Here I am sharing a possible path to follow for improving programming skills :

The first and most important thing is Mathematics required for Computer Science

1. Discrete Mathematics- Mathematics for Computer Science
http://www.4shared.com/office/4uZwco9F/Discrete_Math_-_Mathematics_Fo.html

2. Mikolas Bona - A walk through combinatorics
http://www.4shared.com/office/pRJdYArR/A_walk_through_combinatorics.html

3. Problem Solving Strategies
http://www.4shared.com/file/-f-nFuK7/arthur_engel_problem_solving_s.html

4. Concrete Mathematics - Knuth
http://www.4shared.com/office/EaOiUsOP/Concrete_Mathematics.html

5. Peter Wrinkler and Martin Gardiner books on puzzles.(optional)

These books give very nice insight to problem solving and should be read as much as possible irrespective to semester or preparation level.

For practicing programming(first year)

K&R - The C programming language + C++(use STL as far as possible)
The exercises in K&R are nice and very instructive.

1. Get some basic programs running in these languages and play around with common tasks till you feel comfortable with basic constructs mainly functions and recursion.

2. Pay attention to K&R so that you can have better understanding of memory usage and pointer arithmetic.

3. Write as many programs as possible in 1st/2nd semester. See programs of others and in books to acquire good coding practices in order to write clear and concise code.

4. Refer to http://c-faq.com/ for C pitfalls and basic concepts. Good for revision.

5. Once you are comfortable with basic programming , start practicing at topcoder.com/tc solve almost all Div-2 250 and 500 pointer in practice room or live contest.

From second semester one should start concentrating on Theory of Data Structures and algorithms.

1.The most important and most ignored resource
Introduction to Algorithms - Cormen
Initial chapters are mainly on data structures. You can use MIT video lectures as companion.
Read all the stuff in Cormen and solve all the exercises(very very important) in the language of your choice.Try to write clean and working code and test on your hand made testcases.

2.For algorithms refer again to MIT lectures + nptel lectures
Books most useful other than cormen and its exercises are:
Algorithms - Dasgupta and Vazirani, very intuitive and non mathematical treatment, must read
Algorithm Design - Kleinberg and Tardos

3. As you carry along with step 1 and 2 keep in touch with http://train.usaco.org/usacogate. It has chapter by chapter tutorials and programming questions with automatic judging and you can see best solution after you correctly submit your solutions. However all solutions

can be google searched but do not do that unless you desperately want to move to next level.

4. Step 3 honestly covers it all.

The above steps can be done for complete second year.

For third year.

Now comes time of random problem solving which should not be done before second year. Randomly picking problems and solving doesn't make you systematic in solving new problems, so third year onwards is the right time.

1. Start practicing previous year's ACM ICPC problems of as many sites as possible in increasing order of difficulty, Topcoder Div-2 100 etc. Read as many code of other good coders or solution for ACM ICPC problems.
This process with help in gaining many techniques.

COCI(http://hsin.hr/coci/) is also a great site with solutions to all questions and problem set are indeed very interesting and range from easy to hard so its ideal for anyone to practice.

2. Make a doc of all the major algorithms you use and always try to copy/paste from the doc in contests. Be adapted to type fast and use your doc as far as possible.

Our Doc :
https://docs.google.com/document/d/13W4fS6n9aN43Ft9yNQm0OXcEA8asB99BQgpqIS0NKBo/edit

3. Most importantly always explain to someone your approach for solving a problem and keep discussing/listening to others. Anyone can have different insight to any propble and believe me you can learn a lot

in this process.

4. If you think you know an algorithm just randomly explain it to your friend on paper. You will be amazed by how much you do not know.

5. By this time you will be expert at writing codes on paper and checking correctness/complexity without using computer. In fact I have used paper/pen 10x times more than terminal/keyboard.

Read as many proofs of algorithms as possible.

Participate in ACM-ICPC or any programming contest you get hands on, irrespective of your preparation level. Just go to onsite and feel the heat to get motivated and cool T-Shirts of course :)

Regarding placements(I am not good at them)
If possible get some good pointers.

Do all questions asked in interviews. Easily available at GeeksforGeeks/Careercup/Leetcode.

Most importantly donot just read the solutions. Implement clean and working code on you computer and test with own testcases.
Take each other's interviews :)

Be confident and very friendly in inteve erviews. Ask proper and intelligent questions. Many times if you are good at it, you can make interviewer
spit the working solution for you :)

Steve Yegges blog has better insight on this.

During the terms read some course stuff or listen to teacher in class mainly in OS/FAT/Compilers/DBMS.

One last remark :

Solve as many puzzles you can find and try coursera's courses which you find interesting. Computer Science course pages of Stanford/MIT/CMU/Harvard etc contain a lot of interesting links and tutorials.

I am not very good at these things but I feel these steps are worth following. These came out of the guidance and support that Surendra Meena Sir, Imran Sir, Hemant Verma Sir , Ankul Garg Sir, many other senior and awesome programmers(yes all of them are programmers) of my B2k8 batch gave me.

If you find it useful please share with your classmates and juniors.