <simplecpp>

main_program{

turtleSim();

repeat(10){ // repeat tells the computer to

// repeat whatever is inside the loop that many times

forward(70); left(36);

}

wait(5);

}
```

Repeat is not part of traditional C++ but only part of simplecpp.

Understanding the program more deeply: drawing a regular n-gon

```
#include <simplecpp>

main_program{

int num_sides; // an integer variable to store the number of sides of

// the n-gon

turtleSim();

cout << "enter number of sides of the polygon";  // ask user to enter

// desired number via keyboard - print this on the screen

cin >> numsides; // get user input and store in "variable" called num_sides

repeat(num_sides){ // repeat 'num_sides' number of times

forward(50); left(360/num_sides);

}

wait(5);

}
```

What is a variable?

- It is a small region of the computer's **memory**
- It stores a certain desired **value**. The value may or may not change throughout the program.
- A program needs to **declare** a variable before it is used.
- A program can declare **as many** variables as desired.
- Each variable is of one of many **types**: here it was an integer.
- Each variable has a **name** decided by the programmer - meaningful, descriptive names are desired.
- Try modifying the earlier program by also taking the length of the side of the polygon as input from the user.

The answer...

```
#include <simplecpp>
main_program{
int num_sides; // an integer variable to store the number of sides of  the n-gon
int side_length;
turtleSim();
cout << "enter number of sides of the polygon";  // ask user to enter desired number via keyboard
cin >> numsides; // and store in "variable" called num_sides
cout << "Enter the side length:";
cin >> side_length;
repeat(num_sides){  // repeat 'num_sides' number of times
forward(side_length); left(360/num_sides);
}
wait(5);
}
```

Nested `repeat` loops

- What if you wanted to draw an `n`-gon multiple (say `k`) times, maybe with a different `n` each time?
- You need to write a `repeat` loop which will run `k` times.
- Inside each loop, you will have a `repeat` loop for drawing the n-gon.
- This is called as **nesting** of `repeat` loops.
- If required in a program, you can increase the level of nesting - `repeat` inside a `repeat`, inside a `repeat`, inside a `repeat` ...
- The full program is on the next slide.

Nested `repeat` loops

```
#include <simplecpp>
main_program{
int num_sides; // an integer variable to store the number of sides of  the n-gon
int side_length;
turtleSim();
repeat (5) {
     cout << "enter number of sides of the polygon"; // ask user to enter number via
//keyboard
     cin >> num_sides; // and store in "variable" called num_sides
     cout << "Enter the side length:";
     cin >> side_length;
     repeat(num_sides){ // repeat 'num_sides' number of times
     forward(side_length); left(360/num_sides);
     wait(5);
     } // inner repeat loop ends here
  } // outer repeat loop ends here
}
```

Useful turtle commands: a few more

- `right (angle)`: cause the turtle to turn right by `angle` degrees.
- `penUp()`: raises the pen. If the turtle moves forward when the pen is raised, no line will be drawn.
- `penDown()`: lowers the pen. If the turtle moves forward when the pen is down, a line will be drawn.
- The default state of the pen is **down** when `turtleSim();` is invoked.
- Can you write a program which shows the usage of `penUp()`?

Some tips for programming

- Programming is an art - it requires practice
- Execute the programs from these slides on your own.
- Try asking yourself questions - what if I do this? Then implement those changes and see what happens for yourself.
- For example in the `n`-gon program: suppose I wanted to draw an `m`-gon of much smaller size at each vertex, how would I do it?

Basic Computer Organization

Ajit Rajwade

Parts of a computer

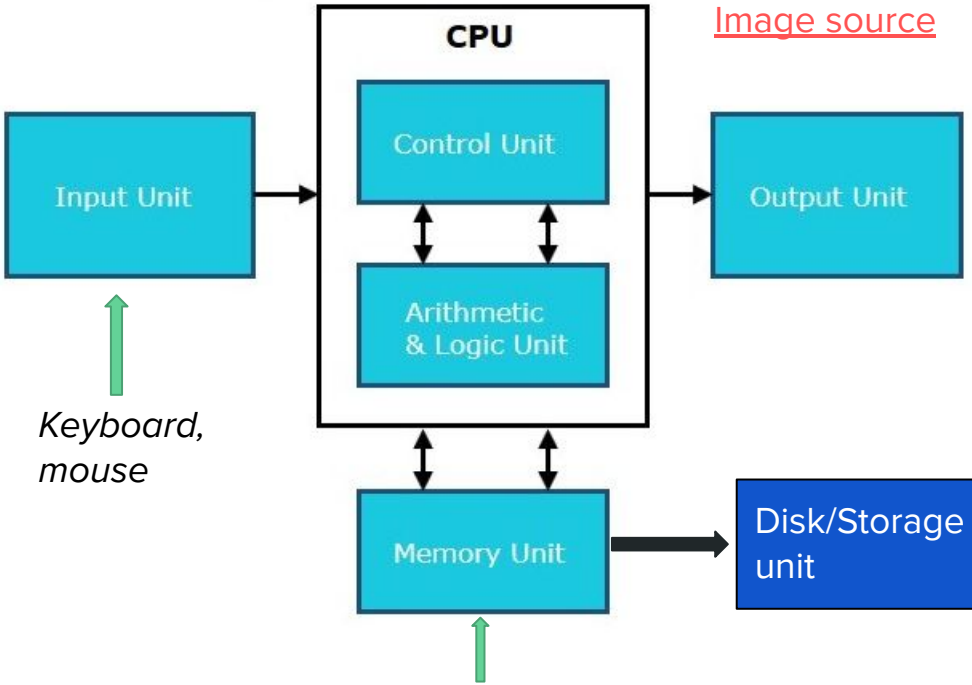


Image source

The job of the computer program is to instruct the CPU what to do

Monitor for display

Keyboard, mouse

This is where the files on your computer are stored, including the .cpp and .out files we saw

This is where the program, and variables you declare in your program, are stored. In fact each variable has an address - an indicator of the storage location in the memory