C++语言开发基础

目录

[1、面向对象 1](#_Toc445590515)

[2、命名空间 - 避免类名冲突 1](#_Toc445590516)

[3、类的继承 – 可以直接访问父类定义的方法 1](#_Toc445590517)

[4、构造方法和析构方法 1](#_Toc445590518)

[5、执行父类的构造方法 1](#_Toc445590519)

[6、执行父类的方法 2](#_Toc445590520)

[7、实函数、虚函数、纯虚函数、函数重写 2](#_Toc445590521)

[8、纯虚类 2](#_Toc445590522)

[9、函数重载 – 函数名称一样，参数不同 2](#_Toc445590523)

[10、运算符重载 3](#_Toc445590524)

[11、伪函数 – 实际上是一个类或者结构体 3](#_Toc445590525)

[12、函数指针 4](#_Toc445590526)

[13、引用 6](#_Toc445590527)

[14、友元类 6](#_Toc445590528)

[15、标准库容器的基本用法 6](#_Toc445590529)

[16、字符串常用操作 7](#_Toc445590530)

[17、文件操作 8](#_Toc445590531)

# 1、面向对象

头文件和静态库，不提供代码分享给其他人

头文件 – 申明

类中 - 实现

new/delete

malloc/free

# 2、命名空间 - 避免类名冲突

创建：

Namespace yx {

}

使用：

yx::类名

using namespace yx;

# 3、类的继承 – 可以直接访问父类定义的方法

this指针

# 4、构造方法和析构方法

构造方法 – 初始化操作 – new创建指针对象

析构方法 – 删除当前类的对象 - 值对象在代码块执行完毕后删除或者指针对象delete

创建值对象时会先调用构造方法再调用析构方法

# 5、执行父类的构造方法

Man::Man(**int** age):yx::People::People(age, 1) { }

# 6、执行父类的方法

在子类方法中写一个方法，调用父类的方法

**void** Man::sayHello() {  
 yx::People::sayHello();

}

直接在main函数中使用时，可以直接指定使用子类方法或者父类方法

Man \*m = **new** Man(21);  
m -> People::sayHello();

# 7、实函数、虚函数、纯虚函数、函数重写

虚函数 – 函数重写 – 动态绑定 – 将基类函数使用virtual声明

纯虚函数 – 基类中方法定义而不实现，由子类实现 – “virtual” “=0”

**virtual void** eat() = 0;

# 8、纯虚类

抽象类 – 在类中包含纯虚函数，那么这个类就叫做抽象类

纯虚类 – 类中的所有成员方法都是虚函数 – 相当于Java中的接口

# 9、函数重载 – 函数名称一样，参数不同

C++中的字符串是string类型

C中的字符串是char\*类型

转换：

string name = **"yx"**;

(**char** \*)name.c\_str()

string str = **"Hello "**;  
str += name; //字符串连接

# 10、运算符重载

**void operator**+=(Point p) {  
 add(p);  
}

Point p = Point(10, 10);//值对象  
p += Point(13, 13);

Point \*p = **new** Point(6,6); //指针对象  
(\*p) += Point(2,2); //都是通过值进行重载

# 11、伪函数 – 实际上是一个类或者结构体

重载“（）”，将伪函数作为变量来传递，通过这种方式将类当做一个函数来对待，方便

**void** hello(){  
 printf(**"Hello\n"**);  
}  
  
**class** Hello{  
  
**public**:  
 **void operator**()(){  
 printf(**"Hello Cpp\n"**);  
 }  
};  
  
**int** main() {  
  
 hello();  
   
 Hello h;  
 h();  
 **return** 0;  
}

# 12、函数指针

**函数指针**是指向函数的指针变量，即**本质是一个指针变量**

指向函数的指针包含了函数的地址，可以通过它来调用函数。声明格式如下：

类型说明符 (\*函数名)(参数)  
 例如：void (\*fptr)();

**指针函数**是指带指针的函数，即**本质是一个函数**。函数返回类型是某一类型的指针

类型标识符 \*函数名(参数表)

例如：int \*f(x，y);

首先它是一个函数，只不过这个函数的返回值是一个地址值。指针函数一定有函数返回值，而且在主调函数中，函数返回值必须赋给同类型的指针变量

#include **<iostream>**#include **<stdio.h>  
  
using namespace** std;  
  
**class** Object;  
**typedef void**(Object::\*SayHi)();  
  
  
**class** Object {  
**public**:  
*// void (Object::\*sayhi)();*SayHi sayhi;  
};  
  
**class** Hello:**public** Object{  
**public**:  
 Hello(){  
*// sayhi = (void (Object::\*)())(&Hello::helloSayHi);* sayhi = (SayHi)(&Hello::helloSayHi);  
 (**this**->\*sayhi)();  
 }  
 **void** helloSayHi(){  
 printf(**"Hello CPP\n"**);  
 }  
};  
  
**int** main() {  
 Hello \*h = **new** Hello();  
 **delete** h;  
 **return** 0;  
}

应用：延时执行某个方法的代码

#include <iostream>

#include <thread>

#include <unistd.h>

class Object;

typedef void(Object::\*SayHi)();

typedef void(Object::\*CallaterHandler)();

void threadFunc(Object \*target,CallaterHandler handler,int delay){

sleep(delay);

(target->\*handler)();

}

void callater(Object \*target,CallaterHandler handler,int delay){

std::thread t(threadFunc,target,handler,delay);

t.join();

}

#define CH(fp) (CallaterHandler)(&fp)

class Object {

public:

SayHi sayHi;

};

class Hello:public Object {

public:

Hello(){

// sayHi = (SayHi)(&Hello::HelloSayHi);

//

// (this->\*sayHi)();

callater(this, CH(Hello::HelloSayHi), 3);

}

void HelloSayHi(){

printf("Hello CPP\n");

}

};

int main(int argc, const char \* argv[])

{

Hello \*h = new Hello();

delete h;

// insert code here...

std::cout << "Hello, World!\n";

return 0;

}

# 13、引用

值传递 – 执行内存拷贝 – 消耗时间

引用传递 – 不执行内存拷贝 – 程序运行高效

也可以使用指针

# 14、友元类

将类声明为友元，来访问另一个类的私有成员

friend class B

# 15、标准库容器的基本用法

#include **<iostream>**#include **<list>**#include **<string>**#include **<map>  
  
using namespace** std;  
  
**int** main() {  
 list<string> l;  
 l.push\_back(**"Hello"**);  
 l.push\_back(**"World"**);  
  
 list<string>::iterator it;  
 **for** (it = l.begin();it != l.end();it++){  
 cout<<\*it<<endl;  
 }  
  
 map<string,string> m;  
 m.insert(pair<string,string>(**"a"**,**"A"**));  
 m.insert(pair<string,string>(**"b"**,**"B"**));  
 cout << m.at(**"a"**) << endl;  
  
 m[**"d"**] = **"D"**; *//C++重载了[]运算符* cout << m.at(**"d"**) << endl;  
   
 **return** 0;  
}

# 16、字符串常用操作

字符串连接：重载了运算符+=

字符串与其他类型连接：

stringstream ss;  
 ss<<**"A "**;

#include **<iostream>**#include **<sstream>  
  
using namespace** std;  
  
**int** main() {  
 string str;  
 str+=**"a "**;  
 str+=**"b"**;  
 cout << str <<endl;  
  
 stringstream ss;  
 ss<<**"A "**;  
 ss<<200;  
 ss<< **"B "**<<**"C"**<<100;  
  
 cout<<ss.str();*//c++语言字符串* cout<<ss.str().c\_str();*//c语言字符串* **return** 0;  
}

# 17、文件操作

#include **<iostream>**#include **<fstream>**#include **<sstream>  
  
using namespace** std;  
  
**int** main() {  
*// ofstream of("data.txt");//写  
// of<<"Hello Cpp\n";  
// of.close();* ifstream inf(**"data.txt"**);*//读  
// char c;  
// inf >> c;* stringbuf sb;  
 inf >> &sb;  
  
*// cout << c;* cout << sb.str();  
  
*// cout << "Hello, World!" << endl;* **return** 0;  
}