C++ Assignment 7

实现思路

- 所有 `sub_ingredient` 从 `ingredient` 继承过来。而 `ingredient` 只保存了价格、量和名字。

  `ingredient` 的析构函数必须是虚函数。因为在存储原料的时候会使用原料的指针保存 `sub_ingredient`，而对后者进行删除需要利用多态性调用后者的析构函数，不能调用原料的析构。

  原料基类的其他实现较为简单，由于没有利用指针，复制构造、析构都可以默认。

- `sub_ingredient` 可以通过宏来进行声明，节省代码量。宏中可以通过 `\` 来换行，`#S` 表示 `"S"`。

```
#define DEFCLASS( NAME , PRICE ) \
    class NAME : public Ingredient {\
    public:\
        NAME( size_t units ) : Ingredient{ PRICE , units } {\
            this->name = #NAME;\
        }\
        virtual std::string get_name() { return this -> name; }\
    };
```

- `EspressoBased` 是咖啡的基类，其是一个抽象类。

  类中构造、复制、`copy assign` 都需要在 `protected` 中，因为在继承后需要能够访问。其 `get_name,price` 都需要在继承后得到实现。

  由于需要对基类指针指向的实际咖啡进行析构，其析构函数必须是虚函数。

  其复制构造函数可以在子类进行复制构造的时候提前被调用。其内容是对所有参数的内容进行复制（开新的空间来存储）。

- 对 `Cappuccino` 和 `Mocha` 的实现差不多。

  实现中复制构造函数可以直接调用基类的复制构造函数。

  析构函数先把所有内部的指针删掉后把向量清空。

  复制构造函数中需要判断 `this` 和即将复制的东西，防止自我复制。

  可以在子类中通过 `EspressoBased::operator = ( cap );` 调用基类的等于构造函数。