FUNCTION DEFINITIONS AND DECLARATIONS

Creating and Describing Functions

Delroy A. Brinkerhoff
• Declaration stores information about a function in the symbol table
  • name of variable or function
  • number and type of parameters
  • type of variable or return value type of function
• Definition uses memory
  • contents of variable
  • store machine instructions generated from function
  • the function’s location in memory is added to the symbol table
### Relationship Between Definitions and Declarations

<table>
<thead>
<tr>
<th>Definition</th>
<th>Declaration / Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>double foo(int x, double y, char z)</code></td>
<td><code>double foo(int x, double y, char z)</code></td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td><code>double foo(int a, double b, char c)</code></td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td><code>double foo(int, double, char)</code></td>
</tr>
</tbody>
</table>

Some code can serve as both a declaration and a definition.
Function prototypes have three components
- Name
- Return value type
- Parameter list

Prototypes permit the compiler to
- Verify that calls are correct (number and type of arguments)
- Perform appropriate conversion on arguments and return values
- C++ requires a declaration or prototype to compile
```c
int main()
{
    double y;
    y = sqr(2);
}

double sqr(double x)
{
    return x * x;
}
```
double sqr(double x)
{
    return x * x;
}

int main()
{
    double y;
    y = sqr(2);
}
double sqr(double x);

int main()
{
    double y;
    y = sqr(2);
}

double sqr(double x)
{
    return x * x;
}
WHY PROTOTYPES (I)?

file1.cpp

```cpp
struct G {
};

int f(G x) {
    ...
}
```

file2.cpp

```cpp
G a = {
};

int b = f(a);
```
WHY PROTOTYPES (2)?

```c
void a()
{
    . . . .
    b();
    . . . .
}

void b()
{
    . . . .
    a();
    . . . .
}
```
FUNCTIONS AND TYPES

- Function definition
  - Has typing information
  - Has a body
- Function prototype (declaration)
  - Has typing information
  - No body; ends with a semicolon
- Function call
  - Does NOT have typing information

- Definition
  - int max(int x, int y) { return (x > y) ? x : y; }
- Prototype (declaration)
  - int max(int x, int y);
  - int max(int, int);
- Call
  - max(10, 20);
  - max(a, b);