

CSC 211: Computer Programming

Introduction to Classes

Michael Conti

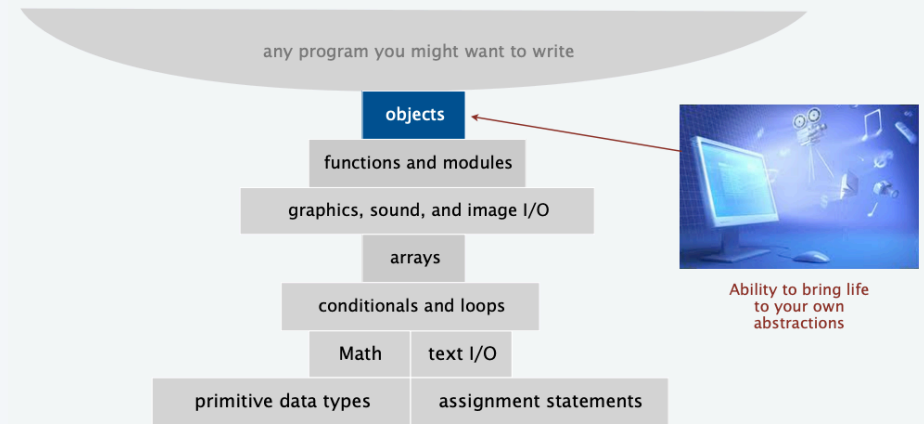
Department of Computer Science and Statistics
University of Rhode Island

Spring 2022



Original design and development by Dr. Marco Alvarez

Basic building blocks



<https://introc.cs.princeton.edu/java/lectures/keynote/CS.9.CreatingDTs.pdf>

2

Classes

- In object-oriented programming (OOP), a **class** is an extensible “datatype” for creating **objects** (“variables”)
- Examples of classes you have already used
 - ✓ `std::string`, `std::istream`, `std::ostream`
- A class can define **member variables** and behavior (called **member functions** or **methods**)
- When an object is created, the resulting object is also called an **instance of the class**

3

C++ Classes

- A class in C++ is a user-defined type declared with the keyword **class**
- A class can define data members and member functions
 - ✓ three levels of access: **private** (default), **protected**, or **public**
- **Private members** are not accessible outside the class
 - ✓ only through methods of the class
- **Public members** form an interface to the class and are accessible outside the class

4

Class declaration

- Similar to structs, however level of access must be specified

```
class MyClass {  
    public:  
        int myNum;  
        string myString;  
};
```

5

Declaration and dot operator

```
#include <iostream>  
#include <string>  
  
class MyClass {  
    // access specifier  
    public:  
        // data members  
        int myNum;  
        std::string myString;  
};  
  
int main() {  
    // creating an object  
    MyClass object;  
  
    // using the dot operator  
    object.myNum = 10;  
    object.myString = "My Message";  
    std::cout << object.myNum << std::endl;  
    std::cout << object.myString << std::endl;  
  
    return 0;  
}
```

6

Methods (member functions)

- Methods must be declared inside the class
 - definition of methods must identify the class they belong to
 - :: is the **scope resolution operator**

```
class Date {  
    public:  
        int month;  
        int year;  
        int day;  
  
        void print();  
};  
  
void Date::print() {  
    std::cout << month << '-' <<  
        << day << '-' <<  
        << year << std::endl;  
}  
  
// declaration  
// definition
```

7

Example

```
#include <iostream>  
  
class Date {  
    public:  
        int month;  
        int year;  
        int day;  
  
        void print();  
};  
  
void Date::print() {  
    std::cout << month << '-' <<  
        << day << '-' <<  
        << year << std::endl;  
}  
  
int main() {  
    Date today;  
  
    today.day = 11;  
    today.month = 16;  
    today.year = 2021;  
    today.print();  
  
    return 0;  
}
```

Must include the object

8

Improving the class declaration

- Making changes to the internal representation of `Date` requires changes to the entire program
- A better declaration of the class `Date` would allow for changes to the class without requiring changes to the program(s) that use `Date`

don't allow the program to directly reference member variables

9

```
#include <iostream>

class Date {
public:
    int month;
    int year;
    int day;

    void set(int m, int d, int y);
    void print();
};
```

10

```
void Date::print() {
    std::cout << month << '-'
               << day << '-'
               << year << std::endl;
}

void Date::set(int m, int d, int y) {
    month = m;
    day = d;
    year = y;
}
```

11

```
int main() {
    Date today;

    today.set(11, 16, 2021);
    today.print();

    return 0;
}
```

12

Encapsulation

- **Encapsulation** is one of the most fundamental concepts of OOP
- In OOP, encapsulation is used to hide the values or state of a structured data object inside a class. It is implemented as a:
 - ✓ language construct that **facilitates the bundling of data with the methods** (or other functions) operating on that data
 - ✓ language mechanism for **restricting direct access** to some of the object's components

[https://en.wikipedia.org/wiki/Encapsulation_\(computer_programming\)](https://en.wikipedia.org/wiki/Encapsulation_(computer_programming))

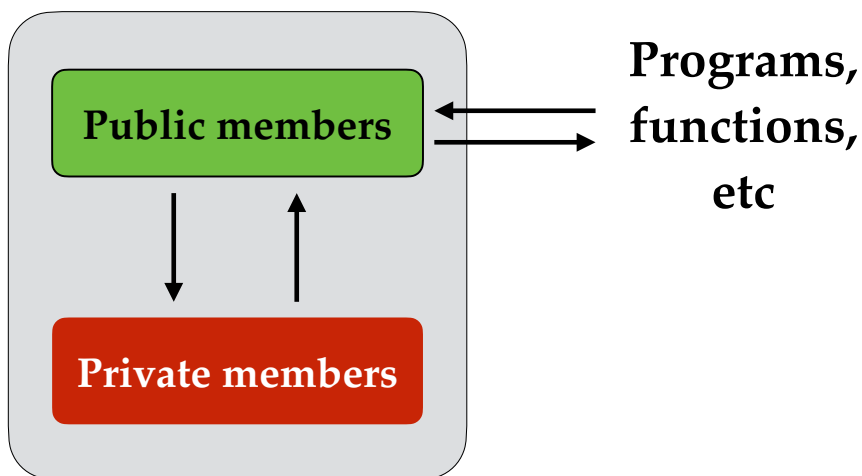
13

Public vs Private

- C++ helps us restrict the program from directly referencing member variables
- Private members of a class can **only be referenced** within member functions
 - ✓ otherwise, the compiler gives an error message
- The keyword **private** identifies the members of a class that can be accessed only by member functions
- The keyword **public** identifies the members of a class that can be accessed from outside the class

14

Object



15

```
class Date {  
    private:  
        int month;  
        int year;  
        int day;  
  
    public:  
        void set(int m, int d, int y);  
        void print();  
};
```

16

```
// https://www.partow.net/programming/bitmap/index.html
#include "bitmap_image.hpp"

int main() {
    bitmap_image image(200,200);

    // set background to orange
    image.set_all_channels(255, 150, 50);

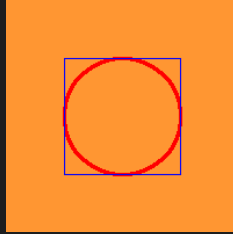
    image_drawer draw(image);

    draw.pen_width(3);
    draw.pen_color(255, 0, 0);
    draw.circle(image.width() / 2, image.height() / 2, 50);

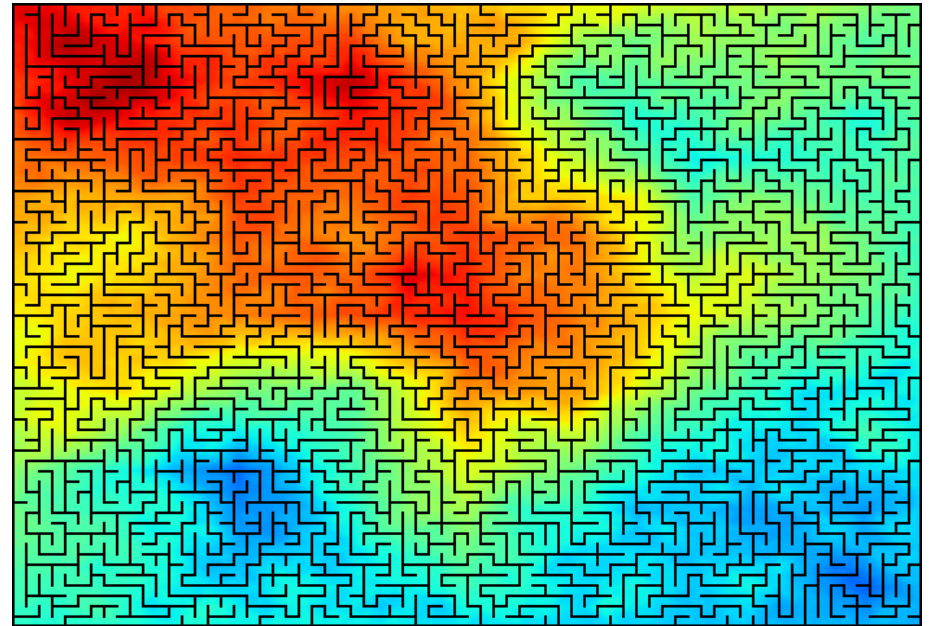
    draw.pen_width(1);
    draw.pen_color(0, 0, 255);
    draw.rectangle(50, 50, 150, 150);

    image.save_image("output.bmp");

    return 0;
}
```



17



<https://www.partow.net/programming/bitmap/index.html>

18

Assignment operator

- Objects and structures can be assigned values using the = operator

```
int main() {
    Date today;
    Date due;

    today.set(11, 16, 2021);
    due = today;
    today.print();
    due.print();

    return 0;
}
```

19

Exercise

- Implement the following public methods for the class date
 - ✓ add_years, which adds a number of years to the current date
 - ✓ add_months, which adds a number of months to the current date
 - ✓ add_days, which adds a number of days to the current date

When accessing data members or methods from other methods inside the class, the object name is not required

20