PD Hw 8 Problem 4

財金所碩三 r10723057 黃元裕

Item:

Item::Item(const char* n, const int mc) 若 n, mc 為 private，則須透過一些 function 才能改動，那這邊加不加 const 都可以。反之，若n, mc 為 public，為了避免傳入的 argument 被改動，應該加上 const

```
{
    name = new char[strlen(n) + 1];
    strcpy(name, n);
    materialCost = mc;
}
```

```
Item::Item(const Item& item)  // copy constructor
{
    name = new char[strlen(item.name) + 1];
    strcpy(name, item.name);
    materialCost = item.materialCost;
}
```

而 operator 成員函數前面若加上 const 通常都是為了避免 (a1 = a2) = a3 這種情況的發生

```
Const void Item::operator=(const Item& item)  //
assignment operator
{
    if (this != &item) // 檢查自我賦值
    {
        delete[] name; // 刪除舊的記憶體
        name = new char[strlen(item.name) + 1];
        strcpy(name, item.name);
        materialCost = item.materialCost;
    }
}
```

上面兩個地方原本就有加上 const，是為了確保我們已經打包好的 item 不會受到改動。

Product:

同上，若這些變數為 private，則可加可不加；若這些變數為 public ，則應該要加上 const 來避免這些變數被改動。

```
Product::Product(const char* n, const int p, const int
lc, const int sq, const int ic)
{
    // 在 Product 的 constructor 中應該初始化 itemList 為一個存
有 itemCnt 個 Item* 的動態陣列
    itemList = new Item* [ic];  // 記得這裡不需要再次聲明
Item**，直接賦值即可
    // 並且將這些指向 Item 的指標都先指向 nullptr
    for (int i = 0; i < ic; i++) {
        this->itemList[i] = nullptr;
    }

    name = new char[strlen(n) + 1];
    strcpy(name, n);
    price = p;
    laborCost = lc;
    salesQty = sq;
    itemCnt = ic;
}

Product::Product(const Product& prod)  // copy
constructor
{
    name = new char[strlen(prod.name) + 1];
    strcpy(name, prod.name);
    this->price = prod.price;
    this->laborCost = prod.laborCost;
    this->salesQty = prod.salesQty;
    this->itemCnt = prod.itemCnt;
    itemList = new Item* [itemCnt];
    for (int i = 0; i < itemCnt; i++) {
        itemList[i] = prod.itemList[i];
    }
}
```

以下三個 member function 傳入的 argument 都是 const，同理，是為了保護我們已經打包好的 Product 不被改動。而 operator 成員函數前面若加上 const 通常都是為了避免 (a1 = a2) = a3 這種情況的發生

```
Const void Product::operator=(const Product& prod)
{
    if (this != &prod) // 檢查自我賦值
    {
        delete[] name; // 刪除舊的 name
        name = new char[strlen(prod.name) + 1];
        strcpy(name, prod.name);

        for (int i = 0; i < itemCnt; i++) { // 先刪除舊的
itemList
            delete itemList[i];
        }
        delete[] itemList; // 刪除 itemList 本身

        price = prod.price;
        laborCost = prod.laborCost;
        salesQty = prod.salesQty;
        itemCnt = prod.itemCnt;
        itemList = new Item*[itemCnt]; // 重新分配空間
        for (int i = 0; i < itemCnt; i++) {
            itemList[i] = prod.itemList[i];
        }
    }
}

bool Product::isInFrontOf(const Product &prod, int
criterion)
{
    switch(criterion)  // 以下的 laborCost 要記得加上
materialCost!!!
    {
        case 1:
            if (this->price > prod.price)
                return true;
            else if (this->price == prod.price)
```

```
            // strcmp(this->name, prod.name) < 0 表示如果
this->name 字典順序上小於 prod.name，則返回 true。
            return strcmp(this->name, prod.name) < 0;
         break;
      case 2:
         if (this->laborCost > prod.laborCost)
            return true;
         else if (this->laborCost == prod.laborCost)
            return strcmp(this->name, prod.name) < 0;
         break;
      case 3:  // 毛利 = price - laborCost
         if ((this->price - this->laborCost) >
(prod.price - prod.laborCost))
            return true;
         else if ((this->price - this->laborCost) ==
(prod.price - prod.laborCost))
            return strcmp(this->name, prod.name) < 0;
         break;
      case 4:
         if (this->salesQty > prod.salesQty)
            return true;
         else if (this->salesQty == prod.salesQty)
            return strcmp(this->name, prod.name) < 0;
         break;
      case 5:  // 總營收 = price * salesQty
         if ((this->price * this->salesQty) >
(prod.price * prod.salesQty))
            return true;
         else if ((this->price * this->salesQty) ==
(prod.price * prod.salesQty))
            return strcmp(this->name, prod.name) < 0;
         break;
      case 6:  // 總利潤 = 毛利 * salesQty = (price -
laborCost) * salesQty
         if ((this->price - this->laborCost) * this-
>salesQty > (prod.price - prod.laborCost) *
prod.salesQty)
            return true;
```

```
          else if ((this->price - this->laborCost) *
this->salesQty == (prod.price - prod.laborCost) *
prod.salesQty)
              return strcmp(this->name, prod.name) < 0;
          break;
    }
    return false;  // 如果當前物件不應該排在傳入的物件之前，則返回
false
}
```

```
void Product::addItem(Item* itemPtr)
{
    this->itemList[index] = itemPtr;
    this->index++;
}

void swapPtr(Product*& p1, Product*& p2)
{
    Product* temp = p1;
    p1 = p2;
    p2 = temp;
}
```