CSC 211: Object Oriented Programming

Header Files and Constructors

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Spring 2020

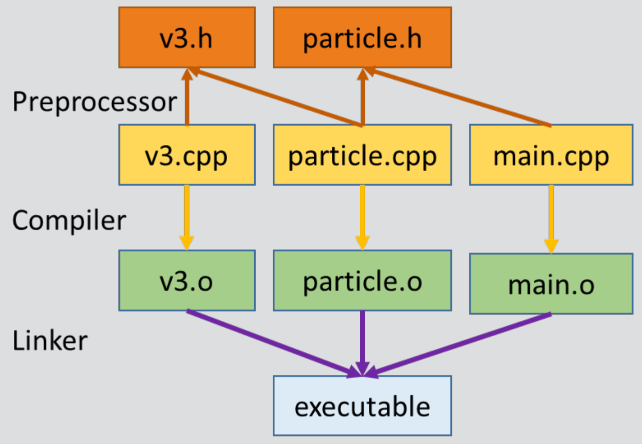
THINK BIG WE DO℠

Header Files

Separate compilation

- Source code can be divided into multiple files
  - source files can be compiled separately
- Classes can be implemented in their own files
  - allows reusing codes in multiple programs
  - source files including class methods and function definitions
  - header files including declarations and global constants

Compiling multiple files



```
g++ v3.cpp particle.cpp main.cpp -o executable
```

https://devblogs.nvidia.com/separate-compilation-linking-cuda-device-code/

#include

- Used for including header files
  - usually contains class declarations, function prototypes, or global constants
- When used with `< >`
  - compiler looks for the file in the system paths
- When used with " "
  - compiler looks for the file in the current folder
- Cannot compile header files directly!

Multiple declarations of classes

- With large projects, multiple declaration of classes must be prevented
- Use `#ifndef`

```
#ifndef DATE_H
#define DATE_H

class Date {
    // ...
};

#endif
```

Constructors

Constructors

- Special `methods` used to initialize data members when objects are created
- A constructor …
  - … is a member function (usually `public`)
  - … must have the same name as its class
  - … is automatically called when an object is created
  - … does not have a return type (not even `void`)

`constructors cannot be called as other methods`

Example

```
class Date {
    private:
        int month;
        int year;
        int day;

    public:
        Date();    No return
                   value

        // ...
};
```

Example: Date

```
class Date {
    private:
        int month;
        int year;
        int day;

    public:
        Date();
        void print();
};
```

```
#include "date.h"

int main() {
    Date mydate;

    mydate.print();
}
```

```
#include "date.h"
#include <iostream>

Date::Date() {
    month = 1;
    day = 1;
    year = 1970;
}

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

```
g++ date.cc main.cc -o exec
```

Overloading constructors

- A constructor with no parameters is also known as the **default constructor**
- Classes may have multiple constructors
  - constructors are **overloaded** by defining constructors with different parameter lists

```
Date();
Date(int m, int d, int y);
```

Synthesized default constructor

- If you don't define any constructor, C++ will define one default constructor for you
- If you define at least one constructor, C++ will not add any other (not even the default constructor)

Initialization lists

- C++ allows for optional initialization lists as part of the constructor definition

```
Point2D::Point2D(int _x, int _y) {
    x = _x;
    y = _y;
    // more statements
}
```

```
Point2D::Point2D(int _x, int _y) : x(_x), y(_y) {
    // more statements
}
```