考试科目名称 高级程序设计 C++ （A 卷）

考试方式： 闭卷 考试日期 2014 年 1 月 12 日 教师 郑滔

系（专业） 软件学院（软件工程） 年级 班级

学号 姓名 成绩

题号	一	二	三	四	五	六	七	八	九	十
分数										

注意：所有作答请写直接写在卷面上。

得分 | | 一、简答题（本题满分 30 分，共三题）

1、 请简述 C++程序设计语言的设计理念、演化历程（包括主要的贡献者），并阐明 C 和 C++的关系（本题 15 分）

强大的表述能力，高效
Father of Simula67、Father of OO programming

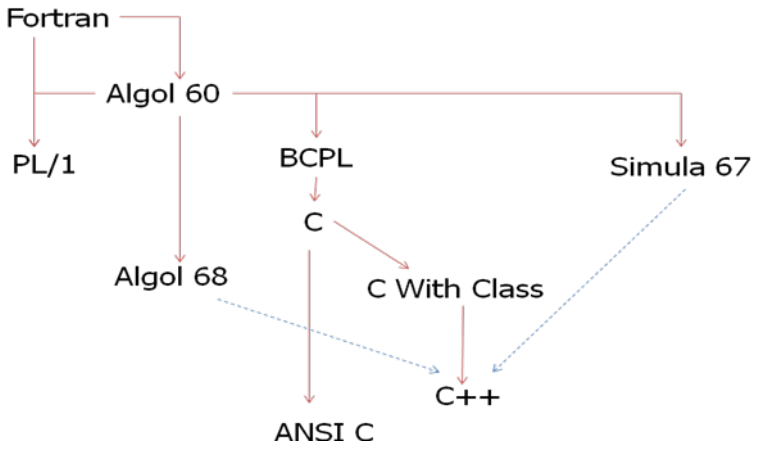
Ole-Johan Dahl、Kristen Nygaard **4 分**

Design PASCAL、Modula2、Oberon
Niklaus Wirth
Structural Programming
E.W.Dijkstra
Father of C、Co-inventing Unix

Dennis Ritchie 、Ken Thompson **2 分**

Design C++

Bjarne Stroustrup **4 分**



5 分

2、 影响表达式值的因素有哪些？请说明之（本题 7 分）

操作符、操作数、优先级、结合性、求值次序、类型转换约定

3、 C++编译系统赋予一个空类，如：class Empty{ } ，哪些成员函数？（本题 8 分）

Empty();
Empty(const Empty&);
~Empty();
Empty& operator=(const Empty&);
Empty *operator &();
const Empty* operator &() const;

得分	

二、请指出以下程序存在的问题（本题满分 10 分，共 2 题）

1、

```
#include <IOSTREAM>
#include <stdlib.h>
#include <time.h>
using namespace std;

class poker{
public:
    unsigned int id;
    poker(){ id = rand() % 13 + 1; } // 产生 1-13 的随机数
};
void main() {
    srand(unsigned(time(NULL)));  // 根据系统时间，设置随机种子值
    poker* p = new poker[5];
    int sum = 0;
    for(int i=0; i<5; i++, p++)
        sum += p->id;
```

```
    cout << "总点数为: " << sum << endl;
    delete p;
}
```

答案：

由于在 for 循环中，移动了 p 指针的位置，导致原先 p 所申请的内存空间的首地址信息丢失，无法归还系统，因此 delete p 时程序会出现异常中止 **5 分**

2、

```
int* get_inputs() {
    int numbers[10];
    int i;
    for(i = 0; i < 10; i ++) {
        cin >> numbers[i];
    }
    return numbers;
}
```

```
int find_max(int* inputs, int size) {
    int i;
    int max = -1;
    for (i = 0; i < size; i ++ ) {
        if (max < inputs[i]) {
            max = inputs[i];
        }
    }
    return max;
}
```

```
void main() {
    cout << "inputs" << endl;
    int* inputs = get_inputs();
    cout << "find the max" << endl;
    int max = find_max(inputs, 10);
    cout << "max: " << max << endl;
}
```

答案：

get_inputs 函数中，返回局部变量。在函数外部使用过程中会造成错误。 **5 分**

得分	

三、请给出以下程序运行结果（本题满分 20 分，共 2 题）

1、

```
#include <IOSTREAM>
using namespace std;

class Vehicle{
public:
    virtual void run(int number = 10){
        cout << "we do not know how to run\n";
    }
```

```
    virtual void stop(){
        cout << “we do not know how to stop\n”;
    }
    void announce(){
        cout << "this is a vehicle\n";
    }
};
class Car:public Vehicle
{
public:
    void run(int number = 60){
        cout << "driving at " << number << " km/h\n";
    }
    void stop(){
        cout <<“brake to stop \n”;
    }
    void announce(){
        cout << "this is a car\n";
    }
};

void main()
{
    Vehicle v1,*v2;
    Car c1;
    v1.run();
    c1.announce();

    v2 = &c1;
    v2->run();
    v2->stop();
    v2->announce();
}
```

答案：

we do not know how to run **2 分**

this is a car **2 分**

driving at 10km/h **2 分**

brake to stop **2 分**

this is a vehicle **2 分**

2、

```
class Error {
public:
    virtual void show(){ cout << "something is error"<<endl;}
};
```

```
class nameError:public Error {
public:
	void show() {
		cout<<"name is error"<<endl;
	}
};
```

```
class ageError:public Error {
public:
	void show() {
		cout<<"age is error"<<endl;
	}
};
```

```
class Person {
private:
	int age;
	char* name;
public:
	void setAge(int a) {
		ageError ag;
		if(a<0||a>100)
			throw ag;
		this->age=a;
	}
	void setName(char* str){
		nameError ne;
		if(str=="exit")
			throw ne;
		this->name=str;
	}
};
```

```
void catcher(int command, Person p){
	try {
		switch(command){
		case 1:
			p.setAge(101);
			break;
		case 2:
			p.setName("exit");
			break;
		}
	}
```

```
	catch(nameError ner){
		ner.show();
	}
	catch(Error er) {
		er.show();
	}
	catch(ageError aer){
		aer.show();
	}
}
```

```
int main(void) {
	Person p;
	catcher(1, p);
	catcher(2, p);
```

```
    cout<<"program end"<<endl;
    return 0;
}
```

答案：

something is error　　**4 分**

name is error　　**2 分**

program end　　**4 分**

得分	

四、程序填充（本题满分 10 分）

下列程序输出如下星状图形，请将程序空白部分补充完整：

```
*
* *
* * *
* * * *
```

```
int printEachLine(int j){
    if (__________)
        return 1;
    else{
        ____________________
        ____________________
        return1;
    }
}

int print(int i){
    if (_________)
        return 1;
    else{
        ____________________
        ____________________
        cout<<endl;
        return1;
    }
}
void main(){
    print(4);
}
```

答案：

```
int printEachLine(int j){
    if (j==0)
        return 1;
    else{
        printEachLine(j-1);
```

1分

2分

```
            cout<<"* ";                                          2分
            return 1;
        }
}

int print(int i){
        if (i==0)                                                1分
            return 1;
        else{
            print(i-1);                                          2分

            printEachLine(i);                                    2分
            cout<<endl;
            return 1;
        }
}
```

得分	

五、（编程题。本题满分 30 分。）

观察者模式：定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。

举个博客订阅的例子，当博主发表新内容的时候，即博客内容发生了改变，那些订阅的读者就会收到通知。博主与读者之间存在种一对多的依赖关系，**博客中可能有多个观察者**（即订阅者），当博客的内容发生变化时，通过 notify 成员函数通知所有的观察者，告诉他们博客的内容更新了。而观察者通过 update 成员函数获取博客的内容信息

请根据以下类图和部分类描述，请用 C++给出相关类的实现（Blog, BlogCSDN，Observer，ConsoleObserver）。

类名Blog：

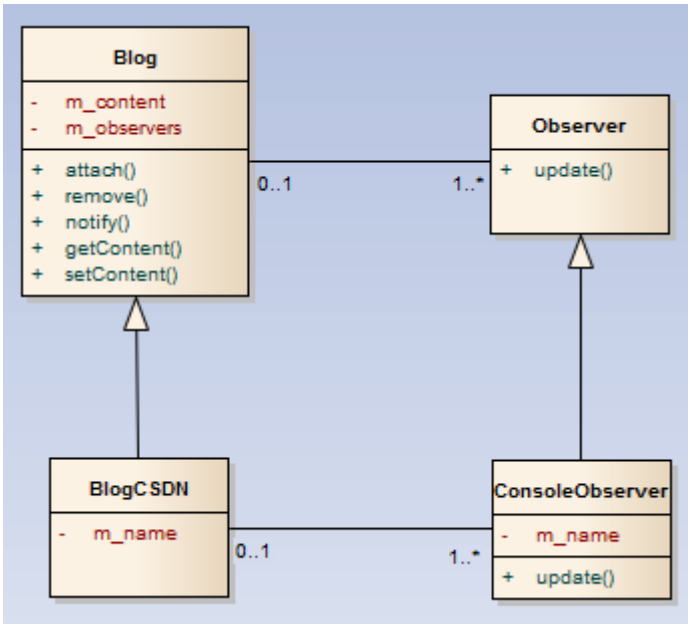
属性	描述
m_content	博客内容（长度< 1024个字符）
m_observers	多个观察者
方法	描述
attach	添加博客听众
remove	删除博客听众
notify	通知所有听众，获取当前博客内容
setContent	设置博客内容
getContent	获取博客内容

类名BlogCSDN:

属性	描述
m_name	博主名称（长度<16个字符）

类名ConsoleObserver：

属性	描述
m_name	观察者名称（长度< 128个字符）
方法	描述
update	获得观察的博客的更新内容，并在控制台显示



答案：

4 个类写出来，并写对继承关系，以及成员变量初始化 **5 分**

写出了 1 对多，即多个观察者 **7 分**

attach、remove、notify、update **每个方法各 4 分**

getContent、setContent **每个方法各 1 分**

```
#include <iostream>
#include <list>

using namespace std;
```

```
//观察者
class Observer
{
public:
	Observer() {}
	virtual ~Observer() {}
	virtual void update() {}
};

//博客
class Blog
{
public:
	Blog() {}
	virtual ~Blog() {}
	void attach(Observer *observer) { m_observers.push_back(observer); }	//添加观察者
	void remove(Observer *observer) { m_observers.remove(observer); }		//移除观察者
	void notify() //通知观察者
	{
		list<Observer*>::iterator iter = m_observers.begin();
		for(; iter != m_observers.end(); iter++)
			(*iter)->update();
	}
	void setContent(char* c) { strcpy(m_content, c); } //设置博客内容
	char* getContent() { return m_content; }	//获得博客内容
private:
	list<Observer* > m_observers; //观察者链表
protected:
	char m_content[1024]; //博客内容
};

//具体观察者
class ConsoleObserver : public Observer
{
private:
	char	m_name[128];
	Blog *m_blog;
public:
	ConsoleObserver(char* name,Blog *blog){
		strcpy(m_name, name);
		m_blog = blog;
	}
	~ConsoleObserver() {}
	void update()	//获得更新内容并输出
```

```
    {
        string content = m_blog->getContent();
        cout<<"Blog content update : " << content << ". My name is : " << m_name << endl;
    }
};

//具体博客类
class BlogCSDN : public Blog
{
private:
    char m_name[16]; //博主名称
public:
    BlogCSDN(char* name){strcpy(m_name, name);}
    ~BlogCSDN() {}

};

//测试用例
int main()
{
    Blog *blog = new BlogCSDN("ZT");

    Observer *observer1 = new ConsoleObserver("GSX", blog);
    Observer *observer2 = new ConsoleObserver("HT", blog);
    Observer *observer3 = new ConsoleObserver("QYG", blog);
    Observer *observer4 = new ConsoleObserver("WLX", blog);

    blog->attach(observer1);
    blog->attach(observer2);
    blog->attach(observer3);
    blog->attach(observer4);

    blog->setContent("C++期末试卷");
    blog->notify();
    blog->remove(observer1);

    delete blog;
    delete observer1;
    delete observer2;
    delete observer3;
    delete observer4;

    return 0;
}
```