CS 32 Week 1
Discussion 1I

UCLA CS

Srinath

Slides by Yiyou Chen

Topics

- Order of Construction
- Const Member Functions for class
- Compiling and Linking
- #include guard
- Circular Dependency

Order of Construction

1.
2.
3.

Order of Construction

1. -----
2. Initialize the data members (built-in: uninitialized; class: default constructor) in order
3. Body of the constructor

Order of Construction

```
1  class Characters {
2    public:
3      Characters(double x, double y, string name);
4    private:
5      double m_x;
6      double m_y;
7      string m_name;
8  };
9
10 Characters::Characters(double x, double y, string name) {
11   m_x = x;
12   m_y = y;
13   m_name = name;
14 }
15
16 class Game {
17   public:
18     Game(const double& size, const double& x, const double& y, const string& name);
19   private:
20     Characters m_character;
21     double m_size;
22 };
23
24 Game::Game(const double& size, const double& x, const double& y, const string& name) {
25   m_character = Characters(x, y, name);
26   m_size = size;
27 }
```

Will it compile?

Order of Construction

```
1  class Characters {
2    public:
3      Characters(double x, double y, string name);
4    private:
5      double m_x;
6      double m_y;
7      string m_name;
8  };
9
10 Characters::Characters(double x, double y, string name) {
11   m_x = x;
12   m_y = y;
13   m_name = name;
14 }
15
16 class Game {
17   public:
18     Game(const double& size, const double& x, const double& y, const string& name);
19   private:
20     Characters m_character;
21     double m_size;
22 };
23
24 Game::Game(const double& size, const double& x, const double& y, const string& name) {
25   m_character = Characters(x, y, name);
26   m_size = size;
27 }
```

Wrong!
Characters has no default constructor
Characters::Characters();

Order of Construction

```
 3 class Characters {
 4   public:
 5     Characters(double x, double y, string name);
 6   private:
 7     double m_x;
 8     double m_y;
 9     string m_name;
10 };
11
12 Characters::Characters(double x, double y, string name) {
13   m_x = x;
14   m_y = y;
15   m_name = name;
16 }
17
18 class Game {
19   public:
20     Game(const double& size, const double& x, const double& y, const string& name);
21   private:
22     Characters m_character;
23     double m_size;
24 };
25
26 Game::Game(const double& size, const double& x, const double& y, const string& name)
27   : m_character(x, y, name)
28 {
29   m_size = size;
30 }
```

Correct!

Order of Construction

```
1  class Characters {
2    public:
3      Characters(double x, double y, string name);
4    private:
5      double m_x;
6      double m_y;
7      string m_name;
8  };
9
10 Characters::Characters(double x, double y, string name) {
11   m_x = x;
12   m_y = y;
13   m_name = name;
14 }
15
16 class Game {
17   public:
18     Game(const double& size, const double& x, const double& y, const string& name);
19   private:
20     Characters* m_character;
21     double m_size;
22 };
23
24 Game::Game(const double& size, const double& x, const double& y, const string& name)
25 {
26   m_character = new Characters(x, y, name);
27   m_size = size;
28 }
```

Correct!

Const Member Functions

```
56 class Characters {
57   public:
58     Characters(double x, double y, string name);
59     void Move(const double& movex, const double& movey);
60   private:
61     double m_x;
62     double m_y;
63     string m_name;
64 };
65 Characters::Characters(double x, double y, string name) {
66   m_x = x;
67   m_y = y;
68   m_name = name;
69 }
70
71 void Characters::Move(const double& movex, const double& movey) {
72   cout << "hi" << endl;
73 }
74
75 void Play(const Characters* character, const double& movex, const double& movey) {
76   character->Move(movex, movey);
77 }
```

Will it compile?

Const Member Functions

```
56 class Characters {
57   public:
58     Characters(double x, double y, string name);
59     void Move(const double& movex, const double& movey);
60   private:
61     double m_x;
62     double m_y;
63     string m_name;
64 };
65 Characters::Characters(double x, double y, string name) {
66   m_x = x;
67   m_y = y;
68   m_name = name;
69 }
70
71 void Characters::Move(const double& movex, const double& movey) {
72   cout << "hi" << endl;
73 }
74
75 void Play(const Characters* character, const double& movex, const double& movey) {
76   character->Move(movex, movey);
77 }
```

Wrong! Move is not defined to be a constant member function.

Const Member Functions

```
56 class Characters {
57   public:
58     Characters(double x, double y, string name);
59     void Move(const double& movex, const double& movey) const;
60   private:
61     double m_x;
62     double m_y;
63     string m_name;
64 };
65 Characters::Characters(double x, double y, string name) {
66   m_x = x;
67   m_y = y;
68   m_name = name;
69 }
70
71 void Characters::Move(const double& movex, const double& movey) const {
72   cout << "hi" << endl;
73 }
74
75 void Play(const Characters* character, const double& movex, const double& movey) {
76   character->Move(movex, movey);
77 }
```

Correct!

Compiling and Linking

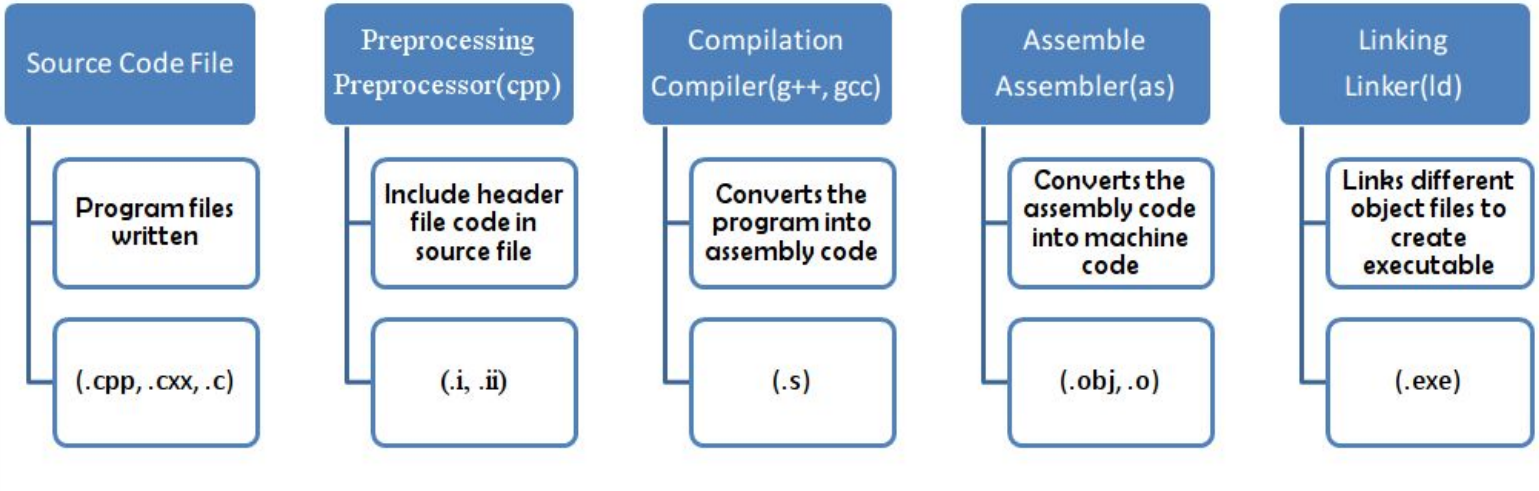

Image: https://www.cplusplus.com

#include guard

To make sure each header file is included once for each source file.

```
“XXX.h”
#ifndef XXX_INCLUDED
#define XXX_INCLUDED
class XXX{
   ….
};
…
#endif
```

Circular Dependency

```
"Characters.h"
#ifndef Characters_INCLUDED
#define Characters_INCLUDED
#include <cstring>
#include "Game.h"
using namespace std;
class Characters {
  public:
    Characters(double x, double y);
  private:
    double m_x;
    double m_y;
    string m_name;
    Game m_game;
};
#endif
```

```
"Game.h"
#ifndef Game_INCLUDED
#define Game_INCLUDED
#include <cstring>
#include <iostream>
#include "Characters.h"
using namespace std;
class Game {
  public:
    Game(double size, double x, double y);
  private:
    Characters* m_character;
    double m_size;
};
#endif

"main.cpp"
#include "Game.h"
#include "Characters.h"
```

Will it compile?

Circular Dependency

```
80 "Characters.h"
81 #ifndef Characters_INCLUDED
82 #define Characters_INCLUDED
83 #include <cstring>
84 #include "Game.h"
85 using namespace std;
86 class Characters {
87   public:
88     Characters(double x, double y);
89   private:
90     double m_x;
91     double m_y;
92     string m_name;
93     Game m_game;
94 };
95 #endif
```

```
 98 "Game.h"
 99 #ifndef Game_INCLUDED
100 #define Game_INCLUDED
101 #include <cstring>
102 #include <iostream>
103 #include "Characters.h"
104 using namespace std;
105 class Game {
106   public:
107     Game(double size, double x, double y);
108   private:
109     Characters* m_character;
110     double m_size;
111 };
112 #endif
113
114 "main.cpp"
115 #include "Game.h"
116 #include "Characters.h"
```

Wrong!

Circular Dependency

```
80 "Characters.h"
81 #ifndef Characters_INCLUDED
82 #define Characters_INCLUDED
83 #include <cstring>
84 class Game;
85 using namespace std;
86 class Characters {
87   public:
88     Characters(double x, double y);
89   private:
90     double m_x;
91     double m_y;
92     string m_name;
93     Game m_game;
94 };
95 #endif
```

```
 98 "Game.h"
 99 #ifndef Game_INCLUDED
100 #define Game_INCLUDED
101 #include <cstring>
102 #include <iostream>
103 class Characters;
104 using namespace std;
105 class Game {
106   public:
107     Game(double size, double x, double y);
108   private:
109     Characters* m_character;
110     double m_size;
111 };
112 #endif
113
114 "main.cpp"
115 #include "Game.h"
116 #include "Characters.h"
```

Will it compile?

Circular Dependency

```
80 "Characters.h"
81 #ifndef Characters_INCLUDED
82 #define Characters_INCLUDED
83 #include <cstring>
84 class Game;
85 using namespace std;
86 class Characters {
87   public:
88     Characters(double x, double y);
89   private:
90     double m_x;
91     double m_y;
92     string m_name;
93     Game m_game;
94 };
95 #endif
```

```
 98 "Game.h"
 99 #ifndef Game_INCLUDED
100 #define Game_INCLUDED
101 #include <cstring>
102 #include <iostream>
103 class Characters;
104 using namespace std;
105 class Game {
106   public:
107     Game(double size, double x, double y);
108   private:
109     Characters* m_character;
110     double m_size;
111 };
112 #endif
113
114 "main.cpp"
115 #include "Game.h"
116 #include "Characters.h"
```

Will it compile?

In this case, Yes!

Circular Dependency

```
80 "Characters.h"
81 #ifndef Characters_INCLUDED
82 #define Characters_INCLUDED
83 #include <cstring>
84 class Game;
85 using namespace std;
86 class Characters {
87   public:
88     Characters(double x, double y);
89   private:
90     double m_x;
91     double m_y;
92     string m_name;
93     Game m_game;
94 };
95 #endif
```

```
 98 "Game.h"
 99 #ifndef Game_INCLUDED
100 #define Game_INCLUDED
101 #include <cstring>
102 #include <iostream>
103 class Characters;
104 using namespace std;
105 class Game {
106   public:
107     Game(double size, double x, double y);
108   private:
109     Characters* m_character;
110     double m_size;
111 };
112 #endif
113
114 "main.cpp"
115 #include "Game.h"
116 #include "Characters.h"
```

Will it compile?

However, it will not compile if main.cpp has the following code instead(notice the reverse ordering of header files).

“main.cpp”
#include “Characters.h”
#include “Game.h”

Circular Dependency

```
"Characters.h"
#ifndef Characters_INCLUDED
#define Characters_INCLUDED
#include <cstring>
#include "Game.h"
using namespace std;
class Characters {
  public:
    Characters(double x, double y);
  private:
    double m_x;
    double m_y;
    string m_name;
    Game m_game;
};
#endif
```

```
"Game.h"
#ifndef Game_INCLUDED
#define Game_INCLUDED
#include <cstring>
#include <iostream>
class Characters;
using namespace std;
class Game {
  public:
    Game(double size, double x, double y);
  private:
    Characters* m_character;
    double m_size;
};
#endif

"main.cpp"
#include "Game.h"
#include "Characters.h"
```

A better practice!

Correct!