C++

PROGRAMMING LANGUAGE

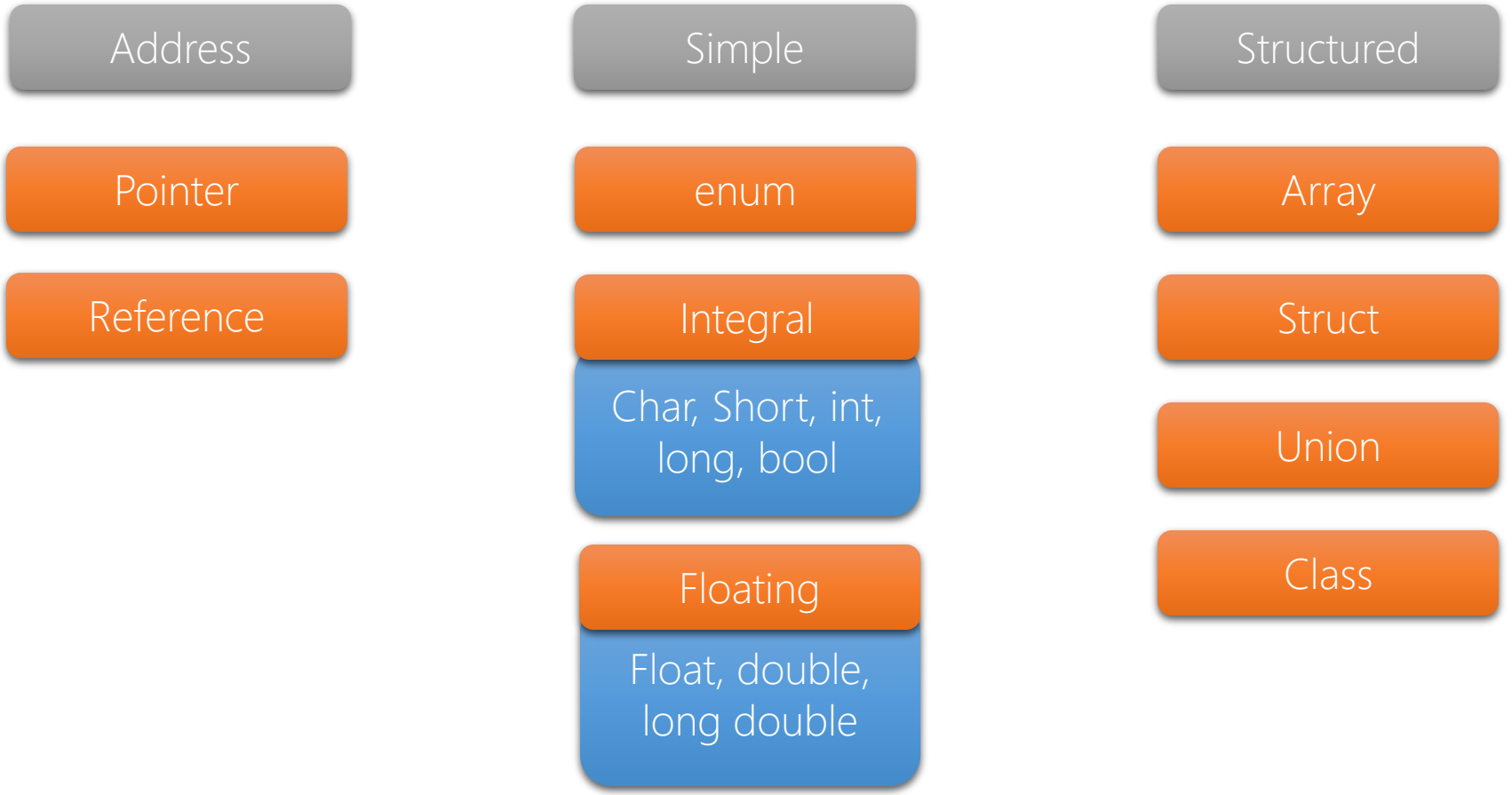
L08-CLASSES,P1

Mohammad Shaker
mohammadshaker.com
@ZGTRShaker
2010, 11, 12, 13, 14

OOP

Object Oriented Programming

C++ data types


Address
Pointer
Reference
Simple
enum
Integral
Char, Short, int, long, bool
Floating
Float, double, long double
Structured
Array
Struct
Union
Class

a class

a Class

- class
  - Concept
    - Information hiding
    - Prevent access to their inner workings
    - Encapsulation (private, public)
  - code

- class
  - Is an extension of the idea of “struct”

- class
  - User defined (programmer defined) type
    - Once class is defined it can be used as a type
    - Object of that class can be defined

- No memory is allocated (like struct)

a Class

- Class package data declarations with function declarations
  - Thus, coupling data with behavior
- Used to implement “ADT”
- Examples:
  - Time, Complex numbers,

Class

- Type:
  - Time
- Domain:
  - Any time value is a time in: Hour, minute, second
- Operations:
  - Set the time
  - Print the time
  - Incrementing / decrementing time
  - Compare times
    - =, <, >

Class

- Members
  - They are the components of the class
    - Attributes
    - Behaviors \ methods

- Class instance
  - Object

Class

- OOP and classes
  - Encapsulation
  - Composition
  - Inheritance
  - Reusable

Class

```
#include <iostream>
using namespace std;

class MyClass
{


};

void main()
{

}
```

Compile & run

```
#include <iostream>
using namespace std;

class MyClass
{
            int i;
};

void main()
{

}
```

Compile and run

```
#include <iostream>
using namespace std;

class MyClass
{


}

void main()
{

}
```

Compiler error. Missing ;

```
#include <iostream>
using namespace std;

class MyClass
{
            int i = 0;
};

void main()
{

}
```

Compiler error

Rule:
You can't initialize a variable when you declare it in classes.

Class

- Function in classes
  - It can directly access any member of the class (WITHOUT PASSING IT AS PARAMETER)
    - Data members
    - Function members

Class

- private:
  - Default access mode
  - Accessible just to the functions of the class and friends
- public:
  - Accessible wherever object of class in scope
- protected:
  - Accessible from members of their same class and friends class and also from their derived class

Class

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time();     // constructor
            void ChangeTime(int,int,int);
private:
            int hour;
            int minute;
            int second;
};

void main()
{

}
```

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time()      // constructor
            {
                        minute = second = hour = 0;
            }
            void ChangeTime(int,int,int)
            {
                        hour = 12;
                        second = 0;
                        minute = 0;
            }
private:
            int hour;
            int minute;
            int second;
};

void main()
{}
```

Class

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time();
            void ChangeTime(int,int,int);
private:
            int hour;
            int minute;
            int second;
};

Time::Time()            // constructor
{
            minute = second = hour = 0;
}
void Time::ChangeTime(int,int,int)
{
            hour = 12;
            second = 0;
            minute = 0;
}
void main()
{}
```

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time::Time();
            void Time::ChangeTime(int,int,int);
private:
            int hour;
            int minute;
            int second;
};

Time::Time()            // constructor
{
            minute = second = hour = 0;
}
void Time::ChangeTime(int,int,int)
{
            hour = 12;
            second = 0;
            minute = 0;
}
void main()
{}
```

Class

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time();
            void ChangeTime(int,int,int);
            void PrintTime();
private:
            int hour;
            int minute;
            int second;
};
Time::Time()            // constructor
{           minute = second = hour = 0; }
void Time::ChangeTime(int x1,int x2,int x3)
{           hour = x1;  second = x2;            minute = x3;}
void Time::PrintTime()
{           cout << hour << ":" << minute << ":"
            << second << endl;
}

void main()
{           Time T;
            T.PrintTime();
            T.ChangeTime(11,1,5);
            T.PrintTime();
}
```

```
0:0:0
11:5:1
```

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time();
            void ChangeTime(int,int,int);
            void PrintTime();
private:
            int hour;
            int minute;
            int second;
};
Time::Time()            // constructor
{           minute = second = hour = 0; }
void Time::ChangeTime(int x1,int x2,int x3)
{           hour = x1;  second = x2;            minute = x3;}
void Time::PrintTime()
{           cout << hour << ":" << minute << ":"
            << second << endl;
}

void main()
{           Time T;
            T.hour = 12;
            T.PrintTime();          T.ChangeTime(11,1,5);
            T.PrintTime();
}
```

```
Compiler error
Attempting to access private data members
```

Class

```
Compiler error
Missing return type of the PrintTime() function
```

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time();
            void ChangeTime(int,int,int);
            void PrintTime();
private:
            int hour;
            int minute;
            int second;
};
Time::Time()            // constructor
{           minute = second = hour = 0; }
void Time::ChangeTime(int x1,int x2,int x3)
{           hour = x1;  second = x2;            minute = x3;}
Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second <<
endl;}

void main()
{           Time T;
            T.PrintTime();
            T.ChangeTime(11,1,5);
            T.PrintTime();}
```

Class

```
class Time
{
public:
            Time();
            void ChangeTime(int,int,int);
            void PrintTime();
            void ChangeHourFromOutside(int);
private:
            int hour;
            int minute;
            int second;
};
Time::Time()            // constructor
{           minute = second = hour = 0; }
void Time::ChangeTime(int x1,int x2,int x3)
{           hour = x1;  second = x2;            minute = x3;}
void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second <<
endl;}
void Time::ChangeHourFromOutside(int x)
{           hour = x;}

void main()
{
            int TempHour = 0;       Time T;
            cout << "Enter The Hour: " << endl;
            cin >> TempHour;        //5
            T.PrintTime();
            T.ChangeHourFromOutside(TempHour);
            T.PrintTime();
            T.ChangeTime(11,1,5);
            T.PrintTime();
}
```

```
Enter The
Hour:
5
0:0:0
5:0:0
11:5:1
```

```
class Time
{
public:
            Time();
            void ChangeTime(int,int,int);
            void PrintTime();
            void ChangeHourFromOutside(int);
            void Clear ();
private:
            int hour;
            int minute;
            int second;
};
Time::Time()            // constructor
{           minute = second = hour = 0; }
void Time::ChangeTime(int x1,int x2,int x3)
{           hour = x1;  second = x2;            minute = x3;}
void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second <<
endl;}
void Time::ChangeHourFromOutside(int x)
{           hour = x;}
void Time::Clear ()
{           minute = second = hour = 0; }
void main()
{
            int TempHour = 0;       Time T;
            cout << "Enter The Hour: " << endl;
            cin >> TempHour;        //5
            T.PrintTime();
            T.ChangeHourFromOutside(TempHour);
            T.Clear();
            T.PrintTime();
            T.ChangeTime(11,1,5);
            T.PrintTime();
}
```

```
Enter The
Hour:
5
0:0:0
0:0:0
11:5:1
```

Class

```
5:0:0
5:0:0
5:0:0
```

```
class Time
{
public:
            Time();
            void ChangeTime(int,int,int);
            void PrintTime();
            void ChangeHourFromOutside(int);
            void Clear ();
private:
            int hour;
            int minute;
            int second;
};
Time::Time()            // constructor
{           minute = second = hour = 0; }
void Time::ChangeTime(int x1,int x2,int x3)
{           hour = x1;  second = x2;            minute = x3;}
void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second <<
endl;}
void Time::ChangeHourFromOutside(int x)
{           hour = x;}
void Time::Clear ()
{           minute = second = hour = 0; }
void main()
{
            Time T1;                Time *T2 = &T1;
            Time* &T3 = T2;
            T1.ChangeHourFromOutside(5);
            T1.PrintTime();
            T2->PrintTime();
            T3->PrintTime();
}
```

```
5:0:0
5:0:0
5:0:0
```

```
class Time
{
public:
            Time();
            void ChangeTime(int,int,int);
            void PrintTime();
            void ChangeHourFromOutside(int);
            void Clear ();
private:
            int hour;
            int minute;
            int second;
};
Time::Time()            // constructor
{           minute = second = hour = 0; }
void Time::ChangeTime(int x1,int x2,int x3)
{           hour = x1;  second = x2;            minute = x3;}
void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second <<
endl;}
void Time::ChangeHourFromOutside(int x)
{           hour = x;}
void Time::Clear ()
{           minute = second = hour = 0; }
void main()
{
            Time T1;                Time *T2 = &T1;
            Time &T3 = *T2;
            T1.ChangeHourFromOutside(5);
            T1.PrintTime();
            T2->PrintTime();
            T3.PrintTime();
}
```

Class

```
class Time
{
public:
            Time();
            void ChangeTime(int,int,int);
            void PrintTime();
            void ChangeHourFromOutside(int);
            void Clear ();
private:
            int hour = 9;
            int minute;
            int second;
};
Time::Time()            // constructor
{           minute = second = hour = 0; }
void Time::ChangeTime(int x1,int x2,int x3)
{           hour = x1;  second = x2;            minute = x3;}
void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second <<
endl;}
void Time::ChangeHourFromOutside(int x)
{           hour = x;}
void Time::Clear ()
{           minute = second = hour = 0; }
void main()
{
            Time T1;                Time *T2 = &T1;
            Time &T3 = *T2;
            T1.ChangeHourFromOutside(5);
            T1.PrintTime();
            T2->PrintTime();
            T3.PrintTime();
}
```

Compiler error, can't initialize when declaring no matter what private or public

Class

- A Getter, Get function:
  - Reads private data
  - “Query” functions

- A Setter, Set function:
  - Modify private data
  - Perform a validity check before modifying private data
  - Notify if invalid values

Class

```
class Time
{public:
            Time();
            void SetTime(int,int,int);          // set functions
            void SetHour(int);
            void SetMinute(int);
            void SetSecond(int);
            int GetHour();                                  // get functions
            int GetMinute()
            ;int GetSecond();
            void PrintTime();
private:    int hour;   int minute; int second;             };

Time::Time()
{           hour = minute = second = 0; };

void Time::SetTime(int xhour,int xminute, int xsecond)
{           SetHour(xhour);         SetMinute(xminute);     SetSecond(xsecond);};

void Time::SetHour(int h) { hour = (h>=0 && h<24)? h: 0; };
void Time::SetMinute(int m) { minute = (m>=0 && m<60)? m: 0; };
void Time::SetSecond(int s) { second = (s>=0 && s<60)? s: 0; };

int Time::GetHour() {return hour;};
int Time::GetMinute() {return minute;};
int Time::GetSecond() {return second;};

void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second << endl; }
```

```
void main()
{
            Time T;
            T.PrintTime();

            T.SetHour(17);
            T.SetMinute(45);
            T.SetSecond(22);

            T.PrintTime();

            T.SetHour(77);
            T.SetMinute(565);
            T.SetSecond(-1);

            T.PrintTime();

}
```

```
0:0:0
17:45:22
0:0:0
Press any key to continue
```

Class

- Utility functions (Helper functions)
  - private
    - Support operations of public member functions
    - Not intended for direct client use
  - Examples

Organizing Your Classes

Organizing Your Classes

- Implementing classes
- Separating Interface from implementation

- Headers (in header file) –if needed-
  - # ifndef Time_H
  - # define Time_H

  - // your code goes here

  - # endif

Organizing Your Classes

In TimeClass.h

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time();
            void SetTime(int,int,int);
            void SetHour(int);
            void SetMinute(int);
            void SetSecond(int);

            int GetHour();
            int GetMinute();

            int GetSecond();

            void PrintTime();

private:
            int hour;
            int minute;
            int second;
};
```

In TimeClass.cpp

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

Time::Time()
{           hour = minute = second = 0; };

void Time::SetTime(int xhour,int xminute, int xsecond)
{SetHour(xhour);        SetMinute(xminute);     SetSecond(xsecond);};

void Time::SetHour(int h) { hour = (h>=0 && h<24)? h: 0; };
void Time::SetMinute(int m) { minute = (m>=0 && m<60)? m: 0; };
void Time::SetSecond(int s) { second = (s>=0 && s<60)? s: 0; };

int Time::GetHour() {return hour;};
int Time::GetMinute() {return minute;};
int Time::GetSecond() {return second;};

void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second << endl; }
```

In MyMain.cpp

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

void main()
{
            Time T;
            T.PrintTime();

            T.SetHour(17);
            T.SetMinute(45);
            T.SetSecond(22);

            T.PrintTime();

            T.SetHour(77);
            T.SetMinute(565);
            T.SetSecond(-1);

            T.PrintTime();
}
```

Live Demo

Class Constructor

Class

- Constructor
  - What's it for?
    - Used to guarantee the initialization of data members of class
  - The same name as class
  - Called implicitly when object is created
    - Whether calling it is statically or dynamically
    - (new operator) *
  - NO return type
  - Can be overloaded
  - Must be declared in public section

---

* That means that we don't need to call the constructor explicitly

Class

- Constructor
  - Two types:
    - Without parameters (Default constructor)
    - With parameters
  - When creating a class object, C++ provides its own constructor for the object
    - Compiler implicitly creates default constructor
    - The programmer should explicitly define it, why?
      - Coz the data otherwise wouldn't be guaranteed to be in consistent state
      - And that doesn't perform any initialization of fundamentals variables

Class Constructor

```
class Time
{
public:
            void SetTime(int,int,int);
            void SetHour(int);      void SetMinute(int);
            void SetSecond(int);
            int GetHour();          int GetMinute();
            int GetSecond();
            void PrintTime();
private:
            int hour;
            int minute;
            int second;
};
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;


void Time::SetTime(int xhour,int xminute, int xsecond)
{           SetHour(xhour);         SetMinute(xminute);     SetSecond(xsecond);
};
void Time::SetHour(int h) { hour = (h>=0 && h<24)? h: 0; };
void Time::SetMinute(int m) { minute = (m>=0 && m<60)? m: 0; };
void Time::SetSecond(int s) { second = (s>=0 && s<60)? s: 0; };

int Time::GetHour() {return hour;};
int Time::GetMinute() {return minute;};
int Time::GetSecond() {return second;};

void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second << endl; }
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

void main()
{
            Time T;
            T.PrintTime();

            T.SetHour(17);
            T.SetMinute(45);
            T.SetSecond(22);

            T.PrintTime();

            T.SetHour(77);
            T.SetMinute(565);
            T.SetSecond(-1);

            T.PrintTime();
}
```

```
2486832:1588607792:2486968
17:45:22
0:0:0
Press any key to continue
```

Cauze we didn't write the constructor ourself so that the variables of the class hasn't been initialized properly

Class Constructor

```
class Time
{public:
            Time();
            void SetTime(int,int,int);          // set functions
            void SetHour(int);
            void SetMinute(int);
            void SetSecond(int);
            int GetHour();                                  // get functions
            int GetMinute()
            ;int GetSecond();
            void PrintTime();
private:    int hour;   int minute; int second;             };

Time::Time()
{           hour = minute = second = 0; };

void Time::SetTime(int xhour,int xminute, int xsecond)
{           SetHour(xhour);         SetMinute(xminute);     SetSecond(xsecond);};

void Time::SetHour(int h) { hour = (h>=0 && h<24)? h: 0; };
void Time::SetMinute(int m) { minute = (m>=0 && m<60)? m: 0; };
void Time::SetSecond(int s) { second = (s>=0 && s<60)? s: 0; };

int Time::GetHour() {return hour;};
int Time::GetMinute() {return minute;};
int Time::GetSecond() {return second;};

void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second << endl; }
```

```
void main()
{
            Time T;
            T.PrintTime();

            T.SetHour(17);
            T.SetMinute(45);
            T.SetSecond(22);

            T.PrintTime();

            T.SetHour(77);
            T.SetMinute(565);
            T.SetSecond(-1);

            T.PrintTime();

}
```

```
0:0:0
17:45:22
0:0:0
Press any key to continue
```

Overloading Constructors

Overloading Constructors

- Constructors can be overloaded
  - must all have the same name as class
- Overloading function in General allows you to use the same name for separate functions that have different argument lists

Overloading Constructors

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time();
            Time(int TimeHour);
            void PrintTime();
private:
            int hour;
            int minute;
            int second;
};
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

Time::Time()
{           hour = minute = second = 0; };

Time::Time(int TimeHour)
{           hour = TimeHour; }

void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second << endl; }
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;


void main()
{
            Time T1;
            T1.PrintTime();

            Time T2(2);
            T2.PrintTime();
}
```

```
0:0:0
2:1588607792:3011112
Press any key to continue
```

Class – Member initializing list

- Two ways to initialize member data by constructor
  - Code inserted in the constructor
  - A member initialization list
    - Given in the constructor definition
      - Not the prototype!
    - Follows the arguments list
    - Tells which member data to initialize with which arguments
    - So, how is the code?
      - Consists of
        - » A colon followed by a comma separated list of member name \ argument pairs

Overloading Constructors

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time::Time():hour(),minute(),second(){};
            Time(int TimeHour):hour(TimeHour),minute(0),second(0){};;
            Time(int TimeHour, int
TimeMinute):hour(TimeHour),minute(TimeMinute),second(0){};
            Time(int TimeHour, int TimeMinute, int
TimeSecond):hour(TimeHour),minute(TimeMinute),second(TimeSecond){};
            void PrintTime();
private:
            int hour;
            int minute;
            int second;
};

void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second << endl; }
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;


void main()
{
            Time T1;
            T1.PrintTime();

            Time T2(2);
            T2.PrintTime();

            Time T3(2,3,06);
            T3.PrintTime();

}
```

```
0:0:0
2:0:0
2:3:6
```

Overloading Constructors

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time::Time():hour(),minute(),second();
            Time(int TimeHour):hour(TimeHour),minute(0),second(0){};
            void PrintTime();
private:
            int hour;
            int minute;
            int second;
};

void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second << endl; }
```

```
Compiler error, missing {}
```

Overloading Constructors

```
#include <iostream>
using namespace std;

class Time
{
public:
            Time::Time():hour(),minute(),second(){};
            Time(int TimeHour):hour(TimeHour),minute(0),second(0){};
            void PrintTime();
private:
            int hour;
            int minute;
            int second;
};

void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second << endl; }
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;


void main()
{
            Time T1;
            T1.PrintTime();

            Time T2(2);
            T2.PrintTime();

            Time T3(2,3,06);
            T3.PrintTime();

}
```

```
Compiler error, no oveloaded function which
takes 3 arguments
```

Classes Destructors

Classes Destructors

- Same name as class
- Preceded by tilde (~)
- No arguments
- No return value
- Can't be overloaded
  - Once only
- When is it called?
  - When the object is destroyed
    - Termination housekeeping
    - new \ delete
- If no explicit destructor was declared,
  - Compiler creates "Empty destructor"

Constructors VS Destructors

- Orders of constructor \ destructor calls
  - Depends on scope
  - Generally, destructor are called in reverse order of constructor call

Constructors VS Destructors

- Global scope object:
  - Constructors:
    - Before any other function (including main!)
  - Destructors:
    - When
      - main terminates
      - exit function is called
    - Not called with abort

Constructors VS Destructors

- Automatic local object:
  - Constructors:
    - When object is defined
      - Each time entering scope
  - Destructors:
    - When
      - Objects leave scope
    - Not called with abort or exit

Constructors VS Destructors

- static local object:
  - Constructors:
    - Exactly once
    - When the execution reaches point where the object is defined
  - Destructors:
    - When
      - main terminates
      - exit function is called
    - Not called with abort

Class Constructor

```
#include <iostream>
using namespace std;

class object
{
public:
            object(int Id,   char *MsgPtr);
            ~object();

private:
            int objectId;
            char *message;
};

object::object(int Id,   char *MsgPtr)
{
            objectId = Id;
            message = MsgPtr;
            cout << objectId << " constructor runs for " << message << endl;
}

object::~object()
{
            cout << objectId << " destructor runs for " << message << endl;
}
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

object first(1,"Global before main before CreateCrazy");
void CreateCrazy();

void main()
{
            static object fourth(4, "static in main");
            CreateCrazy();
}


void CreateCrazy()
{
            object fifth(5, "In CreateCrazy");
}
```

```
1 constructor runs for Global before main before CreateCrazy
4 constructor runs for static in main
5 constructor runs for In CreateCrazy
5 destructor runs for In CreateCrazy
4 destructor runs for static in main
1 destructor runs for Global before main before CreateCrazy
Press any key to continue
```

Copy Constructors

Copy Constructor

- Special member functions
  - That's implicitly called in the following 3 situations
    - Object used to initialize another object
      - obj1(obj2)
    - Passing an object arguments by value to another object
    - Returning a temporary object as the return value of a function

Copy Constructor

- C++ provides its own implicitly copy constructor
  - Overloaded constructor
  - Has no return value
  - Take only arguments as a reference to an object of the same class
  - The compiler provides a copy whose signature is:
  - Class_Name::Class_Name (const Class_Name&)
  - When an object is passed to a function, a simple bitwise (exact) copy of that object is made and given to the function

Copy Constructor

- Assignment "="
  - Assign object to object
  - Is done by
    - Each member in the first object is copied individually to the same member in the second one
    - Done to the object that are value arguments

Copy Constructor

```
#include <iostream>
using namespace std;

class object
{
public: object();
            object(const object& ob);
            ~object();
            void display();

private:    int num, den;};

object::object()
{cout << "constructing" << endl;   num = 0;    den = 1;}

object::object(const object& ob)
{cout << "copy constructing" << endl; num = ob.num; den = ob.den;}

object::~object()
{           cout << "destructing" << endl;}

void object::display()
{           cout << "num = " << num << ", den = " << den << endl;}
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

void Play(object x)
{
            cout << "In function Play" << endl;
            x.display();
}

void main()
{
            object ob1;            object ob2;

            cout << "In main, before Play" << endl;
            ob1.display();
            ob2.display();

            Play (ob1);
            cout << "In main, after calling Play " << endl;
            ob1.display();
            ob2.display();

            ob2 = ob1;
            cout << "In main, after assignemet " << endl;
            ob1.display();
            ob2.display();
}
```

Copy Constructor

```
#include <iostream>
using namespace std;

class object
{
public: object();
            object(const object& ob);
            ~object();
            void display();

private:    int num, den;};

object::object()
{cout << "constructing" << endl;   num = 0;    den = 1;}

object::object(const object& ob)
{cout << "copy constructing" << endl; num = ob.num; den = ob.den;}

object::~object()
{           cout << "destructing" << endl;}

void object::display()
{           cout << "num = " << num << ", den = " << den << endl;}
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

void Play(object x)
{
            cout << "In function Play"
            x.display();
}

void main()
{
            object ob1;            objec

            cout << "In main, before P
            ob1.display();
            ob2.display();

            Play (ob1);
            cout << "In main, after calling Play " << endl;
            ob1.display();
            ob2.display();

            ob2 = ob1;
            cout << "In main, after assignemet " << endl;
            ob1.display();
            ob2.display();
}
```

```
constructing
constructing
In main, before Play
num = 0, den = 1
num = 0, den = 1
copy constructing
In function Play
num = 0, den = 1
destructing
In main, after calling Play
num = 0, den = 1
num = 0, den = 1
In main, after assignemet
num = 0, den = 1
num = 0, den = 1
destructing
destructing
Press any key to continue
```

Cloning Objects

Shallow copy VS Deep Copy

- C++,by default, use shallow copy for initializations and assignments
- With shallow copy, if the object contains a pointer to allocated memory
  - The copy will also point to the memory as does the original object
  - Deleting the original object may cause the copied object to incorrectly disappear!

Shallow copy VS Deep Copy

- If the class data points to a dynamic data
  - You should write your own copy constructor to create a “deep copy” of the dynamic data
  - A deep copy copies not only the class data members, but also makes a separate stored copy of any pointed to data
  - The copy constructor is implicitly called in initialization situations and makes a deep copy of the dynamic data in a different memory location

const Data Members

const Data Members

- Specify object not modifiable
- Compiler error if attempting to modify const object

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

void main()
{
        const Time T1;         // dealcaring cost with default constructor
        const Time T2(11,0,0); // decalring const with another constructor
}
```

const Data Members

- Member function for const object must be const
  - Can't modify object
- Specify const in both
  - Prototype: after parameter list
  - Definition: Before beginning left brace
- Note!
  - Constructors and destructors can't be const
    - Must be able to modify

const Data Members

```
class Time
{public:
              Time();
              void SetTime(int,int,int);              // set functions
              void SetHour(int);
              void SetMinute(inat);
              void SetSecond(int);
              int GetHour() const;                    // get functions
              int GetMinute() const;
              int GetSecond() const;
              void PrintTime1stform() const;
              void PrintTime2ndform();
private:
              int hour;                  int minute;                int second;
};
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

Time::Time()
{             hour = minute = second = 0; };

void Time::SetTime(int xhour,int xminute, int xsecond)
{             SetHour(xhour);            SetMinute(xminute);        SetSecond(xsecond);};

void Time::SetHour(int h) { hour = (h>=0 && h<24)? h: 0; };
void Time::SetMinute(int m) { minute = (m>=0 && m<60)? m: 0; };
void Time::SetSecond(int s) { second = (s>=0 && s<60)? s: 0; };

int Time::GetHour() const{return hour;};
int Time::GetMinute() const{return minute;};
int Time::GetSecond() const{return second;};

void Time::PrintTime1stform ()const
{             cout << hour << ":" << minute << ":" << second << endl; }

void Time::PrintTime2ndform ()
{             cout << hour << ":" << minute << ":" << second << endl; }
```

const Data Members

```
void main()
{// delcaring const
 //with default constructor
           const Time T1;
           Time T2;
}
```

```
Everything is okay till now
```

```
void main()
{
           const Time T1;        // delcaring cost with default constructor
           Time T2;

           T2.SetMinute(20);     // non const - non const          working
           T2.GetHour();                    // non const - const             working
           // T1.SetHour(20);    // const - non const              error
           T1.GetHour();                    // const - const                 working

           T2.PrintTime1stform();                      // non const - const   working
           T2.PrintTime2ndform();                      // non const - non const working
           T1.PrintTime1stform();                      // const - const       working
           //T1.PrintTime2ndform();         // const - non const   error
}
```

```
0:20:0
0:20:0
0:0:0
Press any key to continue
```

const Data Members

- Initializing with member initializer list
  - Can be used for
    - all data members
  - Must be used for
    - const data members
    - Data members that are references

const Data Members

```
class Time
{public:
              Time();
              void SetTime(int,int,int);              // set functions
              void SetHour(int);
              void SetMinute(int);
              void SetSecond(int);
              int GetHour() const;                    // get functions
              int GetMinute() const;
              int GetSecond() const;
              void PrintTime1stform() const;
              void PrintTime2ndform();
private:
int hour;     int minute;   const int second;
};
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;
Time::Time()
{             hour = second = minute = 0;}
void Time::SetTime(int xhour,int xminute, int xsecond)
{             SetHour(xhour);             SetMinute(xminute);        SetSecond(xsecond);};

void Time::SetHour(int h) { hour = (h>=0 && h<24)? h: 0; };
void Time::SetMinute(int m) { minute = (m>=0 && m<60)? m: 0; };


int Time::GetHour() const{return hour;};
int Time::GetMinute() const{return minute;};
int Time::GetSecond() const{return second;};

void Time::PrintTime1stform ()const
{             cout << hour << ":" << minute << ":" << second << endl; }

void Time::PrintTime2ndform ()
{             cout << hour << ":" << minute << ":" << second << endl; }
```

```
Compiler error, const member "second"
must be initialize with member
initializer list
```

const Data Members

```
class Time
{public:
              Time();
              void SetTime(int,int,int);              // set functions
              void SetHour(int);
              void SetMinute(int);
              void SetSecond(int);
              int GetHour() const;                    // get functions
              int GetMinute() const;
              int GetSecond() const;
              void PrintTime1stform() const;
              void PrintTime2ndform();
private:
int hour;     int minute;   const int second;
};
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

Time::Time():hour(0),minute(0),second(0){};


void Time::SetTime(int xhour,int xminute, int xsecond)
{             SetHour(xhour);            SetMinute(xminute);        SetSecond(xsecond);};

void Time::SetHour(int h) { hour = (h>=0 && h<24)? h: 0; };
void Time::SetMinute(int m) { minute = (m>=0 && m<60)? m: 0; };
void Time::SetSecond(int s) { second = (s>=0 && s<60)? s: 0; };

int Time::GetHour() const{return hour;};
int Time::GetMinute() const{return minute;};
int Time::GetSecond() const{return second;};

void Time::PrintTime1stform ()const
{             cout << hour << ":" << minute << ":" << second << endl; }

void Time::PrintTime2ndform ()
{             cout << hour << ":" << minute << ":" << second << endl; }
```

Compile error, function SetSecond change the const second

const Data Members

```
class Time
{public:
              Time();
              void SetTime(int,int);     // set functions
              void SetHour(int);
              void SetMinute(int);
              int GetHour() const;                    // get functions
              int GetMinute() const;
              int GetSecond() const;
              void PrintTime1stform() const;
              void PrintTime2ndform();
private:      int hour;     int minute;   const int second;          };
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

Time::Time():hour(0),minute(0),second(5){};

void Time::SetTime(int xhour,int xminute )
{             SetHour(xhour);            SetMinute(xminute);        };

void Time::SetHour(int h) { hour = (h>=0 && h<24)? h: 0; };
void Time::SetMinute(int m) { minute = (m>=0 && m<60)? m: 0; };


int Time::GetHour() const{return hour;};
int Time::GetMinute() const{return minute;};
int Time::GetSecond() const{return second;};

void Time::PrintTime1stform ()const
{             cout << hour << ":" << minute << ":" << second << endl; }

void Time::PrintTime2ndform ()
{             cout << hour << ":" << minute << ":" << second << endl; }
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

void main()
{
              Time T1;
              T1.SetTime(2,3);
              T1.PrintTime1stform();
              T1.SetTime(1,4);
              T1.PrintTime1stform();
}
```

```
2:3:5
1:4:5
Press any key to continue
```

Quiz

Quiz #1

```
#include <iostream>
using namespace std;

class object
{
public: object();
              object(const object& ob);
              ~object();
              void display();
              void Changeden(int);

private:      int num, den;};

object::object()
{             cout << "constructing" << endl;num = 0;den = 1;}

object::object(const object& ob)
{             cout << "copy constructing" << endl;num = ob.num;     den = ob.den;}

object::~object()
{             cout << "destructing" << endl; }

void object::display()
{             cout << "num = " << num << ", den = " << den << endl;}

void object::Changeden(int x )
{             den = x; }
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

void PlayAgain(object x)
{cout << "In function PlayAgain" << endl;   x.display();}

void Play(object x)
{cout << "In function Play" << endl;        PlayAgain(x);
            x.display();}

void main()
{           object ob1;            object ob2;
            cout << "In main, before Play" << endl;
            ob1.display();
            ob2.display();

            Play (ob1);
            cout << "In main, after calling Play " << endl;
            ob1.display();
            ob2.display();
            ob1.Changeden(3);

            ob2 = ob1;
            cout << "In main, after assignemet " << endl;
            ob1.display();
            ob2.Changeden(5);
            ob2.display();
}
```

Quiz #1

```
#include <iostream>
using namespace std;

class object
{
public: object();
              object(const object& ob);
              ~object();
              void display();
              void Changeden(int);

private:      int num, den;};

object::object()
{             cout << "constructing" << endl;num = 0;de

object::object(const object& ob)
{             cout << "copy constructing" << endl;num =

object::~object()
{             cout << "destructing" << endl; }

void object::display()
{             cout << "num = " << num << ", den = " << den << endl;}

void object::Changeden(int x )
{             den = x; }
```

```
constructing
constructing
In main, before Play
num = 0, den = 1
num = 0, den = 1
copy constructing
In function Play
copy constructing
In function PlayAgain
num = 0, den = 1
destructing
num = 0, den = 1
destructing
In main, after calling Play
num = 0, den = 1
num = 0, den = 1
In main, after assignemet
num = 0, den = 3
num = 0, den = 5
destructing
destructing
Press any key to continue
```

```
#include <iostream>
#include "TimeClass.h"
using namespace std;

void PlayAgain(object x)
{cout << "In function PlayAgain" << endl;   x.display();}

void Play(object x)
{cout << "In function Play" << endl;        PlayAgain(x);
           x.display();}

void main()
{          object ob1;           object ob2;
           cout << "In main, before Play" << endl;
           ob1.display();
           ob2.display();

           Play (ob1);
           cout << "In main, after calling Play " << endl;
           ob1.display();
           ob2.display();
           ob1.Changeden(3);

           ob2 = ob1;
           cout << "In main, after assignemet " << endl;
           ob1.display();
           ob2.Changeden(5);
           ob2.display();
}
```

Quiz #2

```
#include <iostream>
using namespace std;

class object
{
public:
              object(int Id,  char *MsgPtr);
              ~object();

private:
              int objectId;
              char *message;

};

object::object(int Id,  char *MsgPtr)
{
              objectId = Id;
              message = MsgPtr;
              cout << objectId << " constructor runs for " << message << endl;
}

object::~object()
{             cout << objectId << " destructor runs for " << message << endl;    }
```

```
#include <iostream>
#include "Testing.h"
using namespace std;

object first(1,"Global before main before CreateCrazy");
void CreateCrazy();
static object second(2,"Static Global before main after
CreateCrazy");

void main()
{
            object bar(100, "in main");
            {
                        object third (3,"in main");
            }
            static object fourth(4, "static in main");
            CreateCrazy();
            object eigth(8,"after CreateCrazy in main");
}


void CreateCrazy()
{
            object fifth(5, "In CreateCrazy");
            static object sixth(6, "Local static in
CreateCrazy");
            object seventh(7, "In CreateCrazy");
}
```

Quiz #2

```
1 constructor runs for Global before main before CreateCrazy
2 constructor runs for Static Global before main after CreateCrazy
100 constructor runs for in main
3 constructor runs for in main
3 destructor runs for in main
4 constructor runs for static in main
5 constructor runs for In CreateCrazy
6 constructor runs for Local static in CreateCrazy
7 constructor runs for In CreateCrazy
7 destructor runs for In CreateCrazy
5 destructor runs for In CreateCrazy
8 constructor runs for after CreateCrazy in main
8 destructor runs for after CreateCrazy in main
100 destructor runs for in main
6 destructor runs for Local static in CreateCrazy
4 destructor runs for static in main
2 destructor runs for Static Global before main after CreateCrazy
1 destructor runs for Global before main before CreateCrazy
Press any key to continue
```

Local Static is the last one of all time. Just the Global is destruct after it!