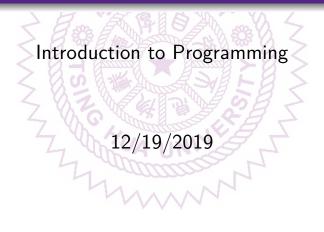
C++ and Object-Oriented Programming



and Object-Oriented Programming

Contents

Introduction to Programming

- Structure programming and object oriented programming
- C++ features
- Classes

12/19/2019

- Computer hardware is getting very powerful nowadays
 - Hardware cost has been driven down very significantly
 - Thanks for Moore's Law and talented electrical engineers
 - General purpose processors for most applications
- Application software development has seen significant progress as well, but to a lesser extent
 - Software cost dominates in many applications
 - Software plays the role of product differentiation as well
 - Software programs sustained for a long time
 - Software maintenance and upgrade are crucial in many applications

and Object-Oriented Programming

Software engineering progress

- High level languages
- Library for reuse

Introduction to Programming

- Structured programming
 - Readability and maintenance
 - Basic components are functions
 - To solve a specific problem
 - C was developed with this intention
- Object-Oriented Programming
 - Basic components are objects that model real world counterparts
 - Attributes and operations -data and functions -
 - Data hiding and implementation hiding
 - Users know how to use them but not how were they implemented
 - Reusability increases so is team work
 - Interface and implementation
 - C++ fits to this paradigm

12/19/2019

Object Oriented Programming

- Define object attributes and operations
 - Data and functions
- Objects such defined can be reused in other projects
- Detailed data storage or function implementation need not be known to users

and Object-Oriented Programming

- Only interface is known
- Clear responsibility
- Easier debugging4
- Enable team work^(*)
- Program still needs algorithmic description and implementation

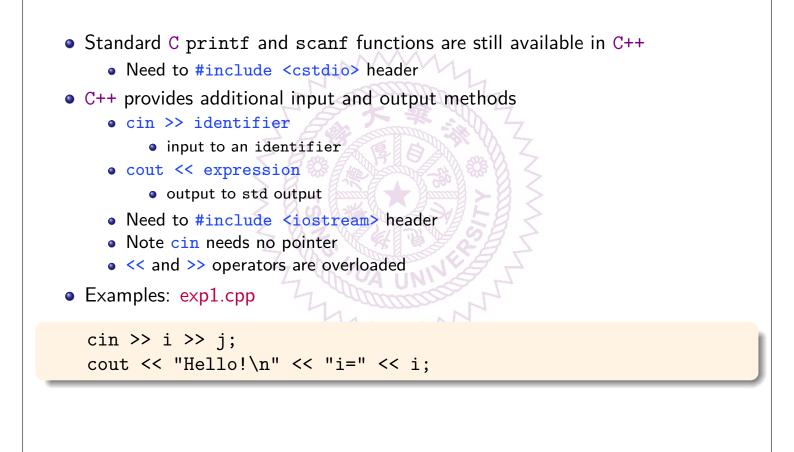
C++ Source File and Compiler

Introduction to Programming

- C++ source files have the file extension of .cpp instead of .c
 - C source files: lab1.c, lab2.c
 - C++ source files: lab1.cpp, lab2.cpp
- Header files have .h file extension
 - The same as C headers
- Compilation of C++ files
 - g++ lab1.cpp
 - Produce a.out program
 - g++ -o lab1 lab1.cpp
 - Produce lab1 program
 - g++ -c lab1.c
 - Produce lab1.0 file

12/19/2019

C++ Input and Output



```
Namespaces
```

Introduction to Programming

- Two properties of a variable: storage duration and scope
- For large programs, it is not difficult to see that we may need many variables and functions

and Object-Oriented Programming

- Name crashes can happen, especially in a large team
- C++ provides a way to manage variable scopes namespace
 - variable in a name space can be referenced by :: operator
 - using preprocessor can simplify accessing to these variables
- Examples: exp2.cpp, exp3.cpp

```
namespace mySpace {
    int i, j;
    double mysqrt(double);
}
```

12/19/2019

Reference Parameters and Variables

- In addition to pass-by-value and pass-by-pointer schemes, C++ provides additional pass-by-reference scheme
- reference parameters of a function will not be copied and they occupy the same memory locations as the referenced variables
 - Value of the referenced variable can be changed
 - Function calls are more efficient
- reference variable within a function also serves as a alias to the referenced variable
 - Same memory location and same value
- The value of a reference variable needs no * operator
- Examples: exp4.cpp

```
void func(int i, int &j) ; // j passed by reference
int i, &j = i; // j is an alias to i
```

Introduction to Programming

and Object-Oriented Programming

12/19/2019 9/25

Functions with Default Parameters

- In C++ functions can have default arguments
- Default value is declared in function definition
- If a parameter is not provided by a function call, then the default value is taken for the parameter
- Only trailing parameters can be default parameters
- Example: exp5.cpp

```
void f(int a = 1, int b = 1, int c = 1) {
    // ...
}
// function calls
    f(i, j, k);
    f(i, j);
    f(i);
    f();
```

C++ and Classes

- The aim of the C++ class concept is to provide the programmer with a tool for creating new types that can be used as conveniently as the built-in types.
- A type is a concrete representation of a concept.
 - For example, float with its operations +, -, *, etc., provides a concrete approximation of the mathematical concept of a real number.
- A class is a user-defined type.
- A program that provides types that closely match the concepts of the application tends to be easier to understand and easier to modify than a program that does not.
- Example: exp6.cpp

C++ and Classes

Introduction to Programming

and Object-Oriented Programming

- A well-chosen set of user-defined types makes a program more concise.
 - It also enables the compiler to detect illegal uses of objects that would otherwise remain undetected until the program is thoroughly tested.
- The fundamental idea in defining a new type is to separate the incidental details of the implementation from the properties essential to the correct use of it.
- Such a separation is best expressed by channeling all uses of the data structure and internal housekeeping routines through a specific interface.

12/19/2019

Classes and Object-Oriented Programming

In C++ classes are the basic components of a program

- Data members for attributes
- Function members for operations
- Example:

```
class Complex {
   public:
        Complex(double, double);
        void printComplex(void);
        double getReal(void);
   private:
        double x, y;
}; // need ;
```

and Object-Oriented Programming

// constructor

12/19/2019 13/25

Class Definition

Introduction to Programming

- Data members
 - Similar to struct's definition (struct itself is also a class)
 - Any type: basic or user-defined, including class

• Function members

- Function declarations should be included
- Function to operate on this class
- public members (data or functions) can be accessed by any functions
- private members (data or functions) can be accessed by member functions only
 - Non-member functions accessing private members is a compilation error
 - Private functions: utility functions

Access Control

- Private members can only be accessed by member functions
- Public members can be accessed by any functions
- A struct is a class with public members only
- Benefits of access control:
 - Easier debugging, localization is done before the program is even run

and Object-Oriented Programming

- Change of the class needs to recompile the member functions only
- Serve as documentation as well

Class Member Function Definitions

- Member functions' definition can be done within class declaration
- Function definition can also be done outside of class declaration
 - Need to prefix with classname and scope resolution operator ::
- Member functions are invoked by
 - object.memberfunction()
 - objectPtr->memberfunction()
- constructor

Introduction to Programming

- Same name as class and no return type (or value)
- destructor
 - ullet \sim className
 - Called explicitly or when variables are released
 - Destructors clean up and release resources
 - Destructors are called, for example, when automatic variables go out of scope

12/19/2019

Class and Memory Allocation

- class similar to struct take actual memory space to store data
 - data member

Introduction to Programming

- Member functions are not duplicated, only one copy exists
- Static data also has one copy only
- Similar to struct, class object can be assigned using =
 - Member-wise copying
 - Each member is copied from **rvalue** object to **lvalue** object

Class Header and Implementation Files

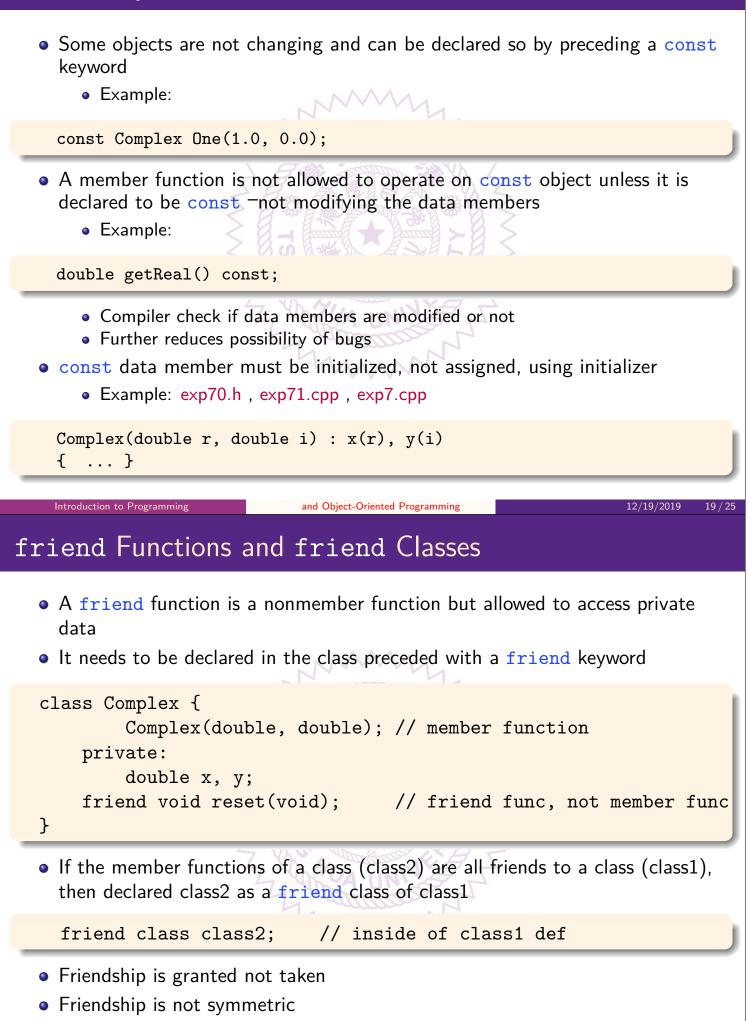
• Class definition and implementation can all be located in the same file as the main function

and Object-Oriented Programming

- In practice, for each object, a header file .h and an implementation file
 .cpp are usually created
 - Interface .h and implementation .cpp are separated
 - Class users need to know the interface but not the actual implementation
- Implementation source file needs not be provided.
 - Object file .o is sufficient to create final program
 - Hiding implementation from users
- With the header and object files, the class can be reused by other programs
- Limiting data member access to the member functions reduces possibility of program bugs

12/19/2019

const Objects and const Member Functions



Friendship is not transitive

this Pointer and Member Functions

- Compiler creates an implicit pointer, this, that points to the object
- All data member can be accessed either directly or through this pointer
- Sometimes we want to return a reference to the updated object so the operations can be chained.

```
class_type & class::func() {
    // ...
    return *this;
```

- }
- ***this** refers to the object of which the function is invoked.
 - this is a pointer to the object
- For const member function, this is

const X* this

• Example: exp80.h, exp81.cpp, exp8.cpp

static Members

Introduction to Programming

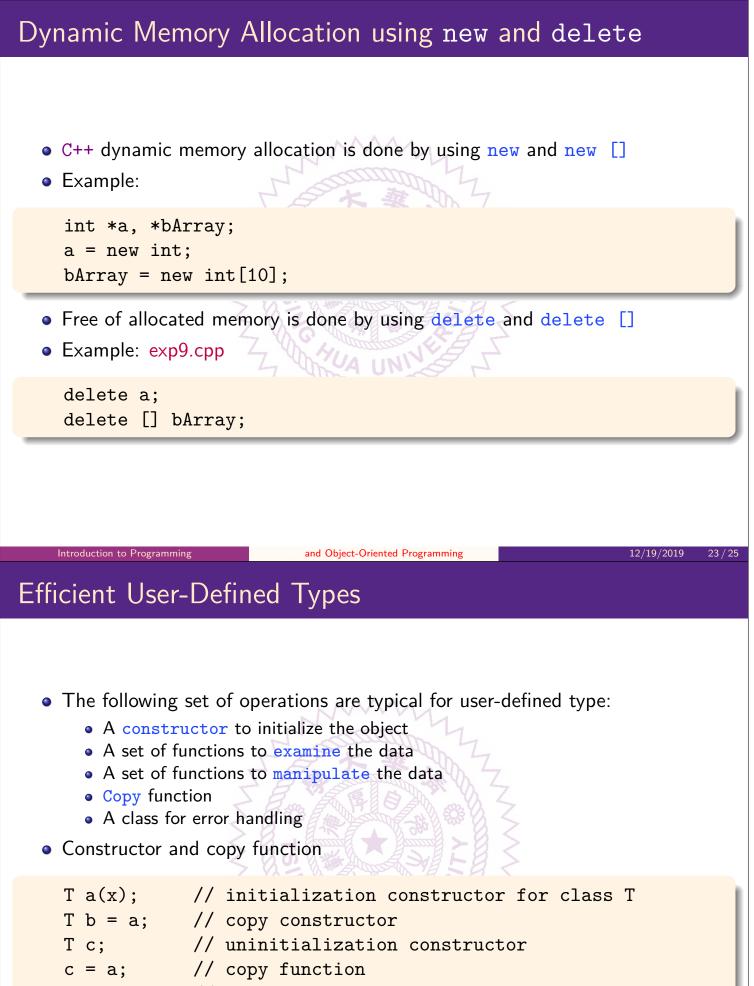
• static member: a variable is part of a class but not part of an object

and Object-Oriented Programming

- There is only one copy of static variable, not one for each object
- Static member functions access to the members of a class not object
- Static members can be accessed using class name as the qualifier
- Static data and functions must be defined somewhere (data initialized)

```
class T {
    static int accessCount;
    static void incAccessCount(void) { accessCount++ };
}
// ...
int T::accessCount = 0;
T::incAccessCount();
```

12/19/2019



// default to memberwise copying

Summary

- Software development and OO programming
- C++ source files and compilation
- C++ input and output
- Namespaces
- Reference parameters and variables
- new and delete
- Classes
- const object and member functions
- friend functions and friend classes
- this pointer and member functions
- Static members
- Class operations

Introduction to Programming

and Object-Oriented Programming

12/19/2019 25/25