2.2C++类

类的定义与实现

类是对某一类对象的抽象，对象是类的具体实例。

C++将对象的属性抽象为**数据成员**，将对象的行为抽象为**成员函数**。数据成员又称**成员变量**，成员函数又称为**方法**。
C++类定义的基本形式：

```
class <类名>
{
  private:
          <私有数据成员和私有成员函数的声明列表>;
  public:
          <公有数据成员和公有成员函数的声明列表>;
  protected;
          <保护数据成员和保护成员函数的声明列表>;
};
```

例子：类的定义

```
// CTime.hpp
class CTime
{
private: //仅可内部使用
    int hour;
    int minute;
    int second;
public:
    void setTime(int h, int m, int s);
    void showTime();
};
```

```
// CTime.cpp
void CTime::setTime(int h, int m, int s)
{
    hour CTime::setTime(int h, int m, int s)
    {
        hour = (h >= 0 && h < 24) ? h : 0;
        minute = (m >= 0 && m < 60) ? m : 0;
        second = (s >= 0 && s < 60) ? s : 0;
}

void CTime::showTime()
{
    cout << hour << ":" << minute << ":" << second;
    cout << endl;
}
```

```
// main
void eg3_1()
```

```
{
  CTime t1, t2, *pTime = &t1;  //用CTime类定义了两个对象和一个指针

  t1.setTime(8, 10, 1);
  t2.setTime(20, 30, 30);
  t1.showTime(); //对象. 方法
  t2.showTime();
  pTime->showTime();
}

int main()
{
  eg3_1();
  return 0;
}
```

建议使用方法

```
#include<bits/stdc++.h>
using namespace std;

// CTime.hpp
class CTime
{
    private:
        int hour;
        int minute;
        int second;
    public:
        void setTime(int hour = 0, int minute = 0, int second = 0);
        CTime(int hour = 0, int minute = 0, int second = 0);

        friend ostream& operator<<(ostream& os, const CTime& time)
        {
            os << setw(2) << setfill('0') << time.hour << ":";
            os << setw(2) << setfill('0') << time.minute << ":";
            os << setw(2) << setfill('0') << time.second;
            return os;
        }
};

void CTime::setTime(int hour, int minute, int second)
{
    this->hour = (hour >= 0 && hour < 24) ? hour : 0;
    this->minute = (minute >= 0 && minute < 60) ? minute : 0;
    this->second = (second >= 0 && second < 60) ? second : 0;
}

CTime::CTime(int hour, int minute, int second)
{
    setTime(hour, minute, second);
}

void eg3_1()
{
    CTime t1(12, 34, 56), t2(23, 59, 59);;
```

```
    CTime *pTime = &t1;

    cout << "t1 = " << t1 << endl;
    cout << "t2 = " << t2 << endl;
    cout << "pTime = " << *pTime << endl;
}

int main()
{
    eg3_1();
    return 0;
}
```

构造函数和析构函数

- 如何**初始化**成员变量?
  - 不能在类定义时进行，为什么?
  - 何时进行? ==>对象创建时
- 成员函数的执行**时机**
  - 一般时机：被调用时
  - 对象创建时：**构造函数()**
  - 对象释放时：析构函数
  - 拷贝（赋值、实参传递）时：拷贝构造函数
  - ……

构造函数和析构函数

- 构造函数的名称与**类名相同**，析构函数的名称必须在**类名前加上"~"**符号。
- 构造函数和析构函数不能指定任何返回值类型，包括void返回类型。构造函数可以有参数，析构函数不能有参数。
- 若没有定义构造函数和析构函数，系统会**自动**为类**添加**默认的构造函数和析构函数（不带参数，函数体为空）。

例子：构造函数和析构函数

```
#include <bits/stdc++.h>
using namespace std;


// CTime.hpp
class CTime
{
private:
    int hour;
    int minute;
    int second;

public:
    void setTime(int hour = 0, int minute = 0, int second = 0);

    CTime(int hour = 0, int minute = 0, int second = 0);
```

```
    ~CTime();


    friend ostream &operator<<(ostream &os, const CTime &time)
    {
        os << setw(2) << setfill('0') << time.hour << ":";
        os << setw(2) << setfill('0') << time.minute << ":";
        os << setw(2) << setfill('0') << time.second;
        return os;
    }
};


void CTime::setTime(int h, int m, int s)
{
    hour = (h>=0 && h<24) ? h:0;
    minute = (m>=0 && m<60) ? m:0;
    second = (s>=0 && s<60) ? s:0;
}


CTime::CTime(int hour, int minute, int second)
{
    this->setTime(hour, minute, second);
    cout << *this << ", constructor" << endl;
}



CTime::~CTime()
{
    cout << *this << ", destructor" << endl;
}




int main()
{

    CTime t1(12, 20), t2(23, 30);
    CTime* pTime = &t1;


    cout << t1 << endl;
    cout << t2 << endl;
    cout << *pTime << endl;
    return 0;
}

```

例子：构造函数的2种格式

V1推荐

```
CTime::CTime(int hour, int minute, int second)
{
    this->hour = (hour>=0 && hour<24) ? hour:0;
    this->minute = (minute>=0 && minute<60) ? minute:0;
    this->second = (second>=0 && second<60) ? second:0;
    cout << *this << ", constructor" << endl;
}
```

V2

```
CTime::CTime(int hour, int minute, int second):hour(hour), minute(minute), second(second)
{}
```

拷贝构造函数

- 是否可以用对象初始化对象?

```
int x = 0;
int y = x;

CTime t1(8, 10, 1);
CTime t2 = t1; //?
```

- 拷贝构造函数（copyconstructor）

```
CTime(CTime& t1);
CTime::CTime(CTime& t1)
{
    this->hour = t1.hour;
    this->minute = t1.minute;
    this->second = t1.second;
}
```

拷贝构造函数

- 若未定义拷贝构造函数，编译系统**自动**为类**添加**一个默认的拷贝构造函数。
- 默认拷贝构造函数采取**浅拷贝**。
- 浅拷贝
  - 只进行**基本数据类型**的拷贝。
  - 当数据成员为指针类型时，存在潜在**危险**，即两个对象的指针指向同一内存区域。
- 调用时机：
  - 用对象初始化对象。
  - 函数参数传递（此时参数为一个对象）

例子：构造、析构、拷贝构造函数

```
void demo_copy_constructor(CTime t1);
void eg3_5()
{
    CTime t1(8);
```

```
    CTime t2 = t1;

    t2.setTime(11, 30);
    cout << "t2 = " << t2 << endl;
    demo_copy_constructor(t2);
}
void demo_copy_constructor(CTime t1)
{
    cout << t1 << ", demo_copy_constructo..."<< endl;
}
```

问题：

- 不初始化，可以吗？
  - 可以
- 不定义构造或析构函数，可以吗？
  - 可以，系统会自动添加默认构造函数和激活函数
- 没有拷贝构造函数，会发生什么？
  - 采用浅拷贝
- 默认拷贝构造函数的风险到底是什么？
  - 浅拷贝存在风险
- C++的构造函数和析构函数的设计，还可以优化吗？
  - 将构造函数和析构函数定义成和类名无关的

例子：没有构造、析构函数

```
class CTime
{
    private:
        int hour;
        int minute;
        int second;
    public:
    void setTime(int hour = 0, int minute = 0, int second = 0)
    //省略输出流重载...
};
//省略setTime()函数的定义...

//没有构造、解析函数
void eg3_6_1()
{
    using namespace N_CTIME_1;
    CTime t1, t2;
    cout << "t1 = " << t1 << endl;
    cout << "t2 = " << t2 << endl;
}
```

```
t1 = -272632624:32766:80758
t2 = 80800:01:-1732163744
```

系统自动加上的构造函数

```
class CTime {
private:
    int hour;
    int minute;
    int second;
public:
    void setTime(int hour=0, int minute=0, int second=0);
    CTime(); //系统自动添加的默认构造函数（定义）
    ~CTime(); //系统自动添加的默认析构函数（定义）
    //省略输出流重载...
};
```

```
#include "CTime.hpp"

CTime::CTime() { //系统自动添加的默认构造函数（实现）

}
CTime::~CTime() { //系统自动添加的默认析构函数（实现）

}
//省略setTime的实现...
```

例子：没有拷贝构造函数的风险

```
1  class CDog
2  {
3      public:
4      char* name;
5      CDog(char* name)
6      friend os & operator << (ostream & os, const CDog & dog1)
7      {
8          os << "I'm a dog, name is "<<dog1.name;
9          return os;
10     }
11 };
12 CDog::CDog(char* name)
13 {
14     this->name = name;
15 }
16 void eg3_6_2()
```

```
{
    using namespace N2_CDOG;
    char name1[] = "Lao Hei";
    CDog dog1(name1), dog2 = dog1;

    dog2.name[0] = 'H';
    cout << "dog1:"<< dog1 << endl;
    cout << "dog2:"<< dog2 << endl;

}
```

浅拷贝

this指针

- this指针是一个**指向当前对象的特殊指针**。
- 每个非静态成员函数隐藏有一个this指针的函数参数，当通过一个对象调用成员函数时，编译器将把当前对象的地址传递给this指针。
-

```
//我们的代码
void CTime::showTime() {
    cout << hour << ':' << minute << ':'
         << second << endl;
}
```

```
//编译器处理的代码
void CTime::showTime(CTiime* const this) {
    cout << this->hour << ':' << this->minute
         << ':' << this->second << endl;
}
```

```
CTime t1(11, 30);
t1.showTime(); //我们的代码
t1.showTime(&t1); //编译器处理的代码
```