Jiangxi University of Science and Technology

Chapter 15
A Brief Introduction to C++

Objectives

- Procedural Programming in C++
- Object-Oriented C++
- Common Programming and Compiler Errors

Introduction

THE C PROGRAMMING LANGUAGE

Table 15.1 Correspondence Between C and C++

Topic	C Statements, Operators, and Approach	C++ Statements, Operators, and Approach
Standard input and output header files	`#include <stdio.h>`	`#include <iostream>` `using namespace std;`
Standard input	`scanf()` function For example: `scanf("%d", &price);`	`cin` object For example: `cin >> price;`
Basic mathematical and assignment operators	`+, -, *, /, =`	`+, -, *, /, =`
Standard output	`printf()` function For example: `printf("Hello World");`	`cout` object For example: `cout << "Hello World";`

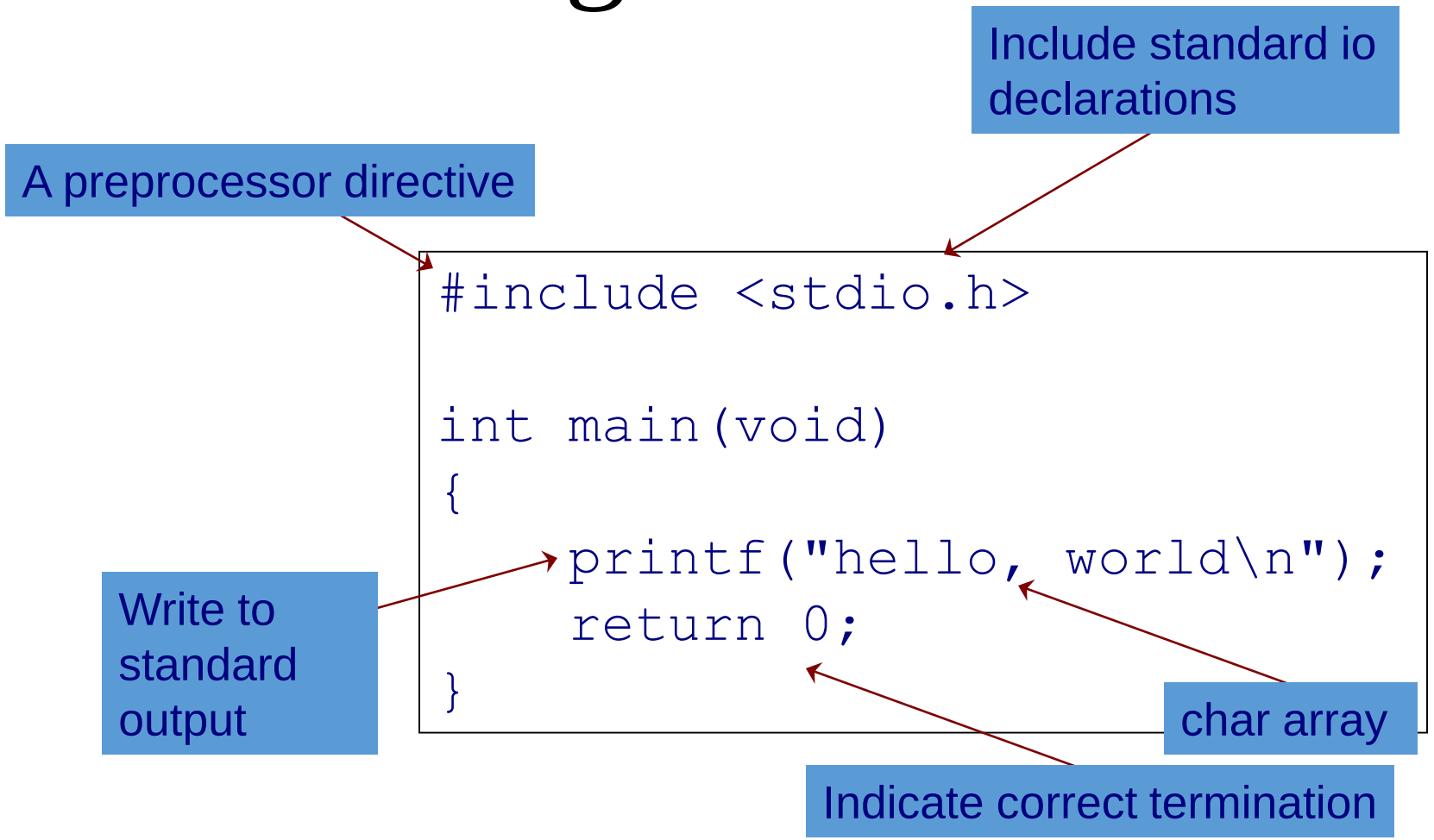
Formatted output	Control sequences
	For example:
	`printf("%5.2f", 2.756);`
		Format manipulators (requires the header file `iomanip`)
		For example:
		`cout << fixed << setw(5)`
		`<< setprecision(2)`
		`<< 2.756;`

Comment delimiters	`/* */`	`/* */` and `//`
New line designation	`'\n'`	`'\n'` and `endl`
Indirect addressing	Pointers	Pointers and references
Program design	Procedural	Object-oriented

Procedural Programming in C++

- C is a procedural language whose statements are used to produce C functions
  - Such C functions can also be constructed in C++
- C++ is sometimes referred to as “a better C”

My first C Program



Difference between C and C++	
C	**C++**
C was developed by Dennis Ritchie between 1969 and 1973 at AT&T Bell Labs.	C++ was developed by Bjarne Stroustrup in 1979 with C++'s predecessor "C with Classes".
When compared to C++, C is a subset of C++.	C++ is a superset of C. C++ can run most of C code while C cannot run C++ code.
C supports procedural programming paradigm for code development.	C++ supports both procedural and object oriented programming paradigms; therefore C++ is also called a hybrid language.
C does not support object oriented programming; therefore it has no support for polymorphism, encapsulation, and inheritance.	Being an object oriented programming language C++ supports polymorphism, encapsulation, and inheritance.
In C (because it is a procedural programming language), data and functions are separate and free entities.	In C++ (when it is used as object oriented programming language), data and functions are encapsulated together in form of an object. For creating objects class provides a blueprint of structure of the object.
In C, data are free entities and can be manipulated by outside code. This is because C does not support information hiding.	In C++, Encapsulation hides the data to ensure that data structures and operators are used as intended.

Difference between C and C++	
C	C++
C, being a procedural programming, it is a function driven language.	While, C++, being an object oriented programming, it is an object driven language.
C does not support function and operator overloading.	C++ supports both function and operator overloading.
C does not allow functions to be defined inside structures.	In C++, functions can be used inside a structure.
C does not have namespace feature.	C++ uses NAMESPACE which avoid name collisions. A namespace is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it. Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries. All identifiers at namespace scope are visible to one another without qualification. Identifiers outside the namespace can access the members by using the fully qualified name for each identifier.
C uses functions for input/output. For example scanf and printf.	C++ uses objects for input output. For example cin and cout.
C does not support reference variables.	C++ supports reference variables.

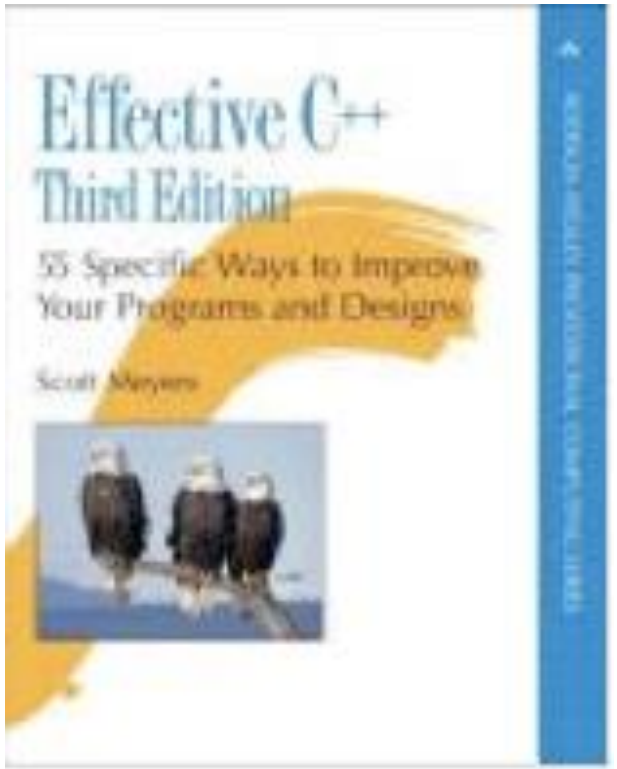
Difference between C and C++	
C	C++
C has no support for virtual and friend functions.	C++ supports virtual and friend functions.
C provides malloc() and calloc()functions for dynamic memory allocation, and free() for memory de-allocation.	C++ provides new operator for memory allocation and delete operator for memory de-allocation.
C does not provide direct support for error handling (also called exception handling)	C++ provides support for exception handling. Exceptions are used for "hard" errors that make the code incorrect.

C++ vs C

Most C programs are also C++ programs.

Nevertheless, good C++ programs usually do not resemble C:

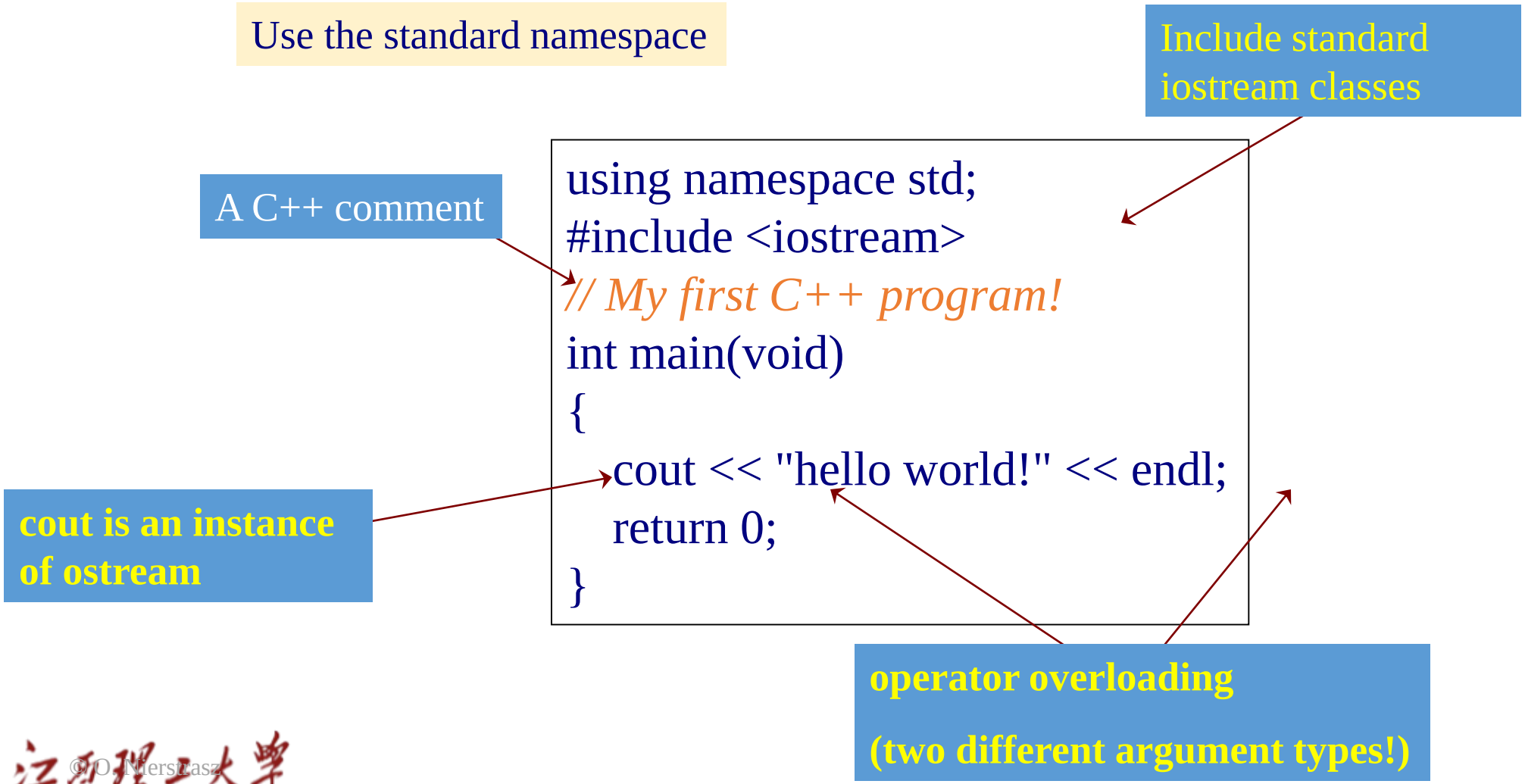
- avoid macros (use `inline`)
- avoid pointers (use references)
- avoid `malloc` and `free` (use `new` and `delete`)
- avoid arrays and `char*` (use `vectors` and `strings`) ...
- avoid `structs` (use classes)



C++ encourages a different style of programming:

- avoid procedural programming
  - *model your domain* with classes and templates

“Hello World” in C++



```
using namespace std;
#include <iostream>
// My first C++ program!
int main(void)
{
   cout << "hello world!" << endl;
   return 0;
}
```

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

Procedural Programming in C++ (continued)

Program 15.1

```
 1  /* this is a C procedural program */
 2  #include <stdio.h>
 3
 4
 5
 6  int main()
 7  {
 8    double price, total;
 9    int units;
10
11    printf("Enter the price: ");
12    scanf("%lf", &price);
13    printf("Enter the number sold: ");
14    scanf("%d", &units);
15
16    total = price * units;
17    printf("The total for the %d units is $%4.2f\n", units, total);
18
19    return 0;
20  }
```

Procedural Programming in C++ (continued)

Program 15.2

```
 1  // this is a C++ procedural program
 2  #include <iostream>
 3  #include <iomanip>    // needed for formatting
 4  using namespace std;
 5
 6  int main()
 7  {
 8    double price, total;
 9    int units;
10
11    cout << "Enter the price: ";
12    cin  >> price;
13    cout <<  "Enter the number sold: ";
14    cin  >> units;
15
16    total = price * units;
17    cout << "The total for the " << units << " units is $"
18         << fixed << setw(4) << setprecision(2)
19         << total << endl;
20
21    return 0;
22  }
23
```

Equivalent to C's newline sequence `'\n'` (which is also available in C++) followed by a `flush()` function call

Procedural Programming in C++ (continued)

C Program 15.1	C++ Program 15.2
`int main()`	`int main()`
`{`	`{`
`  double price, total;`	`  double price, total;`
`  int units;`	`  int units;`
`  printf("Enter the price: ");`	`  cout << "Enter the price: ";`
`  scanf("%lf", &price);`	`  cin  >> price;`
`  printf("Enter the number sold: ");`	`  cout <<  "Enter the number sold: ";`
`  scanf("%d", &units);`	`  cin  >> units;`
`  total = price * units;`	`  total = price * units;`

Object-Oriented C++

- Object-oriented languages make it easier to reuse code in a manner that significantly increases software productivity
- Three features are required for an OO language
  - Class construction
  - Inheritance
  - Polymorphism

Object-Oriented C++ (continued)

- **Class construction** is the ability to create programmer-defined data types
- In a programmer-defined data type, which is known as a **class**, the values defined for the data type can only be accessed and operated on by functions specified as part of the class

Object-Oriented C++ (continued)

- **Inheritance** is the ability to derive one class from another
  - Permits existing code, which has been thoroughly tested, to be reused without extensive retesting
- A derived class is a completely new data type
  - Incorporates all of the data values and operations of the original class
  - Adds new data and operations that create a different and expanded class
- The initial class is the **parent** or **base class**
- The derived class is the **child** or **subclass**

Object-Oriented C++ (continued)

- **Polymorphism** permits the same operation to invoke one set of results on data values of a base class and a different set of results on data values of a derived class
- Using polymorphism, existing operations on a base class can be left alone, without the need to retest and reverify them, while they are extended to a derived class
  - Only the extensions, rather than the complete base class, need to be tested and verified
- Standard Template Library (STL)

Common Programming Errors

- Misspelling the name of a function or C++ supplied name; for example, typing `cot` instead of `cout`
- Forgetting to close a string to be displayed by `cout` with a double quote symbol
- Forgetting to separate individual data items in a `cout` statement with the << symbol
- Forgetting to separate individual data items in a `cin` statement with the >> symbol

Common Programming Errors (continued)

- Omitting the semicolon at the end of each C++ statement
- Not including the C++ header lines

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

at the top of a C++ program

Compiler Errors

Error	Typical Unix-based Compiler Error Message	Typical Windows-based Compiler Error Message

Forgetting to include the statement:
`using namespace std;`
	`S) The name lookup for "cout" did not find a declaration.`
	`(S) The name lookup for "cin" did not find a declaration.`
	`(S) The name lookup for "fixed" did not find a declaration.`
		`:error: 'cin' : undeclared identifier`
		`:error: 'cout' : undeclared identifier`

Using `#include<stdio.h>` instead of `#include <iostream>`	`(S) The name lookup for "cout" did not find a declaration.` `(S) The name lookup for "cin" did not find a declaration.`	`:error: 'cin' : undeclared identifier` `:error: 'cout' : undeclared identifier`
Forgetting to use the "put to" symbol, <<, when placing a variable onto the output stream. For example: `cout << "sum = " sum;`	`(S) The text "sum" is unexpected.`	`:error: syntax error : missing ';' before identifier 'sum'`

Compiler Errors (continued)

Error	Typical Unix-based Compiler Error Message	Typical Windows-based Compiler Error Message

Incorrectly typing the << symbol as < when using cout. For example:
`cout < "Enter price: "`
	`(S) The call does not match any parameter list for "operator<".`
	
	`(I) "template <class _T1, class _T2> std:: operator<(const pair<_T1,_T2> &, const pair<_T1,_T2> &)" is not a viable candidate.`
	
	`(I) "template <class _RI> std:: operator<(const reverse_iterator<_ RI> &, const reverse_iterator<_ RI> &)" is not a viable candidate.`
		`The following warning, not an error, is triggered. :warning: '<' : operator has no effect; expected operator with side-effect`

Incorrectly typing the >> symbol as > when using `cin`.	`(S) The call does not match any parameter`	`:error: binary '>' : no operator found which takes a right-hand operand of type 'int' (or there is no acceptable conversion)`
Forgetting to use the "get from" symbol, >>, when using the input stream: For example: `cin units;`	`(S) The text "units" is unexpected.`	`:error: syntax error : missing ';' before identifier 'units'`

Summary

- C++ is an object-oriented language that also supports procedural programming
- Creating a C-like procedural program using C++ essentially involves changing syntax for the input and output statements
- All object-oriented languages, including C++, must provide the ability to create classes and provide for inheritance and polymorphism
- A class is a programmer-defined data type that includes specification of data values and operations that can be performed on these values

Summary (continued)

- Inheritance is the ability to create a new class by extending the definition of an existing class
- Polymorphism is the ability to have the same function perform different tasks depending on the class it is a member of
- Every variable in a C++ program must be declared as to the type of value it can store

Summary (continued)

- The general form of a statement using cout to display output from a C++ program is
  - cout << expression1 << . . . << expressionn;
- The general form of a statement using cin to accept data input from the keyboard is
  - cin >> variable1 >> . . . >> variablen;
- It is good programming practice to display a message prior to a cin statement, which alerts the user as to the type and number of data items to be entered

Reference

- https://www.codesdope.com/blog/article/int-main-vs-void-main-vs-int-mainvoid-in-c-c/