**Classes**

- A class is a collection of a fixed number of components (like struct!)
- The components of a class are called **members** of the class
- The general syntax of defining a class is:
  ```cpp
  class classIdentifier
  {
    classMemberList
  };
  ```
- A **member** of a class can either be a variable (a **data member** to store some data) or a **function member**.

**Classes members**

- If a member of a class is a variable, it is declared just like any other variable.
- However in the definition of the class, you **cannot** initialize a variable when you declare it.
- If a member of a class is a function, typically the **function prototype** is used to define that member.

---

**Introduction to Classes and Data Abstraction**

- What are classes, members and objects
- private, protected, and public members of a class
- how classes are implemented
- constructors and destructors
- Abstract data type (ADT)
- how classes are used to implement ADT
- information hiding
- how information hiding is implemented in C++
Classes

- If a member of a class is a function,
  - it can (directly) access any member of the class—data members and function members
  - when you write the definition of the member function, you can directly access any data member of the class without passing it as a parameter
  - (but you must declare an identifier before you can use it)

Class is a reserved word and it only defines a data type, no memory is allocated
- The semicolon after the right brace is part of the syntax

Class member types

- The members of a class are classified into three categories:
  - private
  - public
  - protected
- Public and private members of a class:
  - By default, all members of a class are private and cannot access it outside the class
  - public member is accessible outside the class
  - private, protected, and public are reserved words

Variable (Object) Declaration

- Once a class is defined, you can declare variables of that type: a class object or class instance
- The syntax for declaring a class object is the same as that for declaring any other variable
Class Example

class Ratio
{
  public:
    void setRatio(int, int); // it's a function prototype only;
    int getNum() const;
    int getDen();
    double printValue() const;
    // const means during the execution of this function
    // the values of data members cannot be changed
    void invert();
    void printRatio() const;

  private:
    int num, den;
    // "int num =1" is not allowed
    // as num is now a data member of a class
}; // end of Ratio

Variable (Object) Declaration

Once a class is defined, you can declare variables of that type

In C++ terminology, a class variable is called a class object or class instance

The syntax for declaring a class object is the same as that for declaring any other variable

For example
  Ratio r, x;

Accessing Class Members

Once an object is declared, it can access the public members of a class

The general syntax to access the member of a class is:
  classVariableName.memberName

In C++, the dot, . (period), is an operator called the member access operator

Built-in Operations on Classes

Most of C++’s built-in operations do not apply to classes

Arithmetic operators cannot be used to perform arithmetic operations on class objects (unless they are overloaded).

Also, you cannot use relational operators to compare two class objects for equality

The two built-in operations that are valid for class objects are member access (.) and assignment (=)

  x = r;
Class Scope

- A class object has the same scope as other variables
- A member of a class is local to the class
- We access a class member outside the class by using the class object name and the member access operator (.)

Class

- Compare the difference

Constructors

- Constructors
  - We can guarantee the initialization of the data members of a class by using constructors
  - There are two types of constructors:
    - with parameters
    - without parameters
  - The constructor without parameters is called the default constructor
  - A constructor that has no parameters, or has all default parameters, is called the default constructor

Constructors

- 1. The name of a constructor is the same as the name of the class
- 2. A constructor:
  - even though it is a function, has no type
  - it is neither a value-returning function nor a void function
- 3. A class can have more than one constructor, however, all constructors of a class have the same name
- 4. If a class has more than one constructor, they must have different sets of parameters
Constructors

5. Constructors:
   - are automatically executed when a class variable enters its scope
   - since they have no types, they cannot be called like other functions

6. Which constructor executes depends on the type of values passed to the class variable when the class variable is declared

Invoking a Constructor

A constructor is automatically executed when a class variable is declared
A class might have more than one constructor, including the default constructor

Invoking the Default Constructor

The syntax to invoke the default constructor is:
```
className classVariableName;
```

For example:
```
class Ratio
{
    public:
    Ratio() { num = 0; den = 1; }
    . . .
}
Ratio yourRatio;
```
declares yourRatio to be a variable of the type Ratio
In this case, the default constructor is executed and the data members of yourRatio will be initialized to 0 and 1.
Constructor with Parameters

- The syntax to invoke a constructor with parameters is:
  
  ```
  className  classVariableName(argument1, argument2, ...);
  ```

- where argument1, argument2, etc. is either a variable or an expression

## Constructor with Parameters

1. The number of arguments and their type should match the formal parameters (in the order given) of one of the constructors
2. If the type of the arguments do not match the formal parameters of any constructor (in the order given), C++ will use type conversion and look for the best match

For example,

- an integer value might be converted to a floating-point value with zero decimal part
- an ambiguity will result in a compile-time error

Destructors

- Like constructors, destructors are also functions
- The name of a destructor is the character '~' followed by the name of the class
- The name of the destructor for the class Ratio is:
  
  ```
  ~Ratio();
  ```

- A class can have only one destructor and it has no parameters
- The destructor is automatically executed when the class object goes out of scope

Default Parameters

- A constructor can has default parameters
- A constructor that has *no parameters*, or has *all default parameters*, is called the default constructor

Example:

```
class Ratio
{
  public:
    Ratio(int n = 0, int d = 1) { num = n; den = d; }
  private:
    int num, den;
};
```
Functions and Classes

- Class objects can be passed as parameters to functions and returned as function values
- As parameters to functions, classes can be passed either by value or by reference
- If a class object is passed by value, the contents of the data members of the actual parameter are copied into the corresponding data members of the formal parameter

Reference Parameters & Variables

- As a parameter, a class object can be passed by value
- If a variable is passed by value, the corresponding formal parameter receives a copy of the data of the variable
- This operation might require, in addition to a large amount of storage space, a considerable amount of computer time to copy the value of the actual parameter into the formal parameter
- Example
  ```cpp
  bool compareRatio(Ratio r1, Ratio r2);
  ```

Reference Parameters & Variables

- If a variable is passed by reference, the formal parameter receives only the address of the actual parameter
- Therefore, an efficient way to pass a variable as a parameter is by reference
Reference Parameters & Variables

- If a variable is passed by reference, then when the formal parameter changes, the actual parameter also changes.
- You can pass a variable by reference and still prevent the function from changing its value, by using the keyword const in the formal parameter declaration.
- Example:
  ```cpp
  bool compareEqual(Ratio& otherRatio);
  ```

Class Exercise

- Define a class for student, which has the following data members:
  - string ID
  - string lastname
  - string firstname
  - double score1, score2, score3
  - double score4, score5, average

A Struct Versus a Class

- By default the members of a struct are public.
- By default the members of a class are private.
- The member access specifier private can be used in a struct to make a member private.
- Both C++ classes and structs have the same capabilities.

A Struct Versus a Class

- Most programmers restrict their use of structures to adhere to their C-like structure form.
- If all of the data members of a class are public and the class has no member functions, typically a struct is used to group these members.
Abstract Data Types

- Abstract Data Type (ADT) - a data type that specifies the logical properties without the implementation details
- An ADT has three things associated with it
  - The name of the ADT, called type name
  - The set of values belonging to the ADT, called domain
  - The set of operations on the data

Abstract Data Types

- The data type clockType has three data members and the following basic operations:
  1. Set the time
  2. Return the time
  3. Print the time
  4. Increment the time by one second
  5. Increment the time by one minute
  6. Increment the time by one hour
  7. Compare the two times to see whether they are equal

Information Hiding

- The header file has an extension h
- The implementation file has an extension cpp
- The implementation file must include the header file via the include statement
- In an include statement, user-defined header files are enclosed in double quotes while system-provided header files are enclosed between angular brackets

Files

- Visual C++, C++ Builder, and CodeWarrior put the editor, compiler, and linker all into one package
- With one command, the program is compiled and linked with the other necessary files
- These systems also manage multiple file programs in the form of a project
- A project consists of several files, called the project files
Files

- These systems usually have a command, called build, rebuild, or make
- When the build, rebuild, or make command is applied to a project, the system automatically compiles and links all files required to create the executable code
- When one or more files in the project change, you can use these commands to recompile and relink the files

Predefined Classes

- char ch;
- cin.get(ch);
- Other examples:
  - cin.putback(ch);
  - ch = cin.peek();

Predefined Classes

- string str1 = "abcdefg", str2="cd";
- str1.length() //7
- str1.size() //7
- str1.find("bc") //1, index of 'b' in str1
- str1.substr(1,3) // "bcd"
- str1.swap(str2);
  // str1 = “cd”, str2= “abcdefg”