PROGRAMMING OLYMPIADS

Getting Started with Competitive Programming

Set up your environment, learn C++ the way contests actually use it, and get your first accepted solution on a real judge.

SKILL LEVEL: BEGINNER **BUILD TIME:** 1 WEEKEND TO FIRST SUBMISSION **EST. COST:** $0

> Before you start: you don't need to know C++ already — basic logic from any language (Python, Java, Scratch) is enough. What matters is comfort with variables, loops, and functions.

01 Objective

Competitive programming has a notoriously steep-feeling entry curve — not because the first problems are hard, but because beginners usually skip straight to hard problems without a working environment, a template, or any sense of what "Accepted" versus "Wrong Answer" even means. By the end of this blueprint, you'll have a working C++ setup, submitted and passed your first problems on a real online judge, and understand the roadmap of algorithm topics ahead of you — so your next 100 hours of practice go somewhere, instead of in circles.

02 Prerequisites

- Basic programming literacy in any language — variables, loops, conditionals, functions
- A computer that can install a compiler (Windows, macOS, or Linux all work)
- Comfort typing code without heavy autocomplete — contests reward writing code fast and correctly

03 Recommended Toolkit

TOOL	PURPOSE	NOTES
g++ (MinGW on Windows, or built-in on macOS/Linux)	C++ compiler	Free — C++ is the dominant contest language due to speed
VS Code	Code editor	Free, with the C/C++ extension installed
Codeforces account	Main practice & contest judge	Free — the largest active competitive programming community
AtCoder account	Beginner-friendly contests	Free — known for well-written problems and clear difficulty grading
Competitive Companion (browser extension)	Parses problems into your editor automatically	Free, optional but saves setup time
A personal GitHub repo for solved problems	Tracks progress, becomes a portfolio artifact	Free

04 Tools Required

- Computer with g++ and VS Code installed
- A free account on Codeforces (codeforces.com)

> **CONTEST INTEGRITY**
> Never use a second account to farm easier problems for rating, and never submit a solution you copied without understanding it — both are treated as cheating on every major judge and can get accounts permanently banned. The entire value of this practice is what happens in your head, not the rating number.

05 Your Learning Path

Competitive programming isn't one skill — it's language fluency, algorithmic knowledge, and data structure knowledge developing together. Progress on all three, not just the one that feels most fun.

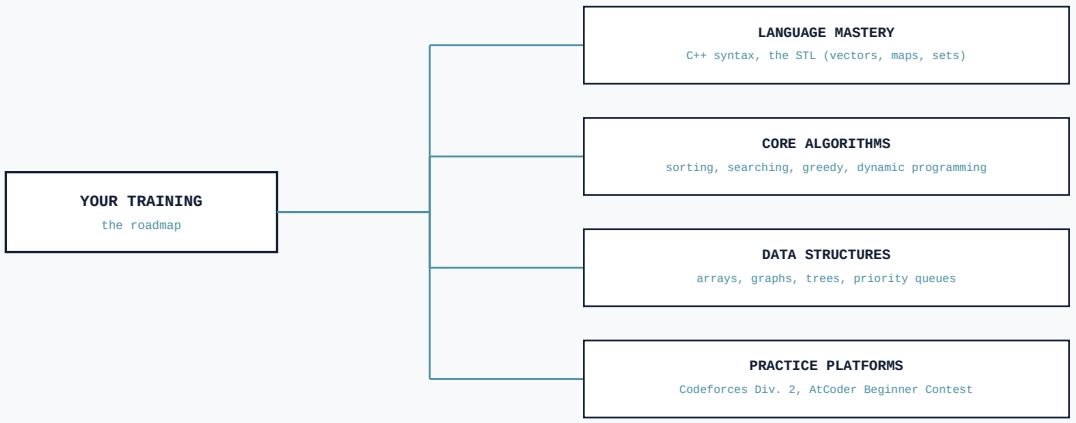

FIG. 1 — FOUR AREAS THAT NEED TO DEVELOP TOGETHER, NOT SEQUENTIALLY

AREA	STARTING POINT	GOAL
Language	C++ basics + STL vector, map, set	Write correct code fast, without looking up syntax
Algorithms	Sorting, binary search, brute force	Recognize which technique a problem is asking for
Data Structures	Arrays and simple graphs	Choose the right structure before coding, not mid-way through
Practice	Codeforces Div. 2 A/B problems	Solve consistently, then gradually raise difficulty

06 Step-by-Step Build

1. **Install g++ and confirm it works.** Compile and run a "Hello World" from the terminal before touching any contest site.
2. **Write a reusable template file.** Fast input/output setup, common includes, and a `main()` skeleton you copy for every problem — see Section 7.
3. **Create your judge accounts.** Codeforces and AtCoder are both free and widely used.
4. **Solve your first 10 problems on the easiest rating tier.** On Codeforces, filter by rating ≤ 800. The goal here is building the submit-debug-resubmit habit, not difficulty.
5. **Learn basic complexity analysis.** Understand why an $O(n^2)$ solution times out on $n = 10^6$ — this single concept prevents most early "Time Limit Exceeded" verdicts.
6. **Set a weekly problem quota.** 3–5 problems a week, consistently, beats 20 problems in one burst followed by a month off.
7. **Enter your first contest — virtually, not live.** Codeforces lets you do past contests "virtually" under real time pressure without the anxiety of a live leaderboard.
8. **Upsolve after every contest.** Go back and solve the problems you didn't finish, using the editorial only after a real attempt — this is where most of the actual learning happens.

07 Code Walkthrough

This is a standard competitive programming template — fast I/O, common headers, and a clean structure you'll reuse for hundreds of problems. Below it, a first solved problem showing the pattern in practice.

```
#include <bits/stdc++.h>
using namespace std;

// Fast I/O — saves time on problems with large input
static void fastio() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
}

int main() {
    fastio();
```

```
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];

    // Example: find two indices whose values sum to a target
    int target;
    cin >> target;
    unordered_map<int, int> seen;
    for (int i = 0; i < n; i++) {
        int need = target - a[i];
        if (seen.count(need)) {
            cout << seen[need] << " " << i << "\n";
            return 0;
        }
        seen[a[i]] = i;
    }
    cout << -1 << "\n";
    return 0;
}
```

This runs in O(n) using a hash map instead of the O(n²) nested-loop approach — the difference between passing and timing out on large inputs.

08 Testing & Troubleshooting

VERDICT / SYMPTOM	LIKELY CAUSE	FIX
Time Limit Exceeded	Algorithm complexity too high for the input size	Check the constraints given in the problem; an O(n²) loop fails once n exceeds ~10^4–10^5
Wrong Answer on test 2 or 3	An edge case wasn't handled (n=0, negative numbers, duplicates)	Manually test small and extreme inputs before submitting
Compile Error only on the judge, not locally	Judge uses a stricter or older C++ standard	Match your local compiler flags to the judge's (commonly C++17)
Rating drops and it feels discouraging	Comparing rating growth to others instead of your own baseline	Track problems solved and concepts learned, not just rating — rating is noisy short-term
Stuck for over an hour on a beginner-tier problem	Missing a specific pattern, not a general skill gap	Read the editorial after a genuine attempt — this is normal, not a sign to quit

09 Where This Goes Next

FROM CONTEST PROBLEMS TO A DOCUMENTED SKILL SET

A Codeforces profile alone isn't a portfolio — a GitHub repo with your solved problems organized by topic, with brief notes on the technique each one taught you, is. Pair this blueprint with **SA-001 (Building Your First Real AI Project)** to turn your algorithmic skill into a deployable tool, or continue with **PO-002 (Data Structures for Contest Programming)** once basic problems feel routine. See The University & Scholarship Playbook module for how competitive programming results are read by admissions committees.