C++ 高级程序设计-经验分享

熊丘桓 软件学院
eaglebear@smail.nju.edu.cn

Sunday 31st December, 2023

目录

免责声明分享目的

- 咨询任课老师和助教以获取考试内容和考试题型。
- 分享个人学习经验和工程经验，供参考之用。

设计理念

1. 效率
2. 实用性优于艺术性严谨性
3. 允许一个有用的特征比防止各种错误使用更重要（相信程序员）

设计的结果

- 相比于 Java 并不优美的 OO 语法
- 相比于 C 增加了更反直觉且难以理解的执行逻辑
- 开发端最紧缺的岗位之一：C++ 工程师

但如果你理解了 C++ 的理念和目的，那么以上问题都能迎刃而解


John Backus,
1953, 1977

FORTRAN


Alan J. Perlis ,
1958,1966


E.W.Dijkstra
1960,1972



Ole-Johan Dahl (1978)
Ole-Johan Dahl
Kristen Nygaard
1962, 2001

Algol 60

PL/1

BCPL


Dennis Ritchie
Ken Thompson
1972,1983

Simula 67

C


Donald E.Knuth
1965,1974


C. A. R. Hoare
1966,1980

Algol 68

C With Classes


Niklaus Wirth
1966, 1994

ANSI C

C++

CLu
Ada
.....

C++20


Barbara Liskov
1987, 2008

三条主要脉络

- Algo60：结构化编程，拒绝意大利面条！
- BCPL & C：系统编程，程序员可以摸到 CPU 和内存
- Simula67：OO 编程，梦开始的地方

不同人眼里的 C++

- 算法竞赛选手：C++ = C + STL
- Java 选手：C++ = C + OO
- 面经魔怔人：C++ = &…*$#@
- 软院选手：C++ = 比 CPL 更简单的机试题，面向往年卷复习
- 苏州校区选手：往年卷？什么往年卷？

我猜的根据软院往年卷归纳的考试重点

1. 多态：重载（overload）与重写（override）
2. 函数的运行机制：传值和传引用
   C 语言：你猜我为什么不支持重载？
3. 宏：上古时期的奇技淫巧魔法
4. 常量指针和指针常量：CPL 笔试的漏网之鱼
5. C++ 为什么比 Java 难：五三原则、虚函数、多继承
6. 面向对象编程十大问题

恭喜你！你已经精通 C++ 啦
下面来做几道例题吧

```cpp
void bar(int i) { cout << "bar(1)" << endl; }
void bar(const char c) { cout << "bar(2)" << endl; }
void func(int a) { cout << "func(1)" << endl; }
void func(char c) { cout << "func(2)" << endl; }
void func(long long ll) { cout << "func(3)" << endl; }
void hum(int i, ...) { cout << "hum(1)" << endl; }
void hum(int i, int j) { cout << "hum(2)" << endl; }
int main() { // 函数重载例题 1
    char c = 'A';
    bar(c);
    short s = 1;
    func(s);
    hum(12, 5);
    hum(10, 12, 1);
    system("pause");
}
```

```cpp
void bar(int i) { cout << "bar(1)" << endl; }
void bar(const char c) { cout << "bar(2)" << endl; }
void func(int a) { cout << "func(1)" << endl; }
void func(char c) { cout << "func(2)" << endl; }
void func(long long ll) { cout << "func(3)" << endl; }
void hum(int i, ...) { cout << "hum(1)" << endl; }
void hum(int i, int j) { cout << "hum(2)" << endl; }
int main() { // 函数重载例题 1
    char c = 'A';
    bar(c); // bar(2)
    short s = 1;
    func(s); // func(1)
    hum(12, 5); // hum(2)
    hum(10, 12, 1); // hum(1)
    system("pause");
}
```

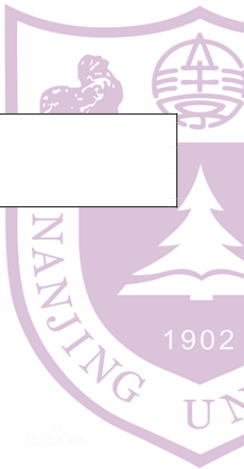
```cpp
int main() { // 常量和指针例题 1
const int c = 128;
    int* q = const_cast<int*>(&c); // 强制类型转换
    *q = 111; // 企图通过变量指针修改常量
    cout << "␣c␣" << &c << c << endl;
    cout << "␣q␣" << &q << q << endl;
    cout << "*q␣" << q << *q << endl;
}
```

```cpp
int main() { // 常量和指针例题 1
    const int c = 128;
    int* q = const_cast<int*>(&c); // 强制类型转换
    *q = 111; // 企图通过变量指针修改常量
    cout << "␣c␣" << &c << c << endl;
    // c 是符号常量，在编译时符号常量已经变为 128，相当于 define
    // 被编译器当作：cout << " c " << &c << 128 << endl;
    cout << "␣q␣" << &q << q << endl;
    cout << "*q␣" << q << *q << endl;
    //Name Addr Value
    // c 0012FF74 128
    // q 0012FF70 0012FF74
    //*q 0012FF74 111
    //对于同一个地址 0x0012FF74，输出了不同的值
}
```

```cpp
class A {
    int val;
    void setVal(int v) {
        val = v;
    }
};

A getA() {
    return A();
}

// 知道风险，并且想要改变新对象，就使用右值引用&&
int main() { // 右值引用例题 1
    int a = 1;
    int &ra = a;
    const A &cra = getA();
    A &&aa = getA();
    A &ab = getA();
}
```

```cpp
class A {
    int val;
    void setVal(int v) {
        val = v;
    }
};

A getA() {
    return A();
}

// 知道风险，并且想要改变新对象，就使用右值引用&&
int main() { // 右值引用例题 1
    int a = 1;
    int &ra = a; // OK, 非 const 引用绑定左值
    const A &cra = getA(); // OK, const 引用绑定右值
    A &&aa = getA(); // OK, 右值引用绑定右值
    A &ab = getA(); // ERROR, 引用不能绑定右值
}
```

```
1 const void show(const A* const this) const { // const 例题 1
2     // const 分别是什么含义?
3 }
```



```cpp
const void show(const A* const this) const { // const 例题 1
    // 第一个 const 修饰指针，表示 this 指针不可修改；
    // 第二个 const 修饰 this，表示 this 指向的对象不可修改；
    // 函数签名当中的 const 相当于参数当中第二个 const
    // 返回值当中的 const：自己去试一试呢？
}
```

新年快乐，期末大吉！
让我看看是谁元旦还要复习考试啊