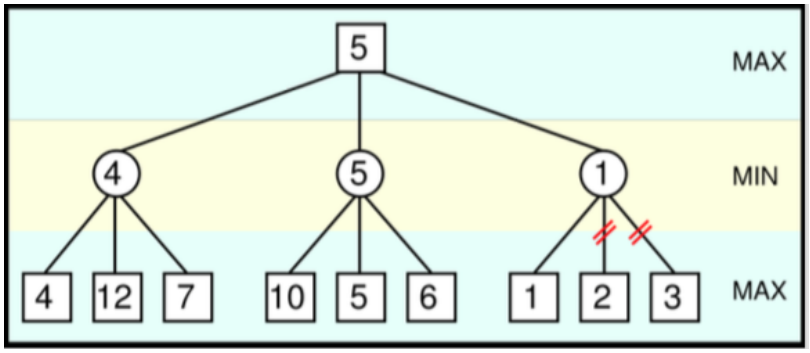
- 程式語言：C++
- 執行環境：Code::Blocks
- 程式說明：

為賦予電腦思考的功能，首先電腦需要有盤面評估函數，才可以針對棋局優劣判斷下手較佳的位置。盤面評估函數即分數給定的機制，在此對於每一種棋型我們皆給予適量的分數(棋型包括活二至活五、死二至死四)。每一回合中，判斷當前局面玩家與電腦全部各有多少棋型組合，尤棋型組合所產生出的分數總合可得玩家與電腦的個別總分，兩者相減即可得當回合玩家與電腦的分數差距。程式實作上給定死二 1 分、活二與死三 10 分、活三 1200 分、活四與死四 30000 分、活五無限大。程式中的 getScore()函式就是盤面評估函數，會在每個棋點位子上判斷橫排、直排和斜排共有多少顆其餘連續的友方棋子，再偵測是否到達邊界、或是有對方棋子是否擋住活路，以判斷棋型的死活。

電腦預測下一步的著手點，就是會分析能得分最多、失分最少的位置。然而，若是電腦下的每一步棋皆只考慮當前的分數，而忽略未來整體的因素，很容易至終因著疏忽整體局勢而輸。因此除了評估函數，我們也需要搜尋多層對局的演算法。此程式以 min-max tree 為架構，做深度為 2 的搜尋機制。以下圖為例：



假設偶數層為電腦下子、奇數層為玩家下子，樹中的根節點表當前棋局，其子節點集合為模擬(simulation)過程中所產生的可能盤面。程式會不斷往下搜尋直至葉節點(深度為 2)時返回，並倒傳(back-propagation)葉節點盤面的評估分數給根結點。邏輯上，當模擬電腦下子時，電腦一定會選得分最多的棋路，而模擬玩家下子時，玩家一定會找使電腦失分最多的棋路；基於以上，符合了 min-max tree 不同回合找最大值、最小值的精神。在此深度設為 2，除了計算上效率較佳，實測出來也足夠有一定的棋力。實作上，我們以遞迴呼叫(recursive call)實現。為節省搜尋成本，搜尋上我們給予條件限制：下一步棋的位置只能坐落在當前棋盤中所有棋子周遭距離為一的範圍，如此限制也能節省資源、避免模擬許多不合理的棋路。

有了預測未來棋路的機制後，我們以當前棋局所算出的當下分數作為探勘分數(exploitation score)，及 min-max tree 所產生的未來盤面評估分數作為開發分數

(exploration score)，兩者做權重所得到的分數成為棋盤的實際分數。經驗上，給予探勘分數的權重為 0.2，而開發分數的權重為 0.8。藉由以上概念，即可模擬出具有一定棋力的五子棋 AI。