C++程序设计作业1

姓名：\_\_\_\_\_\_\_\_ 学号：\_\_\_\_\_\_\_\_ 课程编号：\_\_\_\_\_\_\_\_

1. 选择题

1. （不定项选择题）大型项目开发中，约定合理的编码规范有助于减少编码错误，提高协作效率。以下关于C++的编码习惯描述中，不推荐的有（ABDE）

A. 使用宏#define SQUARE(X) X\*X替换简单平方函数，从而减少过多的函数调用开销。

B. 对于namespace，应尽量多使用using指示语句如using namespace std;来避免单行代码长度过长。

C. 合理的注释可大幅增加代码的可读性，常见的注释场景包含：接口函数注释、临时/需优化逻辑To-do注释、复杂逻辑说明注释等。

D. 项目开发场景中，文件过多会导致编译复杂耗时，因此应该尽量将业务逻辑集中在少量文件、少数大函数中。

E. 全局变量具有多文件可见、读写效率高等优点，应当尽可能多使用无命名空间限定的全局变量进行数据共享。

2. （不定项选择题）C++社区中有一条经典的编码建议：“尽可能使用const”。下述关于const关键字的说法或使用方式正确的有（ADE）

A. 令函数返回常量值，可以减少因使用错误造成的意外，如const int func()函数在判断条件if (func() == val)误输入为if (func() = val)时会在编译阶段报错，避免发生预期外的效果。

B. 对于语句const char\* p1 = str;和char const\* p2 = str;(str为已定义的char数组类型变量)，两者表示的含义相同，均表示指向C风格字符串的常指针（指针const，所指内容无限定）。

C. 类的const成员函数不能修改类对象的任何成员变量（包含普通成员变量及静态成员变量），除非成员变量是mutable变量。

D. 对于自定义类型，在函数传参中，可以使用引用避免参数拷贝，此时增加const限定，则可进一步避免函数内部修改此引用参数值。

E. 使用const变量替换C语言风格的宏常量，可以让编译器支持该常量的静态类型检查。

3. （不定项选择题）static关键字在C++语法中有许多使用场景，下述关于static和全局变量说法错误的有（AB）

A. 类内由static修饰的成员变量称为静态成员变量，静态成员变量只可通过类访问，不能通过对象访问。

B. 全局变量无论是否添加static修饰符，生命周期都和程序的生命周期相同，且所有文件均可见。

C. 内置类型全局变量无显示初始化情况下，会被程序自动初始化为0。

D. 类内由static修饰的成员函数称为静态成员函数，静态成员函数与普通成员函数的主要区别是静态成员函数不包含指向对象本身的this指针，所以无法在静态成员函数中访问非静态成员变量。

E. 若一般函数中包含static修饰的静态变量，则此函数不应作为内联函数使用。

4. （不定项选择题）下列关于C++函数重载的说法正确的有（AC）

A. C++支持函数重载，而C语言不支持，所以在C++中使用C语言函数时，需要添加extern “C”声明。

B. 对于普通函数，函数名相同，参数列表（参数个数、顺序）及返回值任意一处存在区别均可形成函数重载

C. 函数名与参数列表相同的成员函数，可通过const限定符进行重载

D. 类的析构函数可以支持函数重载，所以一个类内可定义多种同函数名的析构函数

E. void func(int)与void func(const int)无法重载，类似的，void func(int\*)与void func(const int\*)也无法重载

5. （不定项选择题）初始化与赋值是一对容易混淆的概念，关于它们，下述说法正确的有（ABCD）

A. 初始化不是赋值，初始化的含义是创建变量时赋予其一个初始值，而赋值的含义是把对象的当前值擦除，以一个新值来替代。

B. 在类的构造函数大括号内部使用=运算符修改成员变量，均为赋值操作。

C. const常量和引用必须进行显式初始化，且之后不能再进行赋值。

D. 应确定对象被使用前已先被初始化，以避免预期外行为，例如内置类型局部变量在不显示初始化情况下，不被初始化，直接使用可能导致结果异常。

E. C++11支持类内初始化，在类内部定义成员变量const int val\_ = 3;（类内初始值），则此类生成的所有对象包含的val\_值均为3。

二、实践题

6. C++11新增了auto关键字的用法，使用auto关键字可自动推断变量的类型。运行参考程序，回答如下问题：

（1）定义变量val1 ~ val10的语句部分存在错误，列出这些存在问题的语句；

auto val1;

auto val6 = new auto{1, 2};

auto val9 = 0, val10 = 3.14;

（2）注释（1）中有问题的语句后，尝试分析得出其他正常初始化变量的类型（提示：可使用typeid(val4).name()命令输出类型参考信息）；

auto val2 = 1; (int)

auto val3 = 2L; (long)

auto val4 = 0.0 + val2; (double)

auto val5 = new auto(1.0f); (float\*)

auto val7 = 0, \*val8 = &val7; (int, int\*)

（3）运行程序，回答在三种范围for循环中，哪些发生了额外的拷贝操作。

for (auto val : vec) {}

参考程序：

#include <iostream>

#include <vector>

class A {

public:

A() { std::cout << "construct" << std::endl; }

~A() { std::cout << "deconstruct" << std::endl; }

};

int main() {

auto val1;

auto val2 = 1;

auto val3 = 2L;

auto val4 = 0.0 + val2;

auto val5 = new auto(1.0f);

auto val6 = new auto{1, 2};

auto val7 = 0, \*val8 = &val7;

auto val9 = 0, val10 = 3.14;

std::vector<A> vec(3);

for (auto val : vec) {}

for (auto& val : vec) {}

for (const auto& val : vec) {}

return 0;

}

7. 在C++中，一般配合使用new([])/delete([])进行堆上自由内存的申请及释放。实际上，还存在一种名为placement new的特殊内存分配方式，运行参考程序，回答以下问题：

（1）buffer数组、p1所指对象、p2所指对象使用的是哪里的内存？（堆/栈/静态区/…）

buffer数组——栈、p1所指对象——堆、p2所指对象——栈

（2）能否像释放p1一样释放p2？尝试解释原因。

不能，p2使用定位new未开辟新内存，使用了buffer已分配的栈内存，若像p1一样释放，函数退出时会再次释放栈内存buffer，双重释放导致错误。

（3）为何要循环调用p2各元素的析构函数？尝试解释原因。

释放p2内结构体A内部b\_分配的内存，防止内存泄漏。

参考程序

#include <iostream>

#include <string>

struct A {

A() : a\_(3), b\_("b") {}

int a\_;

std::string b\_;

};

int main() {

const int size = 3;

const int bufferSize = 128;

char buffer[bufferSize] = {0};

A\* p1 = new A[size];

A\* p2 = new (buffer) A[size];

std::cout << "p1 is " << p1 << std::endl;

std::cout << "p2 is " << p2 << std::endl;

std::cout << "buffer is " << &buffer << std::endl;

delete[] p1;

for (size\_t i = 0; i < size; ++i) {

p2[i].~A();

}

return 0;

}

三、编程题

8. 利用类的封装特性，实现一个最多只能产生2个对象的类型。

(提示：除直接构造类型对象外，拷贝行为也可能产生新的对象。可使用C++11语法= delete禁止拷贝构造和拷贝赋值函数)

思路：

（1）构造函数、拷贝构造函数、赋值函数、移动构造函数、移动赋值函数均设为私有或=delete。（2）使用静态成员函数调用私有构造函数生成对象实例

（3）使用静态成员变量记录已生成对象个数

（4）已有对象个数超过2个时抛出异常或返回null

（5）析构函数内部析构对象结束时需减少静态成员变量记录的数量