Overload Resolution

When you overload a function:

- The parameter **number** must differ, or
- The parameter **types** must differ, or
- The parameter type **order** must differ

You **cannot** merely change the return type of a function. That is an error.

To determine which function is called, your compiler follows a process called **overload resolution**. Resolving which version of the `abs()` function to call is easy, since it only takes one argument. Things are more complex when a function takes several arguments.

Here are the rules:

1. Functions **with the same name** are gathered into a **candidate set**.
2. The candidate set is narrowed to produce the **viable set**: those functions that have **the correct number of parameters** and whose parameters **could** accept the supplied arguments using standard conversions.
3. If there are any **exact matches** in the viable set, use that version.
4. If there are no exact matches, find the **best match** involving conversions. The rules for this can be quite complex. You can find all of the details in the C++ Primer, Section 6.6.

There are **two possible errors** that can occur **at the end** of the matching process:

- There are **no members** left in the viable set. This produces an **undeclared name** compiler error.
- The process **can't pick a winner** among several viable functions. This produces an **ambiguity** compiler error.

When this occurs, **the function definition is not in error**, but **the function call**.