Ch 2: Introduction to C++

- Parts of a C++ Program
- Identifiers
- Data Types
- Variables
- Constants
- Scope
- Arithmetic Operators

Parts of a C++ Program

- Preprocessor Directives
- Comments
- Statements
- Functions

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
  cout << "Programming is great fun!";
  return 0;
}
```

Preprocessor Directives

- line begins with # and is terminated with newline character
- is not considered a C++ statement; is not terminated with semicolon
- backslashes (\) can be used to continue onto succeeding lines
- performs in-line expansion of source code before creating object code

Comments

- compiler ignores comments; they are legal anywhere whitespace is
- useful in documenting program
  - program and function headers
  - inline comments
- single line comments
  - // extends to end of that line in program
- multiple line comments
  - enclosed by /* */ pair
  - can extend across many lines

Statements

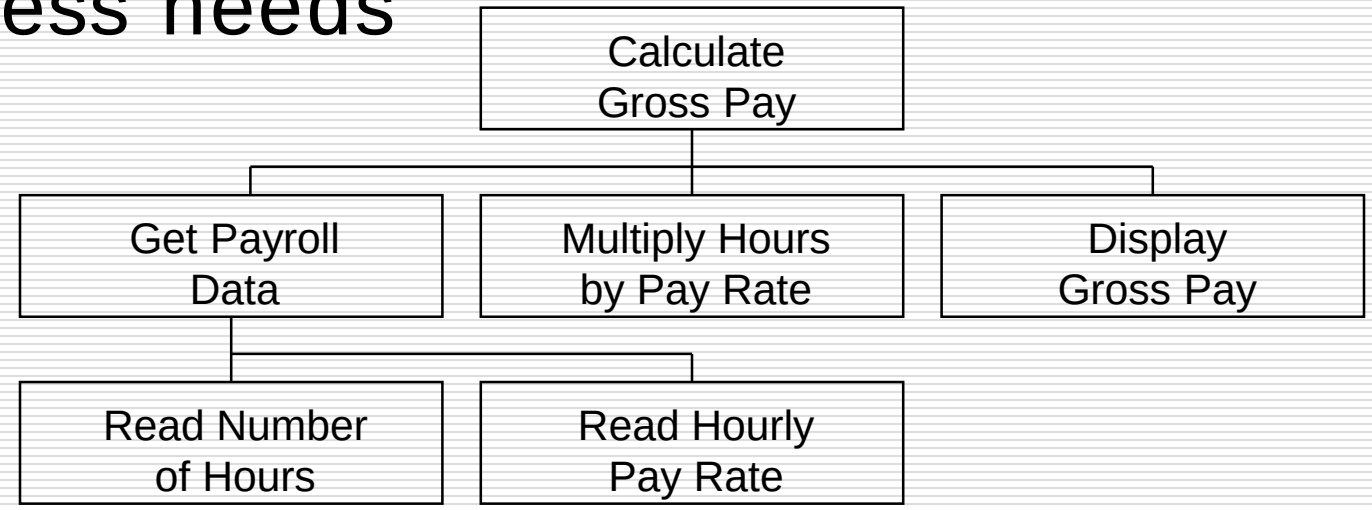
- source code that is terminated by a semicolon ;
  - establish an identifier
  - perform an action
  - act as placeholder
- compound statements enclose zero or more statements within braces { }
  - also called a **block**

Functions

- function header identifies function name and parameters within parenthesis
  - parameters have data type and identifier name
  - parameterless functions must still have parenthesis
- compound statement that performs a related set of actions (modular programming)
  - value-returning
  - void

Functions

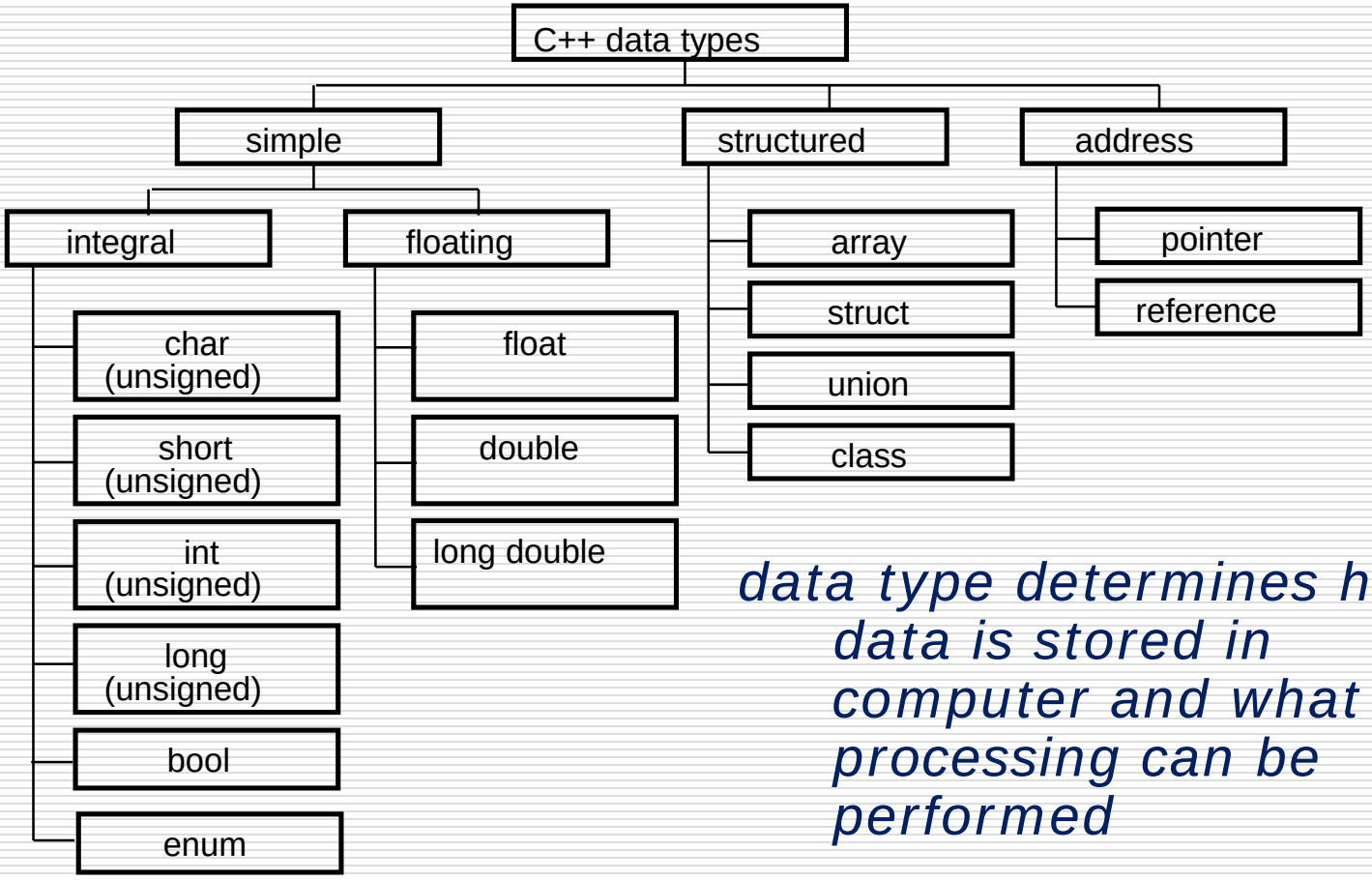
- must have **`main`** function
  - called by operating system
- other functions invoked according to process needs



Identifiers

- Programmer defined name that represents program element
- Naming rules
  - First character must be upper or lower case letter or underscore
    - After first letter, can use upper or lower case letter, number, or underscore
  - Cannot be keyword, but can contain keyword
  - Is case sensitive
  - Standards
    - CamelCase
    - Underscore

Data Types



data type determines how data is stored in computer and what processing can be performed

Data Types

- Numeric data types store numbers
  - Amount of memory determines range of data
  - unary **`sizeof()`** operator return number of bytes required for data storage

```
1<=sizeof(char)<=sizeof(short)<= sizeof(int)<=sizeof(long)

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

- Character data types store non-numeric data

Numeric Data Types

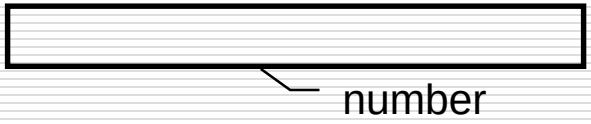
- Integer data types represent whole numbers

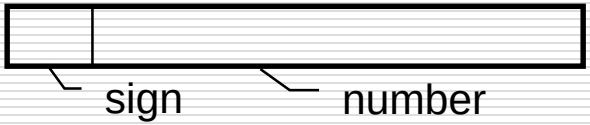
Table 2-6 Integer Data Types

Data Type	Typical Size	Typical Range
`short int`	2 bytes	−32,768 to +32,767
`unsigned short int`	2 bytes	0 to +65,535
`int`	4 bytes	−2,147,483,648 to +2,147,483,647
`unsigned int`	4 bytes	0 to 4,294,967,295
`long int`	4 bytes	−2,147,483,648 to +2,147,483,647
`unsigned long int`	4 bytes	0 to 4,294,967,295
`long long int`	8 bytes	−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`unsigned long long int`	8 bytes	0 to 18,446,744,073,709,551,615

Integral Limits

- unsigned range of values
  - minimum value
    - 0
  - maximum value
    - $2^{(numbits)} - 1$
- signed range of values
  - minimum value
    - $-(2^{(numbits - 1)})$
  - maximum value
    - $2^{(numbits - 1)} - 1$
- default is signed





Numeric Data Types

- Floating point data types represent real numbers

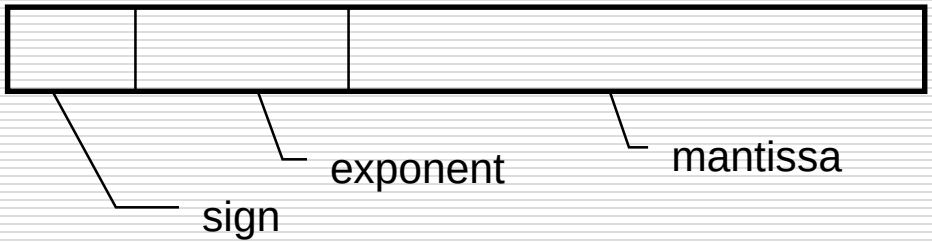
Table 2-8 Floating Point Data Types on PCs

Data Type	Key Word	Description
Single precision	`float`	4 bytes. Numbers between ±3.4E-38 and ±3.4E38
Double precision	`double`	8 bytes. Numbers between ±1.7E-308 and ±1.7E308
Long double precision	`long double*`	8 bytes. Numbers between ±1.7E-308 and ±1.7E308

Floating Limits

- range and precision of values
  - expressed in exponential (scientific) notation
  - minimum and maximum negative and positive values same but opposite sign
  - exponent determines range
  - mantissa determines precision

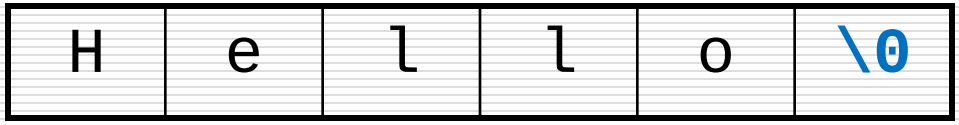
Decimal Notation	**Scientific Notation**
247.91	2.4791 x 10^2
0.00072	7.2 x 10^(-4)



Non-Numeric Data Types

- **`char`** stores individual characters
  - Usually 1 byte of memory
- **`character string`** is a series of consecutive character memory locations terminated by null '\0'

"Hello"



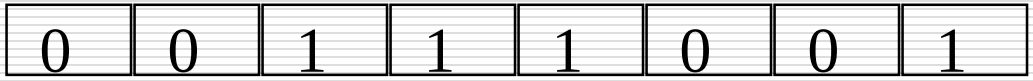
Non-Numeric Data Types

- C++ **`string`** class stores string variables
  - Use **`#include <string>`**
  - Define
    - **`string movieTitle;`**
  - Assign
    - **`movieTitle = “Wheels of Fury”;`**
- **`bool`** stores value of **`true`** or **`false`**
  - Internally stored as small integers
  - true is 1 and false is 0

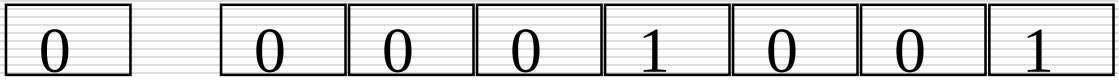
Internal Data Representation

- internal binary storage depends upon data type

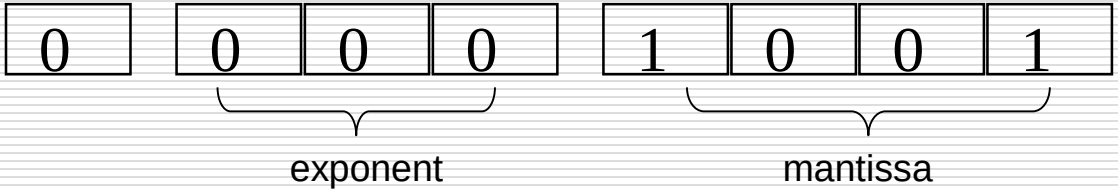
```
char cVar = '9';
// ASCII 57 (see App B)
```




```
int iVar = 9;
```




```
float fVar = 9.0;
```



Variables

- a location in memory, referenced by an identifier, whose value can change during program execution
- must declare identifier before it can be used (*variable scope*)
  - undeclared identifier error
- can only contain data values of type specified in declaration
  - otherwise, implicit **conversion** is performed

Variables

- declaring a variable means specifying both its name and its data type

```
int numStudents;
```

- variables can be initialized in declaration

```
int numStudents = 24;
```

- variables can be assigned values after declaration

```
numStudents = 24;
```

- C++ 11 allows variable declaration and initialization with **auto** keyword to determine data type from initialization value

```
auto numStudents = 24;
```

Literal Constants

- constant values used within an expression
- integral
- floating
- character
  - surrounded by single quotes
  - ‘A’, ‘a’
- string
  - surrounded by double quotes
  - “A”, “hello”

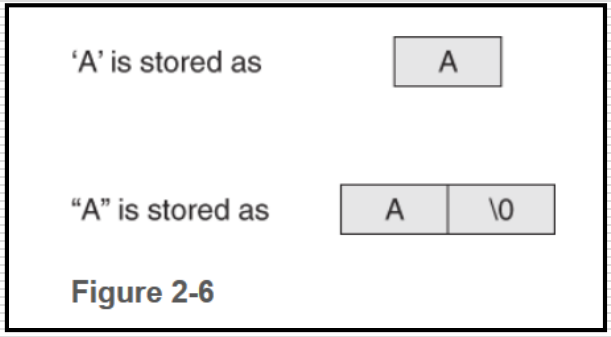


Figure 2-6

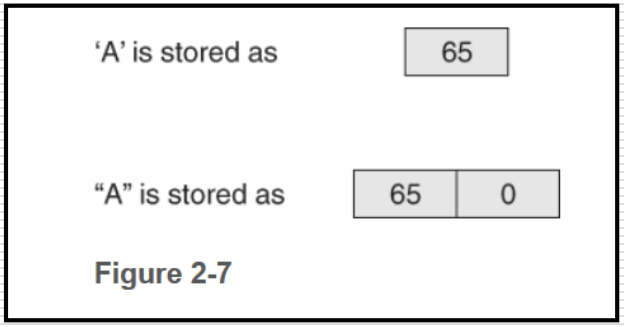


Figure 2-7

Literal Constants: Integral

- defining integral constants
  - decimal (base 10) integer *(*default*)*
    - 256
  - octal (base 8) integer
    - 0400
  - hexadecimal (base 16) integer
    - 0x100
  - unsigned integer
    - 256U
  - long integer
    - 256L
  - unsigned long integer
    - 256UL

Literal Constants: Floating Point

- defining floating point constants
  - float
    - 256.0F
  - double
    - 256.0 *(*default*)*
    - 2.56E+2
  - long double
    - 256.0L

Named Constants

- also called symbolic constants
- using preprocessor directive
  - `#define PI 3.14159`
  - causes code substitution in compiler
- using const variable
  - `const double PI = 3.14159;`
  - location in memory, referenced by an identifier, whose value cannot change during program execution
- C++ 11 allows constant declaration and initialization with **auto** keyword to determine data type from initialization value

```
auto const PI = 3.14159;
```

Scope

- Variable scope is part of program where variable is accessible
  - Cannot be accessed before definition

```
// This program can't find its variable
#include <iostream>
using namespace std;

int main()
{
  cout << value;
  int value = 100;
  return 0;
}
```

Arithmetic Operators

- Used in performing numerical calculations
  - + addition or plus
    - binary or unary
  - - subtraction or minus
    - binary or unary
  - * multiplication
    - binary
  - / division
    - binary; result data type dependent upon operand types
  - % modulus (remainder)
    - binary; integers only