

Function Overloading

Lecture 20
Section 6.14

Robb T. Koether

Hampden-Sydney College

Fri, Oct 12, 2018

- 1 Function Overloading
- 2 Examples
- 3 Ambiguous Function Calls

Outline

- 1 Function Overloading
- 2 Examples
- 3 Ambiguous Function Calls

Function Overloading

- A function name is **overloaded** if the same name is used for two or more distinct functions.
- Function overloading allows the use of the same name for functions with a similar purpose.

Function Overloading

- Consider the square-root function `sqrt()`.
- Without function overloading, we would have to make up a distinct name for the square root function for each different data type
 - **float** `sqrti(int);`
 - **float** `sqrtf(float);`
 - **double** `sqrtd(double);`

The Signature of a Function

- A function's **signature** is the set of characteristics that can be determined from the actual parameter list.
 - The number of parameters.
 - The types of the parameters.
 - The order of the parameters.

The Signature of a Function

- A function's signature does not include those characteristics that cannot be determined from the actual parameter list.
 - The names of the formal parameters.
 - Whether the parameters are reference parameters.
 - Whether the parameters are constant parameters.
 - The function's return type.

Function Overloading

- The compiler uses the actual parameter list to determine which **instance** of an overloaded function to use.
- Therefore, each instance of an overloaded function must have a distinct signature.
- Otherwise, the compiler could not determine which one to use.

Outline

- 1 Function Overloading
- 2 Examples**
- 3 Ambiguous Function Calls

Examples of Legal Overloading

Legal Overloading

```
int max(int x, int y);  
float max(float x, float y);  
float max(float x, int y);  
float max(int x, float y);
```

- It is legal to define simultaneously all of the above functions.

Example of an Overloaded Function

- Example
 - `OverloadedMax.cpp`

Examples of Illegal Overloading

Illegal Overloading

```
int max(int x, int y);  
int max(int x, int& y);  
int& max(int x, int& y);  
int max(const int x, int y);  
const int max(int x, int y);  
float max(int x, int y);
```

- It is illegal to define simultaneously any two of the above functions.

Outline

- 1 Function Overloading
- 2 Examples
- 3 Ambiguous Function Calls**

Ambiguous Function Calls

An Ambiguous Function Call

```
int max(int x, int y);  
float max(float x, float y);  
:  
float m = max(4, 5.6);
```

- The function call in the example below is ambiguous.
- Which instance of `max()` should be invoked?

Type-Casting Parameters

- It is not always necessary that the type of the actual parameter match exactly the type of the formal parameter.
- The compiler will type-cast the parameter, if possible.
- The following will occur automatically.
 - Promotion among the integer types.
 - Promotion among the floating-point types.
 - Conversions between integer and floating-point types.

Type-Casting Parameters

Implicit Type-Casting

```
float max(float x, float y);  
double max(double x, double y);  
:  
float m1 = max(1.2, 3.4);  
int m2 = max(5, 6);
```

- Which function is called?

Type-Casting Parameters

- Implicit type-casting is automatic whenever there is an unambiguous rule, based on constructors.

Explicit Type-Casting

- Explicit type-casting may be used to force a type change.
- There must be a conversion rule available, based on constructors.
- To explicitly type-cast an object, write the name of the new type in parentheses before the object.

Explicit Type-Casting

Explicit Type-Casting

```
int max(int, int);  
float max(float, float);  
int a;  
float b;  
:  
float m1 = max((float)a, b);  
int m2 = max(a, (int)b);
```

- Explicit type-casting may resolve ambiguous references to overloaded functions.

Assignment

Assignment

- Read Section 6.14.