BTE2313

Chapter 2: Introduction to C++ Programming

by
Sulastri Abdul Manap
Faculty of Engineering Technology
sulastri@ump.edu.my

Objectives

- In this chapter, you will learn about:

  1) keywords/reserve words, special symbols, and identifiers in C++

  2) data types

  3) assignment statement

  4) Modifying your first C++ Program

  5) Structure a program, with comments as part of documentation

The Basics of a C++ Program (cont.)

Basic of C++:

- reserved words or keywords (such as *if, while, int,* etc.)
- Consists of symbols (such as *{ } = | <= ! [ ] * & (and more)*
- Consists of *programmer-defined names* known as identifiers for variables, constants and functions

Reserved Words (Keywords)

- Reserved word /keywords:
  - Cannot be changed within program
  - Cannot be used for other than their anticipated use

  Examples:
    - `float`
    - `void`
    - `include`
    - `long`
    - `return`

Identifiers

Identifiers are variables/constants name that are given by *programmer*:

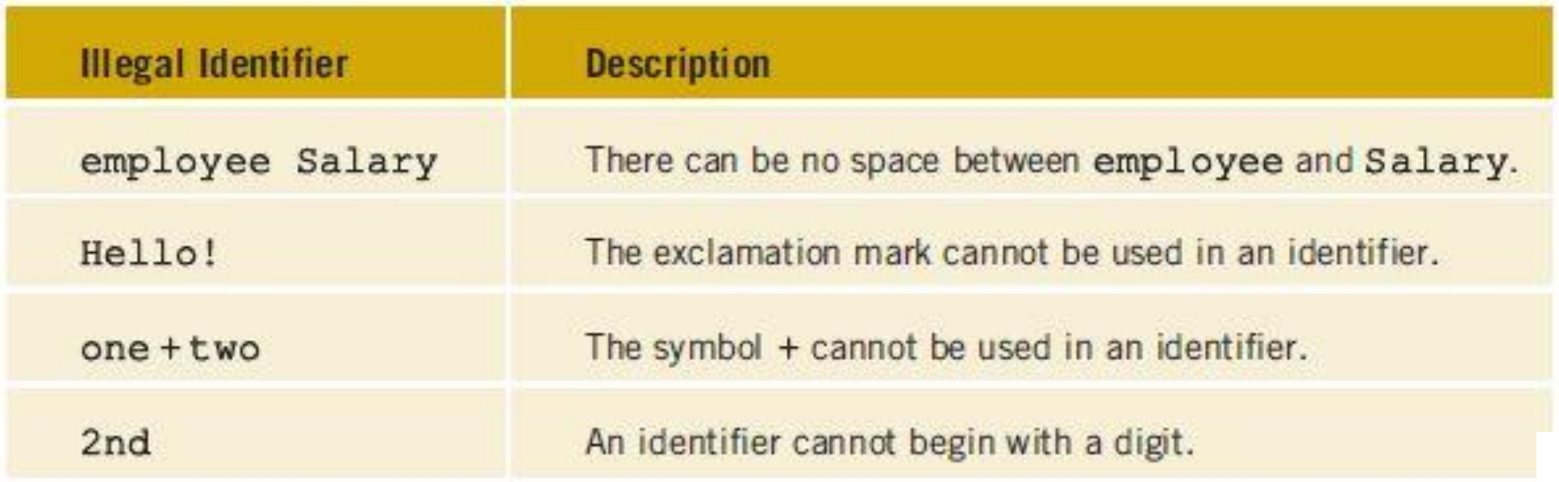
- Must use letter, digits, _ (underscore) and have 1 - 31 chars long
- Must start with a letter or an underscore (_)
- Case-sensitive, where *Mum* is ***different*** than *mum*
- should be meaningful: *studentNumber* is better than *n* or *sn*

Identifiers (cont.)

- Legal identifiers in C++:

```
second
comparison
paySlip
```

TABLE 2-1 Examples of Illegal Identifiers



Illegal Identifier	Description
`employee Salary`	There can be no space between `employee` and `Salary`.
`Hello!`	The exclamation mark cannot be used in an identifier.
`one+two`	The symbol + cannot be used in an identifier.
`2nd`	An identifier cannot begin with a digit.

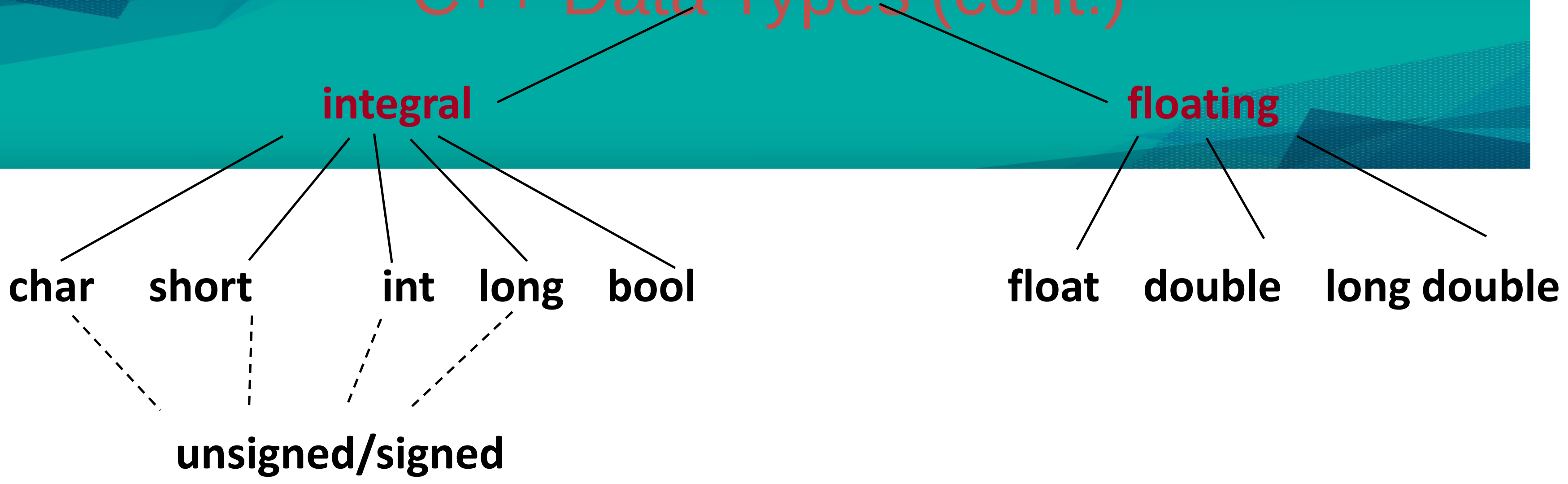
C++ Data Types

- Each data item has a **type** and a **name**
- **type** determines the **range** of possible **values** it can hold and **operations** that can be used on it
- Can we do this?

```
x =  "Joe" + 1 ;
```

- Commonly used data types: int, float, double, char, string.

C++ Data Types (cont.)



Data Type	Size (bytes)	Size (bits)	Value Range
unsigned char	1	8	0 to 255
signed char	1	8	-128 to 127
char	1	8	either
unsigned short	2	16	0 to 65,535
short	2	16	-32,768 to 32,767
unsigned int	4	32	0 to 4,294,967,295
int	4	32	-2,147,483,648 to 2,147,483,647
unsigned long	8	64	0 to 18,446,744,073,709,551,616
long	8	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long	8	64	0 to 18,446,744,073,709,551,616
long long	8	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4	32	3.4E +/- 38 (7 digits)
double	8	64	1.7E +/- 308 (15 digits)
long double	8	64	1.7E +/- 308 (15 digits)
bool	1	8	false or true

`int` Data Type

- *int* keyword is used for integer numbers.
  - → cannot have any decimal point, comma, or other symbol
  - → + or - in the numbers are allowed
  - → example of constants can be coded/written as 1234, -12, +67
- Examples:

```
int x; //data type & identifier’s name without value yet
int x, y; //declares 2 integers without values
int numPeople = 16000; //declare and set initial value
```

float and double Data Type

- Used for *real* numbers that have decimal point, + or – are allowed and no comma or other symbols are allowed.
- Example of constants can be coded/written as 2.11, -78.98, 0.025, 10.5e6 (10.5 times 10 to the 6th power)

- Declaration Examples:

```
float  _pi = 3.1416;     //declares names and sets initial value
float y = 10.5,          //comma here is used to go to next line
z = 15.6543;
double x = 1.5;
double y = 245.643;
Float biggest = 3.28e4;  // exponential notation: 32800
```

`char` Data Type

- Only for one character
- Constants are coded/written using single quotes
  - `'B', 'b', '9', '$', '-', '@', '!'`
- A blank space is considered a character which is written as `' '`
- Example of declaration s:

```
char go;
char selection = ‘s';
```

`String` Data Type

- Strings are used for sequence of characters and written/coded by “string” keyword.
- From 0 to many characters
- Constants for string are written/coded with double quotes

```
"!!Hello world!!"
```

- Examples:

```
string str;
string name = “Daniel”;
```

`bool` Data Type

- `bool` keyword is used for Boolean data, that can have only 2 values; either `true` or `false`
- This data type id used to manipulate logical (Boolean) expressions
- `true` or `false` are logical values and they are all reserved words

C++ Expression

- A C++ expression consists of combination of constants, variables and also operators.
- C++ expression can be evaluated to calculate a value of any data.
- Expression can be a variable or a constant, or an operation such as x + y, or a function call such as CalcArea(2, 4)

Assignment Operator

- An operator to give (assign) a value to a variable.
- Denote as ‘=‘
- Only **variable** can be on the left side.
- An expression is on the right side.
- Variables keep their assigned values until changed by another **assignment statement** or by **reading in** a new value.

Syntax for Assignment Operator

- Assignment operation is used for storing result of an expression into a variable.
- expression on right will be **evaluated** first, and followed by storing the result in the memory location of the variable indicated on left

Assignment Operator Mechanism

- Example:

```
int count = 0;
int starting;
starting = count + 5;
```

0

12345 (garbage)

- Expression evaluation:
  - Get value of **`count`**: 0
  - Add 5 to it.
  - Assign to **`starting`**

5

C++ Program: Example

```
#include <iostream>

using namespace std;
int main()
{
    int number1, number2, summation;
    number1 = 20;
    number2 = 14;
    summation = number1 + number2;
    return 0;
}
```

First C++ Program: Try this!

```
#include <iostream>

using namespace std;
int main()
{
   cout << “Hi BTE2313 students!\n”; //display the string
   return 0;
} //main function ends here
```

Use of Blanks/Whitespaces

- One or more blanks can be used to isolate numbers
- They are also used to separate between reserved words/keywords and also from other symbols
- They cannot appear within a reserved word or identifier
- If blanks are properly utilized, it will make the program more readable.

Semicolons; Brackets () and Commas ,

- All C++ statements must end with a semicolon, which is also known as a statement terminator
- Curly brackets or braces { } are not a C++ statement
  - Can be regarded as delimiters

- Commas separate items in a list

Comments

- In C++, there are comments option, which are not for the C++ compiler, but for the reader.
- There are two types of comments, single line and multiple lines.
- Single line comment begins with //

```
// Example of a single line comment
```

- Multiple lines comments are bounded between /* and */

```
/*
     Example of
     several lines comments
*/
```