```
void print_square(double x)
{
    cout << "the square of " << x << " is " << square(x) << "\n";
}

int main()
{
    print_square(1.234);    // print: the square of 1.234 is 1.52276
}
```

A "return type" **void** indicates that a function does not return a value.

2.2.2 Types, Variables, and Arithmetic

Every name and every expression has a type that determines the operations that may be performed on it. For example, the declaration

```
int inch;
```

specifies that **inch** is of type **int**; that is, **inch** is an integer variable.

A *declaration* is a statement that introduces a name into the program. It specifies a type for the named entity:

- A *type* defines a set of possible values and a set of operations (for an object).
- An *object* is some memory that holds a value of some type.
- A *value* is a set of bits interpreted according to a type.
- A *variable* is a named object.

C++ offers a variety of fundamental types. For example:

```
bool      // Boolean, possible values are true and false
char      // character, for example, 'a', 'z', and '9'
int       // integer, for example, -213, 42, and 1066
double    // double-precision floating-point number, for example, 3.14 and 299793.0
```

```
x+y       // plus
+x        // unary plus
x-y       // minus
-x        // unary minus
x*y       // multiply
x/y       // divide
x%y       // remainder (modulus) for integers
```

So can the comparison operators:

```
x==y      // equal
x!=y      // not equal
x<y       // less than
x>y       // greater than
x<=y      // less than or equal
x>=y      // greater than or equal
```

In assignments and in arithmetic operations, C++ performs all meaningful conversions (§10.5.3) between the basic types so that they can be mixed freely:

```
void some_function()    // function that doesn't return a value
{
    double d = 2.2;     // initialize floating-point number
    int i = 7;          // initialize integer
    d = d+i;            // assign sum to d
    i = d*i;            // assign product to i (truncating the double d*i to an int)
}
```