E1123 Computer Programming (a)

(Fall 2020)

C++ Basics

INSTRUCTOR

DR / AYMAN SOLIMAN

➢ Contents

1) Course Information.

Lectures: Thursday, (14:10 - 14:40 PM) - (15:20 – 15:50 PM)

Office Hours: Thursday, 10:00 ~ 11:50 PM & 13:00 ~ 14:05 & 14:45 ~ 15:15 PM

Prerequisite: E1021 - E1022

References:

- C++ Programming: From Problem Analysis to Program Design, Fifth Edition D.S. Malik
- Object-Oriented Programming Using C++, Fourth Edition Joyce Farrell
- www.learncpp.com

Instructor:

Dr. Ayman Soliman

Ayman.mohamed01@bhit.bu.edu.eg

TAs:

Eng. Nada Elmeligy
Eng. Mai Maher

Eng. Ahmed Ragab
Eng. Mahmoud Osama

2) Objectives

- To be able to write simple computer programs in C.
- To be able to use simple input and output statements.
- To become familiar with fundamental data types.
- To understand computer memory concepts.
- To be able to use arithmetic operators.
- To understand the precedence of arithmetic operators.

3) C++ compiler directives

- Compiler directives appear in green color in C++.
- The **#include** directive tells the compiler to include some already existing C++ code in your program.
- The included file is then linked with the program.
- There are two forms of **#include** statements:

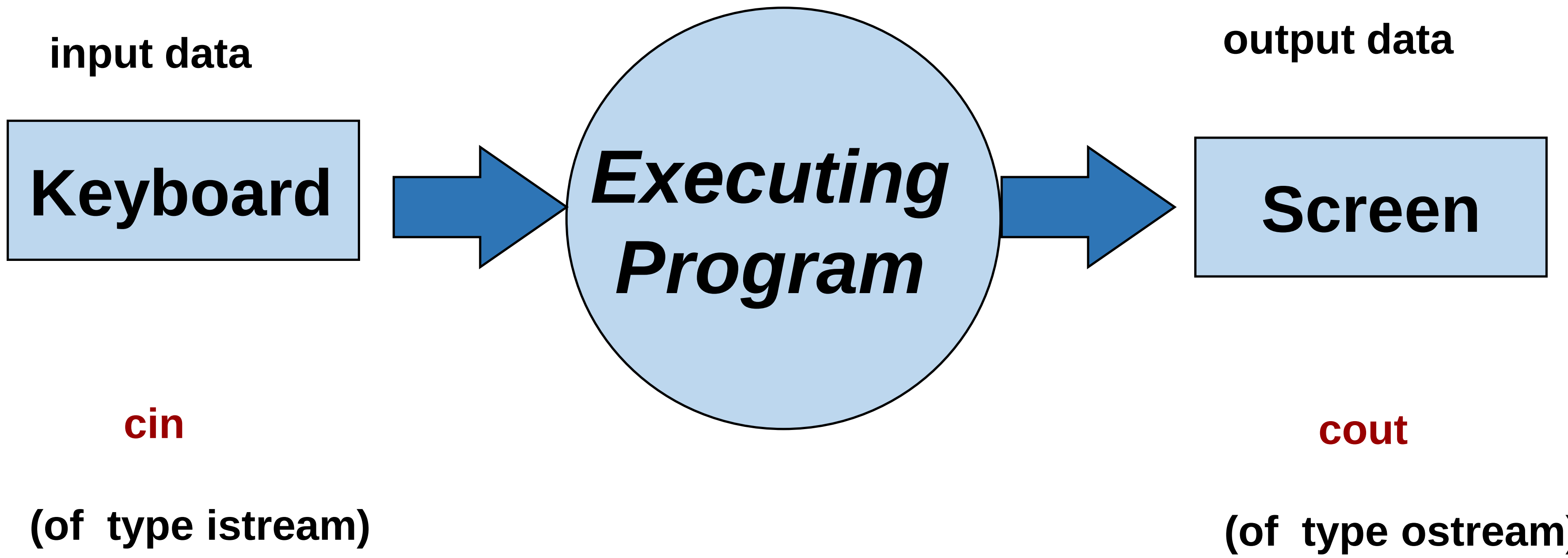
#include <iostream> //for pre-defined files

- the C++ label for a standard header file for input and output streams

#include "my_lib.h" //for user-defined files

Keyboard and Screen, I/O

#include <iostream>



Input

- Variable **cin** is predefined to denote an input stream from the standard input device (the keyboard)

- The extraction operator >> called "get from". The left operand is a **stream expression**, such as **cin--**the right operand is a variable of simple type.

- Operator >> attempts to extract the next item from the input stream and store its value in the right operand variable.

```
cin >> Variable1 >> Variable2 . . . ;
```

Output

- To do input/output, at the beginning of your program you must insert
  #include <iostream>
  using cout; using endl;
- C++ uses streams for input an output
- *stream* - is a sequence of data to be read (*input stream*) or a sequence of data generated by the program to be output (*output stream*)
- Variable **cout** is predefined to denote an output stream that goes to the standard output device (display screen).
- The insertion operator << called "put to".
- The left operand is a stream expression, such as **cout**. The right operand is an **expression** of simple type or a **string constant**.

Output Statements Styles

Syntax

```
cout  <<  Expression1   << Expression2  . . . ;
```

- cout statements can be linked together using << operator.
- These examples yield the same output:

```
cout  <<  “The grades are “ ;
cout  <<  90;
```

```
cout  <<  “The grades are “  <<  90;
```

How Extraction Operator works?

- Input is not entered until user presses <ENTER> key.
- Allows backspacing to correct.
- Skips whitespaces (space, tabs, etc.)
- Multiple inputs are stored in the order entered:

  cin>>num1>>num2;

  User inputs: **5 8**

  Assigns num1 = 5 and num2 = 8

- No difference between a single cin with multiple variables and multiple cin statements with one variable

```
cin>>num1>>num2;
```

```
cin>>num1;
cin>>num2;
```

These examples yield the same output.

Expressions

- An expression is a valid arrangement of variables, constants, and operators.
- In C++, each expression can be evaluated to compute a value of a given type
- In C++, an expression can be:
  - A variable or a constant (area, 22)
  - An operation (x + y, z / 5)
  - Function call (calculaterectanglearea(5, 10))

Comments

- Allow commentary to be included in program
- C++ has two conventions for comments

  // single line comment (preferred)

  /* long comment */ (save for debugging)

- Typical uses

  Identify program and who wrote it

  Record when program was written

  Add descriptions of modifications

Escape sequences

- Escape sequences are used to represent certain special characters within string literals and character literals.

Alert	\a	Makes an alert, such as a beep
Backspace	\b	Moves the cursor back one space
Formfeed	\f	Moves the cursor to next logical page
Newline	\n	Moves cursor to next line
Carriage return	\r	Moves cursor to beginning of line
Horizontal tab	\t	Prints a horizontal tab
Vertical tab	\v	Prints a vertical tab
Single quote	\’	Prints a single quote
Double quote	\”	Prints a double quote
Backslash	\\	Prints a backslash
Question mark	\?	Prints a question mark

Preprocessor directives

The **preprocessor** is a separate program that runs just before the compiler when you compile your program. When you #include a file, the preprocessor copies the contents of the included file into the including file at the point of the #include directive.

Directives are specific instructions that start with a # symbol and end with a newline (NOT a semicolon).

There are two different types of directives

```
// The files or libraries that are part of the C++ standard library
#include <filename>

// You'll generally use this form for including your own header files
#include "filename.h"
```

4) Libraries and the C++ Standard Library

- **A library** is a collection of precompiled code (functions) that has been "packaged up" for reuse in many different programs such as math library, sound library and a graphics library.

- C++ comes with a library called the **C++ standard library** that provides additional functionality for your use and it is divided into areas or libraries that provide a specific type of functionality. One of the **most used** parts of the C++ standard library is the **iostream library**, which contains functionality for writing to the screen (cout) and getting input (cin) from a console user.

5) First Program

Preprocessor directives tell the compiler to add the contents of the iostream header to the program that includes cout and cin.

```
#include <iostream>

int main()
{
cout << "Hello world!" ;
return 0;
}
```

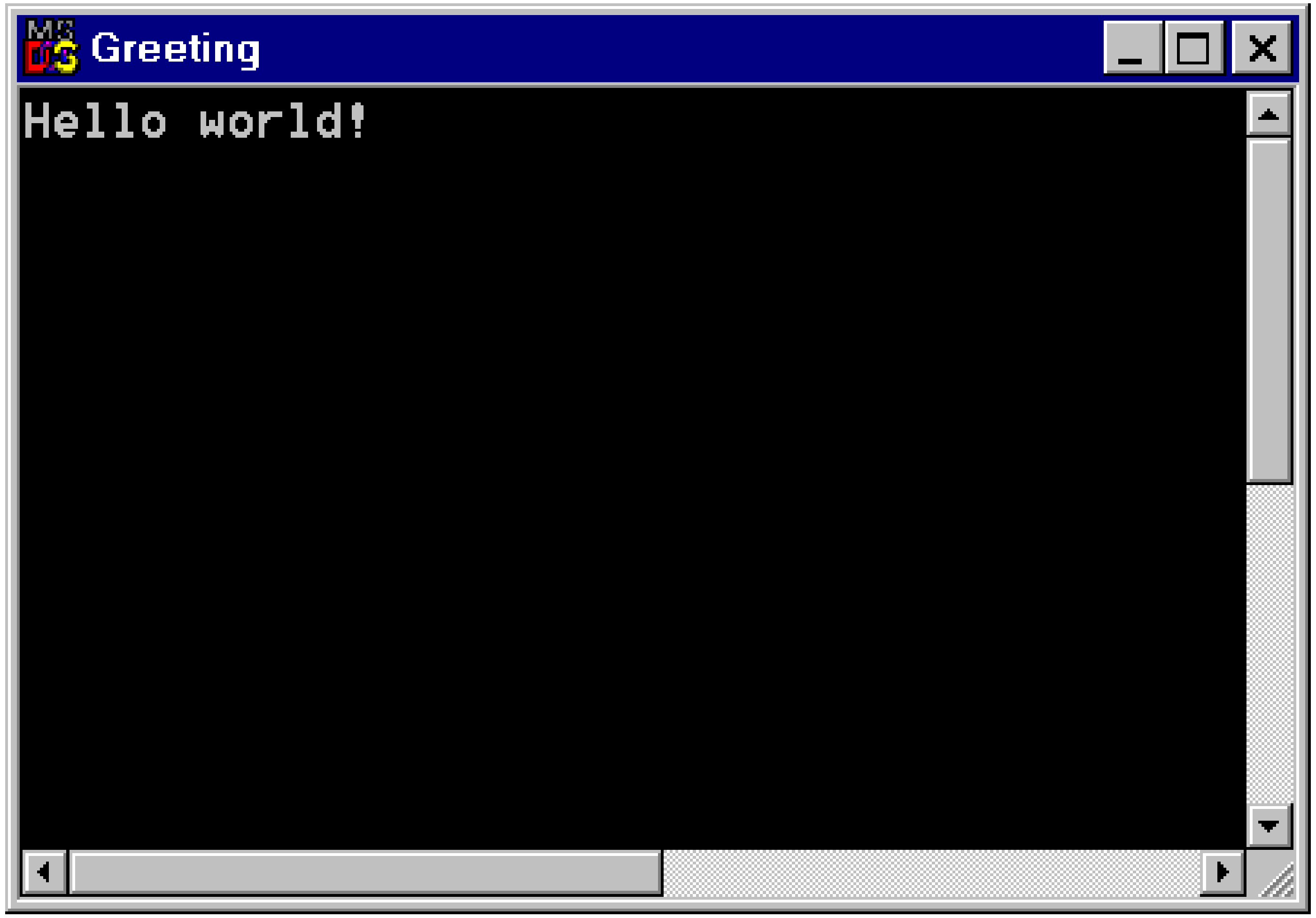
This line is blank, and it is ignored by the compiler.

declaring the **main()** function, which is mandatory. Everything inside curly brace {} is a part of main() function.

The << symbol is an the **output operator**.

A **return statement** sends a value back to the operating system that indicates whether it was run successfully or not.

Greeting Output



```
Hello world!
```

6) Second Program

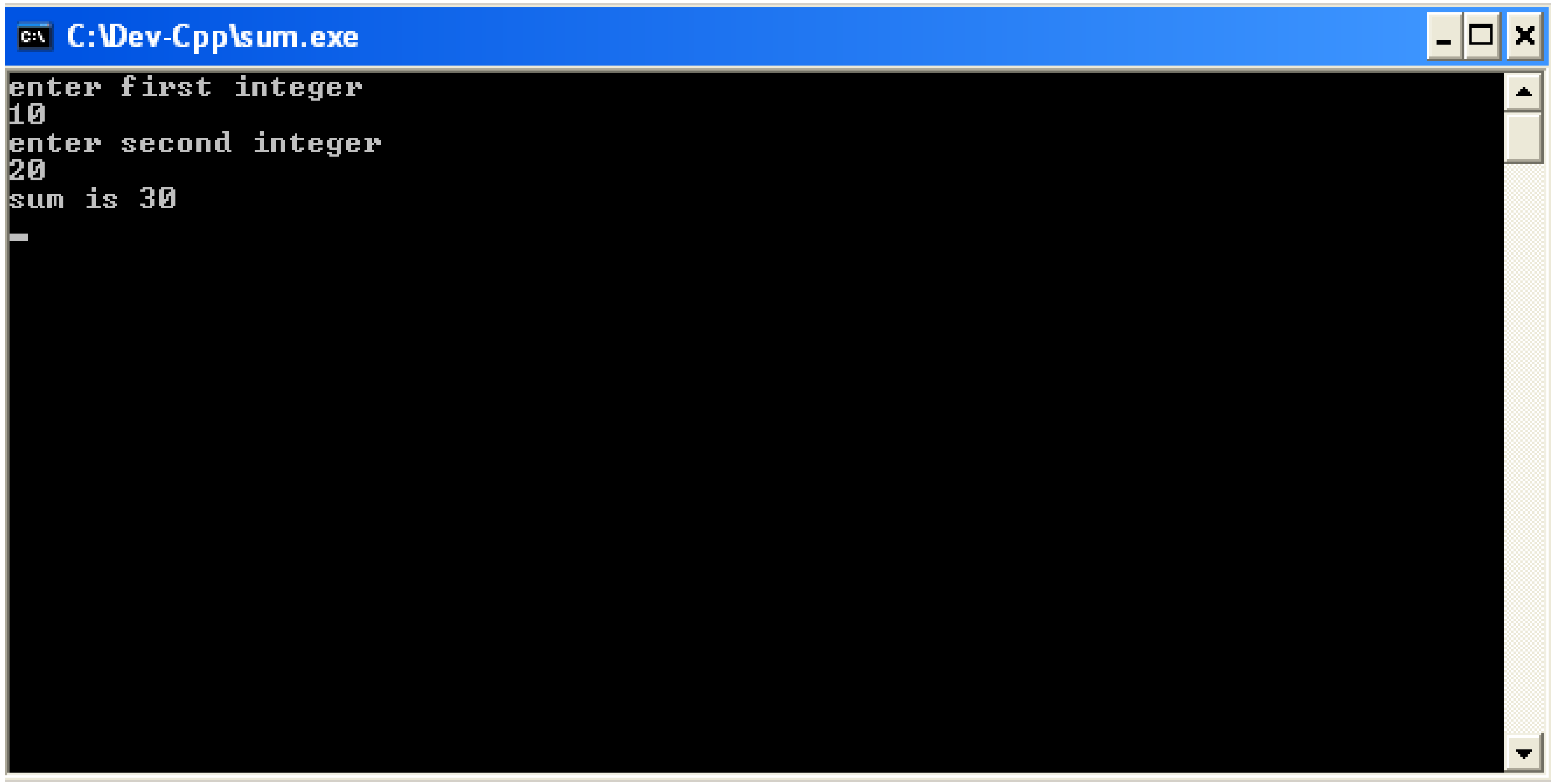
```
//example
// program to add two numbers
#include <iostream.h>

int main()
{
   int integer1, integer2, sum;            // declaration

   cout << "Enter first integer\n";        // prompt
   cin >> integer1;                        // read an integer
   cout << "Enter second integer\n";       // prompt
   cin >> integer2;                        // read an integer
   sum = integer1 + integer2;              // assignment of sum
   cout << "Sum is " << sum << endl;       // print sum

   return 0;                               // indicate that program ended successfully
}
```

Output



```
enter first integer
10
enter second integer
20
sum is 30
```

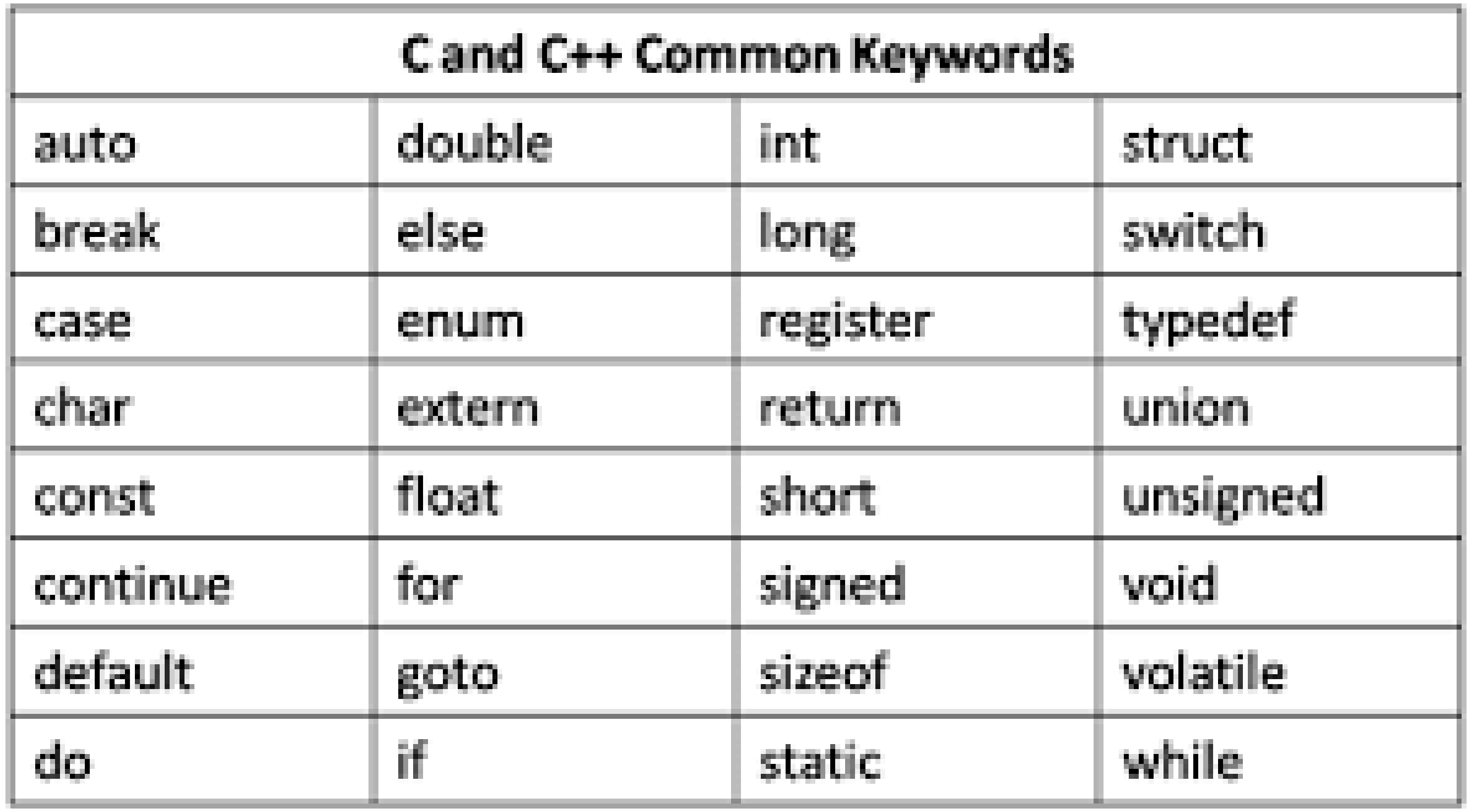
7) Variables and Assignments

- Variables are like small blackboards
  - We can write a number on them
  - We can change the number
  - We can erase the number
- C++ variables are names for memory locations
  - We can write a value in them
  - We can change the value stored there
  - We cannot erase the memory location

identifiers

- Variables names are called identifiers
- Choosing variable names
  - Use short meaningful names that represent data to be stored
  - generally avoid single letter variables
- First character must be
  - a letter
  - the underscore character
- Remaining characters must be
  - letters
  - numbers
  - underscore character
- Identifiers can not be any keywords (reserved words)

C++ keywords



C and C++ Common Keywords			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

- Keywords are words reserved as part of the language
- They cannot be used by the programmer to name things
- They consist of lowercase letters only
- They have special meaning to the compiler

Whitespace and basic formatting

Whitespace is a term that refers to characters that are used for formatting purposes. In C++, this refers primarily to spaces, tabs, and (sometimes) newlines. The C++ compiler generally ignores whitespace, with a few minor exceptions. **The following statements all do the exact same thing:**

```
cout << "Hello world!";
```

```
cout              <<           "Hello world!";
```

```
         cout << "Hello world!“     ;
```

```
cout
<< "Hello world!";
```

```
cout << "Hello ";  cout << "world!";
```

```
int main() { return 0; }
```

```
int main() {
   return 0; }
```

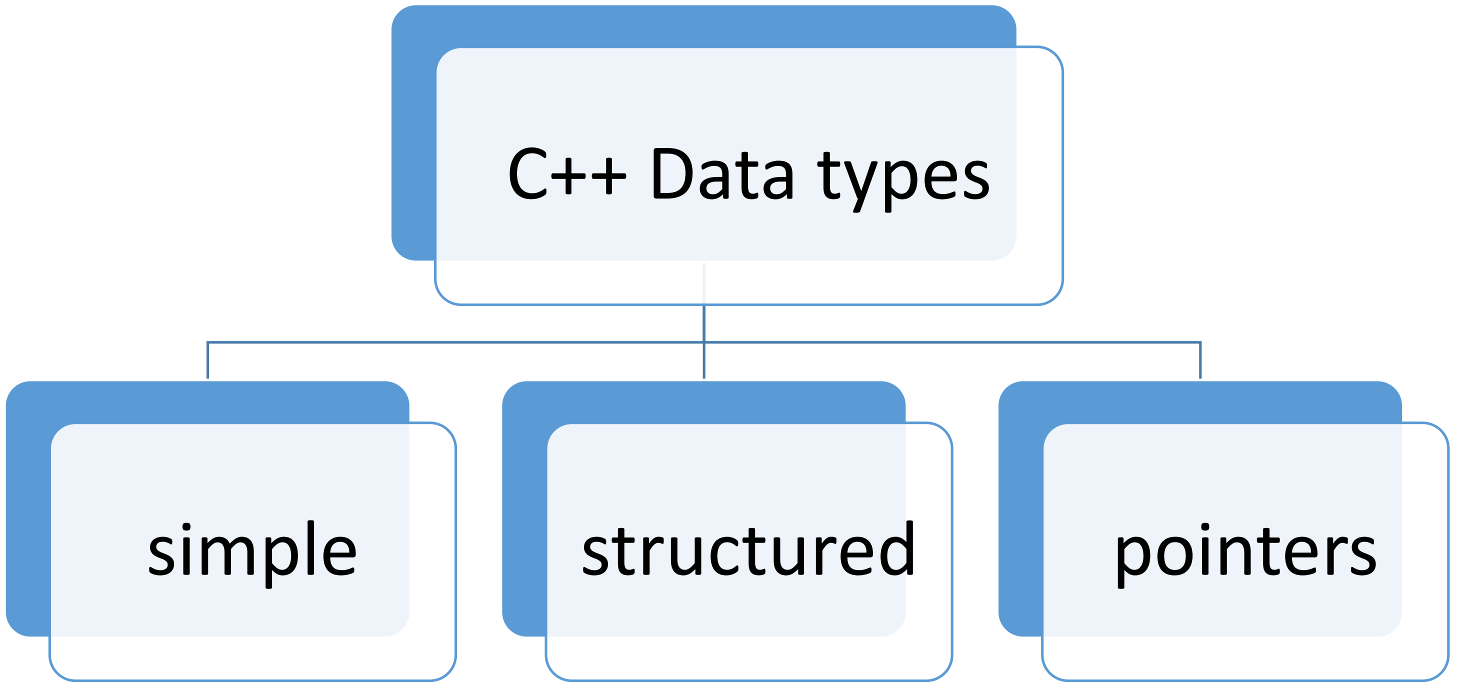
```
int main()
{    return 0; }
```

```
int main()
{
   return 0;
}
```

8) Data types

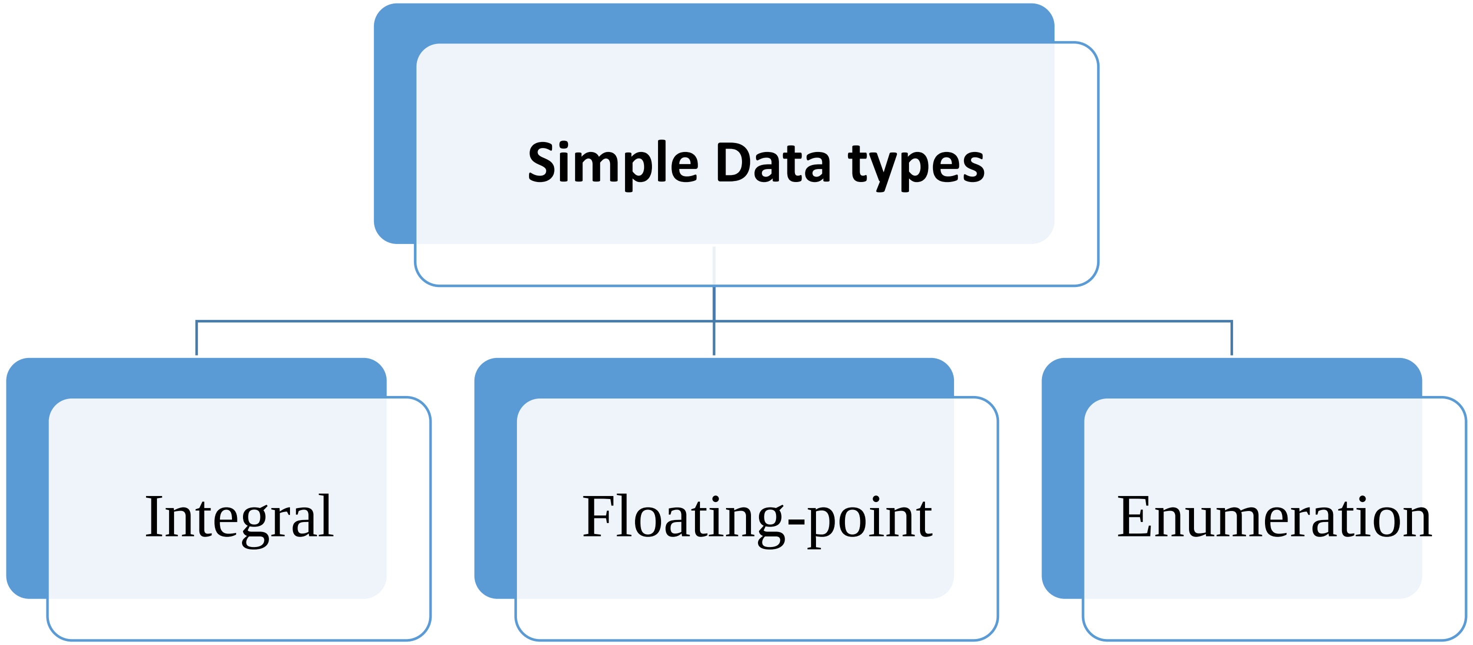
- Data type: set of values together with a set of operations

- C++ data types fall into three categories:



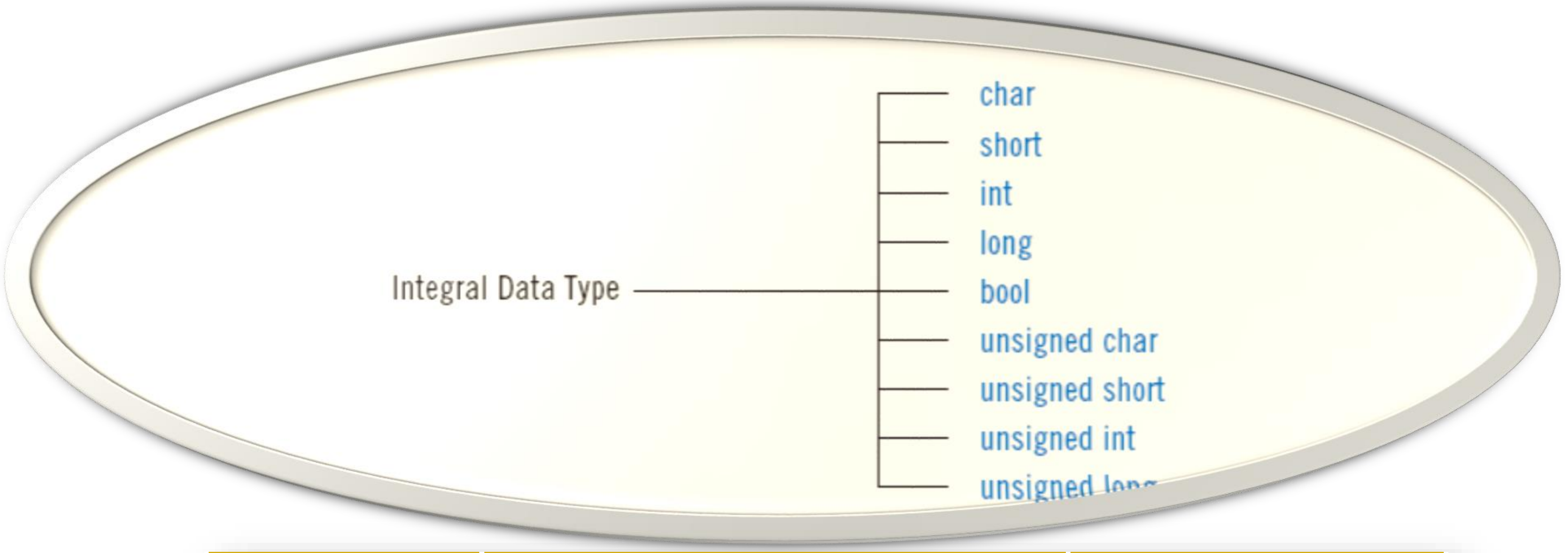
Simple Data types

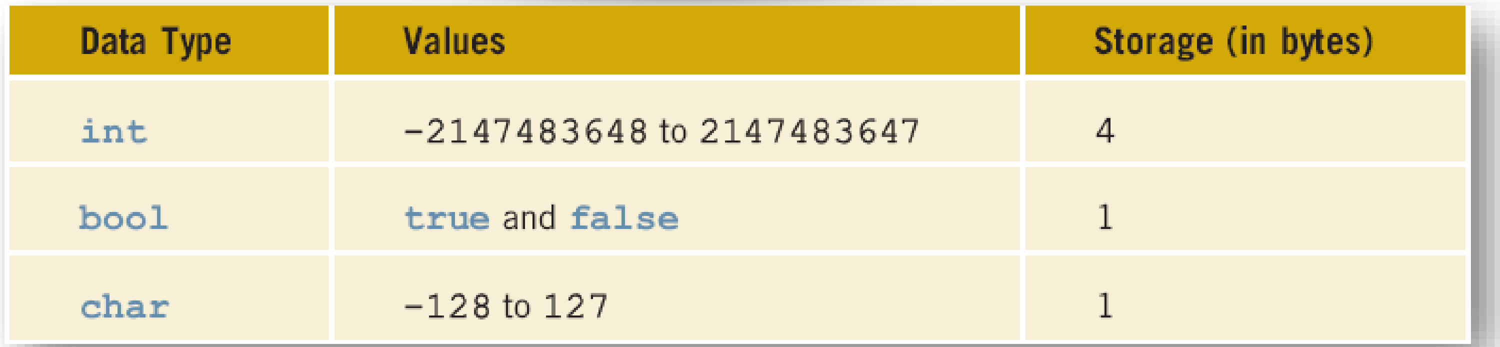
- Three categories of simple data
  - ❑ Integral: integers (numbers without a decimal)
  - ❑ Floating-point: decimal numbers
  - ❑ Enumeration type: user-defined data type



Simple Data types (cont.)

- Integral data types are further classified into nine categories:





Data Type	Values	Storage (in bytes)
int	-2147483648 to 2147483647	4
bool	true and false	1
char	-128 to 127	1

int Data Type

- Examples:

  -6728

  0

  78

  +763
- Positive integers do not need a + sign
- No commas are used within an integer
- Commas are used for separating items in a list

bool Data Type

bool type

- Two values: true and false
- Manipulate logical (Boolean) expressions

true and false are called logical values

bool, true, and false are reserved words

char Data Type

The smallest integral data type

Used for characters: letters, digits, and special symbols

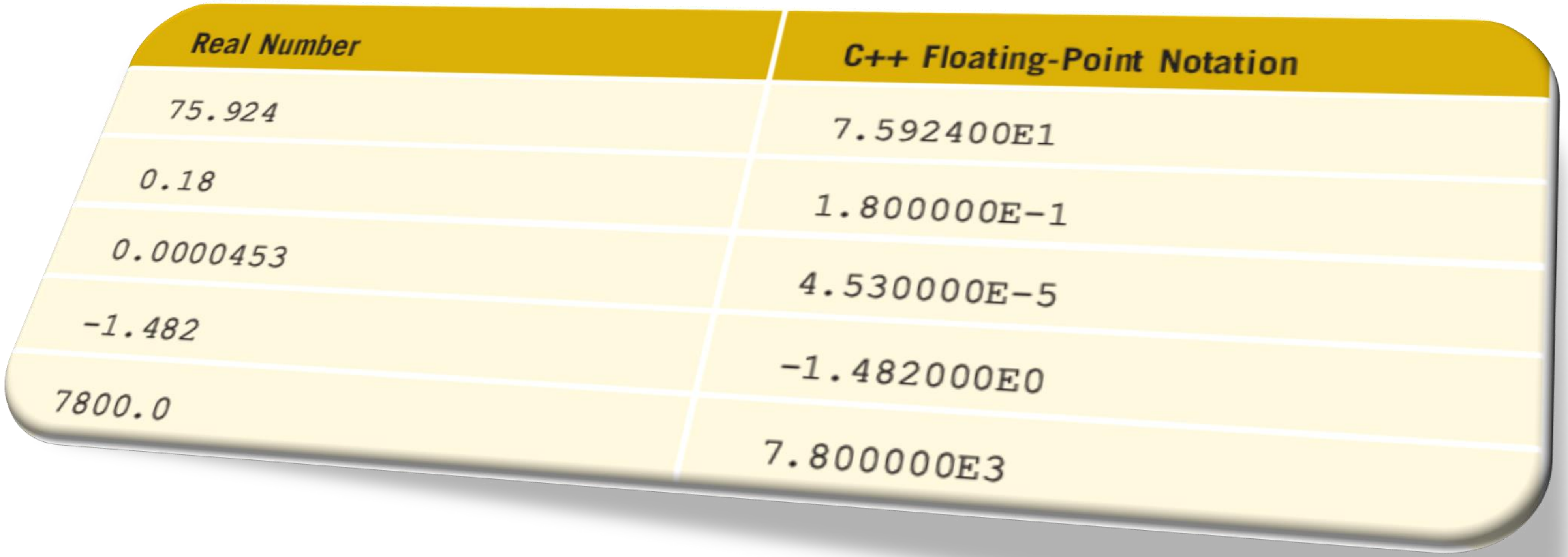
Each character is enclosed in single quotes

- 'A', 'a', '0', '*', '+', '$', '&'

A blank space is a character and is written ' ', with a space left between the single quotes

floating-point Data Type

- C++ uses scientific notation to represent real numbers (floating-point notation)



Real Number	C++ Floating-Point Notation
75.924	7.592400E1
0.18	1.800000E-1
0.0000453	4.530000E-5
-1.482	-1.482000E0
7800.0	7.800000E3

floating-point Data Type (cont.)

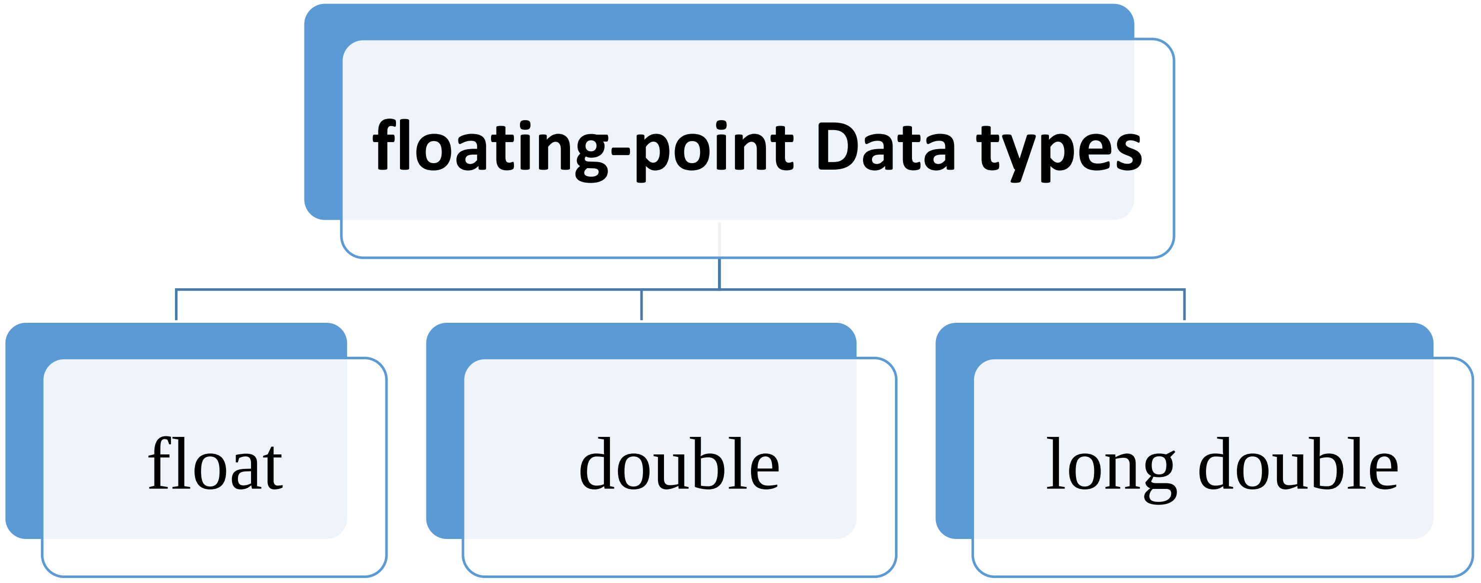
float: represents any real number

Range: -3.4E+38 to 3.4E+38 (four bytes)

double: represents any real number

Range: -1.7E+308 to 1.7E+308 (eight bytes)

On most newer compilers, data types double and long double are same



Arithmetic Operators and Operator Precedence

- C++ arithmetic operators:
    - + addition
    - - subtraction
    - * multiplication
    - / division
    - % modulus operator

- +, -, *, and / can be used with integral and floating-point data types

- Operators can be unary or binary

Order of Precedence

- All operations inside of () are evaluated first
- *, /, and % are at the same level of precedence and are evaluated next
- + and – have the same level of precedence and are evaluated last
- When operators are on the same level
  - Performed from left to right (associativity)
- 3 * 7 - 6 + 2 * 5 / 4 + 6 means
  - (((3 * 7) – 6) + ((2 * 5) / 4)) + 6

Allocating Memory with Constants and Variables

Named constant: memory location whose content can't change during execution

The syntax to declare a named constant is:

In C++, const is a reserved word

Consider the following C++ statements:

```
const double CONVERSION = 2.54;
const int NO_OF_STUDENTS = 20;
const char BLANK = ' ';
const double PAY_RATE = 15.75;
```

```
const dataType identifier = value;
```

Variable: memory location whose content may change during execution

The syntax to declare a named constant is:

```
double amountDue;
int counter;
char ch;
int x, y;
string name;
```

```
dataType identifier, identifier, . . .;
```

Assignment Statement

The assignment statement takes the form:

```
variable = expression;
```

Expression is evaluated and its value is assigned to the variable on the left side

In C++, = is called the assignment operator

```
int num1, num2;
double sale;
char first;
string str;
num1 = 4;
num2 = 4 * 5 - 11;
sale = 0.02 * 1000;
first = 'D';
str = "It is a sunny day.";
```

```
1.  num1 = 18;
2.  num1 = num1 + 27;
3.  num2 = num1;
4.  num3 = num2 / 5;
5.  num3 = num3 / 4;
```

Declaring & Initializing Variables

- Variables can be initialized when declared:

```
int first=13, second=10;
char ch=' ';
double x=12.6;
```

- All variables must be initialized before they are used

  But not necessarily during declaration

Thank you