类、对象注意点

朱晓旭

苏州大学计算机科学与技术学院

类的作用

- 产生对象
  - C++内置类型简单、高效
  - 类为自定义数据类型创造了无限可能
- 作为基类
  - 进行继承派生

例题

- 设有X计算机语言的表达式中允许使用三种括号，分别是（）、[]和{ }，请编写程序让用户输入一个表达式，判断该表达式中括号是否匹配
  - 程序的输入是表达式(字符串长度小于80)
  - 输出是TRUE或FALSE

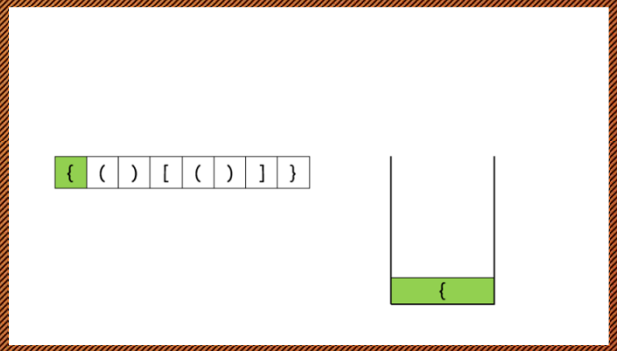
例如：

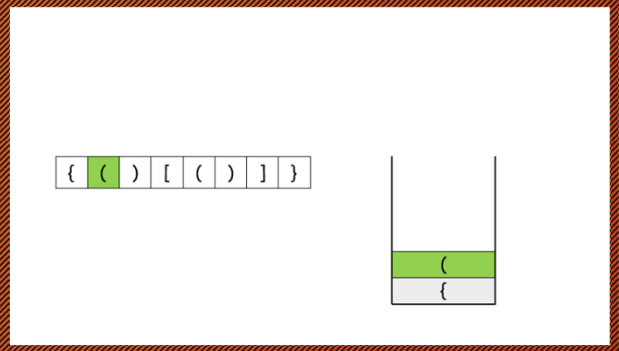
输入	{a+b(c*d)}	输出：	TRUE
输入	{a+b(c*d)	输出：	FALSE
输入	}a+b(c+d){	输出：	FALSE
输入	(a+b)	输出：	TRUE

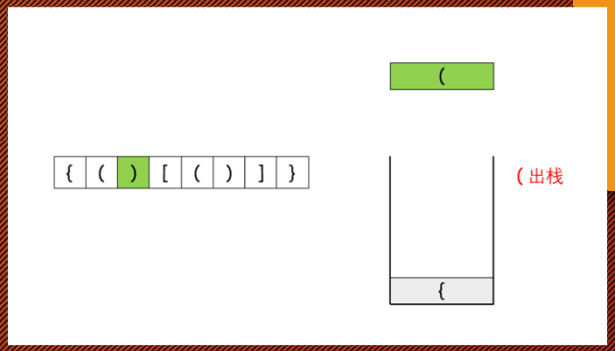
解决方案

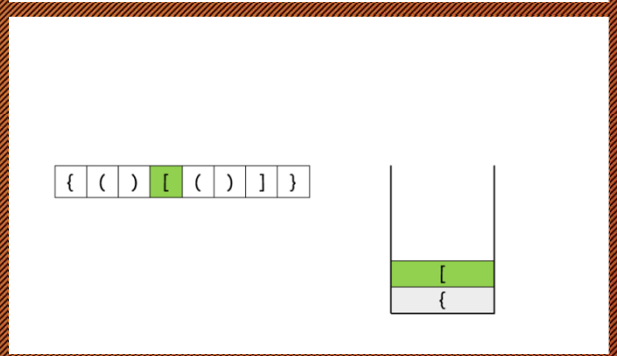
- 一种括号
  - 一个整型变量就可以
- 两种以上的括号
  - 双指针+回文串思想
  - 不完备
    - 仅能处理双端闭合的情况
- 成熟方案
  - 借助栈

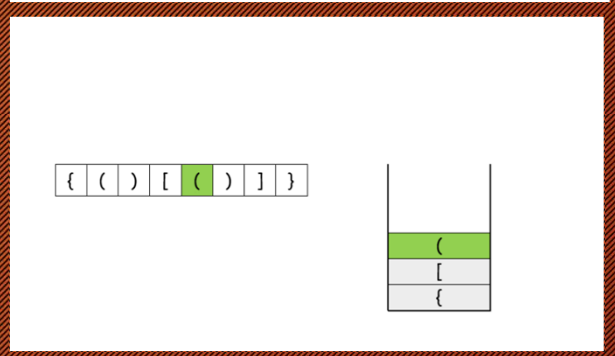
用栈加以解决

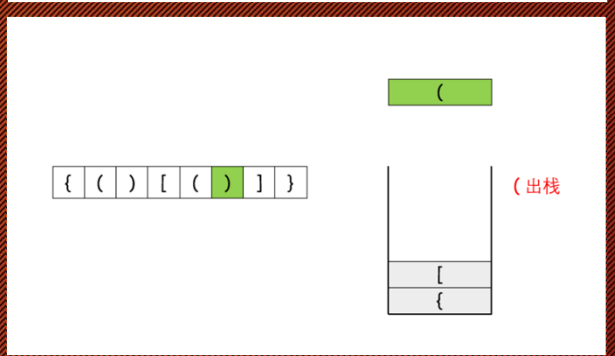


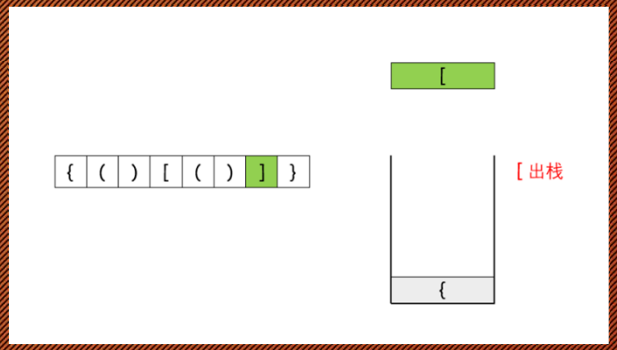


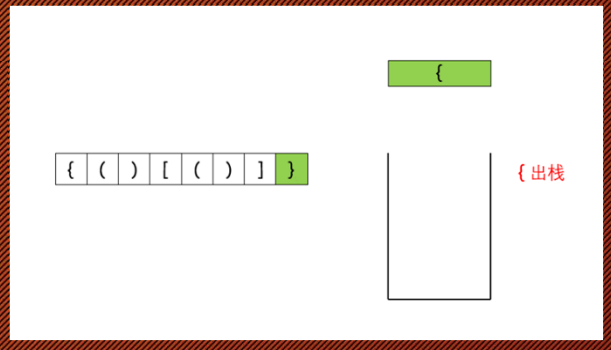


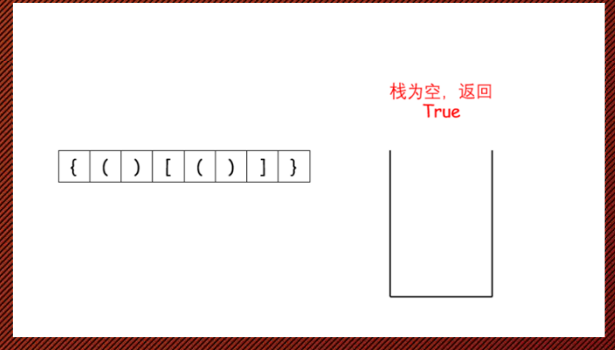












函数返回值

- 返回值
  - 最常见
- 返回引用
  - operator[]
  - operator<<
  - 前置++
- 返回地址
  - 本质上是返回一块内存的首地址

值返回与引用返回

- 值返回
  - 返回临时表达式的值

```
Point operator+(const Point& a, const Point& b)
{
    Point s;
    s.set(a.x+b.x, a.y+b.y);
    return s;
}
```

- 引用返回

```
ostream& operator<<(ostream& o, const Point& d)
{
    return o<<"(“<<d.x<<”,“<<d.y<<”)\n”;
}
```

引用返回

- 不要返回无效的引用
- int & test()

```
{
  int temp=5;
  return temp;
}
```

返回地址（指针）

- 不要返回局部变量的地址
  - 指针悬挂
- 可以返回堆中申请的内存

访问控制

- 访问控制
  - public和private是C++语言提供给程序员的功能
- 成员函数　一般为公有public
  - 公有的成员函数在类的外部可以被使用
  - 外界可以调用成员函数.
- 数据成员　一般为私有private
  - 私有的数据成员在外部不能被访问
  - 外界不能访问对象的数据分量，而只能由成员函数内部去处理
- 公有和私有可任意设定

程序结构

- 多文件项目
  - 一个类
    - .h
    - .cpp
  - 优点：便于协作开发，分工明确
  - 缺点：接口设计要求高

友元作用

- 弥补访问控制符的不足
  - 在外部频繁操作对象(即调用成员函数)，引起调用开销增加
  - 直接访问对象的成员(而不是调用成员函数)
  - 性能明显提高
  - 严重破坏封装

友元作用（续）

- 用在无法成员化的操作符重载中

```
class Point {
  // ...
  friend ostream& operator<<(ostream& o, const Point& d ) {
        return  o<<'('<<d.x<<','<<d.y<< ")\n" ;
    }
};
```

  - 第一操作数为cout，不是Point对象
  - 无法在Point类中将操作符<<成员化
  - 但又想直接访问Point中的私有数据成员

构造顺序

- 局部对象
  - 在C语言中
    - 所有局部变量都在函数开始执行时统一创建
    - 创建顺序根据变量在程序中按语句行出现的顺序
  - 在C++中
    - 根据运行中定义对象的顺序来决定对象创建的顺序
    - 静态对象只创建一次

全局对象的创建

- 同一工程不同代码文件全局对象的创建没有明确顺序规定
  - 对策：
    - 不要让不同文件的全局对象互为依赖
    - 依赖具有先后性，而其全局对象的创建不能保证该依赖性发挥作用
- 全局对象在main函数启动之前生成，而调试则在main函数启动之后.
  - 对策：
    - 调试时，应先将全局对象作为局部对象来运行观察
    - 在构造函数中添加输出语句来观察运行过程

成员对象的构造顺序（组合）

成员对象的构造顺序按类定义的出现顺序，最后执行自身构造函数：

```
class A{
  B b;
  C c;
  D d;
public:
  A(){}
  // ...
};
int main(){
  A a;
}
```

则构造顺序为b->c->d，然后执行A的构造函数的花括号体{}

构造位置

- 全局数据区：
  - 全局对象，静态全局对象，静态局部对象，常对象
  - 类的静态数据成员也存放在该数据区
- 栈区：
  - 局部对象
    - 根据不同编译器的实现方法，临时对象可能在栈区，也可能在动态存储区，也可能一部分在栈区，一部分在动态存储区
- 动态存储区（也称堆区）：
  - 用new申请的对象

析构函数

- 定义析构函数的目的：
  - 要进行对象占有资源的释放工作.
  - 一般来说
    - 若编写复制构造函数，则也应该定义析构函数
    - 因为对象创建中有资源要获得分配，则对象失效前必应先释放资源