1 Studying

Time – Uninterrupted!

Learn

- Watch: lecture / video(s)
- Read: book(s), web, notes
- Teach: Explain to others

Programming

- Write programs
- Write test programs
- Write programs unaided (no notes, web, etc.)

1.1 Some Key Questions

- How do I?
- How do I fix this?
- How does?
- Why?
- What happens if?
- Example of?

2 Common Program Operations

- Add / Insertion
- Delete / Removal
- Traversal / Iteration
- Copy
- Filtering
- Text manipulation (copy / extraction)

3 Program Construction

Building a program involves three steps:

- Editing
- Compiling
- Run

Some would add debugging, but that is another problem.

4 Simple C++ Program

```
/* hello.cpp
 */

#include <iostream>

using namespace std;

int main()
{
   cout << "Hello!" << endl;

   return 0;
}
```

To run this program:

1. `edit hello.cpp # edit (use nano / vi / emacs)`
2. `g++ hello.cpp # Compile`
3. `./a.out # Run`

Find a reference to the editor of your choice (emacs and vi are very popular – not easy).

5 Code

Composed of statements.

- Comments
- `#include`
- `using`

5.1 Statements

- Assignment
- Mathematical
- Function call

5.2 Syntax

- Semi-colons
- Curly braces
- Quotes
- Parentheses ('(', ')', '[', ']')

5.3 Variables

- Naming – case sensitivity(?)
- Declare
- Initialize
- Assign (different than initialize?)
- Scope (local, global, block)
- Constants

5.4 Types

- Basic / Primitive (language)
- User- / System- defined
- `enum`
- `typedef`

5.5 Simple Output

Language dependent. (`cout` in C++)

5.6 Operators

- Simple operators (`+`, `-`, `*`, `/`, `%`)
- `+=`, `++`, `--`
- Logical: `!`, `&&`, `||`, `^`
- Precedence

5.7 Decisions

- `if`
- `switch`
- Ternary operator: `?  :`

5.8 Repetition (aka loops)

- `for`
- `while`
- `do { } while ( );`

5.9 Functions / Procedures / Subroutines

- Naming
- `main`
- Prototypes
- `return` statements (types)
- Parameter passing
  (by value, by reference, by pointer (address))

5.10 I/O (Input / Output)

- Read values
- Write values
- File I/O

5.11 Basic data structures

- Arrays
- Strings (language dependent: arrays, pointers, objects)
- structs
- Function parameters
- Memory – static / dynamic

5.12 Intermediate (to Advanced)

- Interface / Definition (functions, classes, files)
- Lists
- Objects
- Classes
- Methods
- Data structures

5.13 Advanced

- Objects
- Modules / Packages / Files
- Templates
- Meta-templates