2.4 CHARACTER TYPES

A *character type* is an integral type whose variables represent characters like the letter `'A'` or the digit `'8'`. Character literals are delimited by the apostrophe (`'`). Like all integral type values, character values are stored as integers.

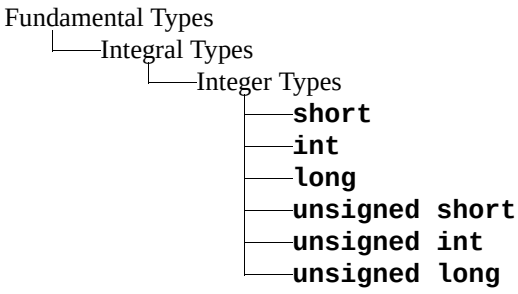
EXAMPLE 2.2 Character Variables

```
int main()
{ // prints the character and its internally stored integer value:
  char c='A';
  cout << "c = " << c << ", int(c) = " << int(c) << endl;
  c='t';
  cout << "c = " << c << ", int(c) = " << int(c) << endl;
  c='\t';  // the tab character
  cout << "c = " << c << ", int(c) = " << int(c) << endl;
  c='!';
  cout << "c = " << c << ", int(c) = " << int(c) << endl;
}
```

```
c = A, int(c) = 65
c = t, int(c) = 116
c =     , int(c) = 9
c = !, int(c) = 33
```

Since character values are used for input and output, they appear in their character form instead of their integral form: the character `'A'` is printed as the letter "A", not as the integer 65 which is its internal representation. The *type cast operator* `int()` is used here to reveal the corresponding integral value. These are the characters' ASCII codes. (See Appendix A.)

2.5 INTEGER TYPES



There are 6 integer types in Standard C++: These types actually have several names. For example, **`short`** is also named **`short int`**, and **`int`** is also named **`signed int`**.

You can determine the numerical ranges of the integer types on your system by running the program in the following example.

EXAMPLE 2.3 Integer Type Ranges

This program prints the numeric ranges of the 6 integer types in C++:

```
#include <iostream>
#include <climits>    // defines the constants SHRT_MIN, etc.
using namespace std;
int main()
{ // prints some of the constants stored in the <climits> header:
  cout << "minimum short = " << SHRT_MIN << endl;
  cout << "maximum short = " << SHRT_MAX << endl;
```

```
    cout << "maximum unsigned short = 0" << endl;
    cout << "maximum unsigned short = " << USHRT_MAX << endl;
    cout << "minimum int = " << INT_MIN << endl;
    cout << "maximum int = " << INT_MAX << endl;
    cout << "minimum unsigned int = 0" << endl;
    cout << "maximum unsigned int = " << UINT_MAX << endl;
    cout << "minimum long= " << LONG_MIN << endl;
    cout << "maximum long= " << LONG_MAX << endl;
    cout << "minimum unsigned long = 0" << endl;
    cout << "maximum unsigned long = " << ULONG_MAX << endl;
}
```

```
minimum short = -32768
maximum short = 32767
maximum unsigned short = 0
maximum unsigned short = 65535
minimum int = -2147483648
maximum int = 2147483647
minimum unsigned int= 0
maximum unsigned int= 4294967295
minimum long = -2147483648
maximum long = 2147483647
minimum unsigned long = 0
maximum unsigned long = 4294967295
```

The header file `<climits>` defines the constants `SHRT_MIN`, `SHRT_MAX`, `USHRT_MIN`, *etc*. These are the limits on the range of values that a variable of the indicated type can have. For example, the output shows that variables of type **`int`** can have values in the range –2,147,483,648 to 2,147,483,647 on this computer.

On this computer, the three **`signed`** integer types have the same range as their corresponding unqualified integer type. For example, **`signed short int`** is the same as **`short int`**. This tells us that the **`signed`** integer types are redundant on this computer.

The output also reveals that the range of the **`int`** type (–2,147,483,648 to 2,147,483,647) is the same as that of the **`long int`** type, and that the range of the **`unsigned int`** type (0 to 4,294,967,295) is the same as that of the **`unsigned long int`** type. This tells us that the **`long`** integer types are redundant on this computer.

The output from Example 2.3 shows that on this computer (a Pentium II PC running the Windows 98 operating system and the CodeWarrior 3.2 C++ compiler), the six integer types have the following ranges:

`short`:	–32,768 to 32,767;	(2^8 values ⇒ 1 byte)
`int`:	–2,147,483,648 to 2,147,483,647;	(2^{32} values ⇒ 4 bytes)
`long`:	–2,147,483,648 to 2,147,483,647;	(2^{32} values ⇒ 4 bytes)
`unsigned short`:	0 to 65,535;	(2^8 values ⇒ 1 byte)
`unsigned int`:	0 to 4,294,967,295;	(2^{32} values ⇒ 4 bytes)
`unsigned long`:	0 to 4,294,967,295;	(2^{32} values ⇒ 4 bytes)

Note that **`long`** is the same as **`int`** and **`unsigned long`** is the same as **`unsigned int`**.

The **`unsigned`** integer types are used for bit strings. A *bit string* is a string of 0s and 1s as is stored in the computer's random access memory (RAM) or on disk. Of course, everything stored in a computer, in RAM or on disk, is stored as 0s and 1s. But all other types of data are formatted; *i.e.*, interpreted as something such as a signed integer or a string of characters.