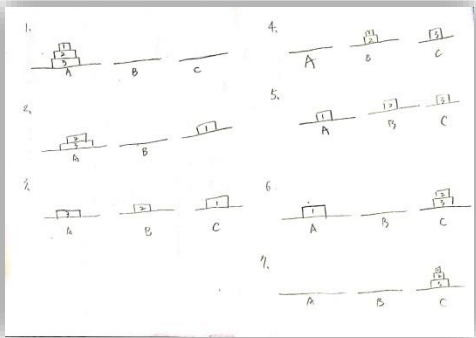


國立虎尾高中 109 學年度第 1 學期自主學習成果表

2020/1/12 19:38

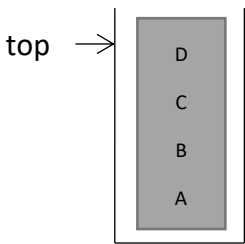
申請人	林詠智	班級/座號	2 年 06 班 16 號
申請學期	1091	申請時數	21
學習夥伴			
協助專家			
計畫名稱	一直 C 一直爽!		
相關學科	自主學習-資訊科技		
所需設備			
內容說明	學習 C++演算法、資料結構；運用所學內容解競賽題程式與製作報告		
預期效益	解兩題程式競賽題，熟悉鏈結串列、佇列和陣列、所學內容的運用		
成果報告形式	書面報告		


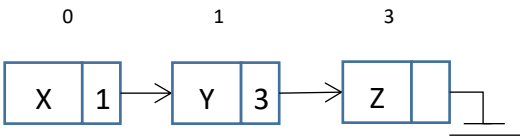
週次	星期	節次	自學內容	學習心得
1	二	4	學習指標	<p>指標 (pointer) 可以透過記憶體映射的方式直接控制硬體。</p> <pre>int *p1 = &a;</pre> <p>p1 指向 a 的記憶體位置(a 前需加&)</p>
2	二	4	學習遞迴	<p>遞迴:呼叫自身函式使用判斷結構直到傳入參數沒有再呼叫自身函式</p> <p>例:</p> <pre>int function(int x,int y){ return function(x-1,y-1); }</pre> <p>以河內塔問題程式實作為例:</p>  <p>程式執行時的過程:</p> <ol style="list-style-type: none"> 1. 首先以 (3, A, B, C) 呼叫 2. 呼叫 (2, A, C, B) 3. 呼叫 (1, A, B, C) print 1 A->C 4. print 2 A->B 5. 呼叫 (1, C, A, B) print 1 C->B 6. Print 3 A->C 7. 呼叫 (2, B, A, C)

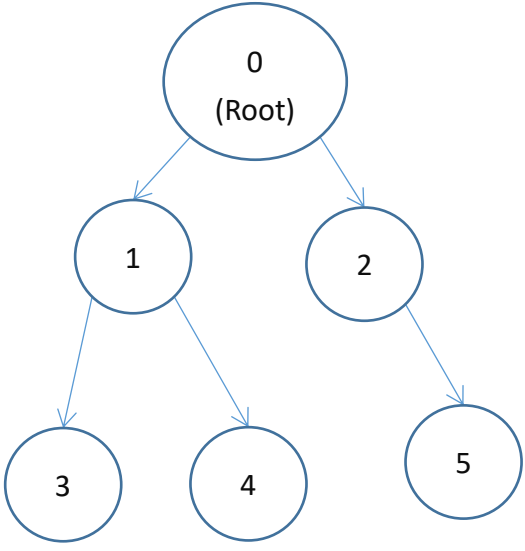
				8. 呼叫(1, B, C, A) print 1 B->A 9. Print 2 B->C 10. 呼叫(1, A, B, C) print 1 A->C																
3	二	4	學習陣列與矩陣	1. 陣列:結構一致(相同資料型態)使用同一個變數名稱,存放在連續空間的儲存體。 I. 一維陣列: a. TYPE *(name) = new TYPE[x] 例:int *a = new int [5]; b. TYPE number[x]; 例:double _a[5]; 都為長度 5 的一維陣列 II. 二維陣列: TYPE **(name) = new TYPE*[x] for (int i = 0; i < x ; i++) { arr[i]=new int[y]; } 為 x 行,y 列的二維陣列 2. 陣列指派: 陣列名稱[註標] = 數值; or 陣列名稱[行註標][列柱標] = 數值; *例: 3. 陣列取用: * 4. 陣列圖型: I. 一維陣列(Array[4]) <table border="1"><tr><td>[0]</td><td>[1]</td><td>[2]</td><td>[3]</td></tr></table> II. 二維陣列(Array[4][3]) <table border="1"><tr><td>[0][0]</td><td>[0][1]</td><td>[0][2]</td><td>[0][3]</td></tr><tr><td>[1][0]</td><td>[1][1]</td><td>[1][2]</td><td>[1][3]</td></tr><tr><td>[2][0]</td><td>[2][1]</td><td>[2][2]</td><td>[2][3]</td></tr></table> 5. 輸入: cin >> 陣列名稱[第幾格]; or cin>>陣列名稱[第幾行][第幾列] = 數值; 輸出: cout << 陣列名稱[第幾格]; or	[0]	[1]	[2]	[3]	[0][0]	[0][1]	[0][2]	[0][3]	[1][0]	[1][1]	[1][2]	[1][3]	[2][0]	[2][1]	[2][2]	[2][3]
[0]	[1]	[2]	[3]																	
[0][0]	[0][1]	[0][2]	[0][3]																	
[1][0]	[1][1]	[1][2]	[1][3]																	
[2][0]	[2][1]	[2][2]	[2][3]																	

				<p>cout<<陣列名稱[第幾行][第幾列] = 數值;</p> <p>6. 矩陣:行等於列的二維陣列。</p>																				
4	二	4	學習選擇排序	<p>排序演算法中以選擇排序、插入排序和氣泡排序最為常見</p> <p>選擇排序(Selection_Sort):</p> <p>將要排序的對象分作兩段，一個是已排序的，一個是未排序的。如果排序是由小而大，從後端未排序部份選擇一個最小值，並放入前端已排序部份的最後一個。</p> <table border="1"> <tr><td>起始</td><td>4</td><td>3</td><td>2</td><td>1</td></tr> <tr><td>第1次</td><td>1</td><td>4</td><td>3</td><td>2</td></tr> <tr><td>第2次</td><td>1</td><td>2</td><td>4</td><td>3</td></tr> <tr><td>第3次</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> </table>	起始	4	3	2	1	第1次	1	4	3	2	第2次	1	2	4	3	第3次	1	2	3	4
起始	4	3	2	1																				
第1次	1	4	3	2																				
第2次	1	2	4	3																				
第3次	1	2	3	4																				
5	二	4	學習插入排序	<p>插入排序(Insertion_Sort):</p> <p>將要排序的對象分作兩段，一個是已排序的，一個是未排序的。每次從後端未排序部份取得最前端的值，然後插入前端已排序部份的適當位置。</p> <table border="1"> <tr><td>起始</td><td>4</td><td>3</td><td>2</td><td>1</td></tr> <tr><td>第1次</td><td>3</td><td>4</td><td>2</td><td>1</td></tr> <tr><td>第2次</td><td>2</td><td>3</td><td>4</td><td>1</td></tr> <tr><td>第3次</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> </table>	起始	4	3	2	1	第1次	3	4	2	1	第2次	2	3	4	1	第3次	1	2	3	4
起始	4	3	2	1																				
第1次	3	4	2	1																				
第2次	2	3	4	1																				
第3次	1	2	3	4																				
6	二	4	學習氣泡排序	<p>氣泡排序(Bubble_Sort):</p> <p>由左而右重複掃描，排序時若是從小到大，最大元素會移至右端，其利用比較相鄰元素的方式，將較大元素交換至右端，所以較大的元素會不斷往右移動，直到適當的位置為止，就好像大氣泡受到的浮力最大而首先冒出來一樣。</p> <table border="1"> <tr><td>起始</td><td>4</td><td>3</td><td>2</td><td>1</td></tr> <tr><td>第1次</td><td>3</td><td>2</td><td>1</td><td>4</td></tr> <tr><td>第2次</td><td>2</td><td>1</td><td>3</td><td>4</td></tr> <tr><td>第3次</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> </table>	起始	4	3	2	1	第1次	3	2	1	4	第2次	2	1	3	4	第3次	1	2	3	4
起始	4	3	2	1																				
第1次	3	2	1	4																				
第2次	2	1	3	4																				
第3次	1	2	3	4																				
7	二	4	學習搜尋	<p>搜尋演算法中以循序搜尋和二元搜尋最為常見</p> <p>循序搜尋:從頭開始比對資料，適合小範圍資料</p> <p>二元搜尋(Binary_Search):</p> <p>需要使用已排序的資料，取中間值後比對大於就往小的找，小於較往大的找，</p>																				

				<p>重複取中間值，直到找到資料，在資料量龐大時可以迅速找出資料位置。</p> <p>例如：資料為 12345，要找到 4 的 Index (如下圖所示)：</p> <div data-bbox="858 365 1374 546" data-label="Diagram"> <pre> graph TD A[第一次，往上找中間值(找到 4)] --> B[1 2 3 4 5] style B fill:none,stroke:#000,stroke-width:1px </pre> </div> <p>所以 $Index = Mid + 1 = 3$</p>
8	二	4	學習物件導向基礎	<p>物件導向三大特性：</p> <ol style="list-style-type: none"> 1. 封裝：讓外人無法直接讀取，設定 private 後，只能在該 class 中使用 2. 繼承：繼承的目的是讓 class 具有像是親屬的垂直關係（父子），子類別（subclass）可以擁有父類別（superclass）的成員（member） 3. 多型：多型像是親屬的平行關係，多個子類別繼承自單一父類別之時，這些子類別就可以用父類別代替 <p>*引言：</p> <p>類別：使用者自訂的資料型別，屬性預設為 private</p> <pre> class _name(){ privity: member.... public: member.... } </pre> <p>物件(obj):</p> <pre> _name obj; //存取 member Obj._num; //函式 void _name:function(){}; </pre>
9	二	4	學習物件指標	<p>在一般情況下，用點運算子 “.” 來訪</p>

			<p>問物件成員；而將類別宣告物件指標並指向某個物件，需使用 -> 運算子存取該物件的屬性和方法。</p> <p>例：</p> <pre>class exe{ public: void set_a(int a){ x=a; } //定義成員函式 show_a，輸出資料成員的值 void show_a(){ cout<<x<<endl; } private: int x; }; int main(){ exe ob; exe *p; // ob.set_a(2); ob.show_a(); p=&ob; //& p->show_a(); //-> (*p).show_a(); //* return 0; }</pre>
10	二	4	<p>學習堆疊與佇列</p> <p>1. 堆疊：為先進後出的結構</p> <p>I. push:輸入 stack</p> <p>II. pop:輸出 stack</p> <div data-bbox="932 1608 1177 1850"></div> <p>III. 範例：(push A B C)</p> <p>pop C A B 順序不行，因 A, B, C 都 push 過 C 才可出，然後 B 又要在 A 後出，所以不可能。</p> <p>2. 佇列：</p>

				<p>為先進先出的結構</p> <p>input A B C D E F</p> <p>output A B C D E F</p> 
11	二	4	學習鏈結串列	<ol style="list-style-type: none"> 鏈結串列(Linked_List): 結構可不一致(不同資料型態)使用指標標註下一位記憶體, 存放在不連續空間的儲存體。 跟陣列比較的優點: <ol style="list-style-type: none"> 利用 Node 串起資料 操作上比 Array 彈性許多, 可以自由加入刪除 Node 在 "有需要的時候" 才 "動態" 配至記憶體節省記憶體空間 跟陣列比較的缺點: <ol style="list-style-type: none"> linked list 不能直接透過索引值找到某節點 鏈結串列圖型: 
12	二	4	學習樹狀結構	<ol style="list-style-type: none"> 樹狀結構(Tree): 每個點之間都可以找到路徑連通, 從 root 開始往下連接 children, 任兩個 Children 沒有相連, 而 Children 的上一個 Node 為 Parent Binary Tree 圖型:

				 <p>在 Binary Tree 中一個 Node 的 children 限定兩個以內。</p> <p>3. 名詞介紹:</p> <ul style="list-style-type: none"> I. Node(節點): 儲存資料的地方 II. root(根): 最上層的 Node III. parent(父母)、children(子): 被指向的為 Children，指向的為 parent IV. leaf: 沒有 child 的 node 稱為 leaf node V. edge: 連接 Node 表示他們之間有關連 VI. hight : root 到 leaf Node 的距離 VII. depth: 某一 node 與 root 之間的 edge 數
13	二	4	<p>程式-1 分析題目和資料結構</p> <p>給定一個 $n \times n$ 的二維陣列，其中 n 為奇數，從中心位置開始，以順時針旋轉方式走訪每個陣列物件一次。</p> <p>輸入格式:</p> <p>輸入整數 n，n 為奇數且不小於三</p> <p>輸入 0-3 的整數, 代表起始方向, 其中 0, 1, 2, 3 分別代</p>	<p>題目解析: 宣告邊長為 N 的矩陣，並使用 switch-case 分別執行從左、上、右、下開始，並使用 for 迴圈、if 控制走訪路線。</p>

			<p>表左, 上, 右, 下</p> <p>輸入陣列內容, 順序為上到下, 左至右, 同行中的數字以空白間隔</p> <p>輸出格式</p> <p>輸出走訪順序的陣列內容, 為一連串數字, 結尾有換行, 數字中不用輸出空白</p>	
14	二	4	<p>完成程式-1</p>	<p>以第一個測資料為例</p> <p>5</p> <p>0</p> <p>3 4 2 1 4</p> <p>4 2 3 8 9</p> <p>2 1 9 5 6</p> <p>4 2 3 7 8</p> <p>1 2 6 4 3</p> <p>先以正中間那個點為初始的(y, x)</p> <p>他的規律以轉完一圈為一個循環</p> <p>第一個循環就是</p> <p>左 1 上 1 右 2 下 2</p> <p>第二個循環</p> <p>左 3 上 3 右 4 下 4</p> <p>先建立一個外迴圈</p> <p>for(int i=1;;i+=2)</p> <p>公差是 2 而且沒有上限!</p> <p>以第一個循環來看</p> <p>左和上都進行一步</p> <p>從 1 開始</p> <p>到 i 就結束</p> <p>右和下都進行二步</p> <p>從 1 開始</p> <p>到 i+1 才結束</p> <p>也就是說有四個小迴圈</p> <p>兩個</p> <p>for (int j = 1; j <= i; j++)</p> <p>兩個</p> <p>for (int j = 1; j <= i+1; j++)</p> <p>但是</p>

				<p>最後一步不是一個完整的循環 這時候就需要一個計數器！ 先輸出一個中心 然後 count 從 1 開始加 每輸出一個就 count++ 一直到 count==n*n Break</p>
15	二	4	<p>程式-2 分析題目和資料結構</p> <p>列出進入結構和離開結構的整是次序，判斷是堆疊還是佇列。</p> <p>輸入： 一個整數表示測試用例數 每個測試用例包括三行， 第一行是整數個數 第二行列出整數表示進入結構的次序 第三行列出整數表示離開結構的次序。</p> <p>輸出： queue 為佇列 stack 為堆疊 如果結構可為佇列和堆疊，則輸出 both，否則輸出 neither</p>	<p>題目解析:若第 i 個進入結構，第 i 個離開，則該結構為佇列；若第 i 個進入結構，倒數第 i 個離開，則該結構為堆疊。</p>
16	二	4	完成程式-2	
17	二	4	將範例程式放上 Github	<p>網址： https://github.com/yungchih0209/Autonomous-Learning</p>
18	二	4	修改成果表	

成果說明與反思

簡述資料結構內容，並加入插圖解釋結構運作方式。

在計畫執行的過程中，我遇到了許多問題，除了專業領域的問題以外，在與指導老師擬訂計畫的過程中也因為難以在難易度上取得平衡，而產生爭論，但是透過老師一次又一次耐心地指導，終於順利完成這份計畫的內容制定和執行，讓我的程式設計能力更上一層之外，也間接地提升了我的邏輯思考和整理資料佐證的能力。

中 華 民 國 109 年 1 月 12 日