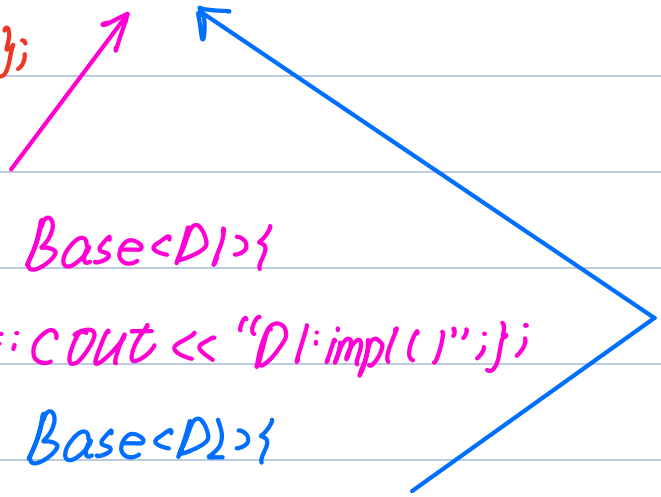
基础9 CRTP和 Expression Templates

①奇异递归模板模式(CRTP)

```
template< typename Derived>
struct Base { void name(){ (static_cast<Derived*>
(this))->impl(); };

struct D1: public Base<D1>{
  void impl(){ std::cout << "D1:impl()"; };
struct D2: public Base<D2>{
  void impl(){ std::cout << "D2:impl()"; };
```



```
template<typename Derived>
void fun(Base<Derived> derived)
{ derived.name(); };

Base<D1> derived1;
Base<D2> derived2;
D1 derived3; D2 derived4;
```

作用：在complie编译阶段确定调用子类接口实现的静态多态能力
(相比于RTTI性能有提升)

②表达式模板(Expression Template) 通过递归实现

作用：①延迟计算表达式，从而可以将表达式传递给函数参数，而非计算对象

②节省表达式中间结果的临时储存空间，减少计算循环次数

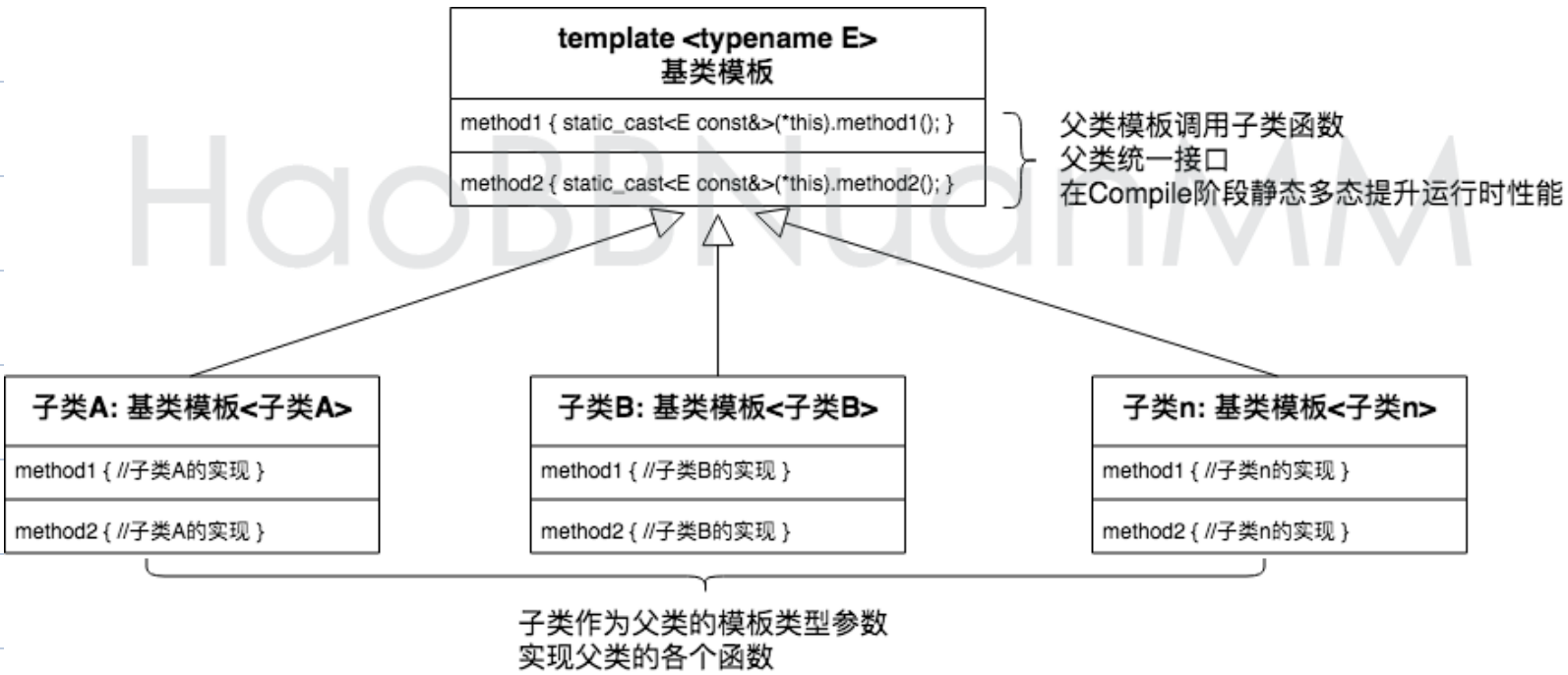
```
V1  V2        S<S<V1,V2>,V3>  V4
S<V1,V2>  V3      S<S<S<V1,V2>,V3>,V4>
```

A.只构造对象 不作计算

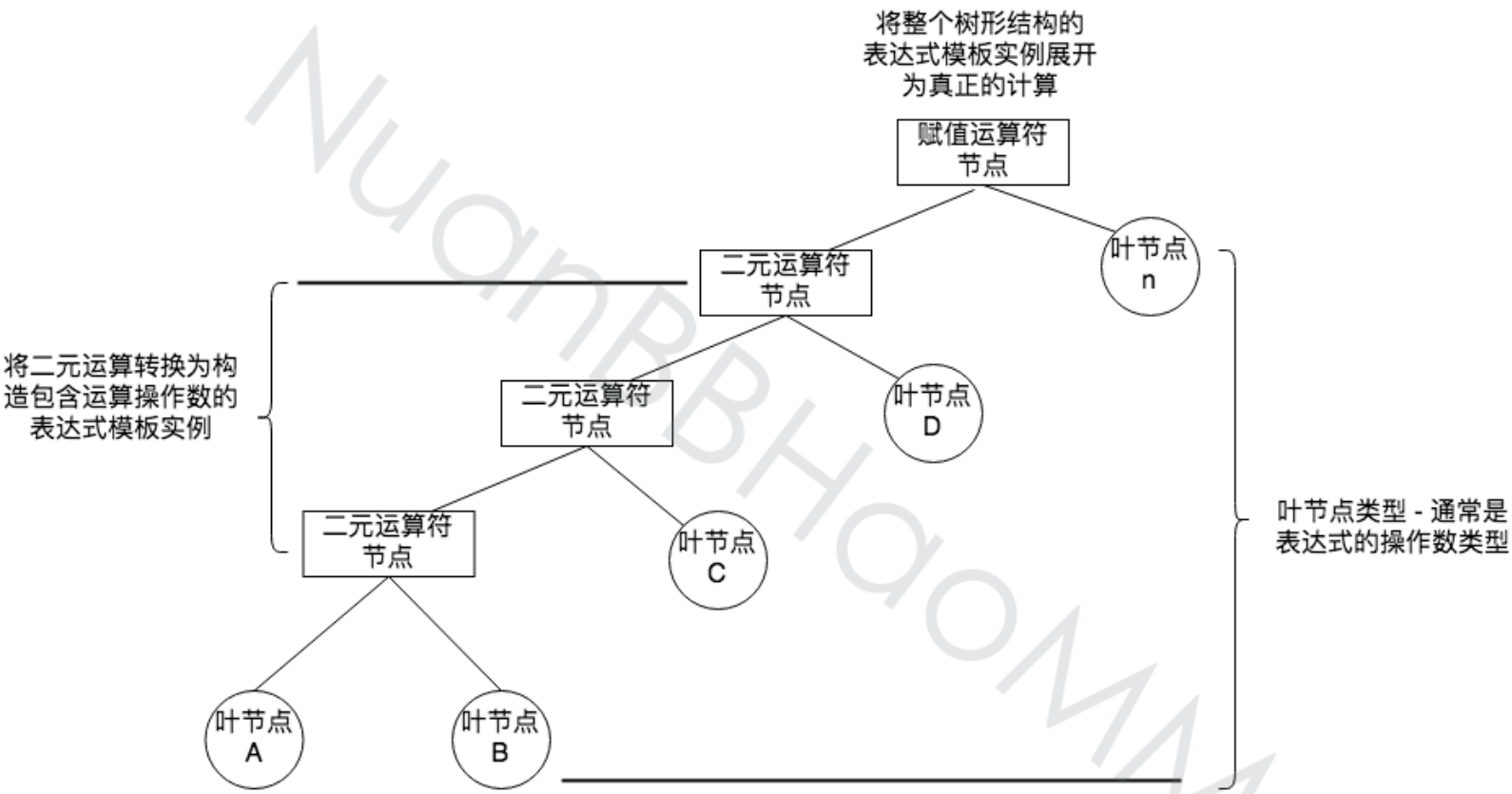
B.传入[i],开始计算 单个对象

vecsum → 调用自己的函数

vec → 返回vec[i]值



奇异递归模板模式（CRTP）原理图



表达式模板编译树原理图