FOPL-Homework 5

昂伟 *PB11011058*

Problem 1

```
Bool Halt(P, n); // true if P halts
                 // false if P does not halt
void K(P)
{
    if (Halt(P, P) == false)
        return true;
    else {
        while(1) { }
    }
}
```

分析　若 K(K) = true, 则 Halt(K, K) = false, 即 K(K) = false, 矛盾.

Problem 2

(a)

1 个类被加入, 即 productExp 类; 没有类被修改.

(b)

没有类被加入,n + 1 个类被修改.

(c)

m 个类被修改,1 个类被加入.

(d)

1 个类被加入, 没有类被修改.

(e)

只增加新的表达式类型而不增加新的表达式操作时, 才用标准做法只须增加一个新的表达式类而不会修改其他类.

(f)

只增加表达式操作而不增加新的表达式类型时, 采用 visitor 设计模式可以只增加一个表达式操作类而不修改其他类.

Problem 3

(a)

Activation Records

(0)	r	(1)
	cp	(3)
(1) Point(...)	access link	(0)
	x	2.1
	y	4.2
	equals	.
	distance	.
(2) Point part of cp	access link	(0)
	x	3.6
	y	4.9
	equals	.
	distance	.
(3) ColorPt(...)	access link	(2)
	c	red
	equals	.
(4) cp.distance(r)	access link	(2)
	q	(r)

Closures
<(1), •>
<(1), •>
<(2), •>
<(2), •>
<(3), •>

Compiled Codes
$|\ code\ for\ equals|$
$|\ code\ for\ distance|$
$|\ code\ for\ cpt\ equals|$

(b)

首先在活动记录 (4) 中搜索 x, 发现没有定义 x, 接着沿着 access link(2) 找到活动记录 (2), 然后在活动记录 (2) 中搜索 x, 发现 x = 3.6

(c)

首先在 cp 指向的活动记录 (3) 中搜索 x, 发现没有 x, 接着沿着活动记录 (3) 的 access link(2) 找到活动记录 (2), 然后在活动记录 (2) 中搜索 x, 发现 x

(d)

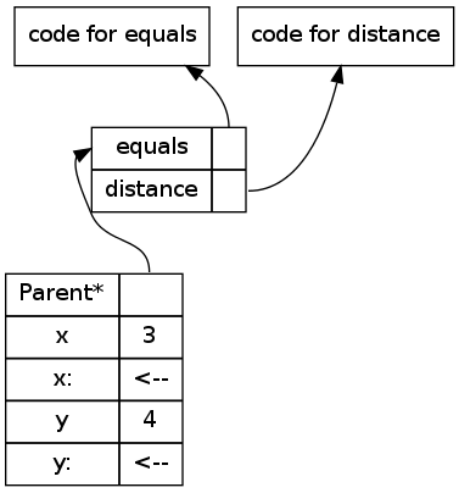
distance 函数定义在 Point 类中, 且没在子类 ColorPt 中重新定义, 并且只用到基类中的数据成员 x,y.

(e)

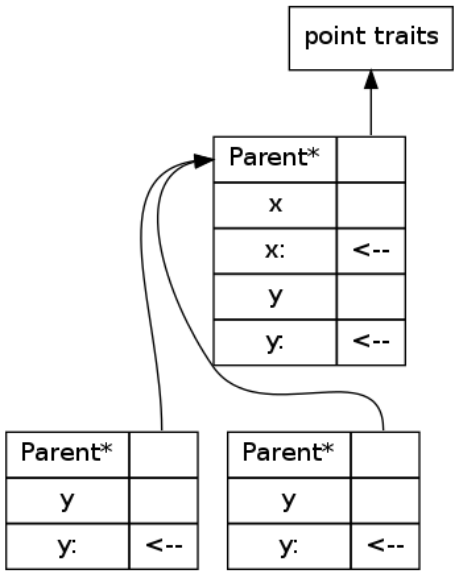
不会放在栈上, 而是放在堆上. 因为只有当程序不再使用对象时才能释放对象占用的活动记录, 放在栈上可能导致提前释放活动记录, 导致引用对象出错.

Problem 4

(a)



(b)



(c)

只复制被复制对象的 fields, 并将被创建对象的父指针指向被复制对象.

(d)

有可能导致改变父节点的对象的子对象的行为发生改变, 即当该对象的子对象引用该对象的父对象的方法时发生问题.

Problem 5

(a)

不需要, 因为可以将子类的 vtable 组织成和父类 vtable 同构的结构.

(b)

2n 个 entries, 其中 n 个 entries 用来存放 offset(虽然单继承时都为 0). 第一种方案实现的编译器对单继承和多继承对象的处理没有不同, 都需要额外的 n 个 entries 来保存 offset.

(c)

采用第一种方案实现的多继承编译器在调用虚函数前先将 *this* 指针加上对应虚函数 entry 中的 offset, 传给该函数作为参数, 除此之外没有其他额外开销.

(d)

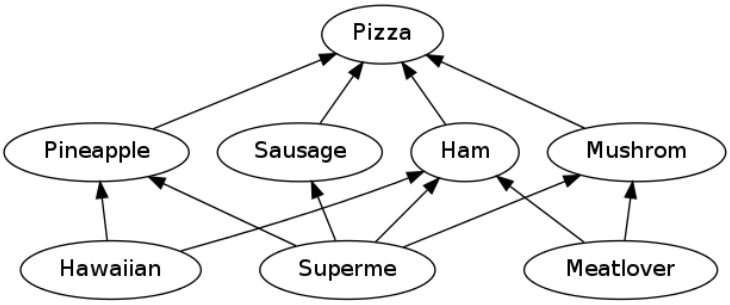
采用第二种方案实现的编译器只需维护 n 个 enries 用于存储虚函数地址. 该方案下, 调用虚函数时可能会发生一次额外的函数调用和一次无条件跳转 (*goto*). 编译器会为使用多继承的函数生成额外的 *chunk*.

(e)

采用第一种方案的编译器不可以链接运行. 第二种方案可以.

Problem 6

(a)



(b)

命名冲突.

(c)

Interface: Pizza, Sausage, Ham, Pineapple, Mushroom.

Class: Supreme, Hawaiian, Meatlover.

(d)

C++ 多继承时无需查表, 程序运行效率. Java 接口实现简单, 直观.