Chapter 2

Introduction to the C++ Language

Brooks/Cole
Thomson Learning™

Background

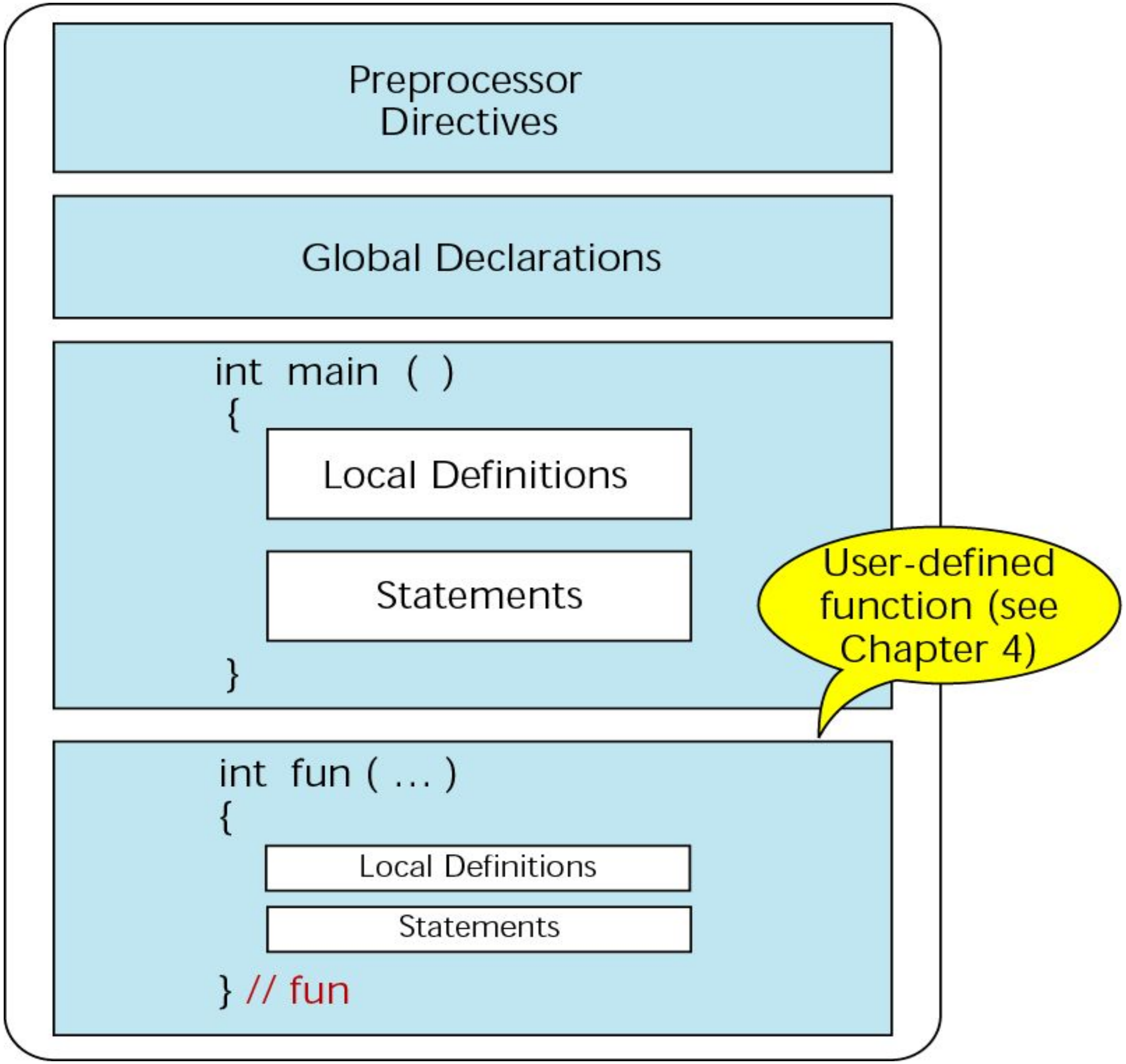
Brooks/Cole
Thomson Learning™

2.2

C++ Programs

Brooks/Cole
Thomson Learning™

Figure 2-2 Structure of a C++ program



Preprocessor Directives

Global Declarations

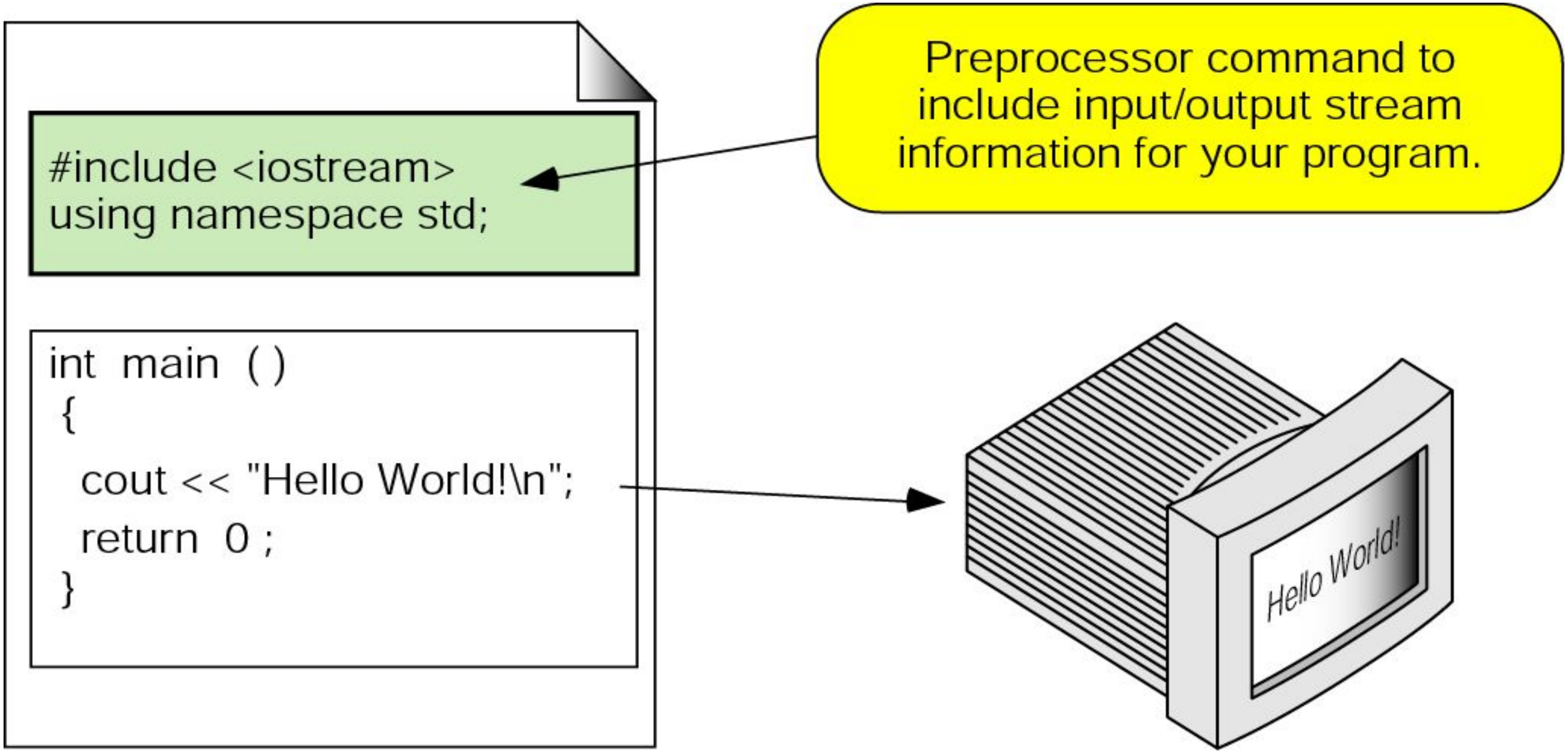
```
int  main  ( )
{
        Local Definitions

        Statements
}
```

User-defined function (see Chapter 4)

```
int  fun ( … )
{
        Local Definitions
        Statements
} // fun
```

Figure 2-3 The greeting program



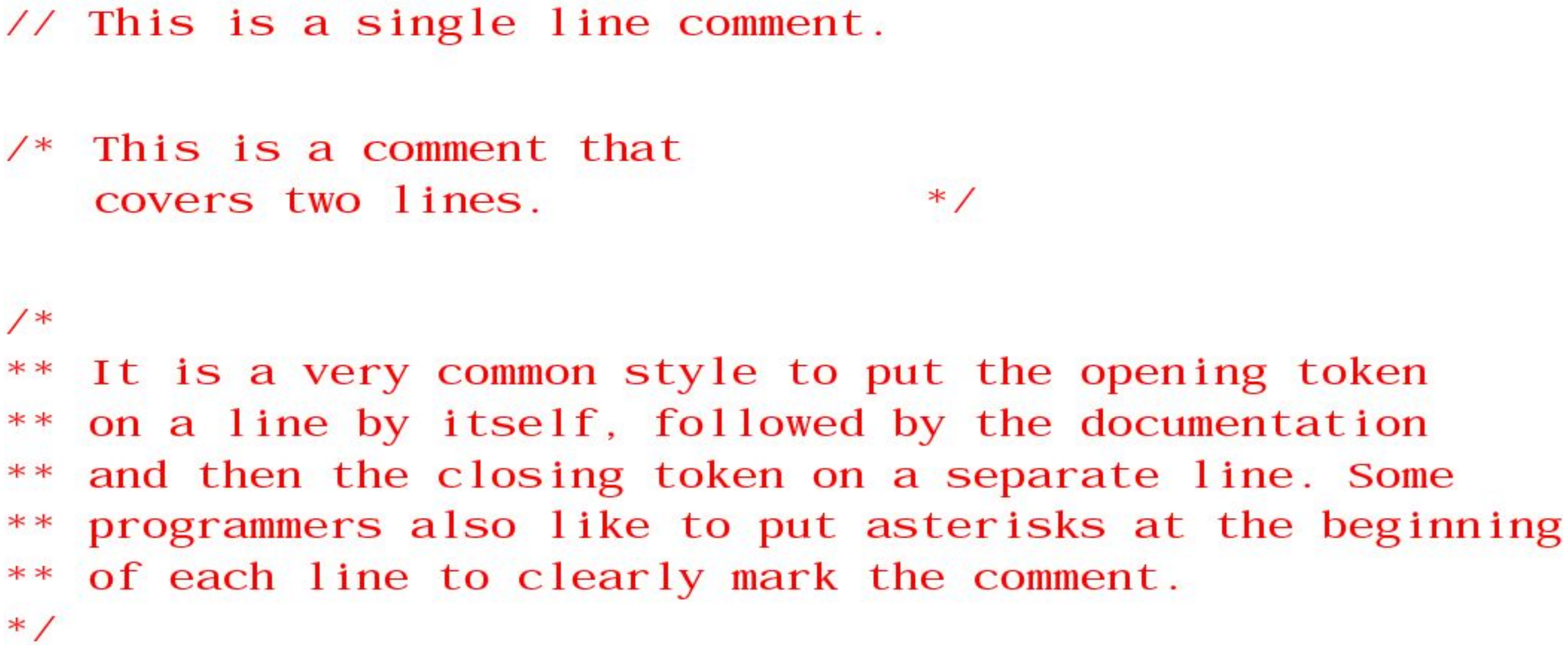
```
#include <iostream>
using namespace std;
```

```
int  main  ( )
 {
   cout << "Hello World!\n";
   return  0 ;
 }
```

Preprocessor command to include input/output stream information for your program.

Brooks/Cole
Thomson Learning™

Figure 2-4 Examples of comments

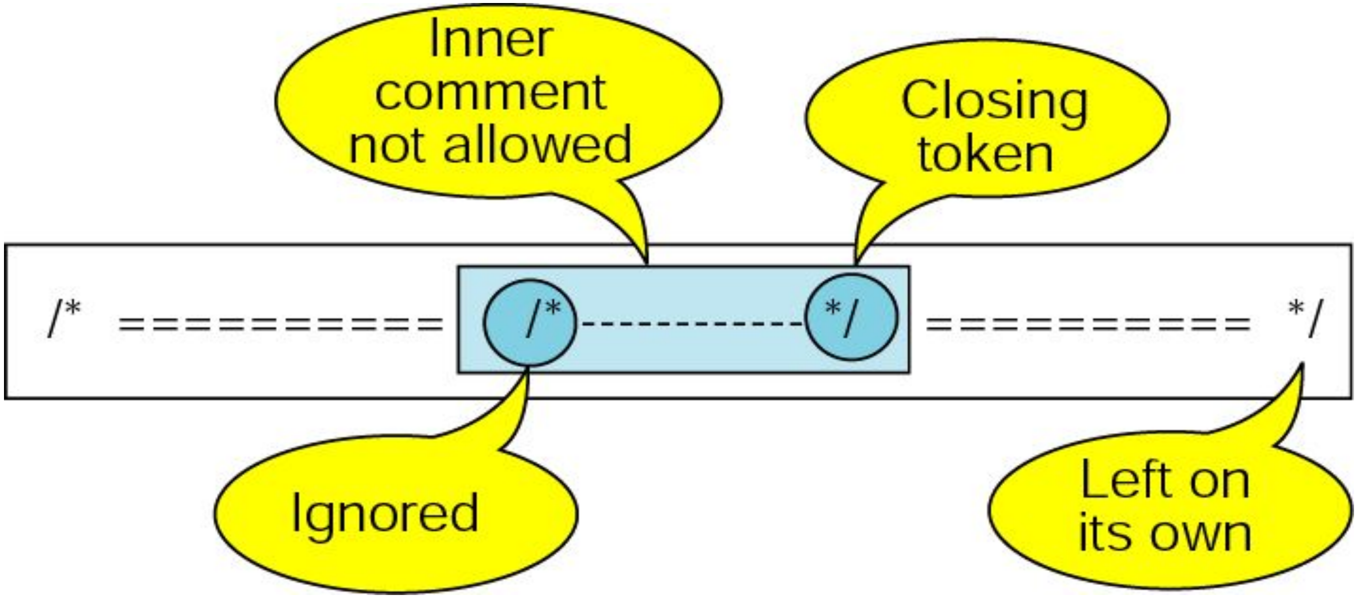


```
// This is a single line comment.

/* This is a comment that
   covers two lines.                */

/*
** It is a very common style to put the opening token
** on a line by itself, followed by the documentation
** and then the closing token on a separate line. Some
** programmers also like to put asterisks at the beginning
** of each line to clearly mark the comment.
*/
```

Figure 2-5 Nested block comments are invalid



Brooks/Cole
Thomson Learning™

2.3

Identifiers

- They allow us to name data and other objects in the program.

Brooks/Cole
Thomson Learning™

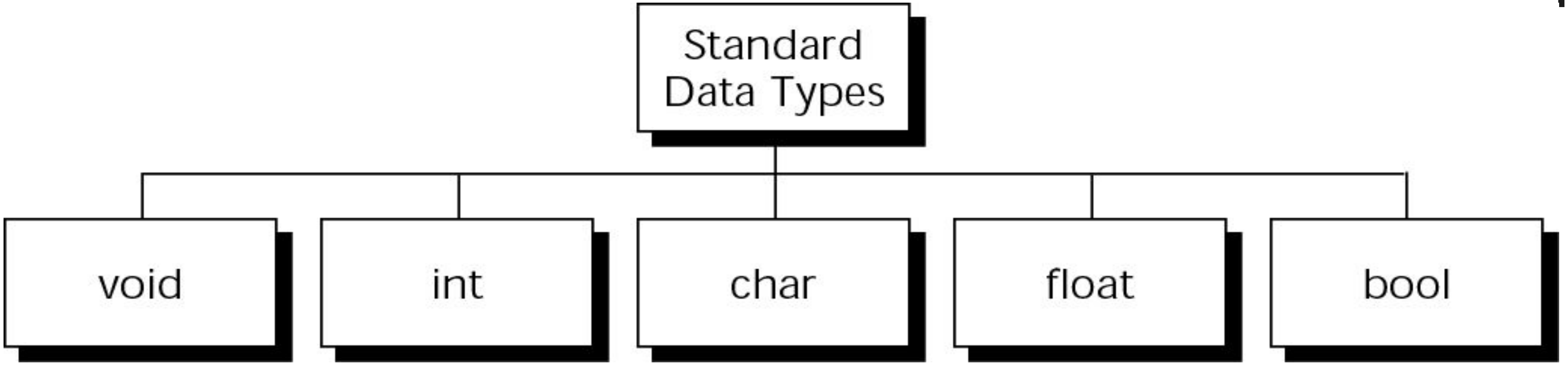
Rules for giving names to identifiers

- Alphabet a to z , A to Z
- Digits 0 to 9
- No space
- No sp ch other than _(under score)
- Cannot start with digit
- Is case sensitive
- Cannot be a key word

Data Types

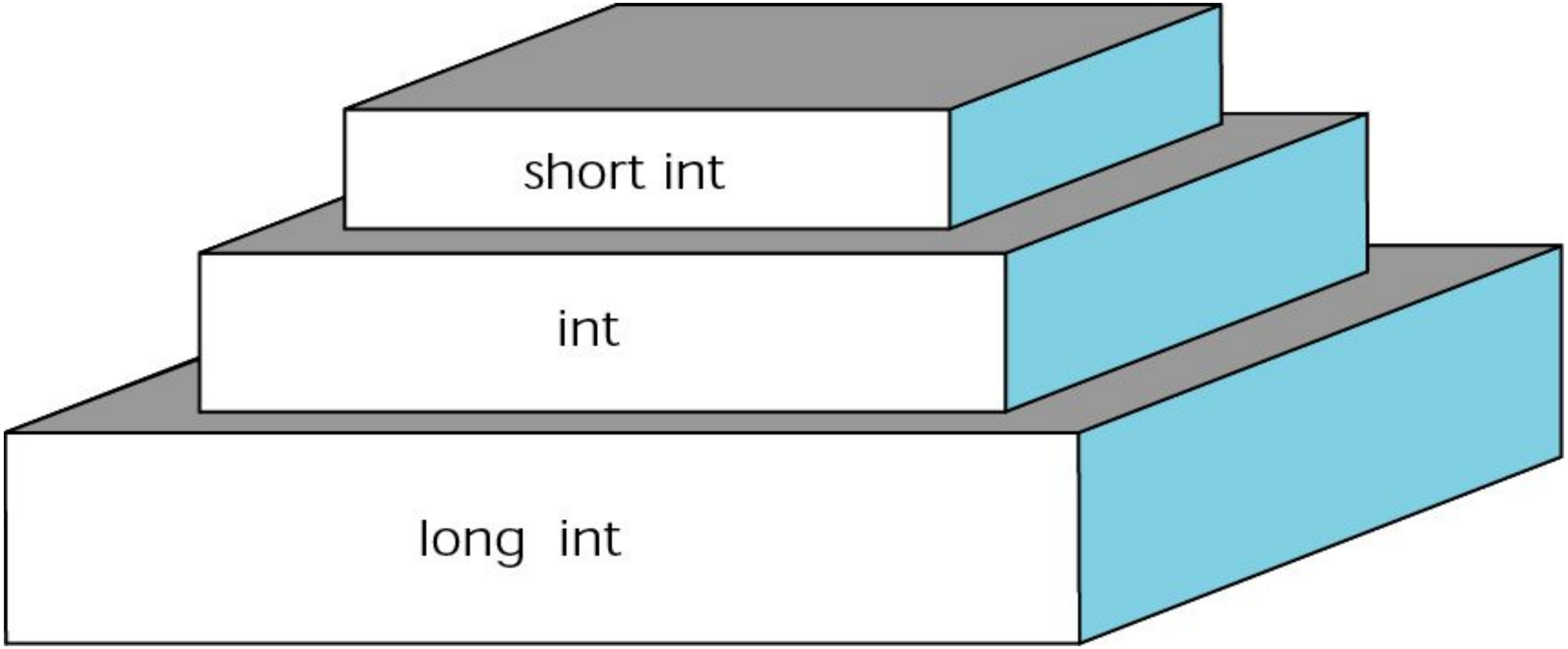
- It defines a set of op that can be applied on those values.
- Set of values for each type is known as the domain of that type

Figure 2-6 Standard data types



- Derived types-Pointers, arrays,structures

Figure 2-7 Integer types



Most microcomputers, minicomputers, and mainframes use the integer sizes shown in Table 2-5.

Type	Sign	Byte size	Num of bits	Minimum value	Maximum Value
short int	signed	2	16	-32,768	32,767
	unsigned			0	65,535
int (PC)	signed	2	16	-32,768	32,767
	unsigned			0	65,535
int (mainframe)	signed	4	32	-2,147,483,648	2,147,483,647
	unsigned			0	4,294,967,295
long int	signed	4	32[a]	-2,147,483,648	2,147,483,647
	unsigned			0	4,294,967,295

[a] Some computers use 48, 64, or more bits.

Table 2-5 Typical integer sizes

C++ provides an operator, *sizeof*, that will tell you the exact size of any data type. We will discuss this operator in detail in Chapter 3. Although size is machine dependent, C++ requires that the following relationship always be true:

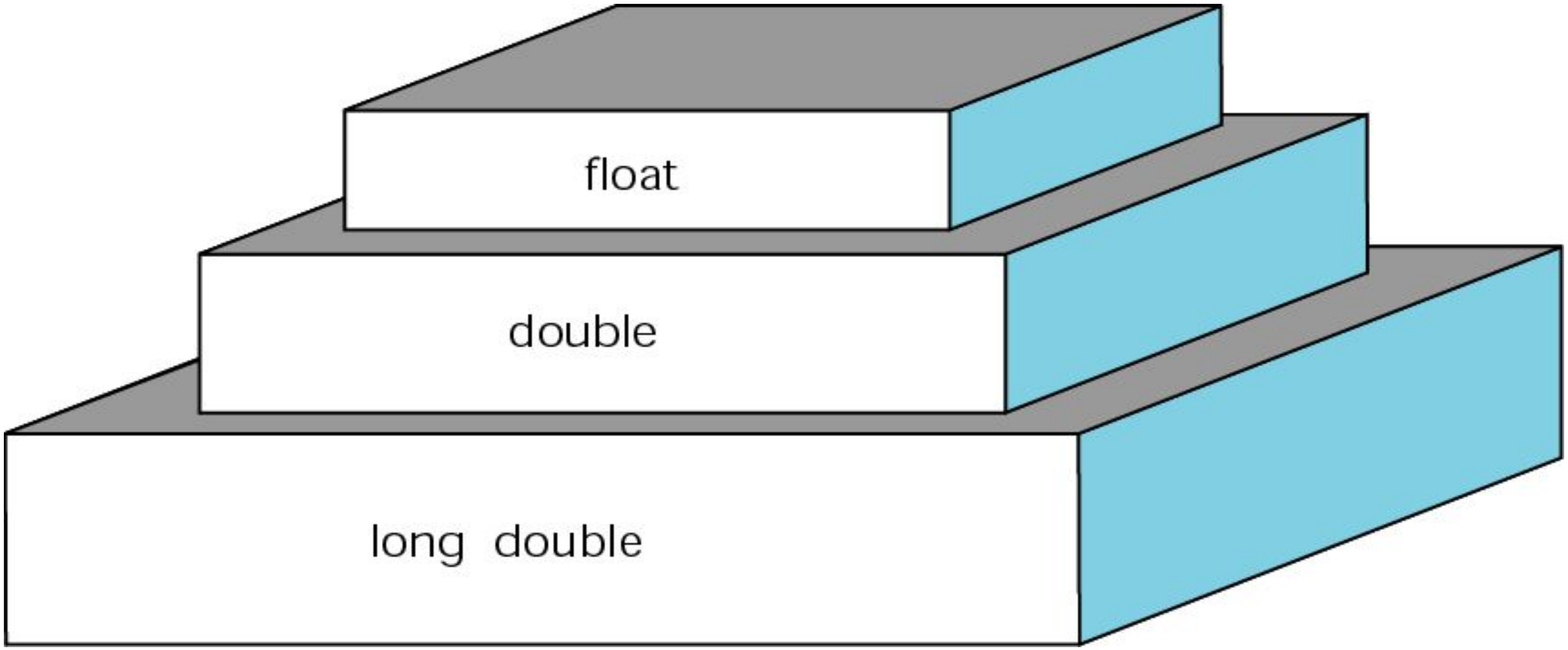
CHAR TYPE

- Any single character from the ch set.
- Uses one byte.

Note:

A character in C++ can be interpreted as a small integer (between 0 and 255). For this reason, C++ often treats a character like an integer.

Figure 2-8 Floating-point types



Type	Byte size	Number of bits
float	4	32
double	8	64
long double	10	80

Table 2-6 Typical float sizes

Regardless of machine size, C++ requires that the following relationship must be t

```
sizeof (float) <= sizeof (double) <= sizeof (long double)
```

Another difference between *float* and *int* types is that *float* is always signed. A

Note:

In C++ the Boolean constants are true and false. Additionally, following traditional standards, any nonzero number is considered true, and zero is considered false.

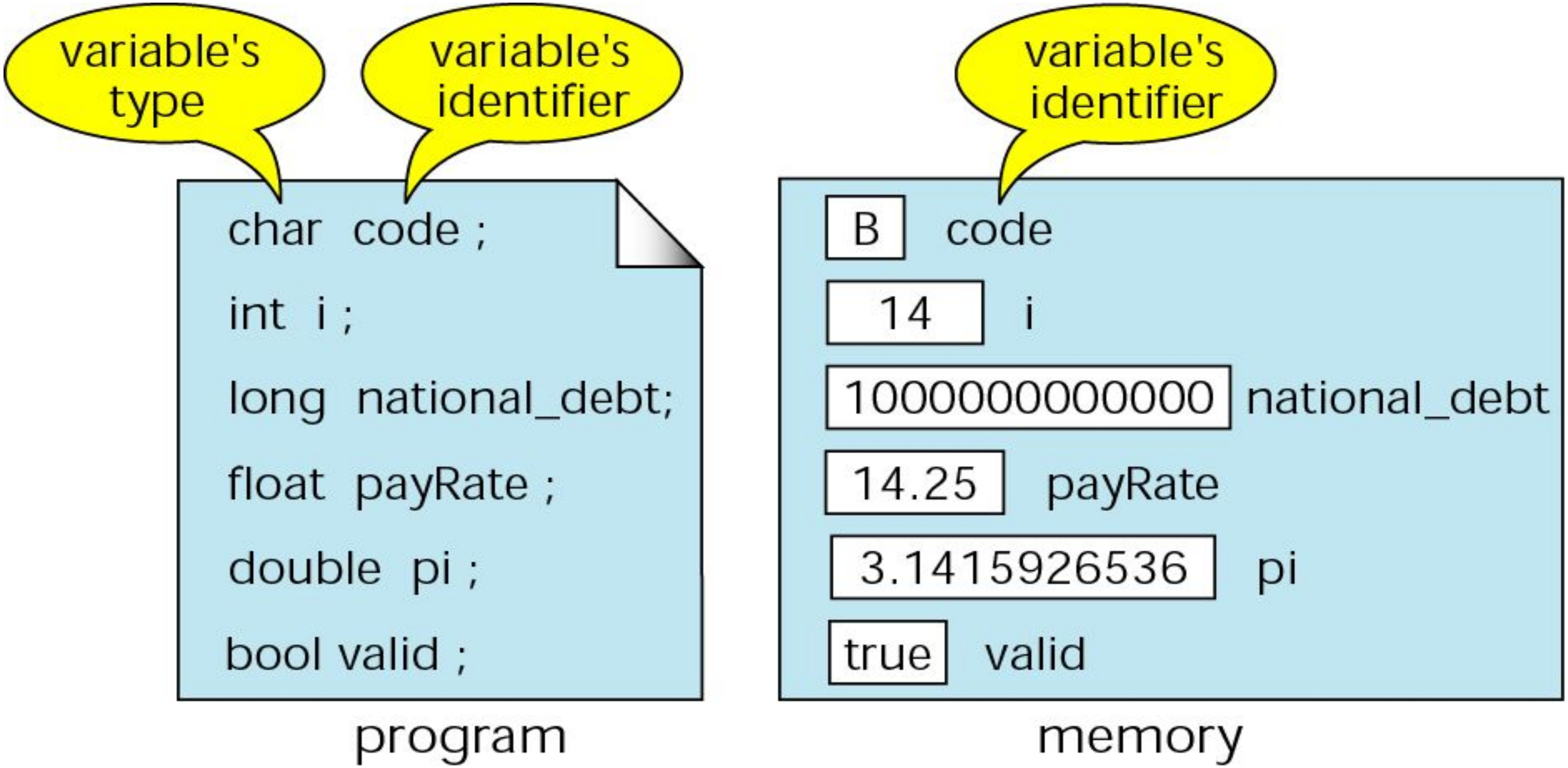
A summary of the five st

Data type	C++
void	void
character	char
integer	unsigned short int unsigned int unsigned long int short int int long int
floating point	float double long double
boolean	bool

Variables

- Variable names
  - Correspond to locations in the computer's memory
  - Every variable has a name, a type, a size and a value
  - Whenever a new value is placed into a variable, it replaces the previous value - it is destroyed
  - Reading variables from memory does not change them

Figure 2-9 Variables in memory



```
char   code ;
int  i ;
long  national_debt;
float  payRate ;
double  pi ;
bool valid ;
```

program

B	code
14	i
1000000000000	national_debt
14.25	payRate
3.1415926536	pi
true	valid

memory

Brooks/Cole
Thomson Learning™

Note:

When a variable is defined, it is not initialized. The programmer must initialize any variable requiring prescribed data when the function starts.

Variable Initialization

- int count=0;
- int a,b=0;
- int a=c=d=0;

2.6

Constants

and use them in everyday life. Thus, the value fifteen is simply coded as 15.

If you code the number as a series of digits, its type is signed integer, or long integer if the number is large. You can override this default by specifying unsigned (*u* or *U*) and long (*l* or *L*) after the number. The codes may be combined and may be coded in any order. Note that there is no way to specify a *short int* constant. When you omit the suffix, it defaults to *int*. While both upper- and lowercase codes are allowed, we recommend that you always use uppercase to avoid confusion (especially with the *l*, which in many cases looks like the number 1). Table 2-9 shows several examples of **integer constants** The default types are typical for a personal computer.

Literal	Value	Type
`+123`	`123`	`int`
`-378`	`-378`	`int`
`-32271L`	`-32,271`	`long int`
`76542LU`	`76,542`	`unsigned long int`

Table 2-9 Examples of integer constants

Float constants are numbers with decimal parts. They are stored in memory as tw parts: the significand and the exponent. The default type for float constants is *doubl* If you want the resulting data type to be *float* or *long double*, you must use a code specify the desired data type. As you might anticipate, *f* and *F* are used for *float* ar *l* and *L* are used for *long double*. Do not use the l

The default type for floating-point literals is double. Floating-point literals of type float or long double can be specified by adding one of the following suffixes:

Suffix	Type
f or F	float
l or L	long double

```
3.14159L // long double
6.02e23f // float
```

Note:

A character constant is enclosed in single quotes.

Note:

Use single quotes for character constants.

Use double quotes for string constants.

Note:

The only bool types constants are true, printed as 1, and false, printed as 0.

Other literals

Three keyword literals exist in C++: true, false and nullptr:
true and false are the two possible values for variables of type bool.
nullptr is the null pointer value.

```
bool foo = true;
bool bar = false;
int* p = nullptr;
```

Figure 2-11 Null characters and null strings

'\0' → Null character

"" → Empty string

2.7

Coding Constants

- literal constant---unnamed constant

‘a’, 5, “hello”

Most common form of constant

- Defined constant

#define tax_rate 10

- Memory constant

Const float pi=3.141

```
const char tab = '\t';
```

Input statements

```
cin >> variable-name;
```

Meaning: read the value of the variable called <variable-name> from the user

Example:

```
cin >> a;
cin >> b >> c;
cin >> x;
cin >> my-character;
```

Output statements

```
cout << variable-name;
```

Meaning: print the value of variable <variable-name> to the user

```
cout << “any message “;
```

Meaning: print the message within quotes to the user

```
cout << endl;
```

Meaning: print a new line

Example:

```
cout << a;
cout << b << c;
cout << “This is my character: “ << my-character
<< “ he he he”
 << endl;
```