

分组:\_\_\_\_1\_\_\_\_



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:	展家慧
学号:	201711010111
班级:	计工本 1 班
教师:	张庆科
时间:	2018-10-5

面向对象程序设计实验报告

姓名	展家慧	班级	计工本 1 班	学号	201711010111	组号	1
时间	2018-9-27	地点	信息楼 E312	周次	4	页码	共 3 页
源码	<input type="checkbox"/> 无源码 <input type="checkbox"/> 文档源码 <input checked="" type="checkbox"/> 托管源码						
报 告 内 容	<p>实验目的：</p> <p>一、     熟悉 VS 编程环境。</p> <p>二、     进一步掌握 C++对 C 的扩充和扩展。</p> <p>三、     掌握将源码上传 GitHub 个人仓库的方法。</p> <p>实验要求：</p> <p>四、     总结第二章知识点。</p> <p>五、     实现第二章例程序的代码。</p> <p>六、     将代码上传至 GitHub 个人仓库内。</p> <p>实验过程：</p>						
	<p>第二章知识点总结</p>						
	<div><div>· 综述</div><div><div>C++对C的改进及扩展</div><div><div>改进</div><div>1.基本控制 2.特色函数</div></div><div><div>扩展</div><div>1.新增引用 2.动态内存 3.异常处理</div></div></div><div>· 改进</div></div>						

## 一、基本控制

1. c++源程序的扩展名：.cpp

2. 用 I/O 流输入输出

- 需要#include<iostream.h>或者#include<iostream>及 using namespace std;
- cin>>变量 1>>变量 2……; (这种方式不能输入带空格的字符串)
- cout<<表达式 1<<表达式 2……; (endl 与换行符\n 等效)

3. 注释

- 新增//单行注释方式
- /\*……\*/不能嵌套，其他的可以尽情嵌套

4. const

- 与宏不同，#define PI 3.1415926  

const double PI=3.1415926;
- 一定要加分号;;;
- 与指针？

比较项目	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;	类型名 const 指针名=地址值;	const 类型名 const 指针名=地址值;
示例	int x=5; const int *p=&x;	int x=5; int * const p=&x;	int x=5; const int * const p=&x;
可修改	p , x	*p , x	x
不可修改	*p	p	*p , p
说明	指向常量的指针在定义形式参数时最常用,以保护对应实参不被修改。 例如: double Sum( const double *p); 与p对应的实参数组中所有元素将不会被修改		

指向常量的指针：

- 指针本身可以改变，可以再次指向另外的对象。
- 不能通过指针修改其所指向的对象的值。
- 对象本身的值可以直接修改。

常指针：

- 指针本身的值不能改变，即不能再指向另外的对象。
- 可以修改指针所指对象的值。

<p>报 告 内 容</p>	<div><div>5. 强制类型转换</div><div><ul style="list-style-type: none"><li>源表达式加括号，形如：</li></ul><pre>float f=100.00;  int x=int(f);</pre></div><div>6. bool 类型</div><div><ul style="list-style-type: none"><li>1，真，true</li><li>2，假，false</li><li>bool 类型只占 1 字节</li><li>boolalpha true/false</li><li>unboolalpha 0/1</li></ul></div><div>7. 名字空间 namespace</div><div><ul style="list-style-type: none"><li>namespace 里什么类型都可以放</li><li>右大括号后不加分号</li><li>三种方法</li></ul></div><div>二、特色函数</div><div>1. 局部变量</div><div><ul style="list-style-type: none"><li>随用随定义</li><li>如果同名，按最近范围变量优先处理</li><li>代码较长时，一般在靠近变量使用的位置定义</li><li>代码较短时，在函数开始处定义</li></ul></div><div>2. 域解析符::（扩大全局变量的可见范围）</div><div><ul style="list-style-type: none"><li>加在同名变量前，访问隐藏的同名全局变量</li></ul></div><div>3. 形式参数可带有默认值</div><div><ul style="list-style-type: none"><li>形参默认值一定要从右往左依次指定</li></ul><p>例如：void fun(int i ,int j = 5,int k = 10)</p><ul style="list-style-type: none"><li>默认值在声明时指定。若无声明，函数在主函数前被定义，则在函数定义首部给出。</li><li>若一个参数既有形式参数的默认值，有提供了实际参数，则实参优先。</li></ul></div></div>
----------------------------	---

#### 4. 内联函数 inline

- 在函数前加 Inline 代替宏定义

宏定义: #define multiply(x,y) x\*y

内联函数: inline int multiply(int x,int y)

{return x\*y;}

- 内联函数的定义和声明放在一起, 在主函数之前的位置
- 内联函数比宏安全的原因是编译器会考虑代码的语义
- 工作原理: 编译时用内联函数代码替换掉调用表达式
- 内联函数不允许用循环语句或者开关语句。递归函数不能用内联函数
- 内联函数实际上适用于 1—5 行的小函数

#### 5. 函数重载

- 形参的个数、类型、顺序不同的几个同名函数 (同名不同参)

### · 扩展

## 一、新增引用

#### 1. 引用&

- 别名, 用作函数参数及函数返回值
- 声明时用&, 使用时不用&
- 声明一个引用的时候必须**初始化**
- 不另外占用空间。与大名共用一个空间
- 只能建立单变量的引用, 不能建立 void 类型引用、引用的引用、指向引用的指针、引用数组

- 小名与大名的值始终保持一致

• int x = 5;

int &r = x;

#### 2. 引用作为形式参数

- 可以通过这种方式修改实际参数

#### 3. 引用作为返回值

- int &fun(int &x,int &y,int z)

```
fun(a,b,c);  
D=fun (a,b,c) ;  
fun(a, b, c)=20;
```

- 三个特别要求

## 二、动态内存

- **库函数：**头：<stdlib.h> malloc 与 free 配对
- **关键字：** new 与 delete 配对
- 使用 new 或者 new[] 之后，一定要用 delete 或者 delete[] 释放内存，否则会产生内存垃圾
- void 作为指针类型时，表示不确定的类型

## 三、异常处理

- 错误分为**语法错误**和**运行错误**两大类，**语法错误**包括**编译错误**和**链接错误**，**运行错误**包括**不可预料的逻辑错误**和**可以预料的运行异常**。
- 异常处理的实现：
  - 检查异常（try）
  - 抛出异常（throw）
  - 捕捉异常（catch）

第二章源代码地址：<https://github.com/zhanjiahui/c-homework.git>

--	--

--	--



--	--

④：可根据内容自行拓展页面