组号： 11

山东师范大学
SHANDONG NORMAL UNIVERSITY

信息科学与工程学院课程实验报告

《面向对象程序设计》

姓名： 陈冰

学号： 201711010250

班级： 计工本 1702

教师： 张庆科

时间：2018/10/15

面向对象程序设计实验报告

姓名	陈冰	班级	计工本 1702	学号	201711010250	组号	11
时间	2018/10/15	地点	E315	周次		页码	8
源码	☐ 无源码 ☐ 文档源码 ☑ 托管源码						

报告内容

实验报告要求：请围绕实验目的、实验内容、实验过程及步骤(可添加文字、矢量图)、实验结论与分析进行撰写。凡涉及源代码内容可给出完整源码并附上源码 Github 托管网址（请务必按照条目书写）。

1. **实验目的**

熟悉上机编程环境，掌握代码编译和调试过程。

强化课堂知识点，提高编程实战能力。

2. 实验内容

a. 源码提交要求：请将第二章的各个案例对应的程序项目上传到“个人 github 作业仓库”内；

b. 实验报告提交要求：报告以本章知识点总结形式展开，实验报告务必附带项目托管网址，采用 pdf 格式上传报告班级云课堂。

3. 知识总结

- **C++ 中 cin,cout 包含在头文件 #include<iostrem> 及 using namespace std 或者 #include<iostrem.h>中**
- **C 语言换行:\n，C++换行：endl**
- **C++用//单行注释；C 语言用/*……*/注释，C++也支持**
- **嵌套：/*……*/不能嵌套，其他都可**
- **const [常量类型] 符号常量名=表达式；**
- **const 与指针结合**

指针名 / 比较方面	指向常量的指针	常指针	指向常量的常指针
定义形式	const 类型名 *指针名=地址值	const * 类型名 指针名=地	const 类型名 * const 指针名=地址值
实例	int x=5; const int *p=&p;	int x=5; int*const p=&p	int x=5; const int * const p=&x;
可修改	p,x	*p,x	x

不可修改	*p	p	*p,p

报告内容

- 强制转换类型：

  C：X=(int)f,X=(int)(f+23.4)

  C++ : x=int (f)

- boolalpha:输出为 true 或 flase

  noboolalpha:输出恢复为 1 或 0

- 名字空间的定义：namespace 名字空间名称{……；}

  使用名字空间的内容:

  - ✧ Using namespace std;（一劳永逸）
  - ✧ 名字空间名称：：局部内容名（随用随取）
  - ✧ Using namespace 名字空间名称：：局部内容名（按需打包）

- 局部变量随用随定义

  函数代码较长时，在靠近使用变量的位置定义

  函数代码较短时，将局部变量集中在函数开始时定义

  若出现同名的局部量，按最近范围变量优先原则处理

- 域解析符：：扩大全局变量的可见范围，对被隐藏的同名全局变量进行访问
- 形式参数可带有默认值

  形式参数的指定顺序是从右到左

  默认参数值只能给定一次。如果在原型中已经给定，则在函数定义首部不能再提供参数值；如果定义在先的函数无需原型声明，则默认参数值在函数定义首部直接给出。

- 内联函数 inline

  考虑了代码的语义

  在内联函数内不允许用循环语句，递归语句，开关语句等

- 函数重载（同名不同参：相同的函数名，形式参数的个数类型顺序的一个或某个方面不同）

报告

项目 / 实例组	函数原型 1	函数原型 2	重载是否正确	原因分析

内容

1	void Fun(int);	void Fun();	正确	形参个数不一样
2	void Fun(int);	void Fun(double);	正确	形参类型不一样
3	void Fun(int，double);	void Fun(double，int);	正确	形参顺序不一样
4	void Fun(int);	void Fun（double，char）;	正确	形参个数和类型都不一样
5	void Fun(int);	int Fun（int）;	错误	两者仅仅返回值类型不一样
6	void Fun(int);	void Fun（）;	错误	两者虽然形式参数个数不同，但调用时不提供实参，将无法判断哪一个函数被调用
7	void Fun(int);	void Fun(int &);	错误	二者是值形式参数和引用参数的区别，如果有 Fun（a）则无法判断调用哪个函数

注：返回值是否相同不是函数重载的判断条件

- 引用 --给变量起别名

  数据类型 & 引用名=一个已定义的变量名；

  温馨提示：不能建立 void 类型的引用，引用的引用，指向引用的指针，引用数据

```
int x=10,a[10];
void & r=x;          //错误，不能建立 void 类型引用
int &&r=x;           //错误，不能建立引用的引用
```

```
int &*p=x;          //错误，不能建立引用的指针
int &ra[10]=a;      //错误，不能建立引用数组
```

- 引用作为形式参数　　--给实际参数变量起别名

作为函数的形式参数，用于在被调用时成为实际参数变量在被调函数中的别名，通过对引用的访问和修改达到对实际参数变量进行操作的效果或者说引用参数使得实际参数变量的作用域“扩大”到原先无法进入的被调函数中

注：与引用形式参数对应的只能是变量，而不能是常量和表达式

同时可以在引用参数之前加 const 使其成为常引用

比较方面 \ 形式参数	常引用	值形式参数
实际参数	变量	可以是常量，变量，表达式
是否需要重新分配内存空间	否	是

- 引用和指针的区别

指针或引用	引用	指针
实质	变量（对象）的别名	指向变量（对象）的地址
初始化	必须进行初始化	不是必须，时候可以赋值或申请地址
可变性	一旦成为某变量的别名，程序中不可更改	可以随时更改，指向不同的变量
是否有空值	不可为空，必须是某变量的别名	可以获得空指针 NULL，表示不指向任何变量
空间占用情况	不另外占用空间，只是所指代变量的别名	另外占用 4 字节空间，存放某变量的地址
访问方式	简洁方便，直接用引用名	相对繁琐一些，需要用指针名配合
安全有效性	安全，总是有效，因为是某	不安全，不一定总有效，可

	一变量的别名	能误指或出现野指针

- 引用作为返回值
  - ✧ 类型名 & 函数名（形式参数表）；
  - ① 可以作为独立的函数调用语句，如 Fun（a,b,c）；
  - ② 可以作为表达式中的某一个运算对象使用，如 d=Fun（a,b,c）；
  - ③ 可以作为左值使用（即将函数的调用放在赋值号左边，当做变量使用），如 Fun（a,b,c）=20
  - ✧ 对于引用返回值函数的 3 个要求
  - ① return 后面只能跟变量，不能是常量和表达式
  - ② return 后面变量的内存空间在本次函数调用后应当仍然存在
  - ③ return 后面返回的不能是常引用
- 动态内存空间管理

比较方面 \ C/c++	C	C++

申请动态内存方式	int *ptr;
	ptr=(int*)malloc(sizeof (int));
	ptr=(int*)molloc(10*sizeof (int));
	Ptr=(int*)calloc(10,sizeof (int));
		int *ptr;
		ptr=new int;
		ptr=new int[10];

释放动态内存方式	free(ptr);	delete ptr 或 delete []指针变量名

- 异常处理

  通过 throw 抛出异常，try-catch 块检测异常，捕捉并处理异常

  try 在前 catch 在后，try 和 catch 之间不能有其他语句，一个 try 可以对应多个 catch 块

4. 实验源码和源码地址

```
git@github.com:dreamm2776181586/Chapter-II-Source-Code.git
```

㊟：可根据内容自行拓展页面，作业内容尾部尽量不要留有空白