INTRODUCTION TO CLASSES AND OBJECTS

ABSTRACT DATA TYPE

- Book Definition:
- An abstract data type is a data type that specifies the values the data type can hold and the operations that can be done on them without the details of how the data type is implemented.

ABSTRACTION

- Abstraction: A general model of something
  - Includes general characteristics of something without specific details
- Consider a car
  - Most of you know how to drive a car but do we need to know everything that is happening under the hood
- Details of internal components, organization, and operation of cars, microwaves, blue-ray players, etc. are kept separate from the description of how they work
- This is Abstraction!!

ABSTRACTION CONTINUED

- You have already experienced abstraction
- printf – a function you have already been using – we can find information about it
  - http://man7.org/linux/man-pages/man3/printf.3.html
  - https://en.cppreference.com/w/cpp/io/c/fprintf
- Do either of these tell us how printf is implemented? NO!!!
  - We know what it is suppose to do
  - We know how to use it
  - We know if we use it correctly it will do what we want it to do
  - We DO NOT know nor care how it is implemented

  This is abstraction!!

ABSTRACTION CONTINUED

- Abstraction applies to data types as well
  - With data types, we really only need two things
    - What Type
    - What are the operations that can be performed on them
      - ex: mod can only be used with an int not a float or double
      - ex: char vs string

PROGRAMMING METHODS

- There are many programming methods
- Two common programming methods used today
  - Procedural programming
    - Driven by a series of computational steps carried out by calls to functions
    - Step by step algorithms
    - Examples of procedural languages
      - C, Fortran, Pascal, BASIC, Ada, Assembly
    - Worked well for programmers for years
      - Programs have gotten larger and more complex which is much harder to manage with procedural style programming
      - Harder to debug and make needed changes

PROGRAMMING METHODS

- Object-oriented programming
  - Large and complex programs influenced the development and shift from procedural to OOP
  - OOP is centered on creating and using objects

OBJECT-ORIENTED PROGRAMMING

- What is an object?
  - BOOK DEFINITION – a software entity that combines both data and the procedures that work with it in a single unit.
- What does an object consist of?
  - Attributes – member variables
  - Member functions (sometimes called methods)
- The bundling of an object’s data and procedure together is called **encapsulation**

EXAMPLE

- Think about a circle, square, and triangle
- What member variables will you need for the shape you have?
- What functions do you think you might need for your shape?
- The answer to these questions are used to describe the
  - Circle object
  - Square object
  - Triangle object
- We bundle the member variables and functions together to make an object **class**

DATA HIDING

- An object provides the ability to hide its data from code outside the object
- Only an object of a class can directly access and make direct changes to its data
- Outside code can only access data of an object through the objects member functions.

DATA HIDING

- Why is information hiding a good thing?
  - Protects the objects data from accidental or intentional corruption
  - Code not part of the object’s class does not need to know how, when, or why something is changed – sorta like I don’t need to know exactly how my car works I only need to know how to drive it properly
- Data hiding and information hiding is an important concepts in software development

INTRODUCTION TO CLASSES

- Before we can use an object you must first create the class – How do we do this?

GENERAL FORMAT OF A CLASS DECLARATION

```
class ClassName
{
    Declare member variables
    Declare member functions
};  semicolon is required
```

CIRCLE CLASS DECLARATION

```
class Circle
{
   private:
      double radius;
   public:
      void setRadius(double r);
      double getArea();
};
```

Access Specifiers

Setter (mutator)

Getter (accessor)

Public member variable can be accessed by functions outside the class, and a public member function can be called by functions outside the class.

A private member variable can only be accessed by a function that is a member of the class.

Default access type is private
The order of the access specifier does not matter.

CREATING AND USING OBJECTS

- A class declaration is like a blueprint
- A blueprint provides a description of a house
- When you use the blueprint to build a house you are constructing an instant of the house (class)
- Can construct as many of the houses that you want

prog7_1.c

CLASS MEMBER FUNCTIONS

- Defined similarly to C-style functions
- Can be implemented either inside or outside the class declaration
- Include:
  - return type
  - function name
  - parameter list (could be empty)
  - the statements that carry out the actions of the function
  - all contained with in {} brackets

INLINE FUNCTION

- When you implement a function within the class declaration
- Can also use the inline key word when declaring a function
  - What
    - Inline function is an optimization technique used by the compilers to reduce the execution time
    - When a function is declared as inline the compiler will go to where that function is being called and replace the call with the entire function
  - Why/why not
    - reduces execution time -- take a little longer to compile
    - increases size of executable
- Only use inline functions if the function is small (just a couple lines) Usually it is used when the function does some mathematical calculation.
- If the function is to large the compiler may ignore the request to

INLINE EXAMPLE

```
Class A
{
    public:
    int add(int a, int b) {
    return (a + b);
    }
};
```

```
Class A
{
    public:
    int add(int a, int b);
};
inline int A::add(int a, int b) {
 return (a + b);
 }
```

SCOPE RESOLUTION OPERATOR ::

- It is not common to define a function within the class declaration
  - This defeats the purpose of information hiding or encapsulation
  - Functions should be defined outside of the class declaration
  - Must use the scope resolution operator :: (double colon)
    - double Circle::getArea()
    - NOTE: THE CLASS NAME AND SCOPE RESOLUTION OPERATOR ARE AN EXTENSION OF THE FUNCTION NAME. WHEN A FUNCTION IS DEFINED OUTSIDE THE CLASS DELARATION, THESE MUST BE PRESENT AND MUST BE LOCATED IMMEDIATELY BEFORE THE FUNCTION NAME

  prog7.3.cpp

IN-CLASS EXERCISE

Let’s write a simple class called Car

CONSTRUCTORS

- A constructor is a member function that is automatically called when a class object is created
- If you do not provide a construct, C++ will provide one
- Have no return value
- The C++ default constructor basically initializes all member variables to 0
- In your constructor you can provide code (instructions) other than just giving values to the variables – when you create an instance of the class your constructor and any other instructions you provide will be executed

prog7_4.cpp

CONSTRUCTORS

- The name of the constructor has to match the name of the class

  Ex. Consider the Circle class – the constructor for the Circle class would be called Circle()

- Constructor with default values - use date class to demonstrate.
- If you write the constructor in the class declaration it will be an inline function
- You can also write the constructor outside the class declaration, as follows: Circle::Circle() { ……..}

prog7_2.cpp

prog7_6.cpp

DESTRUCTORS

- A member function that is automatically called when an object is destroyed
- They are in public space
- Preceded by ~ (tilda)

prog7_7.cpp

PRIVATE MEMBER FUNCTIONS

- Concept: Private member functions may only be called from a function that is a member of the same class.
- So far all our functions have been public – meaning they can be called by code in programs outside the class
- Often you may have a function that should not be called outside the class – you will make these functions private
- prog7_8.cpp

PASSING OBJECTS TO FUNCTIONS

- We can pass primitive data types to a function.
- We can pass structs to a function.
- It should not surprise you that you can pass an object to a function.

Prog7_9.cpp

RETURNING OBJECTS FROM FUNCTIONS

- We can also return an object from a function.

prog7_10.cpp

COMPOSITION

- It is possible for a class to have a member variable that is an instance of another class.
- This is called composition.

prog7_11.cpp

C++ STRUCTURES

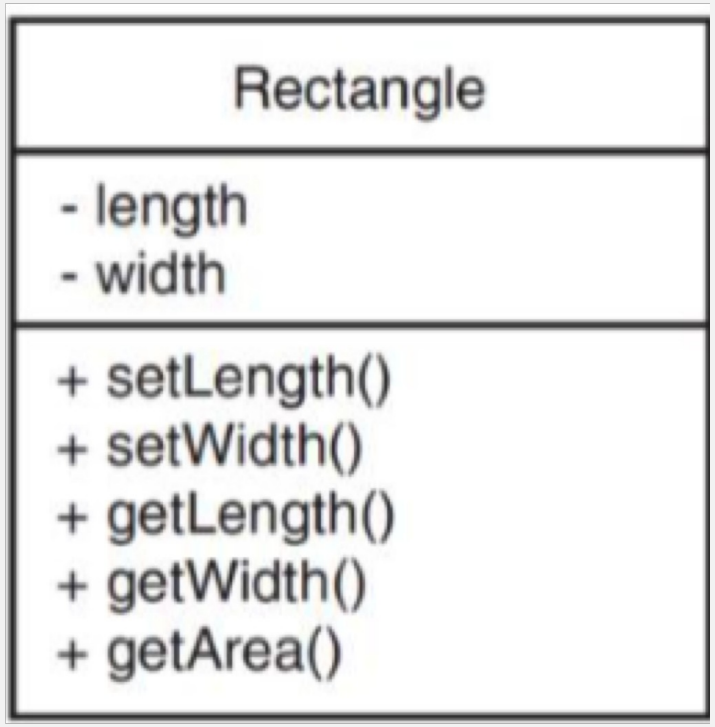
- C++ Structures do not use typedef
- They can also have constructors
- They can also have an instance of another structure as an element

    prog7_14.cpp

    prog7_15.cpp

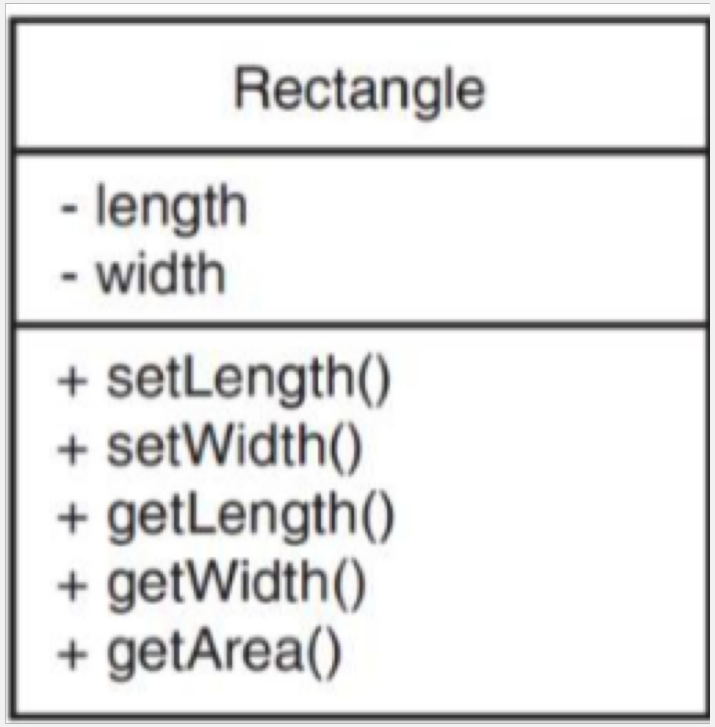
UML (UNIFIED MODELING LANGUAGE)

- Standard diagrams for depicting classes.
- • Class name, attributes (data members) and
- methods are specified.
- • Private members have a prefix of “-” and
- public members have a prefix of “+”.



UML (UNIFIED MODELING LANGUAGE)

- Standard diagrams for depicting classes.
- • Class name, attributes (data members) and
- methods are specified.
- • Private members have a prefix of “-” and
- public members have a prefix of “+”.



UML(UNIFIED MODELING LANGUAGE

- If more details are desired, the data types are
- specified using a colon.
- • Also, data types of method parameters and
- return value are specified using a colon.