Explanation

The first thing I would like to make note of is the fact that an armor set is not defined by, but includes, one piece of each type of armor. We will want to look at every combination of armor that makes up a set, and so my approach uses four nested for-loops to walk through four arrays, each one corresponding to one armor type. Within the inner-most for-loop, we cycle through all pieces of armor that we may use as the extra piece that completes a set with while loops. Although the same could’ve been done with more for-loops, I wanted the separation of importance and structure between the four first pieces and the extra piece to be visually clear. We do this in an order that lines up with how the arrays and loops are ordered – helmets, chests, leggings, and then boots. Furthermore, the extra piece will “start” at the value of each outer loop index and increment until the end of each array to avoid some needless checks. If the extra piece was ever earlier in an array than any index in any outer loop, we would end up double-checking for many cases such as H0C1L1B1H2 and H2C1L1B1H0. As we inspect each complete armor set, we compare its armor value to the highest one we have seen so far. If we find a set that is stronger, we will take down the information if we can afford it. We continue until we have inspected all possible combinations of armor and then we are done. All that’s left is to display the relevant information for the best set of armor if we found one. If we didn’t find one throughout the whole process, it means that Geralt wasn’t able to afford one and that is shown instead.

Thoughts

I believe that my program would be successful with other inventories. The amount of money Geralt has and the items the armorer has for sale can be changed but the logic would stay the same. Although this program was made specifically for four armor types, I think it is worth mentioning that it wouldn’t take much effort or many changes to adjust the number of armor types present if we needed to. The worst part in my mind regarding reusability is the simple data entry of the armor pieces and their respective details. Mistakes in data entry could very easily lead us to an incorrect answer. While that isn’t something wrong with the program, it should be noted nonetheless. If the mistake is just a typo for a price or armor value, the program will run, and the error could potentially remain unknown to us. If the shop had a massive number of items, one would have to enter a massive amount of data. If the information always came to us in a strictly structured way like a text file with convenient delimiters or if we could pull it from an API or something of that nature, the program could be altered so that there would be no data entry but that is not the case here.