Chapter 1: The Basics

1.2 Programs

- c++ is a compiled language
- c++ is statically typed. Every entity (object, value, expression, etc) must be known to the compiler at its point of use

int main()

This is the minimal c++ program. It defines a function called *main*, which takes no arguments and does nothing

- {} curly braces express grouping in c++
- // begins a single line comment
- every c++ program must have *exactly one* function named *main()*, the program starts by executing *main()*
- *int* value returned by *main()* (if any) is the program's return value to the "system". Non-zero value indicates failure

Hello World!

```
#include <iostream>

int main()
{
    // std:: specifies that cout is found in the standard library namepsace
    std::cout << "Hello, World!\n";
}
```

1.3 Functions

A function declaration gives the *name* of the function, the *type* of return value, and the *number* and *types* of the arguments that must be supplied in a call.

```
double sqrt(double); //double argument, returns a double
```

Function Overloading

Defining multiple functions with the same name is called *function overloading*, and it is essential to generic programming. Each function of the same name should implement the same semantics.

1.4 Types, Variables, and Arithmetic

Every name and expression has a type that determines the operations that may be performed on it. A *declaration* is a statement that introduces an entity into the program and specifies a type for the entity.

- *type* defined a set of possible values and a set of operations (for an object)

- *object* is some memory that holds a value of some type
- *value* is a set of bits interpreted according to a type
- *variable* is a named object

Some Basic Types

`auto`: variable type will be deducted from its

Type	Description
bool	*Boolean*, possible values are *true* and *false*
char	*character*, eg. 'z', 'A', '9'
int	*integer*, eg. -273, 43
double	*double-precision floating-point number*, eg. -126.123, 3.14, 6.626e-34
unsigned	*non-negative integer*, eg. 0, 1, 999

1.4.1 Arithmetic

Arithmetic operators can be used for appropriate combinations of the fundamental types:

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 comparison operators:

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

Logical operators are provided:

x & y	bitwise and
x | y	bitwise or
x ^ y	bitwise exclusive or
x && y	logical and
x || y	logical or
!x	logical not (negation)

1.6 Constants

`const`: used primarily to specify interfaces so data can be passed to functions using pointers and references without fear of it being modified. The value of a *const* can be calculated at run time.

`constexpr`: allow placement of data in read-only memory, where it is unlikely to be corrupted and for performance.

The value of a *constexpr* must be calculated by the compiler

1.7 Pointers, Arrays, and References

`*`: means “contents of”
`&`: means “address of”

1.8 Tests

control flow: `if, while, switch, for`

- like `for`, `if` can introduce a variable and test it: `if(x = v.size(); x != 0)`