Prototypes

Kevin Schmidt, Susan Lindsey, Charlie Dey

Spring 2019

Prototypes and forward declarations

A first use of prototypes is *forward declaration*:

```
int f(int);
int g(int i) { return f(i); }
int f(int i) { return g(i); }
```

Prototypes for separate compilation

```
// file: def.cxx
int tester(float x) {
  .....
}
```

```
// file : main.cxx
int tester(float);

int main() {
  int t = tester(...);
  return 0;
}
```

Compiling and linking

Your regular compile line

```
icpc -o yourprogram yourfile.cc
```

actually does two things: compilation, and linking. You can do those separately:

1. First you compile

   ```
   icpc -c yourfile.cc
   ```

   which gives you a file `yourfile.o`, a so-called *object file*; and
2. Then you use the compiler as *linker* to give you the *executable file*:

   ```
   icpc -o yourprogram yourfile.o
   ```

Dealing with multiple files

Compile each file separately, then link:

```
icpc -c mainfile.cc
icpc -c functionfile.cc
icpc -o yourprogram mainfile.o functionfile.o
```

Prototypes and header files

```
// file: def.h
int tester(float);
```

The header file gets included both in the definitions file and the main program:

```
// file: def.cxx
#include "def.h"
int tester(float x) {
  .....
}
```

```
// file : main.cxx
#include "def.h"

int main() {
  int t = tester(...);
  return 0;
}
```

Class prototypes

Header file:

```
class something {
public:
  double somedo(vector);
};
```

Implementation file:

```
double something::somedo(vector v) {
   .... something with v ....
};
```

Strangely, data members also go in the header file.

Review quiz 1

For each of the following answer: is this a valid function definition or function prototype. Are any of them a constructor?

- `int foo();`
- `int foo() {};`
- `int foo(int) {};`
- `int foo(int bar) {};`
- `int foo(int) { return 0; };`
- `int foo(int bar) { return 0; };`
- `foo();`
- `foo() {};`