This question has been answered countless times by recruiters, employees and candidates in full capacity. But I will summarize it for you based on what I did and my mistakes.

* 1. Revise the working and use  of popular data structures like linked list, trees (binary, treap, (red  black, avl : optional)), priority queues, heap (min, max heaps), stack,  hashtable
  2. Learn the equivalent of these theoretical data structures in the language of your choice. e.g. in Java you have HashMap, LinkedHashMap, TreeSet, HashSet, etc.
  3. Learn the implementation differences in these as you have to write real code, not pseudo code during the interview.
  4. Go through standard algorithms which you learn in school e.g. different sorting algorithms, graph traversals, MST, shortest path, etc.
  5. Learn **Big O** notation and make sure you can deduce the time complexity of any algorithm (at least the ones you write)
  6. Pay attention to space complexity. ***Edit 1:*** *This is really important. You will realize this when you actually write code. The amount of space wasted in auxiliary data structures, and random arrays and hashmaps for various purposes is phenomenal.*
  7. Practice implementing **1**, **2** in the language of your choice. During implementation, try to write code without external reference for things like getting console input, file operations and so on. If you do refer, make sure you don't have to look again. ***Edit 1:*** *This would really get you into the comfort zone of the language that you, till date, thought you were an expert at.*
  8. Implement different versions each optimized for time or space complexity as you will be given problems in which input takes a lot of space and need a lot of time to parse. So you need to think in 2 dimensions (time and memory footprint. ***Edit 1:*** *For example, try to implement in-place versions of the various sorting algorithms you studied.*

1. Practice interview questions. Pick up a book like Cracking the coding interview or Programming Challenges by Skiena and solve a considerable number of problems both on PC with IDE as well as on paper/ whiteboard. **Force yourself to solve it to the end irrespective of how lame your answer is. *Edit 1 :*** *Start out with a brute force solution. Always. Once you have it penned down, you will be able to find things which can be done faster. I believe optimization is more structured when done incrementally. This also gives the interviewer a chance to see your logic. Finally, DO NOT forget the f\*\*kin HashMap, cuz it's just awesome. An example of incremental optimization would be: doing a brute force n^2 search -> n log n sort, followed by binary search -> HashMap,*
2. Practice solving problems on whiteboard and try to speak out your logic as you arrive at the solution. You have to have sound explanations for choosing a particular algorithm or a data structure. ***Edit 1:*** *Once you have done this a few times, time yourself. Simply because you can't take forever. My last interview was a screw up because I just took too long, and I didn't articulate my ideas clearly.*

These are the most important things IMO. I made a mistake by not practicing a lot. Try to finish steps 1,2 in 1.5 weeks max. Spend at least a week on solving problems unknown to you on paper or notepad (without code complete). 1/2 days on whiteboard coding.  
  
Mental preparation

1. It is equally important to not panic. It might help if you purposefully try to think that you are better than Google standards.
2. If you panic or get intimidated by the sheer awesomeness of the work they do or the rockstar computer scientists they have on-board, you will most certainly not make it.
3. Stay calm, stay focused. Do not waste time.
4. Assume things will go bad. This way, the only emotion you can ever experience during the interview is excitement from having figured out how to solve the problem.
5. Ask probing questions about which team and what technologies the interviewer is working for/on. This gives them a chance to talk and feel good about themselves. ***Show some interest..***

Be nice to the recruiter, not in a boot-licking way, but just be nice and courteous. Reply to mails on time. Be jovial, as they see loads of Google fanatics daily.  
  
My answer might not be perfect. But I think you will do well if you plan your preparation this way