|  |  |
| --- | --- |
| Elevator Customer scheduler  [文件副標題] | 作業系統期末專案  第十二組 103703033 吳綠庭  103703035 賴宥安  103703047 黃彥魁 |

**目錄**

**ㄧ、報告主題簡介…………………………………………………P.3**

**二、組員分工……..…………………………………………………P.3**

**三、執行緒(Thread)類型以及功能………………………P.4**

**四、共有資源與共享變數……………………………………..P.5**

**五、關鍵區域程式（使用C++）………………………………P.6**

**六、文字顯示狀態與結果……………………………………..P.7**

**七、GUI呈現………………………………………………………..P.8**

**八、參考資料………………………………………………………..P.9**

**九、Github…………………………………………………………..P.9**

**ㄧ、報告主題簡介**

**本組的實作題目是Elevator Customer Schedule Problem。我們必須實作一個趨近無限快、每一層樓間只需花好100 ticks的電梯。當沒有人按電梯時它會處於休眠狀態，收到來自Passenger的通知後便會醒來並前往指定樓層載客。**

**而Passenger有兩個參數：startFloor和endFloor，分別代表按下電梯的樓層和欲前往的樓層。在隨機的時間點會產生隨機變數的Passenger，電梯必須判斷現在應該前往哪一個樓層接獲送出乘客。**

**二、組員分工**

* **103703033吳綠庭： 主程式優化（電梯演算法優化、Poisson隨機分佈時間生成乘客）、GUI圖片製作**
* **103703035賴宥安 ： 主程式建構（電梯載客雛形、乘客生成雛形、Multi-Thread管理）**
* **103703047黃彥魁 ： GUI建構、主程式細節調整**

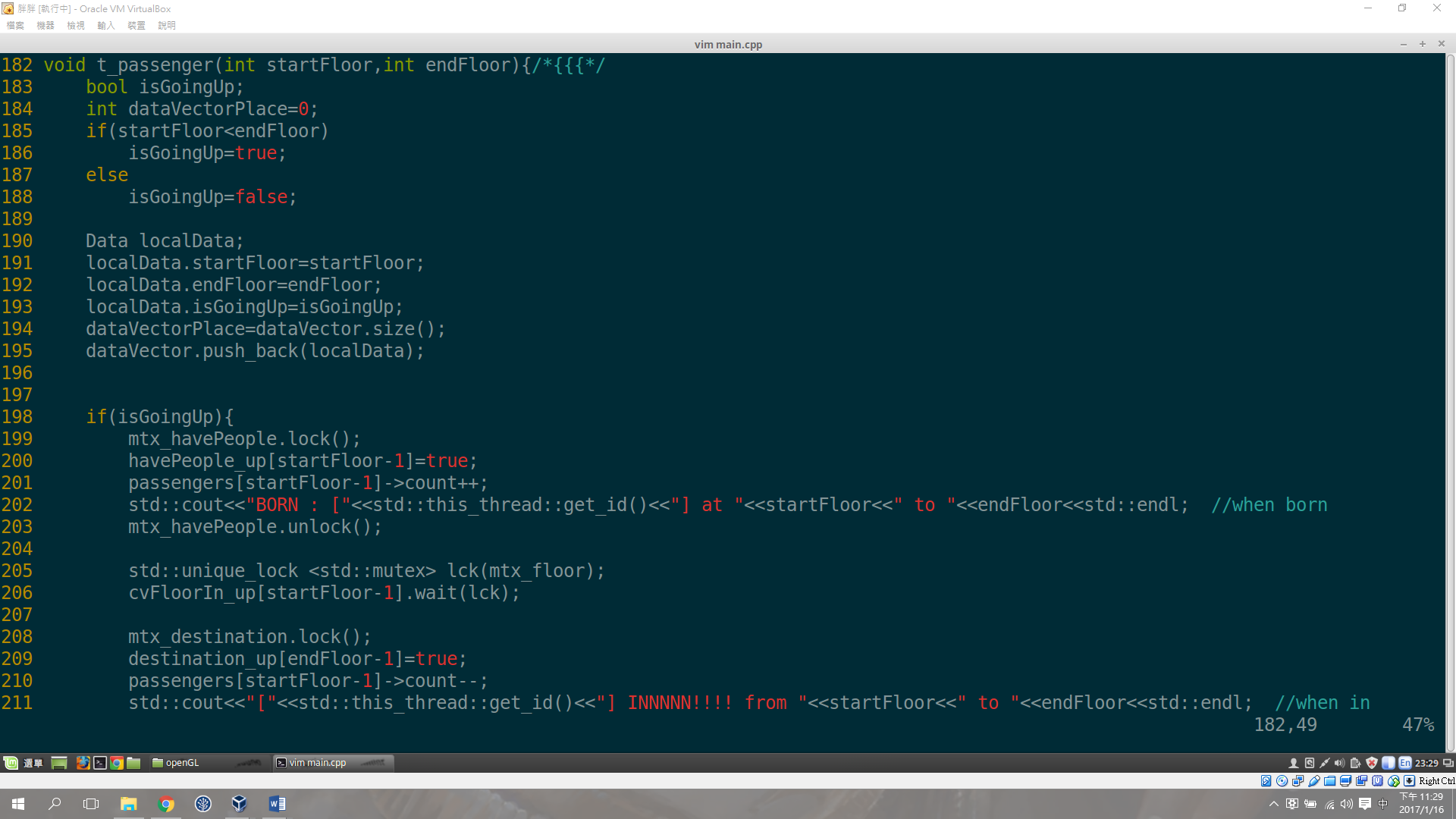
**三、執行緒(Thread)類型需求以及各自功能**

依照題目要求，我們將要搭乘電梯的乘客與電梯分別做成執行緒

1. 乘客的執行緒(void t\_passenger(int startFloor,int endFloor))：

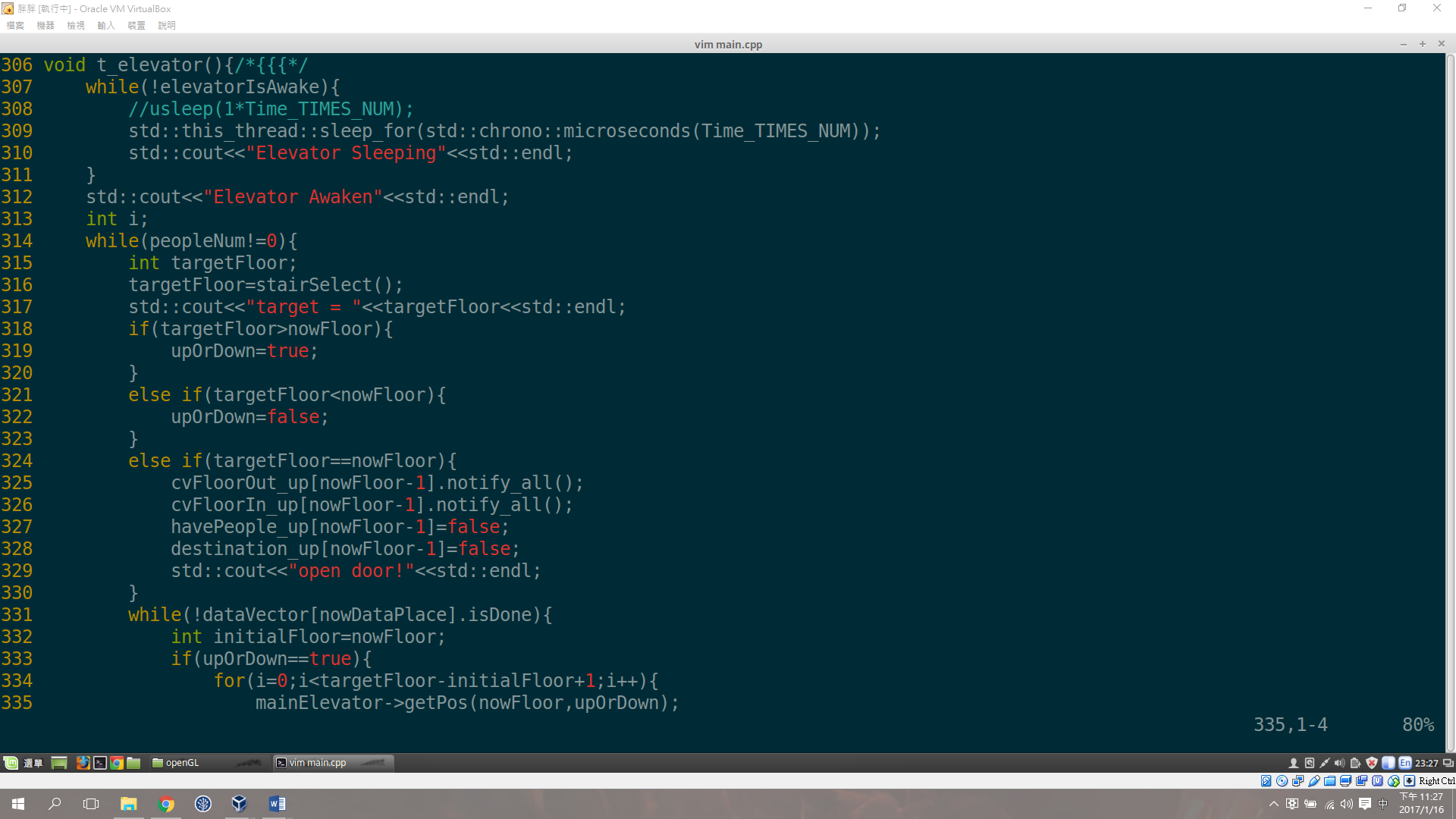
每個隨機產生出來的乘客，都各自擁有獨立的thread，裡面做的事情依序為:按電梯(上樓或下樓)->等電梯->進電梯並按下欲前往之樓層->等待電梯到達目的地->出電梯，當此執行緒被判定為出電梯之後，就結束。

（此圖僅上樓程式碼，下樓與上樓雷同，為免冗長故不列出）



1. 電梯的執行緒(void t\_elevator()):

此執行緒是用來模擬電梯的移動透過Shared Variables來取得資訊並移動，而電梯會有上樓、下樓，或者是沒人按時停止不動，此執行緒並不會因為沒有乘客而結束，而是繼續等待下一個隨機產生出來的乘客。

（此圖僅上樓程式碼，下樓與上樓雷同，為免冗長故不列出）

**四、共有資源與共享變數(Shared Variables)需要進行Multi-Thread (or -Process)之間的同步與合作**

1. Shared Variables:

bool havePeople\_up[NumberOfFloors];

有要上樓的乘客的樓層

bool destination\_up[NumberOfFloors];

上樓乘客要去的樓層

bool havePeople\_down[NumberOfFloors];

有要下樓的乘客的樓層

bool destination\_down[NumberOfFloors];

下樓乘客要去的樓層

以上的Shared Variables是為了要讓電梯與乘客去做溝通，例如:乘客在五樓想要去七樓，那麼havePeople\_up[4]就會被改成true，進電梯後，因為已經載到人了，電梯執行緒又會把它改成false，然後乘客執行緒就會將destination\_up[6]改成true，電梯將人載到目的地後，再改成false，以此類推。

1. Mutex:

std::mutex mtx\_havePeople;

為了不要讓havePeople\_up[NumberOfFloors]和havePeople\_down[NumberOfFloors]被乘客與電梯同時修改。

std::mutex mtx\_destination;

為了不要讓destination\_up[NumberOfFloors]和destination\_down[NumberOfFloors]被乘客與電梯同時修改。

std::mutex mtx\_floor;

與condition variable配合使用

1. Condition Variable:

std::condition\_variable cvFloorIn\_up[NumberOfFloors];

notify\_all()在N樓等上樓電梯的乘客進電梯

std::condition\_variable cvFloorOut\_up[NumberOfFloors];

notify\_all()要上到N樓的乘客下電梯

std::condition\_variable cvFloorIn\_down[NumberOfFloors];

notify\_all()在N樓等下樓電梯的乘客進電梯

std::condition\_variable cvFloorOut\_down[NumberOfFloors];

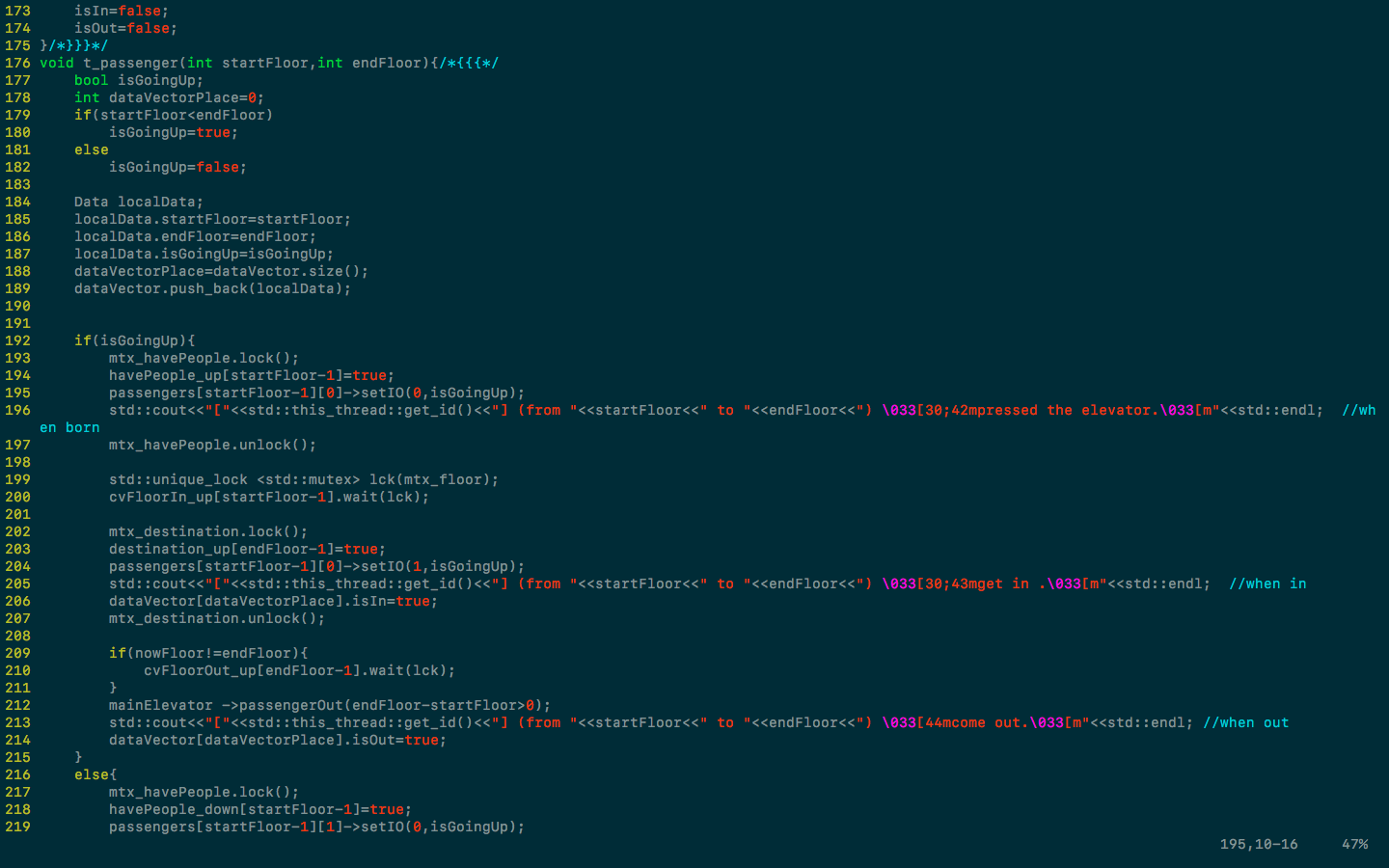
notify\_all()要下到N樓的乘客下電梯

1. Multi-Thread:

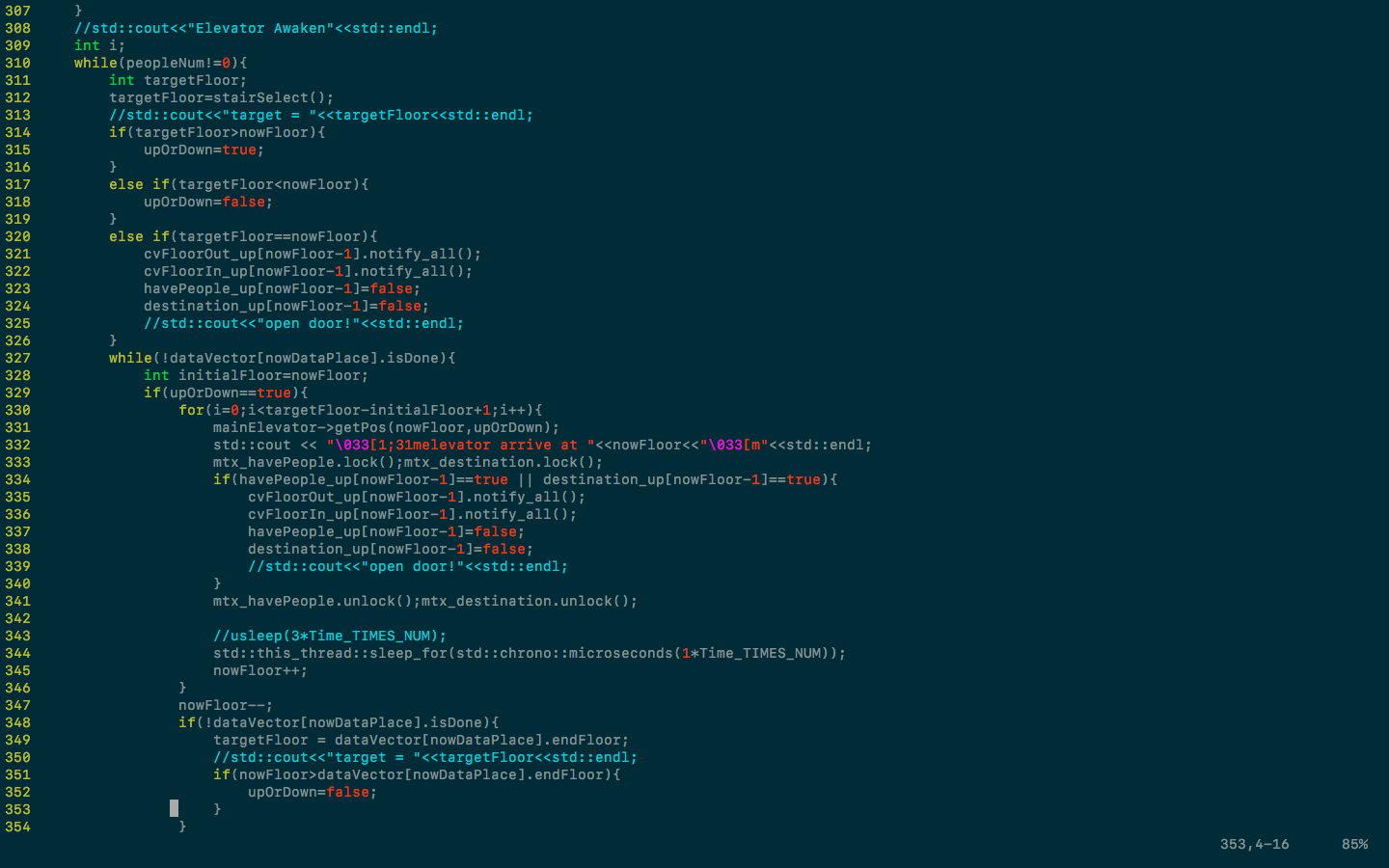
電梯先等待乘客，一有乘客之後，就取得data並且前往載客，並用Condition Variable來告訴乘客何時該進出電梯。

**五、關鍵區域程式（使用C++）**

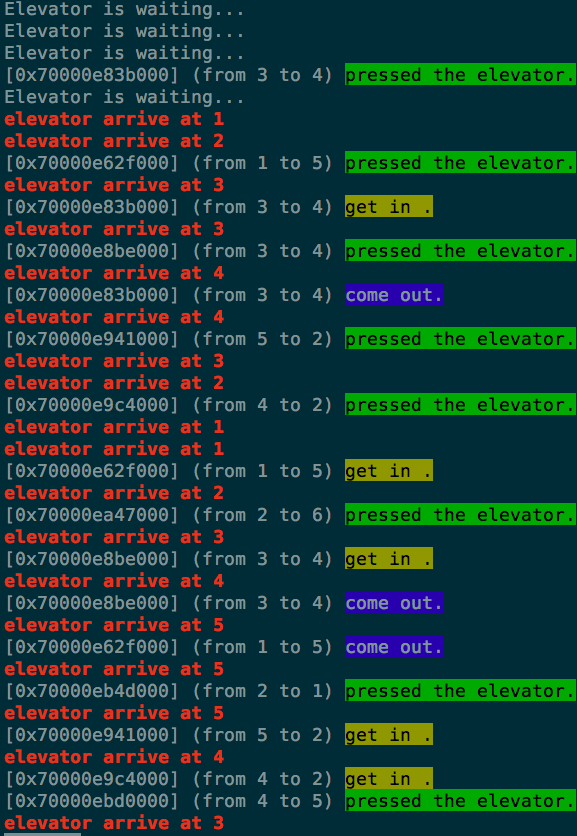
t\_Passenger底下，每個時間點只能有一個passenger按電梯

****

t\_Elevator底下，passenger出電梯後電梯要改變樓層狀態，所以同一時間點不能有passenger按電梯

****

**六、文字顯示狀態與結果**



**七、GUI呈現**

ＧＵＩ一開始試圖使用Qt實作，但在主程式出來後發現不大能把std::thread接到Qt上，也許是對Qt比較不熟的緣故，在試了幾次後放棄改使用OpenGL



**八、參考資料**

[Poisson Distribution](http://www.cplusplus.com/reference/random/poisson_distribution/)

[C++ 的多執行序程式開發 Thread：基本使用](https://kheresy.wordpress.com/2012/07/06/multi-thread-programming-in-c-thread-p1/)

[C++11 并发指南一(C++11 多线程初探)](http://www.cnblogs.com/haippy/p/3235560.html)

**九、Github**