The runtime complexity of my code can be analyzed as follows:

1. **calcOblongCorner function:** The calcOblongCorner function has a time complexity of O(1) since it executes a number of operations regardless of the input size.
2. **arbitraryMinMax function:** This function operates in the manner. Its time complexity remains constant denoted as O(1). It accomplishes this by generating a number from a range.
3. **doRectanglesOverlap function:** Similar to the functions this particular one also has a time complexity that remains constant denoted as O(1). This means that it executes a predetermined number of operations without regard to the magnitude of the input.
4. **main function:** The main function has a loop that runs a number of times determined by the user's input. Inside this loop there is another loop that checks if any of the created rectangles overlap with each other. In the worst case scenario when all rectangles overlap with each other it would take a time complexity of O(n^2) where n represents the number of rectangles.
5. **Writing to file:** The process of saving the information, for each rectangle, to the file requires a time complexity of O(n) with 'n' representing the number of rectangles.

After analysis, I determined that my code's worst-case time complexity is O(n^2) due to the nested loop in the function. In contrast, the best-case scenario occurs when none of the rectangles overlap, resulting in a linear time complexity of O(n log n), where n represents the number of rectangles.