Printed Circuit Board (Spoiler)

It should be quite obvious that this task could be modeled as a graph coloring problem. Indeed, we could create a graph with a vertex for each conductor and an edge between vertices for conductors that would intersect if they were in the same layer. Then obviously a vertex coloring (an assignment of colors to vertices such that no edge would connect two vertices of the same color) could be used to distribute the conductors to layers.

The problem with this approach is that vertex coloring with minimal number of colors is known to be NP-complete. Thus, the best known algorithm for solving the problem in the general case is exhaustive search, which is much too slow to consider for a graph with 100,000 vertices. Such a solution is given in the file `pcbsol1.pas` and would score about 40 points.

Only a small subset of all graphs are models of boards that could appear as the input in this task, but it is not at all obvious how to exploit this fact to speed up the search for a vertex coloring. Thus, it is probably more fruitful to look for another way to model the input data.

And indeed, if we sort the conductors in the order of the X-coordinates of the endpoints on the bottom side of the board, then the potential intersections are defined by the X-coordinates of the endpoints on the top side of the board. In fact, the number of layers needed is exactly the length of the longest decreasing subsequence in the X-coordinates of the top-side endpoints, when the X-coordinates of the bottom-side endpoints are in increasing order.

The longest decreasing subsequence of a given integer sequence can be found using a simple dynamic programming solution: for each element A_i, we compute Z_i, the length of the longest decreasing subsequence that would end with that element; to compute Z_i, we only need to find the largest Z_j such that $j < i$ and $A_j > A_i$. This can be done trivially using a linear scan of all $j < i$, for a solution with $O(N^2)$ running time. Such a solution is given in the file `pcbsol2a.pas` and would score about 70 points.

For a more efficient solution, we could build an index on top of the known values of Z_i. One way to do this is to create a binary tree with $W + 1$ leaves and all non-leaf nodes containing the maximum values in their corresponding subtrees. Initially we fill the tree with zeroes. Whenever we compute a new Z_i, we store it in the leaf number A_i in the tree and update the maximums on the path from the updated leaf to the root in $O(\log W)$ time. To compute Z_i, we only need the maximum value currently in the tree among the leaves numbered greater than A_i, which we can also compute in $O(\log W)$ time. Therefore we can solve the whole problem in $O(N \log W)$ time. Such a solution (using heapsort to order the conductors) is given in the file `pcbsol2b.pas` and would score full points.