Dynamic Arrays

M023010011 廖志倫

***PFArrayD:***

1.constructors:

PFArrayD()

PFArrayD(int capacityValue)

這邊的2項就是constructors的部分，

第一種是使用者沒有給定array的大小時，會使用預設的大小，來定義這個dynamic array。

第二種則是給定array的大小為capacityValue。

因為使用者可能有各種疏忽，所以我們定下constructor來防止，”因為使用者的大意，使得只有宣告，卻忘了給予memory”，而導致的資料無法寫入。

C++所提供的這種做法也方便在宣告時，只需在後面的括號中填入，就給予指定的初始值，而且依照填入的值的不同，可以給予不同的初始值。

2.copy constructor:

PFAarrayD:: PFArrayD(const PFArrayD& pfaObject):

capacity(…).used(…)

**{**

a=new double[capacity]

for(int i=0;i<used;i++)

{…}

**}**

省略了部分程式碼

預設的=只有成員直接對應的處理，像capacity與used這兩項，就可以在透過get解決完private之後，直接對應，而得到值。但a就不同了，它是point，它所指到的目標是array，array裡面的數字無法透過預設的=來完成，所以要另外寫出這部分的複製，也就是for loop中所作的事，即將原先的pfaObject指到的array中的值，copy到現在這個a所指到array中，所以稱為copy constructor，如此一來就能將原先的pfaObject完全複製過來了，一般的constructor在{}中是空白，沒有執行任何動作，而copy constructor就是將這些動作放在這個{}中。

3.destructor:

PFArrayD::~PFArrayD()

{

Delete [] a;

}

預設的destructor是刪除只有刪除指定的物件，但a刪掉之後，a所指到的array仍然存在，而且無法找到位址，變成浪費空間，如果持續這種操作，最後會耗盡memory，所以c++提供改寫destructor的方法。

利用這個方法改寫destructor,以便將a所指到的array一併刪除，{}中所寫的Delete [] a即是刪除a所指到的array，[]指的就是陣列，而a是位址。

4.assignment operator:

double& PFArrayD::operator[](int index)

{…}

double& PFArrayD::operator =(const PFArrayD& rightSide)

{…}

這部分就是operator的改寫，有時候需要處理一些疊加，或相乘，希望用”+”或”\*”的符號表示，但這又與一般運算中的+與\*有所不同，此時，就能利用operator的改寫來為這個類別的+與\*重新定義，以便達到要求，舉例而言，像是向量加法中的+，與一般四則運算中的+，以及矩陣的+都有所不同，此時，我們就能分別為這些類別定義他們專有的operator。

而在這邊，=的使用受到a所指到的array的大小影響，所以他重新定義，先將=左邊的array大小變為與右邊相等，再將右邊的array的值存入左邊。

[]的操作則是用來檢察index所指到的array中的位址，是否有被使用過，如果是為使用過的位址，就通知使用者。

***PFArrayDBak:***

C++提供了名為 inherit的方法，這方法能讓我們在寫新的class時，可以繼承舊有的class的member，如果這兩個class有某種程度的相關，我們就能用這個方法來實現，舉例而言，以”中山大學的學生”為基本的class，然後”電機的學生”、”資工的學生”等，都可以直接繼承”中山的學生”這個class，再額外添加自己這個class的專有屬性，

這個作法最大的好處在維護容易，如果有意天，我們希望作一個基本性的大更動，在以繼承方式實現的class例子中，我們只要對基本的class(中山的學生)作更動，繼承的class(電機、資工等)都會一併更動。但若我們一開始試分別定義”電機”、”資工”等的class，那要改動時，就必須全部一起更改。

這些function的用意與PFArrayD類似，只是多了處理新增的member部分，所以這邊就只介紹C++的運用，不再提為什麼需要這些操作。

1. constructors:

由於繼承了PFArrayD所以在作constructor時，可以呼叫前面的PFArrayD()、PFArrayD(int capacityValue)等PFArrayDD中的constructor，然後再對新增的部分作處理，就能完成這個繼承的class的constructor了。

但也是必須這麼做，因為原先的class中的private部分在繼承下來後，無法經由這個子class來獲得，所以必須透過父(母)方的class來作constructor。

1. copy constructor:

這部分的情況與constructor類似，必須透過PFArrayD的幫助來完成。

b也是個指向array的point，所以這部分與PFArrayD類似，都要為了寫copyconstructor。

1. destructor:

這邊要將b所指到的array一併刪除，所以要寫destructor，而a所指到的array，由於C++會往上的class看。當我們刪除PFArrayDBak時，C++會發現這是繼承自PFArrayD所以，他也會執行PFArrayD中的destructor，將a所指到的array也刪除，所以不必在重複寫一次delete [] a。

1. assignment operator:

這部分也是，呼叫了繼承的PFArrayD中的operator，在為自己新增的b的部分寫好operator，就能完成這部分的操作。