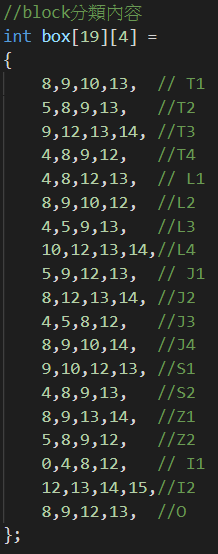
Data Structure Project1:Tetris俄羅斯方塊

NTHUPHYS 106022114 游惟翔

* **前置作業**

1. **定義block**

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Block的資訊儲存在box的二維陣列中(圖1.)，而所有的block皆可由4\*4的小matrix表示，因此我先將box做0-15共16個elements在陣列中(表1.)。

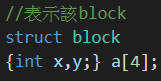
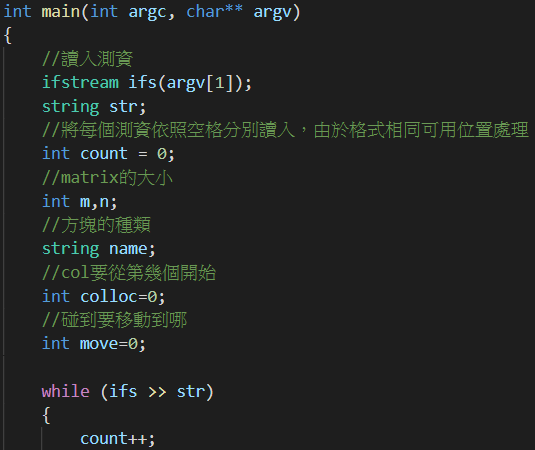
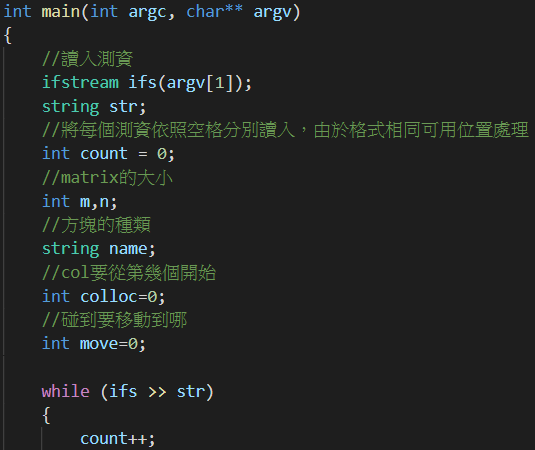
 再透由讀取方塊時的方塊種類判定，將x.y加以標示，此處我們先定義block內涵x.y兩個元素(圖2.)，而它有四項(因每個block由四個小方塊組成)

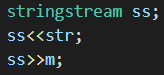
表1.(左上)陣列意識圖 圖1.(最右)各block的資料 圖2.(右下)定義block的x.y

1. **ifstream**

透過ifstream輸入來自目標檔案的資料(圖3.)，用迴圈將每個以空格分開的element讀進來做判斷。利用檔案具有的特定格式，新設一整數count來計算現在進行到檔案中的哪一個element，每當進行完該迴圈後



count+1，並讀入下一個element的訊息。 圖3. 以argv[]讀入測資

1. **stringstream**

而ifstream所取的資料型態都為string，因此使用stringstream將資料轉變成int整數型態做使用。

圖4. stringstream的表示方式

1. **判斷matrix大小**

讀入的第一row資料為m個rows和n個columns，剛好對應到count=1,2讀到的elements，因此用條件式if判斷分別得到代表遊戲進行的行和列。

1. **判斷程式終結**

當讀入的str讀取到End時，跳出迴圈。

* **讀取方塊**

|  |
| --- |
| **方塊種類判別**  **name** |
| **判別方塊種類**  **進行colloc開始落下的欄位位置** |
| **開始落下的欄位位置**  **colloc** |
| **移動至指定位置**  進行***checkinvalid***判斷(是否block放置位置超出matrix)  是，此列不合法，跳到測資下一行**進行name方塊種類判別**  列出：該"row in testcase is the invalidcase!!!"  "-->initial column location is out of boundary"  否，以***checktouch***為落地停止條件執行**下落**  是，**落地**，進行move碰撞後的移動距離 |
| **碰撞後的移動距離**  **move** |
| 進行***checkinvalid***判斷(是否block移動時會撞到其他block或超出matrix)  是，此列不合法，跳到測資下一行**進行name方塊種類判別**  列出："row in testcase is the invalidcase!!!"  " -->the motion will hit existing blocks or out of boundary"  否，執行**橫移**  橫移後：  以***checktouch***為落地停止條件執行**下落**  是，**落地**  落地後：  執行**消去並再落下**，再進行***checksky***判斷(是否block超出天花板)  是，跳出迴圈，遊戲結束  列出："over height"  否，完成此方塊輸入，跳到測資下一行**進行name方塊種類判別** |

表2. 讀取方塊流程圖

**進行讀取方塊**

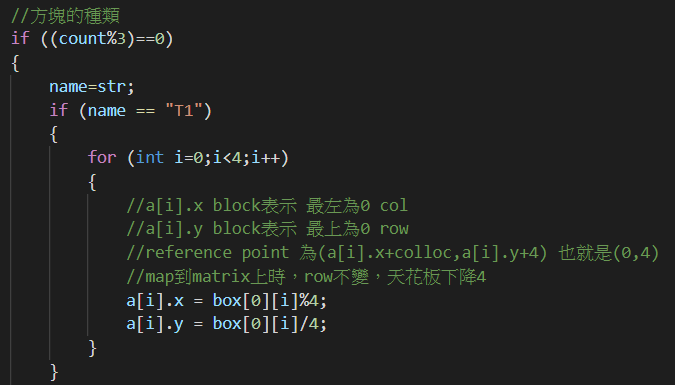
而接下來從測資讀取到的資訊都為每一個方塊block所表示的訊息：

**方塊種類(name)**、**開始落下的欄位位置(colloc)**、**碰撞後的移動距離(move)**

而觀察count變數，當資料為方塊種類時，count會是3的倍數；落下的欄位位置是除3後餘1；移動距離的count是除3後餘2。因此我們用以上的特性，用條件式if 對各個讀到的element做分類。

1. **方塊種類(name)：(count%3)==0**

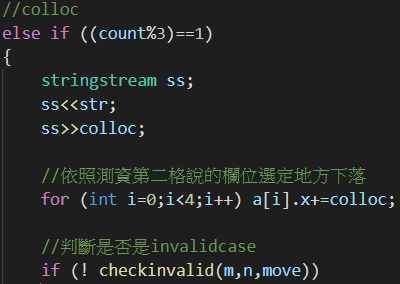
block的資訊儲存在box的二維陣列中，而所有的block皆可表示在4\*4的小matrix上。所以我將box陣列，依照%4轉換成水平軸x:向右為正，/4轉換為垂直軸y:向下為正的4\*4二維座標(表3.)，並將x.y座標輸入block內(圖5.)。



|  |  |  |  |
| --- | --- | --- | --- |
| (0,0) | (1,0) | (2,0) | (3,0) |
| (0,1) | (1,1) | (2,1) | (3,1) |
| (0,2) | (1,2) | (2,2) | (3,2) |
| (0,3) | (1,3) | (2,3) | (3,3) |

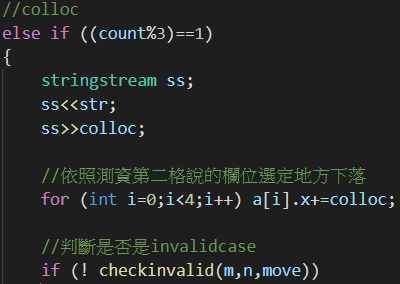
表3. 小matrix意識圖 圖5. 指派x.y資訊到以a為名的block型態class

1. **開始落下的欄位位置(colloc)：(count%3)==1**



確認完block的種類後，即可以從測資當中跳下一個資料-開始落下的欄位，並且將值加到原本第1步**判斷方塊種類**時設的小matrix上，因此這個小matrix平移到正確的欄位(圖6.)，block初始的落下位置確定。

圖6. 移動block到指定欄位

隨後進行這個移動是否合法，有時部分的方塊會移動過度，導致超過左右邊界，因此在此使用***invalidcase***進行檢查(圖7.)，合法後開 圖7. 判斷是否擺錯block

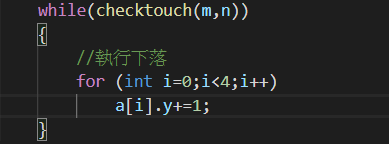
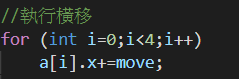
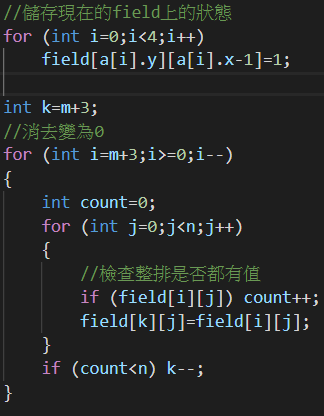
始進行落下(圖8.)直到第一次撞到地面(以***checktouch***進行判斷)。

圖8. 開始落下直到撞地面

1. **碰撞後的移動距離(move)：(count%3)==2**

 第一次觸地後，從測資讀取橫移資料，開始進行橫移(圖9.)，並且在此以***invalidcase***檢查是否合法(圖7.)，在橫移過程中沒有碰觸到既有的方塊，一切通過以後，在進行下落直到觸地停止 圖9. 橫移

(以***checktouch***進行判斷)。

在進行消去前，先將該block儲存到大的matrix上，再執行消去。

透過count計算matrix每一列已經擁有的值，因此當count和欄位總數n的值相符時，則可以表示該行已完全被填滿，可以進行消去，並將上消去之後的上層資料下拉做取代(再落下)。

最後，以***checksky***判斷是否超出天花板，若是都通過，則跳回下一個測資繼續進行block的判別……直到遊戲結束。

圖10. 儲存block並進行消去

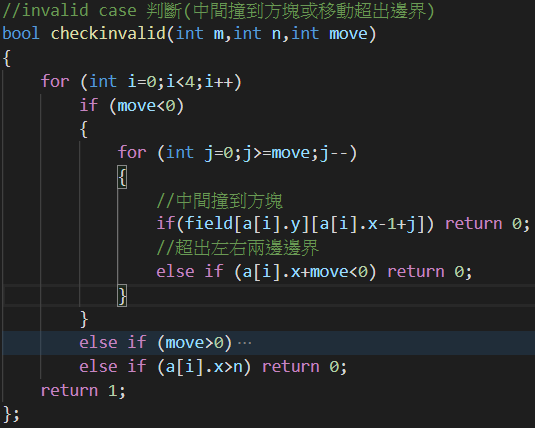
* **判斷式**

1. ***checkinvalid***判斷

block放置位置超出matrix /**落下的欄位位置(colloc)**

移動時會撞到其他block或超出matrix/**碰撞後的移動距離(move)**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |



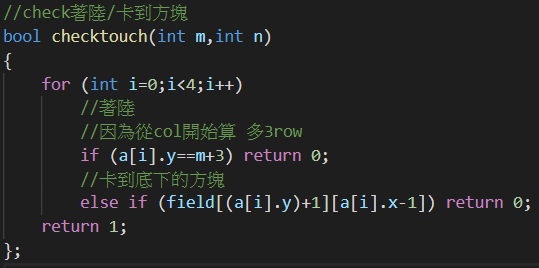
3 2

Z2 1 1

表3.解釋落下位置設定錯誤的狀況，code為圖11.else if (a[i].x>n的)

圖11. ***Checkinvalid***程式碼

1. ***checktouch***判斷

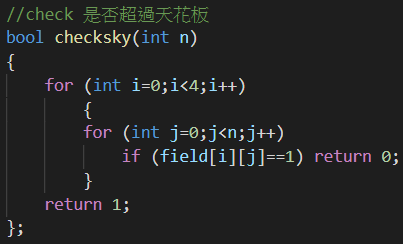
著地判斷/**落下的欄位位置(colloc)**

/**碰撞後的移動距離(move)**

由於我的matrix是從colloc後，由box分配成的小matrix拓展而成(可以想成在matrix上還有一個4\*n的區域)，因

此判斷式if中的列高從m改為m+3。 圖12. ***checktouch***程式碼

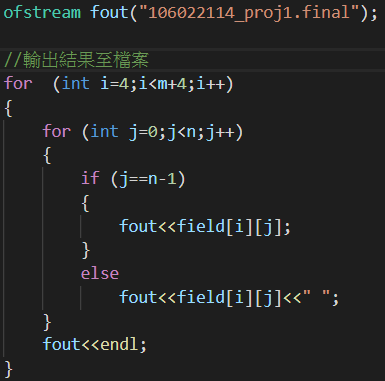
1. ***checksky***判斷

天花板判斷/**碰撞後的移動距離(move)**

整個讀取方塊流程迴圈的最後一步，和2. ***Checktouch***的理由一樣，因為matrix式拓展而成的，因此判斷時只需要看最上面拓展的部分是否在程式全部執行完

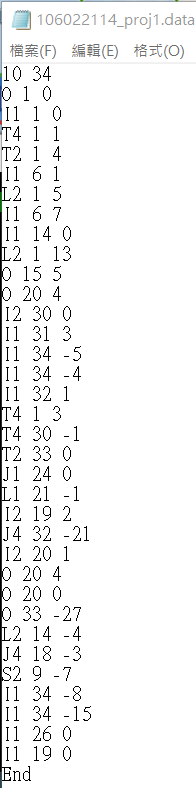
以後還有值。 圖13. ***checksky***程式碼

* **輸出結果**



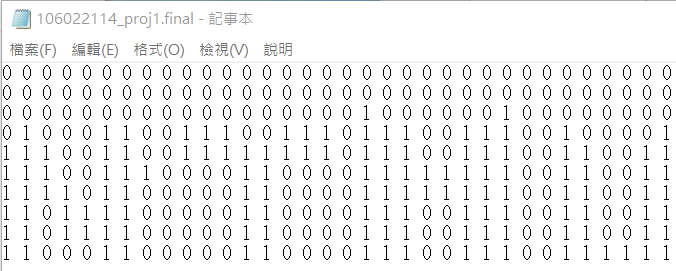
最後由ofstream指令，生成新的檔案：“106022114\_proj1.final”，將field內的最終資料以迴圈的方式寫入檔案中，並在每個element之間以空格做區分。

而和判斷式中***checktouch***、***checksky***當中提及的理由一樣，因為field是將測資提供的matrix大小+生成block用的4\*n大小的matrix，因此在此需要進入檔案的elements從第四row開始取。(圖14.) 圖14. Ofstream輸出檔案



* **TestCase 設計**

希望透過落地以後的左右移動，將”NTHU”四個英文字母呈現出來，所以這段testcase沒有削去的列，都是在觸地以後開始進行移動。

 而執行的結果雖沒有預期的明顯字樣，已有雛型，而testcase內也有著大量需要移動的block，可以增加程式執行時移動的次數。