From C to C++

Hsuan-Tien Lin

Dept. of CSIE, NTU

February 21, 2013

Memory Refreshment: C Language

```
#include <stdio.h>

int main(){
  printf("Hello␣World"); /* comment */
  return 0;
}
```

For Comparison: C++ Language

```c
#include <stdio.h>

int main(){
  printf("Hello World"); /* comment */
  return 0;
}
```

```cpp
#include <iostream>

int main(){
  std::cout << "Hello World"; // comment
  return 0;
}
```

Types (Section 1.1.2)

- C types: `char`, `short`, `int`, `long`, `float`, `double` (and `enum`, `void`)
- C++ types: all above + `bool`

true, false

Pointers, Arrays, Strings, Structures (Section 1.1.3)

same, except

- string can be implemented by
  - character array, like C
  - extended type

```
#include <iostream>
#include <string>

int main(){
  std::string s = "Hello␣";
  std::string t = "World";

  std::cout << (s + t); //comment
  return 0;
}
```

- dynamic memory can be implemented by
  - `malloc` and `free`, like C
  - `new` and `delete` (read Section 1.1.3 by yourself)
  - **Warning: do not mix the two**

Named Constants, Scope, Namespace (Section 1.1.4)

- `const`: C++ style constants
- local scope: can declare variable with lifecycle within each {}
- namespace: can gather variables, functions, ... within a {}, accessed with `::`

```
        std::string s = "Hello␣";
```

Expressions and Control Flow (Sections 1.2, 1.3)

SAME, except casting (read on your own)

Functions (Section 1.4)

mostly same, except (read on your own if we cannot tell you every detail)

- some more sophisticated argument passing possible
- function overloading
- operator overloading
- inline

Classes (Section 1.5)

roughly, an extended structure that allows you to define functions along with the variables

```
class pos_rationale{
public:
  unsigned int num;
  unsigned int denom;

  int floor(){ return num / denom; }
}

pos_rationale r;
r.num = 5; r.denom = 3;
cout << r.floor();
```

learn more in your HW1