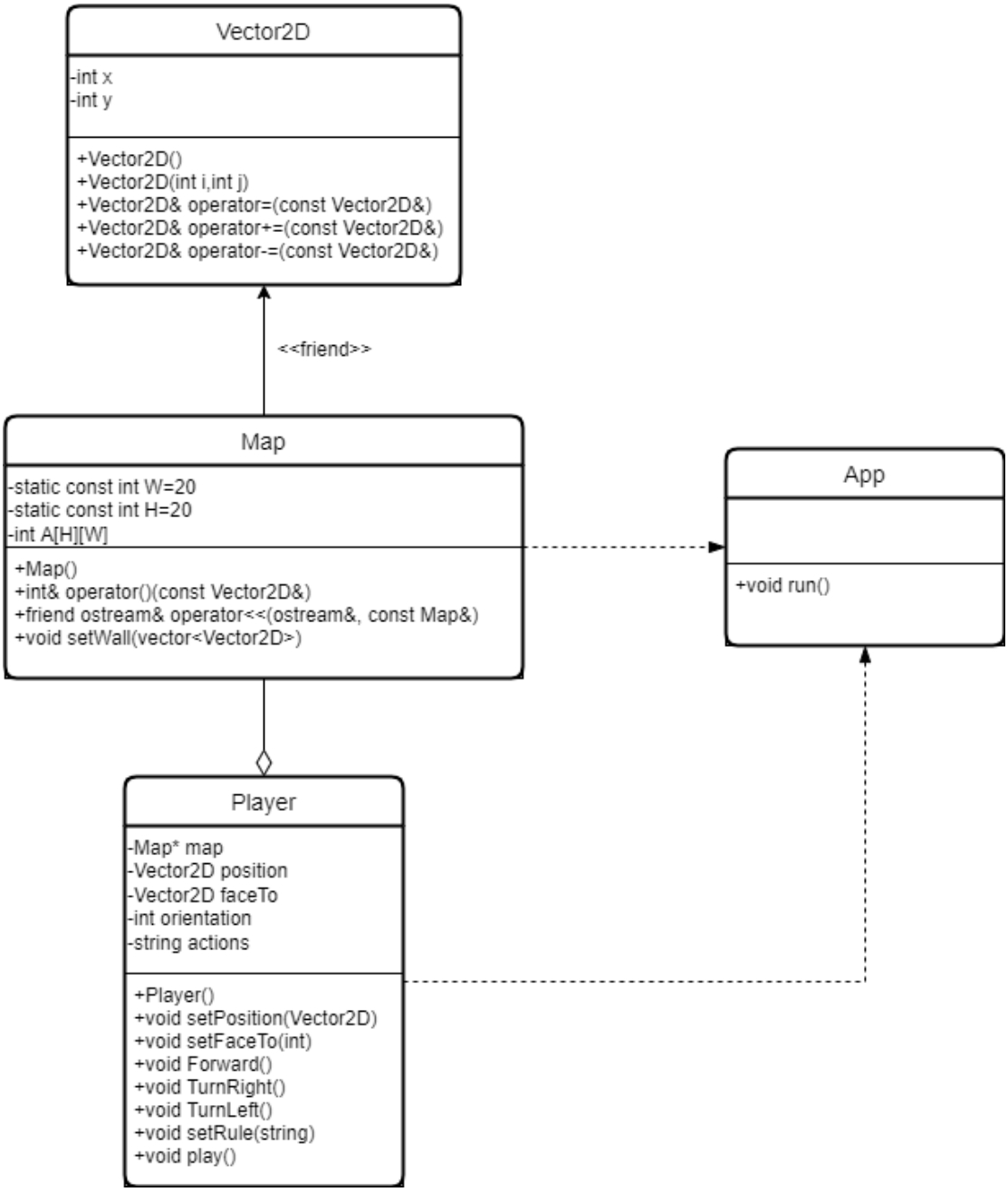
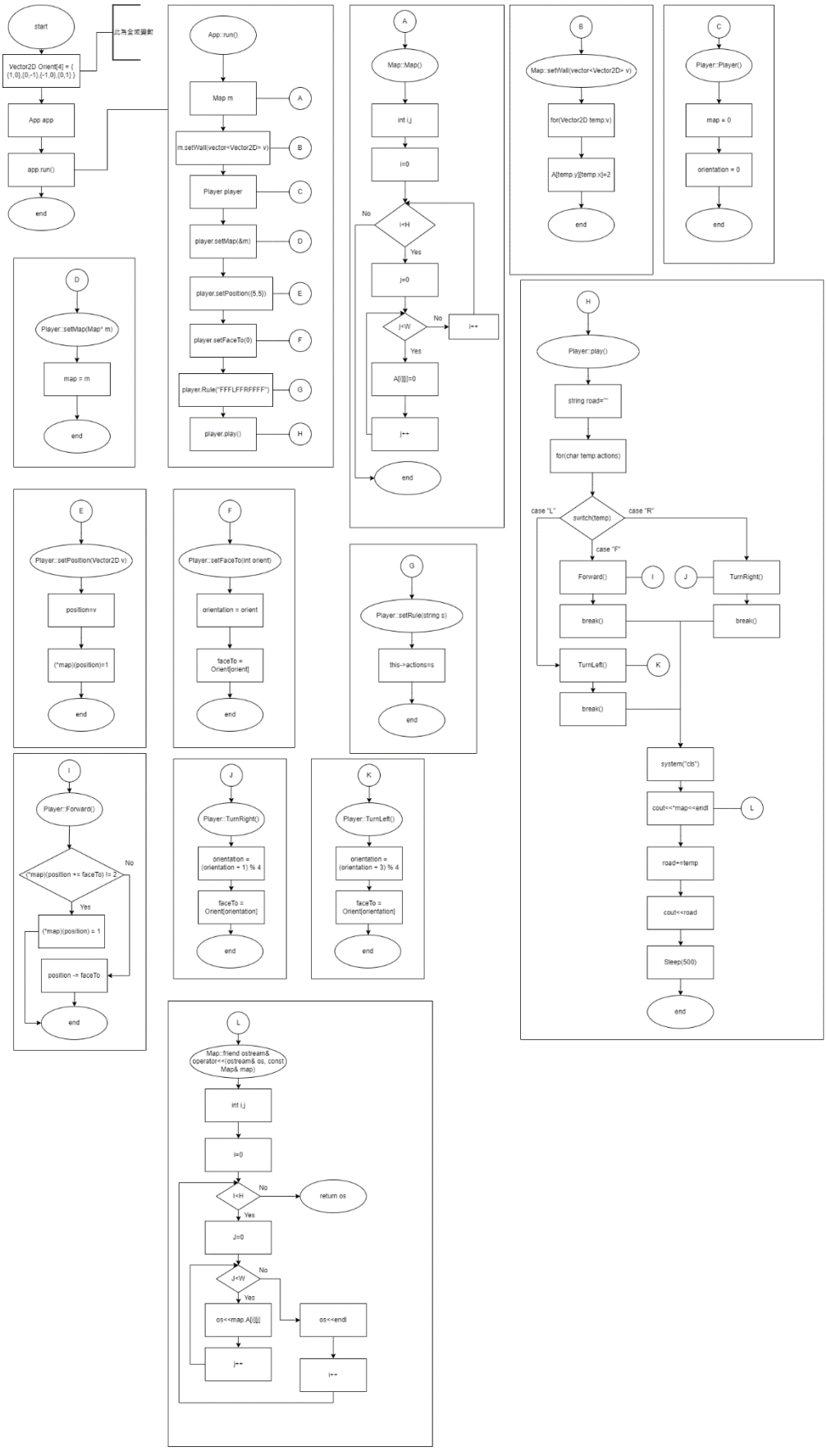
程式設計作業

1072041 陳泳志

類別圖


Vector2D
-int x
-int y
+Vector2D()
+Vector2D(int i,int j)
+Vector2D& operator=(const Vector2D&)
+Vector2D& operator+=(const Vector2D&)
+Vector2D& operator-=(const Vector2D&)
<<friend>>
Map
-static const int W=20
-static const int H=20
-int A[H][W]
+Map()
+int& operator()(const Vector2D&)
+friend ostream& operator<<(ostream&, const Map&)
+void setWall(vector<Vector2D>)
App
+void run()
Player
-Map* map
-Vector2D position
-Vector2D faceTo
-int orientation
-string actions
+Player()
+void setPosition(Vector2D)
+void setFaceTo(int)
+void Forward()
+void TurnRight()
+void TurnLeft()
+void setRule(string)
+void play()

流程圖


start
Vector2D Orient[4] = { {1,0},{0,-1},{-1,0},{0,1} }
設為全域變數
App app
app.run()
end
App::run()
Map m
A
m.setWall(vector<Vector2D> v)
B
Player player
C
player.setMap(&m)
D
player.setPosition({5,5})
E
player.setFaceTo(0)
F
player.Rule("FFFLFFRFFFF")
G
player.play()
H
A
Map::Map()
int i,j
i=0
No
i<H
Yes
j=0
No
j<W
i++
Yes
A[i][j]=0
j++
end
B
Map::setWall(vector<Vector2D> v)
for(Vector2D temp:v)
A[temp.y][temp.x]=2
end
C
Player::Player()
map = 0
orientation = 0
end
D
Player::setMap(Map* m)
map = m
end
H
Player::play()
string road=""
for(char temp:actions)
case "L"
switch(temp)
case "R"
case "F"
Forward()
I
J
TurnRight()
break()
break()
TurnLeft()
K
break()
system("cls")
cout<<*map<<endl
L
road+=temp
cout<<road
Sleep(500)
end
E
Player::setPosition(Vector2D v)
position=v
(*map)(position)=1
end
F
Player::setFaceTo(int orient)
orientation = orient
faceTo = Orient[orient]
end
G
Player::setRule(string s)
this->actions=s
end
I
Player::Forward()
(*map)(position += faceTo) != 2
No
Yes
(*map)(position) = 1
position -= faceTo
end
J
Player::TurnRight()
orientation = (orientation + 1) % 4
faceTo = Orient[orientation]
end
K
Player::TurnLeft()
orientation = (orientation + 3) % 4
faceTo = Orient[orientation]
end
L
Map::friend ostream& operator<<(ostream& os, const Map& map)
int i,j
i=0
No
i<H
return os
Yes
J=0
No
J<W
Yes
os<<map.A[i][j]
os<<endl
j++
i++

說明

```
class Vector2D {
private:
	int x;
	int y;
	friend class Map;
public:
	Vector2D() { x = y = 0; }
	Vector2D(int i, int j) { x = j; y = i; }
	Vector2D& operator=(const Vector2D& other) {
		x = other.x;
		y = other.y;
		return *this;
	}
	Vector2D& operator+=(const Vector2D& other) {
		x += other.x;
		y += other.y;
		return *this;
	}
	Vector2D& operator-=(const Vector2D& other) {
		x -= other.x;
		y -= other.y;
		return *this;
	}
};
```

類別 Vector2D，有兩個 int 變數用來存上下和左右，就像二維陣列那樣，運算子重載是讓這個類別也能直接用=,+=,-=

```
class Map {
private:
     static const int W = 20;
     static const int H = 20;
     int A[H][W] = {};
public:
     Map() {
          int i, j;
          for (i = 0; i < H; i++)
               for (j = 0; j < W; j++)
                    A[i][j] = 0;
     }
     int& operator()(const Vector2D& v) {
          return(A[v.y][v.x]);
     }
     friend ostream& operator<<(ostream& os, const Map& map) {
          int i, j;
          for (i = 0; i < H; i++) {
               for (j = 0; j < W; j++)
                    os << map.A[i][j];
               os << endl;
          }
          return os;
     }
     void setWall(vector<Vector2D> v) {
          for (Vector2D temp : v) {
               A[temp.y][temp.x] = 2;
          }
     }
};
```

類別 Map，兩個 int 用來存地圖的長寬，A[H][W]就是地圖

重載()運算子，用途是改 A 特定位置的值

重載<<運算子，輸出 A 裡面的所有內容

setWall，傳入一個 vector<Vector2D>，然後用 for 迴圈根據傳入的這些 Vector2D，把 A 在這些位置的值改成 2，代表牆壁，後面移動的時候如果是牆壁就不會移動

```
Vector2D Orient[4] = { {1,0},{0,-1},{-1,0},{0,1} };
```
這個存的是4個面朝的方向分別是下、左、上、右

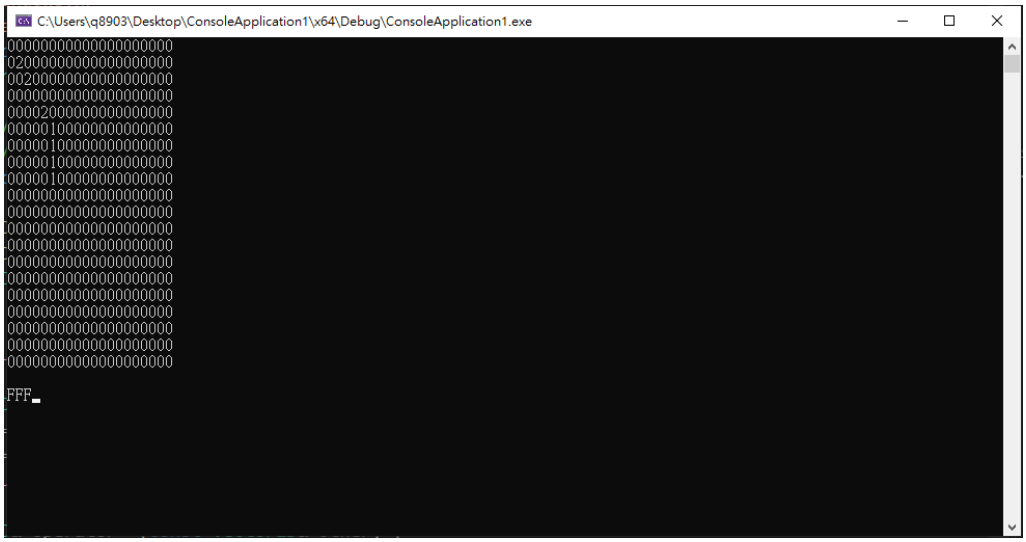
```
class Player {
private:
    Map* map; 地圖
    Vector2D position; 當前位置
    Vector2D faceTo; 面朝方向
    int orientation; 面朝方向的編號
    string actions; 路線
public:
    Player() { map = 0; orientation = 0; }
    void setMap(Map* m) {
        map = m; 設定地圖
    }
    void setPosition(Vector2D v) {
        position = v;設定位置
        (*map)(position) = 1;將該位置改成1
    }
    void setFaceTo(int orient) {
        orientation = orient;設定面朝方向編號
        faceTo = Orient[orient];根據編號設定面朝方向
    }
    void Forward() {
        if ((*map)(position += faceTo) != 2) { 先往前走，然後看是不是牆壁
            (*map)(position) = 1;                不是的話就把該位置改成1
        }
        else {
            position -= faceTo;                  是牆壁的話就往回退
        }
    }
    void TurnRight() {
        orientation = (orientation + 1) % 4;
        右轉就是編號+1，因為是四個方向所以取4的餘數，這樣就可以循環
        faceTo = Orient[orientation];
    }
    void TurnLeft() {
        orientation = (orientation + 3) % 4; 因為是循環所以左轉就是編號+3
```

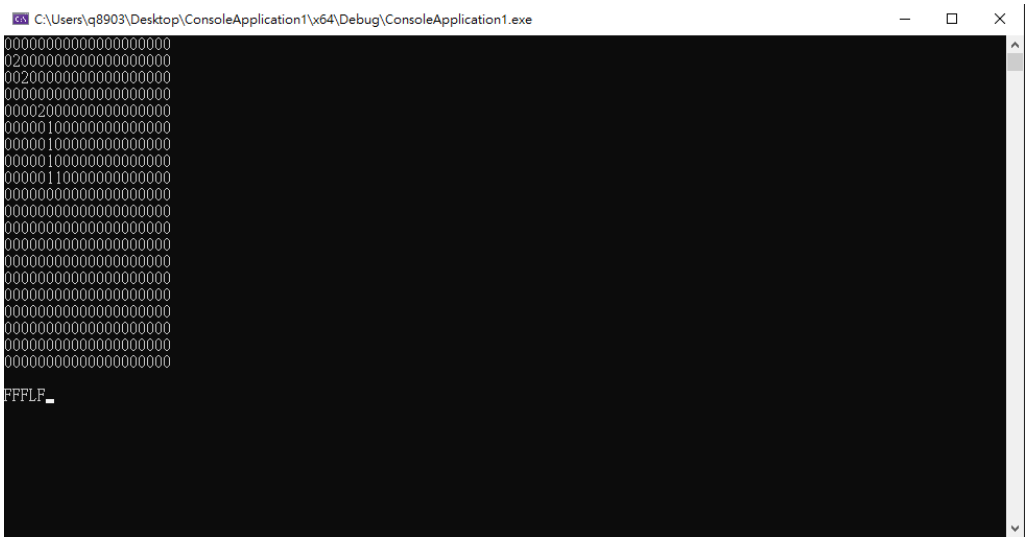
```
        faceTo = Orient[orientation];
    }
    void setRule(string s) {
        this->actions = s;設定路線
    }
    void play() {
        string road = "";已經走過的路線
        for (char temp : actions) { string其實就是char陣列，所以用一個for迴圈遍歷每個
            switch (temp) {          字元，再用switch判斷應該怎麼走就好
            case 'F':
                Forward();
                break;
            case 'L':
                TurnLeft();
                break;
            case 'R':
                TurnRight();
                break;
            default:
                break;
            }
            system("cls");           這邊我希望能可以一步一步顯示執行的畫面，所以每
            cout << *map << endl;    次顯示地圖前先清空在輸出，我也希望能顯示走到哪
            road += temp;            步，所以每走一步就把走過的步記錄起來然後輸出，
            cout << road;            用sleep讓程式每0.5秒跑一次
            Sleep(500);
        }
    }
};
```

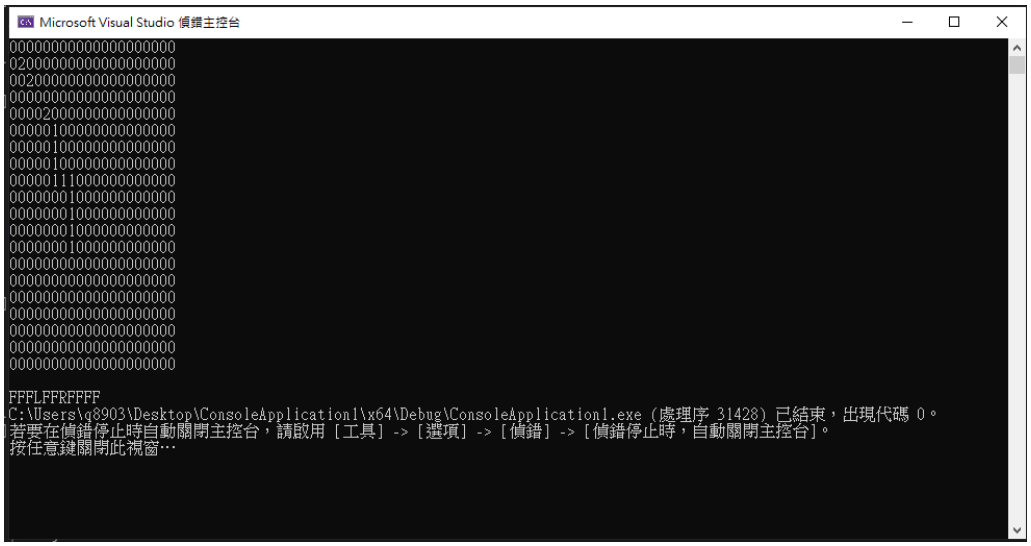
```
class App {
public:
	void run() {
		Map m;
		m.setWall(vector<Vector2D>({ {1,1},{2,2},{4,4} }));
		Player player;
		player.setMap(&m);
		player.setPosition({ 5,5 });
		player.setFaceTo(0);
		player.setRule("FFFLFFRFFFF");
		player.play();
	}
};
int main() {
	App app;
	app.run();
}
```

這邊就是執行程式而已

執行結果


C:\Users\q8903\Desktop\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe
FFF


C:\Users\q8903\Desktop\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe
FFFLF


Microsoft Visual Studio 偵錯主控台
FFFLFFRFFFF
C:\Users\q8903\Desktop\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe (處理序 31428) 已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [偵錯] -> [偵錯停止時，自動關閉主控台]。
按任意鍵關閉此視窗...