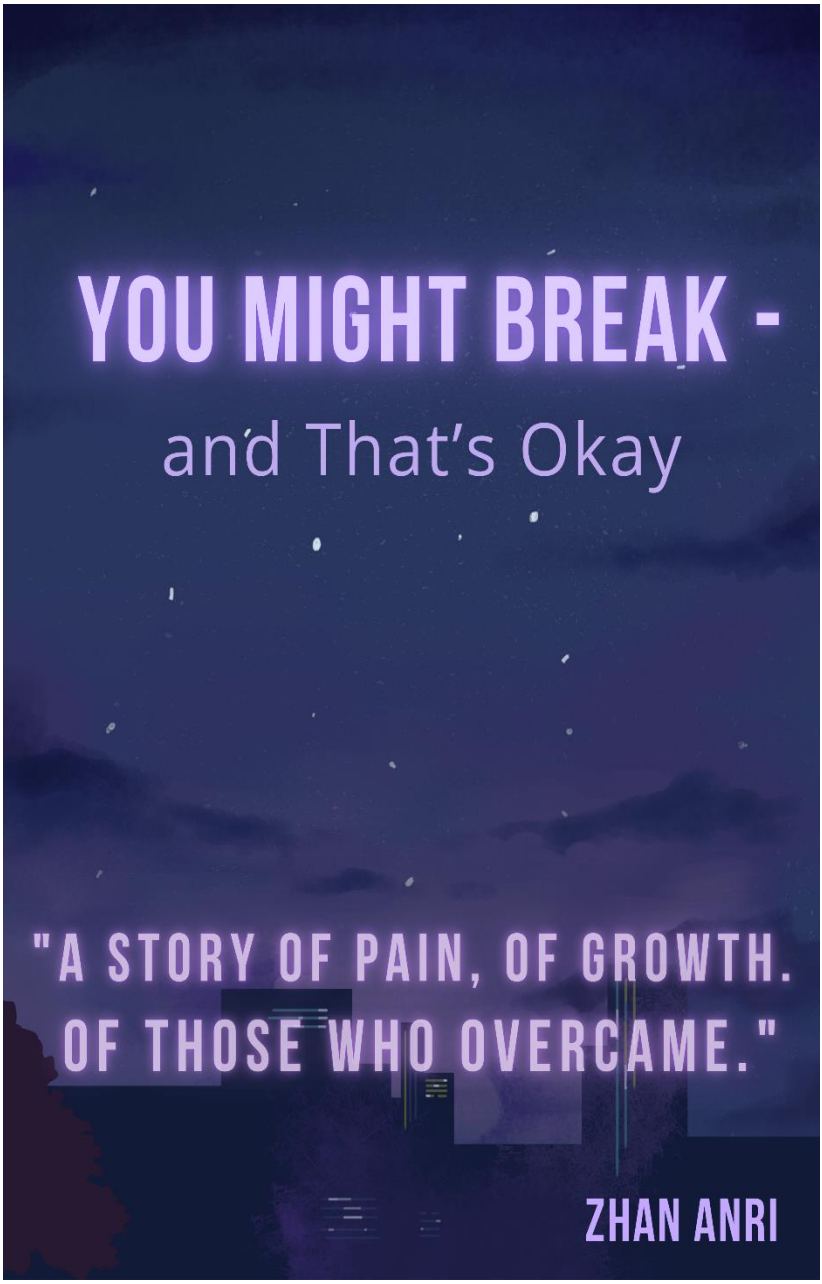
YOU MIGHT BREAK -
and That's Okay
"A STORY OF PAIN, OF GROWTH.
OF THOSE WHO OVERCAME."
ZHAN ANRI

Foreword

Competitive programming isn't about sitting with a laptop and clicking a mouse. It's survival. It's that moment when there are 15 minutes left in the contest, you've got one shot, and your hand is shaking from adrenaline. It's when you slam your head against a problem for two hours, then finally find a solution — and suddenly the world makes sense again. It's a place where no one cares how old you are, who your parents are, or what school you go to. One thing matters: you solve — or you don't.

My name is Zhan Anri. I've been through all of that.

I'm one of only three people in Kazakhstan's history to win gold at the European Junior Olympiad. My name is at the top of Codeforces contributors worldwide — and number one in Kazakhstan. But trust me, this isn't bragging. It's a warning. Because I know what it's like to burn out, to fail, to cry in the kitchen at 2 a.m. after a contest — and still open the laptop to keep solving.

This book is not a magic pill. Not a checklist for "how to become a master in a month." It's a set of honest, sometimes

raw, but working strategies. It’s the kind of advice I wish I’d heard when I was just starting out. You’ll find motivation here, and pain, insights — and a lot of specifics.

If you’re here, it means you either want to grow stronger, or you’re already on that path. Either way — I’m with you. I’m not a guru, not a god of algorithms, not a professor. I’m just a person who once sat down to solve his first problem — and couldn’t.

But didn’t give up.

And if you don’t give up — you’ll make it.

Let’s go. ⚡

Table of Contents

Chapter 1

Introduction

What even is this?

Competitive programming isn't just about "writing code." It's an intellectual sport where you need to come up with a solution, optimize it, implement it — and all that while a timer's ticking and your nerves are about to snap.

There's no time to stare at one line of code for ten minutes. You have to think fast, clear, and flexibly.

The problems? They can be anything:

- Classic algorithms and data structures
- Logic puzzles, game theory, probabilities, complex constructions
- Sometimes — nearly pure olympiad-level math under constraints

But that's not the main point. The main thing — is what it does to your brain.

I saw it myself at the 2025 National Olympiad. People who train in CP solve logical problems faster, handle unexpected situations better, and get out of dead ends quicker. And it's not "talent." It's training.

Why? Because CP builds a lot of things:

- Logic → Problems stop being scary. You read them like a scenario, not a riddle
- Thinking → You learn to see the essence even in messy conditions
- Focus → Your attention becomes razor-sharp
- Memory → When you need to hold 10 variables and constraints in your head — your brain adapts

And it's also a school of stress resistance.

The round starts, and you can't solve the first task. The timer ticks. Others are submitting. You keep getting Wrong answer. And still — you don't scream, you don't panic — you keep looking for a solution. That's how you build composure.

The same kind that helps you stay cool in real life when things go sideways.

Competitive programming isn't just about algorithms. It's about how to think, how to solve — and how to take a hit.

Python vs C++

You can write in anything: Python, Java, Kotlin, even Pascal still survives somewhere.

But if you want results — not just "nice syntax" — forget all that. The standard is clear: old-school, strict, and reliable — C++.

And here's why:

Speed is everything

At olympiads, it comes down to milliseconds. Doesn't matter how elegant your solution is — if it runs too slow, you get Time Limit Exceeded, and it's over.

Python? Sure, it's nice to write in. Looks like English. But it's often 5x slower. Sometimes more.

Imagine: you come up with a great solution, implement it, test it — perfect. But it just doesn't pass in time. That's a real failure.

With C++, that almost never happens. If your algorithm is right — it gets accepted. That's a huge deal.

Power tools

People fear C++ because "it's hard." But in reality — it's just a different style.

And it gives you pro-grade tools: set, map, priority_queue, lower_bound, sort, pair, vector — all of it comes with STL, out of the box.

You save time on implementation. You can write complex solutions 2–3x faster than building them from scratch in other languages.

C++ isn't about comfort. It's about control, performance, and speed — exactly what you need at olympiads.

Bottom line:
If you're serious about winning in CP — C++ should be your main weapon. Not a toy. Not a backup. A reliable, sharp tool you feel confident with.

Yes, it'll hurt at first. Yes, you'll meet Compilation error again and again.

But one day, you'll not only think fast — you'll code like a machine.

And then you won't just be a participant. You'll be a force to be reckoned with.

Learning the Language

Before solving tasks on Codeforces, do the obvious thing most skip — learn the language you're going to use. Without that, you're a soldier without a weapon.

Where to begin?

Stop just reading syntax. Start coding from day one.

Wanna learn C++? Here's what really works:

- Stepik — short courses that make you code immediately
- YouTube — find formats that force you to think, not just watch
- Coursera — if you want a solid foundation without fluff

But the key is this: Don’t watch passively. Sit down, write code with the author.

Solve micro-tasks: sorting, loops, arrays, functions. Don’t jump into olympiad jungles until you feel solid with the basics.

Got the basics? Great. Time to battle.

Codeforces is your main training ground. It’s real: timers, rankings, pressure, competition. Start with 800-level problems. They’re simple, but require precision and clean code.

Solved one? Move on. Messed it up? Review it, understand it, finish it — then go forward.

Mistakes aren’t failures. They’re investments in your skill.

Get used to the routine from day one: Learning the language → solving → failing → reviewing → fixing → improving

Not just doing problems, but growing through the grind.

You won’t become a strong programmer just by watching 10 videos. You’ll become one by falling, getting up, and coding again.

Codeforces

Codeforces isn’t just a site. It’s a universe.

Thousands of problems. Rated contests. Blogs, analysis, memes, discussions. You don’t just solve — you live in a community where everyone’s chasing progress.

Contests happen multiple times a week, usually in the evening. It’s perfect: you get home, turn on your laptop, sit down — and in a minute, you're in the fight. Timer ticking. Brain buzzing. Fingers flying. Then — rating up or down. Emotions. Growth.

In fall 2024, I had the honor of writing a contest for Codeforces. And it wasn’t just cool experience — it blew my mind.

My team submitted a batch of problems — nearly all were rejected. The coordinator said: Not good enough. Too boring. Too standard.

Only after dozens of revisions did we assemble six polished problems that passed.

Each problem? Three days of work. Minimum. For one. Now imagine how much work goes into every contest you casually solve in two hours.

That's why Codeforces is elite. Problems don't appear by accident. Each one is filtered through dozens of minds.

Why should you care?

- Compete → your rating grows
- Read blogs → absorb insights
- Upsolve → lock in skills
- Write your own → learn to think like an author

And you see the progress: handle color, position on the scoreboard. It motivates. It pulls you in.

But the biggest thing — you're not alone.

Thousands like you gather here: smart, hungry, ambitious. You grow inside a living, sharp, smart community.

Want to grow? Go where it's hot.
Codeforces burns every evening.

Alright, you've learned C++, made an account, opened Codeforces — now what? How do you not get lost among thousands of problems? How do you start strong without burning out?

Easy. Here's the plan.

Start with the Problemset

Go to the Problemset section — your home base. Thousands of tasks, rated from 800 to 3500. The rating is a rough measure of difficulty:

- 800 — beginner
- 1000–1200 — basic olympiad logic
- 1500+ — things get serious
- 2000+ — only if you're in shape

Where to begin? Start with 800. Don't flex. These are made for beginners and often test attention, accuracy, and logic — not fancy algorithms.

To use your time wisely:

- Sort by number of solutions — do the most popular ones

- Stuck? Read the editorial, search hints — but don’t copy code
- Finished? Move on — don’t obsess over one forever

Raise the bar step by step: Solved 10 at 800? Try 900. Feel confident? Push into 1000+.

The key — don’t stay in place. Keep challenging yourself.

You’ll fail. A lot. You’ll rage at Wrong answer, Time limit exceeded, compilation errors.

That’s when the growth kicks in.

Read the editorial, rewrite the code, understand → solidify → apply on the next task.

Consistency + effort + mistakes + review = your skill.

Codeforces is a gym for your brain. The more you go — the stronger you get.

Codeforces Rating

If you're serious about CP, one of the first things you'll notice on Codeforces is your rating. It's not just a number. It's a reflection of your form, mistakes, wins, growth, and losses.

How does it work? Contests happen regularly — 5 to 9 tasks in a timed round (usually 2 hours).

Problems go in increasing difficulty. For example, in Div 2:

- A — easiest (800)
- B — 1000–1200
- …
- F or G — deep dark forest of 2400+

Your contest performance affects rating:

- Solve a lot — rating goes up
- Mess up early — it drops
- Average? It stays put

At the time of writing this book, my rating is 2214.

Top participants in the world? 3700–3800+. No joke. That's outer space.

But rating isn’t everything. You’ll lose rating. Sometimes from dumb mistakes. Sometimes bad luck. Sometimes the problems just don’t click.

And that’s fine.

Rating is not the goal. It’s a byproduct of constant growth.

The key: compete regularly, analyze mistakes, improve. Do that — and rating will come. Chase only numbers — and you’ll burn out.

Why care at all?

- Motivation: seeing progress feels good
- Competition: you know where you stand
- Selection: camps and contests look at rating
- Self-worth: you know your level by facts, not vibes

Rating doesn’t make you smart. But it shows how many times you’ve fought — and didn’t quit.

If It’s Not Yours — Let It Go

Let’s be real: CP isn’t for everyone. Not because you’re not smart enough or weren’t born a genius. No. It’s just a certain

mindset, pace, lifestyle — even vibe. And if you don't feel it — don't force yourself.

I've seen dozens of people torture themselves. Solving without joy. Grinding for someone else's expectations. For ratings. For guilt. For the portfolio.

All they got was burnout, lost motivation, and hate for code. Some were exceptions.

Don't follow their path.

If every time you open a problem you feel dread, if every contest leaves you empty, if that voice inside keeps saying I don't like this — stop. Ask yourself:

Do I really want to do this? Or am I just afraid to admit I want something else?

There's no weakness in that.

Weakness is staying in what's destroying you. Strength is saying enough, closing Codeforces — and going to find what's truly yours. Maybe it's design. Maybe backend. Maybe physics or music. Maybe something you haven't even discovered yet.

You don’t have to be a CP star. You do have to be yourself.

And if one day you leave CP and feel happier — know that you did the right thing.

Conclusion

Now you know how to begin: What language to choose. Where to train. How rating works. And most importantly — why you’re doing this.

CP isn’t for geniuses or nerds.

It’s for people who move forward even when it’s hard, boring, or scary. For those who fall, rise, and fall again.

Don’t be afraid to be a beginner. Be afraid to stay one too long.

Now it’s your move.

Chapter 2

Rewiring the Brain

If you want to progress in competitive programming as fast as possible, you'll need to change not only your learning methods, but your way of thinking. Just solving problems and hoping for "natural growth" is like going to the gym and lifting dumbbells randomly with no plan. Sure, it might feel easier over time. But did you really get stronger? Maybe not at all.

Beginners often think that if they just grind problems on the usual sites nonstop, they'll skyrocket in a few months. But that's an illusion. If it worked like that, we'd be seeing new grandmasters on Codeforces every day. Instead, we see thousands of "green" users with a thousand problems solved — and zero progress.

So what actually works? To figure that out, let's start with the key: how your brain works — and why sometimes it works against you.

Short Videos Are Poison

In competitive programming, problem-solving often happens through logical chains in the brain. Sometimes you build them consciously, step by step. Sometimes your brain pulls the solution from your subconscious, based on past experience. Either way, the faster and more clearly you can build those chains — the better your results.

So the question is: how do you make your brain build those chains faster? The answer might sound weird, but the first step is: **cut short videos from your life.**

Yes, it sounds unexpected. But if you're constantly scrolling TikTok, Instagram Reels, or YouTube Shorts, your brain is bombarded with so much fragmented content that after a few minutes, it forgets everything from earlier clips. Why? Because your brain adapts to only remembering the last 1–2 minutes. The rest — trashed.

That mindset is toxic for olympiads. If you can't hold complex chains in your head, if you can't stay focused on one task, or return to earlier thoughts — your progress will be capped.

You've probably felt it: your teacher explains something for 10 minutes — but you can't remember the beginning. Or you read a problem statement, get to the end — and realize you forgot the first paragraph.

That's what short videos do to your brain.

But the problem runs deeper.

Short videos don't just kill focus. They wreck your dopamine system.

You become hooked on the endless feed because every new clip gives you a tiny hit of pleasure. Instant. Effortless.

In real life, that turns into this: doing hard things becomes boring. Studying — boring. Reading a whole problem — boring. Even thinking — boring. Your brain wants that easy hit, and everything else feels dull.

You turn into a zombie, uninterested in anything that doesn't entertain.

If you want to grow fast, you need to break that cycle. Don't worry — it's possible. And you can do it step by step.

Step 1: Realize that short videos are ruining your life

You have to see it for what it is. It's not a harmless habit. It's something that's truly ruining your quality of life. Killing focus. Smearing your attention. Draining your motivation. Making even exciting things feel like a grey swamp.

I've seen hundreds of people live in TikTok. Scrolling every spare minute — on the bus, in line, during class, even while studying. Sound familiar?

If you recognize yourself — good. Because it means you've already made the first step. If you're reading this book, it means something inside you already said "stop." That matters.

Now think: how much could you do in a month if you spent all that scrolling time on something useful? You could start a new hobby. Level up your English. Solve tons of problems. Or just get real sleep.

Most importantly — you could rewire your brain and actually start to grow.

Step 2: Reduce the dose

Quitting TikTok or Reels cold turkey is nearly impossible. Like with any addiction, going hard all at once leads to relapse.

So reduce the dose gradually.

- Set limits using Screen Time (iPhone) or Digital Wellbeing (Android)
- Check how much time you currently spend on short videos
- Cut 15–20 minutes — and hold it for a few days
- Then cut another 20 minutes. Step by step

Don’t try to quit in one day. Don’t fight yourself. Be strategic.

In the meantime, replace short videos with content that has a storyline: 10+ minute YouTube videos, movies, anime, podcasts. Let your brain rebuild the habit of following structure.

Step 3: Enjoy real life again

After a while, you’ll feel things become brighter. Silence will return to your head. Your brain will start thinking again. You’ll enjoy simple things: a walk, a good problem, a real conversation, peace.

You'll stop twitching every 30 seconds for the next hit. You'll come back to yourself.

And with that — ambition returns. Interest returns. You stop procrastinating. You move forward.

You don't have to be the zombie with TikTok in hand. You can be the one who controls their own attention. And that's a superpower.

Deep Focus

Now that we've handled short videos — there's a bigger question: how do you *restore* deep focus?

Because even if you turn off your phone, remove distractions, and sit with a problem — your brain won't instantly switch on. If you've spent years jumping every 30 seconds, silence alone won't fix it.

You have to retrain your brain to enter "flow" — that state where you get so deep into a problem that time disappears. You forget hunger, noise, even the fact that you're sitting.

Here are three real techniques that work.

Method 1: Remove everything that breaks focus

Focus doesn't come from nowhere. It needs conditions.

You can't concentrate if you're constantly interrupted. Every time you glance at your phone, your brain loses 10–15 seconds of focus. Do that 10 times an hour — that's over 2 minutes gone. Add the time to recover your flow — and you've basically never entered it.

What to do:

- Turn on Do Not Disturb mode
- Flip your phone face down. Better yet — out of sight
- If the room is noisy — use headphones. Pick whatever works: instrumental music, white noise, nature sounds
- Never check your phone while solving a problem. Before — sure. After — go ahead. But not during

Your brain needs 5–10 minutes to enter flow. If you check your phone every 5 minutes — congrats, you never get there.

I've seen dozens of people complain they "can't concentrate." They all had one thing in common: phone within reach. Remove it — and within a week, your brain starts obeying.

Method 2: Solve problems in your head, no scratch paper

Sounds odd? Actually one of the most powerful brain workouts.

When you solve in your head, you're training working memory. The ability to hold logic chains and problem constraints without losing the thread.

What to do:

- Choose simple but conceptual problems
- Work through them mentally: solution steps, algorithm complexity, memory usage
- Compare approaches — all in your mind

Note: this is training, not how to write a contest. In a real round — of course, use paper. But outside? It's like doing sit-ups — hard, but effective.

Why it works: working memory is what separates top coders. The more you can hold mentally, the faster you think. The easier it is to plan "two steps ahead."

Method 3: Play smart games (but wisely)

Yes, games hurt your dopamine system. But let's be real — we're not robots. We can't be perfect all the time. Sometimes games are a way to unwind. And that's okay. Personally, I think they're way less harmful than TikTok or Reels.

But if you're going to play — play smart.

What to do:

- In Clash Royale — count elixir, read opponents, memorize decks
- In chess, Catan, any strategy game — forecast moves, evaluate outcomes, plan long-term
- Try puzzles, logic games — anything that trains attention and thinking

Why it works: you're learning to keep focus on one challenge — in a fun way. That's the same skill you need in olympiads: to think long, deep, and sharply.

Deep focus isn't talent. It's habit. And you can rebuild it. Just start.

Work Out

Now here's something that sounds crazy at first: "Sports? For a programmer? Seriously?"

Yes. Very seriously. Exercise isn't just a bonus. It's one of the **main keys** to productivity and thinking.

Want to think faster, focus deeper, and burn out less? Then welcome to the game called "move your body."

Why sport = faster brain: Your brain isn't isolated. It's part of your body. If your life is "code → eat → sleep," barely moving for hours — your brain starts lagging.

- Fatigue hits faster
- Focus collapses
- You blank out on simple problems

But add physical activity — and everything changes:

- Better blood flow → more oxygen to the brain
- Hormonal balance → less laziness, more energy
- Stress resilience → you stay sharp longer

Fact: science has proven that exercise speeds up neural activity, boosts memory, and strengthens concentration. Exactly what we need.

Sport is the best cure for burnout

Burnout isn't "laziness." It's when your brain refuses to keep going. You sit there, but the problems just don't get solved. Not because you don't know how — but because your mind is fried.

Why? Because you're overloading your brain and leaving your body in hibernation. The balance breaks. Your nervous system overheats — and you shut down.

What sport does:

- Releases mental tension through physical effort
- Lowers cortisol — the stress hormone
- Restores emotional balance
- Gives you the power to sit through 4-hour contests without breaking

My experience: how sport changed everything

When I started training seriously for olympiads, I could only focus for 5–6 hours a day. Then I'd crash. I'd lose focus, get

frustrated. So I looked for a fix: “How do I boost my brainpower?”

The answer was simple but uncomfortable — healthy habits. Sleep properly. Eat decently. And most of all — move.

I tried. And the results shocked me.

Mountains, running, walking — they didn’t just give me strength. They gave me stability.

Now I can do olympiad programming all day — and not feel exhausted.

And I got pulled into it:

- Conquer every peak around Almaty
- Run in marathons

Not for checkboxes. For energy.

What sport to choose?

You don’t need to hit the gym or run 15km. Just find what you enjoy and do it regularly. Here’s what works:

- Running — clears the mind, reduces stress, energizes

- Swimming — relaxes and resets the brain
- Gym — builds stability, endurance, tone
- Table tennis, chess — focus, reaction speed, strategy
- Hiking, mountains — exercise + fresh air + mental clarity

30–40 minutes of activity, 3–4 times a week — and you'll feel the difference.

Want to think better, focus stronger, and feel alive — start with the body. Because without it, even the strongest brain burns out.

Solving Problems the Right Way

Now we get to the most… boring-looking part of the chapter — how to solve problems the *right* way. Yeah, that section every coach and strong contestant mentions. Yet somehow, progress still stalls for most people.

So how do you actually solve problems to level up your skill — not just your problem count?

I'll use Codeforces as the example, because 99% of my journey happened there.

Here’s where the chaos starts: choosing a strategy.

You hear all kinds of advice:

- “Only solve problems at your rating!”
- “Never read editorials, no matter what!”
- “Do 10 problems a day — or you’re not improving!”
- “Grind upsolving, only through pain!”

Sometimes tips contradict each other. Sometimes they’re too generic. Sometimes they just don’t fit you.

This book isn’t a forum. I’m not here to argue.

My goal is to give you techniques that work. Tested by me — and many others.

1. Learn to reduce problems to known patterns

When you solve lots of problems, you start seeing patterns. Sometimes you read a statement and think:

“Aha! This is classic greedy, just disguised as a constructive.”

That’s where the magic is.

The goal is to recognize familiar ideas in new wrappers. That speeds you up a lot — you're not starting from scratch, you're unpacking what you already know.

But keep in mind: at the higher levels (Master and up), patterns start to vanish. That's where the real grind begins: weird combinations, heuristics, raw logic without templates. But at beginner and intermediate levels — patterns are gold.

What to do:

- After each solved problem, ask: "What was the pattern? How was it different from the classic version?"
- Tag it mentally: "This was greedy with scanline" and so on
- Build your own pattern collection — even just in your head or notebook

2. Solve problems of varying difficulty

Classic beginner mistake: "I'll do 1400–1500 problems, then 1600–1700, and so on."

Sounds logical. But it's a trap. You start guessing solutions because you know the expected level. Your brain stops really thinking.

What to do:

- Choose problems at different levels — without seeing the rating
- Solve them by instinct — no rating bias
- Use the random feature on Codeforces or just pick by name

Why it works: you're training not just problem-solving, but problem *recognition*, adaptation, flexibility. That's core skill territory.

3. Read editorials — but always implement

One of the most common mistakes: reading the solution — and not writing the code.

Here's how it looks:

- Read the editorial.
- "Ah, makes sense."
- Move on.

Three days later:

- "Wait… how did that work again?"

Understanding ≠ skill. You can grasp an idea, but if you don't code it — you'll forget it.

What to do:

- Read the editorial? Immediately code it out.
- Run your version, debug it, print stuff — play with the problem till the end
- Don't read editorials right away. Give yourself time to struggle. That's how you grow your heuristic thinking — the ability to find weird paths.

Leveling up isn't "more problems." It's "more thoughtful problems."

Conclusion

If you want to win olympiads — it's not enough to "be smart."

You need to weaponize your mind.

- Remove what steals your attention
- Train your brain to focus
- Get your body working for your mind
- Solve not for volume, but for growth

You don’t grow by accident. You grow when you build the right environment for it.

Most importantly — now you know how.

Chapter 3

Naive Mistakes, Harsh Truths

Beginners in competitive programming often make the same mistakes. And we're not talking about syntax errors — but much deeper things: strategy, habits, mindset. These mistakes not only slow down progress, but can kill motivation, lead to burnout, or even make someone quit programming for a long time.

The paradox is — almost everyone goes through this. But some learn and grow. Others keep hitting the same wall again and again. Let's break down the traps that catch beginners, and how to avoid them.

Locking Yourself In and Programming 16/7

You start solving problems. You get hooked. Then the thought hits you: "If I code 10 hours a day, I'll become a master twice as fast!"

No. You won't. You'll become a burned-out zombie who doesn't even remember why you started this in the first place.

I've seen people with fire in their eyes declare "6 problems a day, every day this month!" Two weeks later — fatigue, frustration, apathy. A month in — they quit. Sound familiar?

It's not weakness. It's just reality. People aren't robots. You're not a compiler that runs non-stop. Your brain needs variation. Your soul needs rest. If you erase everything else for one goal, that goal will eventually feel like your enemy.

Diversity is your ally. Sports, music, books, walks, real conversation — these don't take away from your training. They make it stronger. Because rest isn't procrastination. It's part of growth.

When you regularly give your brain space to breathe, it works better. It returns to problems refreshed. It finds creative angles. Sees ideas that once flew right past.

And if you have multiple passions — you're no longer dependent on one source of validation. When programming feels off — you don't fall apart. You pivot. And return when ready.

With variety, you also gain:

- Less doom-scrolling on YouTube, TikTok, or games — because your day is full of real interests
- More creativity — crucial for solving unique problems
- Protection from burnout — which means steady, conscious growth

Don't try to carry 100kg of mental load all at once. Competitive programming is a marathon, not a sprint. Want to make it to the finish? Learn to protect yourself along the way.

Trying to Cheat in Codeforces Contests

Every second beginner has cheated in a Codeforces round at least once. Let's not pretend. I've done it too. I'm not proud — but I'm honest. Because the "why" matters more than the "what."

Most people don't cheat to learn. Not even for the problems. They do it for rating. For that number next to their handle. For the illusion of being "smart" — as confirmed by a blue or purple profile.

And yeah, at first, it feels good. You refresh the page. See +150. Feels like you're a genius. Then — boom. A terrible round. Your color disappears. Along with your confidence, motivation, and mood.

The problem isn't rating. It's tying your self-worth to numbers.

But it's not hopeless. You can get out. I built a roadmap for this. Yes, it's tough. Yes, you'll relapse. But it's real.

Step 1: Reframe why you're here

Close the rating tab and ask yourself: *Why am I doing this at all?*

You came for the beauty of problems. For that feeling when you find a solution yourself. For the mental growth. Not for a fake color boost that means nothing if you can't solve without help.

Cheating gives you only illusion. The illusion of strength — but deep down, you know the truth. And the truth is: your skill is only what you can do alone.

Step 2: Detach from rating

Rating is just a number. It can be helpful — as feedback. But it should never define your worth.

Because:

- You can be a “master” on Codeforces and still fail your national olympiad
- You can be “green” and still think better than half the “blues”
- You can be “gray” and become “yellow” fast — but not instantly

Learn to see rating as a side effect, not a goal. When you stop chasing it — ironically, that’s when it starts to grow. Honestly this time.

Step 3: Play fair

Try playing one round for real: timer, IDE, phone off, full silence. No peeking. No Googling. Just you and the problems.

It’ll hurt. But it’ll be real. And even if you solve just one out of five — it’s yours. A month later — maybe two. A year later — all of them.

You'll learn to handle pressure. You'll understand what real olympiads feel like. And you'll know: *"I'm not pretending to be strong — I'm becoming strong."*

Cheating is easy. Growing is hard. But if you want something that lasts — this is the way. And it starts with honesty. With yourself — and your keyboard.

Only Studying Topics

Some beginners fall in love — not with problems, not with logic — but with *topics*. It's like collecting stamps: "I've done graphs," "Next month I'll do DP," "After that — heuristics."

Seems logical. Systematic. Neat. But here's the issue...

A month later, they open a problem that doesn't say "use segment tree" or "apply DP." It's just a raw, creative task. A mix of ideas. No label.

And then — paralysis. Panic. "I haven't studied this." So they can't solve it?

You might know how every tool works. But forget why you entered the workshop.

How to avoid this? Stop living by “one topic per week.” Start from real problems. Solve rounds. Solve olympiads. If a new idea shows up — *then* learn it. Not before.

Can’t solve it? Read the editorial. Find out what technique was used. Lock it in with 2–3 similar problems. No need for a “DP month.” You need to learn how to think — not how to tag problems by keywords.

In olympiads, nobody says “this is a DP problem.” You figure that out yourself.

Real problems are messy. A bit of logic. A data structure. Some brute force. Maybe a trick.

Winners aren’t the ones who know all topics. They’re the ones who *guess* how to apply what they know.

Topics are tools. Important — but secondary. Primary is thinking. Analysis. Instinct. And you only build that in battle.

Learning topics is easy. Learning to solve is hard. But that’s what separates participants from champions.

Dropping Programming for a Week Every Month

I don’t know how many times I’ve said this already. Maybe three. Maybe ten. But I’ll say it again. Because this is one of the sneakiest, deadliest progress-killers.

You’re climbing. Solved 5 problems one day. Next day — 3. Then 1. Then you’re tired. You think: “I’ll take a couple days off.” Makes sense, right?

A week passes. You haven’t opened your IDE. Haven’t touched Codeforces. Haven’t seen a single problem.

You return — and feel like a stranger. Everything familiar is now foreign. Even easy problems feel hard.

Why? Because you lost rhythm. And rhythm isn’t just habit — it’s your foundation.

It’s like riding a bike uphill. You need momentum. Stop — and you roll back. Sure, you can regain it. But it gets harder every time.

So what’s the fix?

Solve one problem a day. Even the easiest. Even with hints. Even when you're wiped after exams. ABC from AtCoder, a past C problem, a basic task from old topics — anything.

The key is to stay connected. With your mind. With your fingers. With your logic.

Because getting back from "0 problems/week" is always 10x harder than continuing with "1 problem/day."

Remember:

- Strength isn't solving 30 problems and burning out
- Strength is showing up every day — even when nothing's working

Steady rhythm beats chaotic spikes. Always.

Real progress isn't a sprint. It's a habit you protect — even on bad days.

"I LoOoOve DP"

I've met tons of beginners who obsess over one topic. Usually DP or data structures. They fall in love with them — solve 20

problems in a row, memorize all the states in dp[i][j][k], know how Li Chao Tree differs from a segment tree.

But give them a constructive, or a game theory problem, or something purely logical — and boom. Stuck. Panic. “I didn’t learn this.”

The problem? Mastery of one topic ≠ true skill. It’s lopsided growth.

Sure, you might be a DP god. But if you can’t think outside templates — you’ll crumble on the first creative problem at regionals.

How to avoid it? Solve real rounds. Unfiltered. Let randomness hit you. That’s where growth starts.

Broaden your mind. Learn to *understand* problems — not just recognize patterns.

Train flexibility. Three problems with different logic > fifteen of the same theme.

Topics are tools. But solving starts with an idea.

If you want to be a real olympiad fighter — you must think beyond your comfort zone.

DP is cool. Thinking is cooler.

Conclusion

Mistakes are normal. What matters is noticing them — and fixing them.

The strongest programmers aren't the ones who never messed up. They're the ones who never let mistakes stop them.

Chapter 4

How Olympiads Are Won

Want to crush it at an olympiad? Forget about being distracted. Without focus, you're not a participant — you're just a tourist. Focus is your "God mode," when all the noise disappears and only the task, the code, and the timer remain. That's when you catch bugs instantly and don't lose points over stupid mistakes.

So how do you build that skill? Here are tried-and-true methods that help even when your brain is boiling and your forehead has already stamped seven marks onto your keyboard.

1. Cut Out External Noise

Think you can chill with a TV show or anime the night before an olympiad? Think again. Those things leave a mental residue. Even if you feel fine, your brain will start flashing random scenes at you. And suddenly, instead of thinking

through a `for` loop, you're analyzing how noble that character was in episode 23.

Yeah, it's happened to me. I'd be thinking about some chunk of code — and boom, my head's off in the plot, the emotions, the drama. Turns out, it was because I had watched an intense anime the night before.

Even worse if you watch something right before bed. The emotional spikes, plot twists — they ruin deep sleep. And if you don't recover overnight, you won't be at your best. Especially risky are high-energy genres like thrillers, comedies, or heavy dramas.

Instead, read something light. Take a short walk. Put on some wordless music. Or just talk to someone in real life. The point is: don't clog your head the night before a day when every byte of attention matters.

An olympiad is a battle. You want to show up with a clean mind — not one full of leftover drama from yesterday.

2. Mental Warm-Up: Drawing

This tip came from a national-level chemistry coach — during

the 2025 national informatics olympiad. And you know what? I didn't use it then, but the idea is genius.

An hour before the contest, grab a piece of paper and start drawing. Doesn't matter what — patterns, lines, stick figures, random doodles. It's not about art — it's about the process.

Why does it work? Drawing calms you down, lowers anxiety, and gently activates the brain. Neuropsychologists use this for a reason — it "switches on" your brain without overloading it with logic or formulas. Like warming up before a race: you're not sore, but you're ready.

Plus, it grounds you. When you're nervous and don't know what to do with your hands, drawing pulls you out of anxious thoughts and brings you into the moment. Helpful? Hugely. Especially before it's go time.

Don't like drawing? No problem. Try folding origami, coloring something, spinning a Rubik's cube. Just don't overload your brain with hard logic an hour before the start. Let it slide gently into work mode.

3. Fresh Air

You know that clear-headed feeling after a quick walk? That's

not magic — it's biology. When you move, your brain breathes. Literally: more oxygen, better circulation, and your thinking sharpens.

I noticed this myself: I'd step outside, breathe a bit, come back — and suddenly, problems seemed 20% more solvable. Later I looked it up: walking before a contest is like restarting your system before opening heavy software. 10–15 minutes and you're no longer a baked potato — you're a processing unit.

Why it works:

- Less stress
- Better blood flow = more brain fuel
- Sharper thoughts
- And if it's morning — it charges you for the day

Bonus: if there's sun, it's a double boost. Vitamin D doesn't just lift your mood — it keeps your brain from zoning out. Low D = sluggishness, poor focus, the "read the problem and only understood the first word" effect.

No time for a walk? At least crack a window, stand up, take a few deep breaths. Let your brain know you're not just solving — you're alive.

A contest isn't just a mental marathon. It's a sprint powered by a sharp brain. And that brain needs air.

4. Nutrition and Vitamins

Many underestimate nutrition before an olympiad. Big mistake. I started noticing: eat something light and healthy — your brain runs like clockwork. A cup of latte and a solid lunch — and energy's back. But eat something heavy — and bam, brain fog, scattered thoughts, no focus.

Conclusion? What you eat directly affects how you think. On important days, go for a light but nutritious breakfast: proteins, healthy fats, slow carbs. Classic combos — oatmeal with nuts and fruit, eggs with veggies, yogurt. It's not just food — it's fuel.

Some nutrients that actually help:

- **Omega-3 (fish, flaxseed)** — boosts memory and clarity
- **Magnesium (nuts, spinach, cocoa)** — helps with stress
- **Vitamin B complex (meat, eggs, grains)** — supports brain function
- **Iron (red meat, legumes, buckwheat)** — more energy and oxygen

- **Vitamin D (fish, sun)** — sharpens focus and mood

During the olympiad, a small snack helps too — a handful of nuts, a banana, or a few squares of dark chocolate can bring back clarity and strength.

Food isn't just background. It's your hidden ally. Feeding your brain properly means respecting your potential.

5. Meditation

Olympiads are stressful — especially when the stakes are high and the problems look like they were written by aliens. You can know everything, be 100% prepared — and still freeze at the key moment. You know what helps? Meditation. Yeah — it's not just for monks on mountaintops.

If you feel your hands getting cold at the start, your breathing gets shallow, and your thoughts are bouncing around like pointers in a memory leak — you need to stop. Just for a minute. Pause. Notice your breathing. Regain control.

One of the simplest and most powerful techniques is called 4-7-8: inhale for 4 seconds, hold for 7, exhale slowly for 8. Repeat a few times. In just one minute, you can go from "panic" to "collected."

This isn’t magic — it’s nervous system control. You’re taking back the steering wheel. Instead of riding the stress wave, you set the rhythm yourself.

The contest isn’t won by the smartest person. It’s often won by the one who doesn’t break under pressure.

6. Sleep

Let’s say tomorrow is the big day — finals, qualifiers, or that one exam deciding your whole semester. You’ve studied, solved problems, struggled — and now you feel almost ready. But there’s one enemy that can ruin it all, even if you’ve memorized the entire Knuth-Morris-Pratt algorithm in your sleep. That enemy is poor sleep.

Sleep isn’t just rest. It’s literally part of your preparation. It’s when your brain organizes information, files things away, restores resources. If your sleep sucks — forget productivity. Your brain slows down, you start making dumb mistakes in places you’d never mess up, and instead of a clean solution, you end up with a mess that doesn’t even compile.

A few rules that’ll help you not sabotage your final push:

- **Sleep in total darkness:** Melatonin is your sleep hormone. It helps you fall asleep, deepens your sleep, and lets your brain reset. But it has one enemy — light. Especially screen light. When you scroll TikTok or chats before bed, your body thinks: “Oh, day’s not over yet.” Melatonin production drops.
- **Keep the right temperature:** Ideal is around 22°C (72°F). Too hot or too cold — and your body can’t fully relax.
- **No training 3 hours before bed:** Late workouts rev up adrenaline. And your brain thinks it's game time instead of sleep time.
- **Can’t sleep? Don’t stress:** Try melatonin, or just get up, walk, read — don’t doom-scroll or lie awake panicking.
- **Consistent schedule:** Sleep and wake at the same time daily, especially near the olympiad. Body clocks don’t like chaos.

7. Manage Cognitive Resources

You could be a genius, know all the algorithms, and solve problems blindfolded — but if your brain is overloaded, you’re done. Even the fastest processor lags when running nonstop.

Science says the brain can only absorb so much daily. If you flood it with tasks, theory, code, deadlines, and stress — it starts to glitch. Thoughts blur. Focus vanishes. Even basic actions feel like heavy lifting.

You've felt this before: 5–6 hours into a grind, and suddenly you can't even multiply two numbers. You're not dumb — you're just in energy-saver mode.

Instead of pushing through, take breaks:

- Don't cram the night before — it's too late. Review what you know, then rest.
- Don't write long code late at night — do it fresh in the morning.
- Don't force 20 problems in one evening — solve 5 deeply.

Mental resets help more than you think: a walk, fresh air, a few minutes of silence — they reboot your system. And suddenly, that stuck solution clicks.

Save your strength for where it matters. Olympiads are won not by who worked hardest — but by who arrives in peak shape.

8. Writing Code

Anyone who's been in a contest knows this: you write the code, hit submit — and... error. Then another. And you're not solving problems — you're battling bugs like invisible ninjas.

It starts with a simple compilation error. Then half an hour of debugging. Before you know it — half the contest is gone and your nerves are shot. And the problem wasn't even hard — just poorly implemented.

I tried every trick: more `cout`, more testing, more speed. Thought writing fast saved time. It didn't. Some days worked. Others — disaster.

Then I heard this from Altynbai Nazarbek, winner of Kazakhstan's 2024 Championship:

"Write code slowly. Very slowly. But be sure of every single character."

It sounded weird at first. Contests are about speed! Why slow down? But then I saw how he submitted huge problems in 20–30 minutes. While I was still debugging mine after an hour.

I tried his method. At first, it felt like slow-mo. But then I realized — time isn't lost in typing. It's lost in fixing errors. When you code with attention, you make fewer mistakes. Fewer mistakes = less debugging. And results come faster.

Of course, bugs still happen. Nobody's perfect. But this approach gives you control. You don't just mash code — you build it. Like an engineer, not a gamer speedrunning a level.

Over time, this "slowness" becomes confidence. You don't hesitate — because you don't make careless errors. And confident code = fast code = good results.

Olympiads aren't won by those who code fast. They're won by those who code right.

9. Why the Best Ideas Come in the Bathroom

Honestly? My best insights didn't come at the keyboard — they came while washing my face. Or walking. Or yes, sitting in the bathroom. It's not coincidence — it's science.

You're stuck on a problem. Staring at the screen. Nothing works. You step away — and boom, the answer pops up.

Why?

1. **Movement = Oxygen = Clarity:** Walking boosts blood flow, feeds the brain. Your thinking clears.
2. **New Setting = New Links:** Change of scene — even the bathroom — jolts your senses. The brain reconnects differently.
3. **Switching Tasks = Less Pressure:** Trying too hard blocks thinking. Letting go gives your subconscious space to work.

This is called incubation. You step away, but your brain keeps chewing. And when you return — it clicks.

How to use it:

- Feel stuck? Get up. Walk.
- Don't be shy to go to the bathroom. That's strategy.
- Wash your face, open a window — give your brain a reset.

Don't panic when a problem doesn't solve immediately. Sometimes the answer appears not in your IDE — but next to the sink.

Conclusion

You can know all the theory. You can ace greedy algorithms

and graphs. But if you show up tired, unfocused, with your head full of drama — you've lost before you've typed `#include`.

Olympiads aren't just about smarts. They're about state. How you prepare. How you rest. How you manage energy.

This chapter isn't about magic. It's about real strategies to help your brain show up in peak form, when it matters most.

Don't underestimate the little things. A short walk, a glass of water, a dark room, a deep breath — can matter more than a week of late-night cramming.

You're not just solving problems. You're stepping into battle. And you prepare for battle with your body, your mind, and your discipline

Chapter 5

Burnout

At first, it’s a rush. You dive into competitive programming and it feels like a game: each problem is a boss fight, each rating increase is a level-up. Everything glows, everything burns, everything excites. You feel like you're at the start of a grand journey. Things are clicking. Motivation is high. Energy is through the roof.

But then — like someone flipped a switch — it all goes dim. You keep solving, keep competing, but something inside fades. No joy. No meaning. Just fatigue and a blank stare at the next problem’s statement.

Welcome. This isn’t a bug. It’s burnout. And it hits everyone who stays on the path long enough. But here’s the good news — you can deal with it. You just have to stop ignoring it.

Let’s break it down: how it feels, why it happens, and most importantly — how to recover your second wind and avoid burning out for good.

Over time, you spend hours solving problems daily, training, competing in olympiads, entering contests. Day after day. Month after month. And eventually… you feel nothing.

You open a problem — and don't even want to read it. Just recently you loved this stuff. Now? No emotion. No spark. No drive. Just exhaustion.

That's burnout. And it's not the exception — it's practically a rite of passage.

What Is Burnout?

Burnout isn't just being tired. It's when everything inside feels scorched to ashes. You don't just not want to solve — you're annoyed by it. Code? Frustrating. The timer? Infuriating. Even a simple task feels like a final boss.

Physically, you're fine. But mentally, your soul is asleep. Your brain doesn't work like it used to. Thoughts won't come. Solutions stay hidden. And the things you used to love — now spark only disgust.

Then come the doubts:

- "Maybe I overestimated myself."

- “What if this just isn’t for me?”
- “I don’t feel any progress. Maybe I’ve peaked.”

It’s not whining. It’s a signal — the red warning light on your dashboard. Ignore it, and the engine seizes. So don’t act tough. Admit it: yes, you’re tired. Yes, you’re burned out. And yes — there’s a way out.

Why Does It Happen?

Take a pause. Reflect. Burnout usually comes from a few key sources:

- **You’re pushing too hard.** Setting unrealistic goals, trying to solve 5 problems daily, doing 3 contests per week. But the brain isn’t a machine — it can’t grow endlessly without rest.
- **You’re comparing yourself to others.** Seeing someone hit master in six months and thinking, “I’m too slow.” Jealousy and anxiety kill motivation.
- **You’re solving because you have to, not because you want to.** When the process loses meaning and becomes routine, the brain rebels.

- **You're not truly resting.** You take breaks, but still check Codeforces, still think about problems, still stress about the next step. That's not rest — it's pseudo-rest.

Is Taking a Break a Mistake?

Not always.

In the chapter on beginner mistakes, I warned against random long breaks. A lot of people take a "one-week break" because they're tired or lazy — and never come back. That's dangerous.

But burnout isn't laziness. It's a sign of overtraining. In this case, rest isn't a mistake — it's a necessity.

The key is to know the difference between procrastination and recovery.

- If you're taking a break because you're lazy — you just don't want to work.
- If you're taking a break because you're drained — you're healing.

And in that case, a conscious break for a few days isn't a step back. It's an investment in your future.

My Experience

I've been there too. There was a time I didn't solve anything for almost a month. Not because I didn't want to — I just couldn't. I'd open a problem… and just stare at it. No thoughts. No emotions.

In the past, I would've called that weakness. But now I know — it was a signal. A sign I'd gone too long without rest.

So I stopped. Didn't touch a single problem for a few days. Then I watched videos — not for learning, just for fun. Then the interest returned. And then the training.

And you know what? I came back stronger.

Watch for the Signs:

- You no longer feel joy from solving.
- You're irritable and impatient.
- You don't want to open your laptop.
- You're constantly comparing yourself to others.
- You're solving on autopilot, without being present.

- You feel like “none of this matters.”

If that’s you — stop. Let yourself recharge before you break.

How to Avoid Burnout

- **Mandatory days off:** Pick one day a week where you solve nothing. Not a single problem. No Codeforces. Let your brain feel what it’s like to live without pressure. You’ll be amazed how it resets you.
- **Solve for fun:** Sometimes pick an old favorite, or a problem you always wanted to crack. Not for rating. Not for growth. Just for joy.
- **Find a second hobby:** You can’t live on olympiads alone. That’s a fast track to narrow-minded burnout. I started running and hiking. When I’m in the mountains — I think of nothing. It clears my head.
- **Talk to people like you:** Burnout often feels lonely. But hearing “I’ve been there too” makes it lighter. Talk to friends. Message others who’ve gone through it. Share — don’t isolate.
- **Stop chasing others’ progress:** Comparing your chapter 3 to someone else’s chapter 20 will always make you feel behind. Everyone’s pace, context, and

path is different. What matters is moving forward — your way.

Burnout happens to everyone. And it doesn't make you weak. It just means you're human. You're not a robot. You're not a terminator. You're not required to be at your peak every day.

Burnout isn't the end. It's a signal. It's that moment when you tell yourself: "Stop. Time to recharge."

So don't play the hero. Real strength isn't dragging yourself forward on fumes. It's knowing when to pause, reset, and return strong.

Winners aren't those who never fall. They're those who know how to rise and keep going.

You don't have to be perfect. You don't have to carry everything every day. You don't have to solve when you're exhausted. But you do have to take care of yourself.

Because if you burn out — no rating will save you.

Conclusion

Olympiads are a game. And the long game is won not by flooring the gas nonstop — but by knowing when to brake, breathe, and return refreshed.

The winner isn't the one who never tires. It's the one who knows how to recover — calmly, confidently, without guilt. Because they know: rest isn't weakness. It's strategy.

Chapter 6

The Champion's Mindset: What It's Made Of

Want to become a winner? Forget the image of a nerd with glasses and a stack of notebooks. A winner isn't a tryhard. A winner is a predator. Quiet, bold, confident. He's not necessarily the fastest solver — he's just unbreakable.

You look at people like that and think, "Their brain must be wired differently." But no. They have the same brain. It's just tuned differently. While others sit there stuck, praying for the problem to solve itself, the winner is already digging their way out. They're not a superhero. They just don't fall apart under pressure.

They make mistakes. They blank out. They fail. But unlike the rest — they don't spiral into self-pity. They reset, reflect, and move forward.

And here's the craziest part: you can be smarter than the winner — and still lose. Because the olympiad isn't an intelligence contest. It's a battle for self-control.

You don't win with IQ. You win by staying composed when the problem breaks you. By holding pace when your brain's flying off to Haiti. By not drowning in emotion. By standing up and finishing when everything inside wants to quit.

Why? Because olympiads don't reward intellect. They reward the ability to manage yourself when:

- time is short,
- pressure is high,
- the problems are unclear,
- your brain isn't helping — it's fighting back.

A winner isn't always the one who gets the idea first. Sometimes it's the one who didn't give up when the idea didn't come. That's mental toughness. And it can be trained.

The winner is the one who finds clarity in noise.

What Does It Mean to Be a Winner?

It doesn't mean winning every contest. It means:

- not crumbling after a mistake;
- not panicking on problem three;
- not believing you're worthless when you can't solve;
- not comparing yourself to others every second;
- not basing your self-worth on ratings.

Being a winner is about how you act when everything goes wrong. It's about habits, mindset, and attitude. It's about not letting one performance define your worth.

The Winner Before, During, and After the Contest

Before the contest, they don't spiral with "what ifs." They're calm. They don't cram last-minute — they sleep. They're not afraid of underperforming, because they know success isn't just about points — it's about effort. If they're underprepared, they don't panic — they do what they can. They take care of themselves. They don't show up after 4 hours of sleep — they know the body and mind are allies, not tools to burn.

During the contest, they don't scream inside. Even if the solution doesn't work, even if the subtasks are trivial — they keep going. Step by step, calmly, relentlessly. It's like they say: "This problem isn't a monster. I'll break it down." Fall? Get up. Didn't work? Rewrite. Panic? Not their mode. It's not

about how fast you solved the first problem — it's about how fast you recover from bombing the second.

After the contest, they don't start a meltdown. They don't post: "I suck, quitting programming." They analyze. Reflect. Move on. Because they've got a goal — and it doesn't vanish with one loss. They can say: "Yeah, I messed up. But I'm still in the game. That's what counts." They don't obsess over emotions. They find meaning. They draw lessons — not conclusions about their worth.

5 Traits of a Winner

1. Resilience to failure They tanked a contest? Happens. But they don't spiral, delete accounts, or rage-quit. They analyze and keep going. Like a climber — they slipped, but they're still holding the rock. They know growth isn't a smooth climb — it's stumbles and recoveries.

2. A cold head under pressure Contest is ticking. The task is brutal. Clock's flying. But they don't spiral. They don't waste 10 minutes beating themselves up. They go: "Okay. Let's think. No emotions. What do I know? What can I try?" Winners know how to calm themselves — not with mantras, but with action.

3. Peace with others' success Someone else ranks up? They don't think: "I'm worse." They think: "Nice, they did it. I'll get there too." Winners stay in their lane. They know comparison kills progress. It's not about who's beside you — it's who you were yesterday vs. who you are today.

4. Love for problems, not just results Winners get joy not from medals, but from the moment they wrestled an idea for 20 minutes and everything clicked. They're addicted to problem-solving. If the result doesn't come — they're still satisfied. Because they've grown. Results fade. Process stays.

5. Independence from outcomes Even after a bad contest, they don't conclude "I suck." Because winning isn't about never failing. It's about staying in the game when it all falls apart. One bad result is a dot on the graph — not the whole trajectory.

Personal Story

Here's a story I've mulled over for a while: Zhautykov Olympiad 2024. My first international olympiad — and it was solo format. I barely had experience. But the biggest problem? I thought like a loser. Not a joke.

Before day one, I couldn't sleep. Tossed, turned, thoughts spiraling. Got five hours, tops. Woke up foggy, empty.

During the contest, I downed two cans of Monster — and that sealed my fate. I was shaking — from caffeine and nerves. Every line of code was a battle. Time flew. My mind was noise. My body — panic. My screen — uncertainty.

Still, I finished day one mid-bronze. And I said to myself: "Tomorrow — silver. No matter what."

I believed I could. It felt real. But day two… was a cold slap. I couldn't sleep again. At 4 a.m., I lay in the dark, frozen in panic. No melatonin. No tools. Just dread.

Morning came, and I knew it would be rough. And it was. I underperformed. My rank dropped. Silver — gone. I landed at the lower end of bronze.

What's the point of this story?

Simple: a winner's mindset isn't about heroics. It's about preparing yourself — for real. If I'd known myself better, if I'd prepped mentally and physically — fixed my sleep, learned

how to rest, stopped chasing silver and focused on performing well — things might've gone differently.

My biggest mistake wasn't in my code. It was chasing medals instead of mastery. When I missed the prize, I broke.

Now I know: the best mindset is "I'll give it my all," not "I have to win." The first is in your control. The second isn't.

I didn't win a medal that day. But I won something bigger: the understanding of what a real winner's journey looks like. Not medals on your chest — but lessons in your head.

How to Train Yourself Not to Break During a Contest

1. **Run "no-emotion contests"** Do one contest a month where you forbid yourself from getting upset. No matter what happens — reflect, move on. It's a simulation of resilience. Over time, you'll treat failure as routine — not disaster.
2. **Journal "How I Think"** After each contest, don't write "I was stupid." Write:

- What did I learn?

- Where did I lose focus?
- How did I react when it got tough? This builds awareness. You'll start seeing patterns, psychological traps, and gain real mental control.

3. **Recognize effort even without results** If you sat, thought, struggled, didn't quit — you already won. Even if you didn't rank up. Winning is in the effort. Just keep going. Sometimes your weakest day becomes the base for your strongest one.
4. **Don't treat contests as life-or-death** The brain turns contests into final bosses: "If I fail, it's over." No. It's just one round. Not a tragedy. Not your worth. Just a trial. Winners zoom out. They see the whole arc — not just one dip.
5. **Talk to people at your level** You don't have to do it alone. Everyone messes up. Everyone crashes. Talk. Share. Listen. It's not just support — it's grounding. Winners don't isolate. They know when to reach out and when to speak up.
6. **Measure effort, not just outcomes** If you gave it your all — that's a win. Sometimes the real victory isn't first place — it's not giving up on problem three. Quiet victories are still victories.

Conclusion

You become a winner not when you wear a medal — but when you develop the grit to face any storm. A winner isn't someone who shines onstage. It's someone who doesn't fall apart in the dark.

It's not a status. It's a habit. To not whine. Not crumble. Not quit. To turn failure into a ladder upward. A winner doesn't wait for someone to say "well done." They know when they did great. They're their own judge — and biggest fan.

They don't have a success guarantee. But they do have this: they won't back down. Not on problem one. Not on problem two. Not on problem thirty. A winner keeps going when others stop. Stays quiet when others scream. Thinks when others surrender.

And you can be that.

Not tomorrow. Not after a diploma. Right now.

With the first problem you don't give up on. With the first thought: "I'm not weak. I just haven't figured it out yet." With

the first step forward — not for points, but because you want to be stronger.

A winner isn't a trophy. It's a mindset. It's you — when you don't run from the hard, when you face the pressure and learn to strike true.

So here's the play: Open the problem. Don't flinch. Don't posture. Don't whine. Solve. Fall. Rise. Learn. Burn.

Because you're not background noise. You're not a spectator. You're a player. You've got this. And you're already in the game.

Chapter 7

How an Olympiad Thinker Operates

Okay, let's be real. Have you ever wondered what goes on in an olympiad student's head when they look at a problem? Take a normal, everyday problem — not even an olympiad one. What makes them keep going while everyone else is still stuck in paragraph one?

1. They're not a genius — they just see differently

Most people think olympiad kids are born geniuses, calculators in human form. But in reality, they're just trained to dig deeper. They don't settle for the first answer. They're not trying to be the smartest — they're trying to *really* get it.

When you saw the quadratic formula, you thought, "Cool, $D = b^2 - 4ac$. Let's plug it in." The olympiad kid squints: "Where does this come from? Why minus 4ac? What if the coefficients are negative? What if $D = 0$?"

They chew on the formula. Taste it. Spit it out. Go again. Because they don't want to *memorize* — they want to *understand* how it works, when it works, and when it doesn't.

They're not after the grade. They're after the "I get it now." That's the real difference. Not magic — just mindset.

2. They don't drown — they slice

You read a geometry problem and panic. Bisector angle? Median? Circumcenter in some weird quadrilateral? Your brain yells, "HELP!"

The olympiad thinker stays calm. They breathe. "Okay. What do we know? Where's the trick? What can I throw away?" They start cutting — not the problem, but the noise. They break it down into parts. Like a chef with a pile of ingredients already thinking soup.

Ten-line problem? Doesn't matter. They laser in on what matters. Filter out fluff. Simplify to the core.

They don't freeze in chaos. They attack it — with a logical scalpel.

3. They ask "why" nonstop

The average student: “Triangle angles add up to 180°. Got it. Moving on.”

Olympiad kid: “Why 180? Can I prove it myself? What if the triangle’s weird? What if it’s on a sphere?”

They question everything. Not because they’re smartasses — but because their brain *needs* the reason. “Why is that the rule? Who said so? Could it be different?”

They don’t take facts as truth. They test them. Twist them. Push the limits. They need to own the knowledge — not just rent it.

4. They always have a plan (even when it looks like they don’t)

You hit a test like roulette. Spin the wheel. Hope for the best.

They walk in like a strategist. They’ve got a game plan. Not just “study everything” — they optimize. Flashcards, diagrams, memes for memory — they’re building their own learning engine.

Even in class — while others rush to solve, they pause.
"What's important here? Is there a trap?"

They think before they act. And that's why they move faster — less flailing, more precision.

5. They play with examples

Most students memorize rules. Olympiad thinkers *test* them.

"If A divides B and B divides C, then A divides C." Cool. But does it *always* work? Let's try A = 12, B = 6, C = 3. Okay. Try the reverse. Try negative numbers. Try decimals.

They experiment. They poke the rule until they believe it. Because believing isn't about trust — it's about *proof*.

6. They fail — but don't drown in it

You mess up and spiral: "I'm dumb. I suck. End me."

They mess up and shrug: "Oops. Lesson learned."

They don't cry over mistakes — they *collect* them. Like XP points. Each fail is data. And they move on.

They don’t fear mistakes. They fear stagnation.

7. They think of themselves as a system

You make a mistake and immediately hear sirens in your head: "I'm stupid. It's over." That's the mindset of an average student. Their self-worth is like a house of cards—one mistake, and the whole thing collapses.

An olympiad thinker? They go, "Hmm, my brain’s not working today. Time to reboot." They don’t start yelling at themselves like, “I’m worthless!” They just diagnose: “What went wrong?”

Maybe they didn’t sleep well. Maybe the lesson was boring, or the teacher sounded like a robot. Maybe they started off on the wrong foot. They look at the situation like a system—like a mechanic fixing an engine. They don’t yell at the engine for breaking. They take it apart and find the fault.

They don’t blame themselves. They look for root causes. If they didn’t understand a topic, it’s not a personal failure — it’s just feedback. A signal that the strategy needs adjusting.

They're not fixing "themselves" as some broken student. They're fixing the *situation*. The environment. The method. Because they know: the brain isn't stone. It's flexible. It can grow. And that's not magic — that's mindset.

8. They don't wait to be spoon-fed

You sit there, staring at a problem, hoping the teacher pities you enough to drop a hint. Like, "I'm struggling, help!"

An olympiad thinker? Already in motion.

They don't wait for mercy. They become their own Google, YouTube, and curriculum planner. They sketch diagrams, look things up, ask peers, rewatch concepts. Their mindset isn't "waiting" — it's *searching*. Even if they're totally lost, they're still moving.

Wrote down some ideas. Sketched a structure. Asked a neighbor. Tried edge cases. They don't wait for answers to descend from the heavens — they're building their own ladder.

They don't have a voice in their head saying, "I can't." They have one that says, "I'll figure it out." They're not afraid of

looking dumb — because they know what's really dumb is doing nothing.

The olympiad brain is a combine harvester. It chews through rumors, guesses, sketches, and theories. It doesn't need clarity — just a foothold. The rest will roll from there.

That's real thinking. Not when you *know* — but when you *try to know,* no matter what.

9. They don't just memorize — they understand deeply

Everyone gets assigned a definition. Most students go, "Fine, I'll memorize the exact words." The olympiad thinker asks, "Can I put it in my own words? Because this official phrasing sounds like spaghetti."

While others cram formulas like a checklist, they stop and ask: "Wait. What does this actually mean? Where does it come from? Why does it work this way?"

Most students learn with this mindset: "We were told to. It'll be on the test." Olympiad kids go, "Why bother learning

something I don't get? That's like chewing cardboard — you might swallow it, but you're not gaining anything."

They strip things down to the bones. They don't just write "Area of a triangle = 1/2 base × height." They ask, "What if it's not a right triangle? What if it's degenerate? What if the height is outside the triangle?"

They don't believe in the magic of knowledge. They believe in *their own* understanding. And for them, understanding isn't "I memorized it" — it's "I can explain it to my little brother and he'll get it."

Because their goal isn't just to pass. Their goal is to *see how things tick*. To feel the gears moving. To recognize a formula not as a line of symbols, but as an idea — their idea.

That's why they learn deeper. It takes longer. It's harder. But when the test comes, they don't *recall* the answer — they just *know* it. It's embedded. Not in their memory — in their thinking.

They:

- Don't fear mistakes.

- Think before they act.
- See the essence, not the fluff.
- Compare, rewrite, rephrase.
- Stress-test every rule.

They don't want to be an "A student." They want to be smarter than they were yesterday.

Conclusion

Remember this: each of these thinking styles is powerful. One breaks down the complex. Another sharpens the core idea. A third keeps clarity in chaos. There's no "best" one. Each has its strength — and each can lead to mastery if you know how to use it.

Being an olympiad thinker isn't about grinding through thousands of problems or studying till 3 AM. It's a style. A philosophy. It's when you look at a boring workbook question and, instead of "ugh," your brain says, "Hmm, what if...?"

You study not for grades — but to *understand*. To *feel* the concepts. To live with an idea until it clicks. Not to tick a box, but to grasp the heart of it.

Quiz coming up? Fine. New topic no one gets? Even better. You don't fear tough stuff. Because you don't wait to be fed answers. You chew through it yourself. You bring order to chaos.

And yeah, maybe you're not the fastest in class. But when the time comes — you'll be the most resilient. Because you don't quit after the first mistake. You don't wait for inspiration — you *build* it. You're not just a student. You're a player. An explorer. A brain still firing when everyone else has tapped out.

That's what it means to be an olympiad thinker. Not medals. Not ratings. But the habit of thinking deeply. Questioning the obvious. Digging when others have given up.

And you can start *any time*. Don't wait for someone to knight you "Olympiad Kid." Be one now. With your first "why?" With your first refusal to swallow the answer without tasting it. With the first problem you couldn't solve — but didn't walk away from.

So come on. No hype. No fear. Open the next problem. Don't ask "Can I do it?" Ask: "How is it built?" That's how it all begins.

Chapter 8

Gold at the Cost of Childhood

It all started when I got into RFMSH in the 7th grade. Not right away — I was on the reserve list. Like, "if someone drops out, we'll take you." Funny to remember how anxious I was. But enough people left, the door cracked open — and I walked in.

I'd always loved solving puzzles. Logic, riddles, unusual problems — all of it was my jam. That fall, the school held a selection for olympiad reserves. From our whole class, I was the only one who made it in for informatics. I actually wanted to do math at first — it seemed more "prestigious." But something clicked, and I started coding instead.

That's how it began: me, an informatics group, and three lessons a week. Every class was a struggle — we hadn't even gone beyond basic C++, and I was already burned out. After two months, I quit programming. Just dropped it.

7th grade was a time of adapting. New school, new people — tough times, to put it mildly. But somehow I managed.

I only truly got into competitive programming in November of 8th grade. Something inside lit up: "You have to get better. You have to stay ahead."

That's when it began. I started solving 3 Codeforces problems a day. Sometimes it took half an hour. Sometimes — all night. I began with 800-rated problems and kept raising the bar.

Did I burn out? Of course. On those days, I'd just do three easy problems in 20 minutes to keep the streak alive. Then, in winter, I joined a competitive programming group. That's when everything changed.

I met someone who changed my life — Mikhail Kapitov. My most valuable coach. Our olympiad reserve had just 4 people in a tiny room, and he found a unique approach for each student. The result? Two of us skyrocketed. The third couldn't handle it — studies, pressure, and most importantly, he slept only 3–5 hours a day. He left programming and transferred to another school. I feel for him — he tried hard, but he burned out.

My first real result came half a year later — at the national junior olympiad in informatics. No hype, no clue who my competitors were. I just sat down and gave it my all. And I won gold.

I remember the feeling — like hitting a jackpot. Was I lucky? Maybe. One of the country's top contenders didn't show, and two others messed up the round. But that's part of the game. I had only been training for six months and beat kids who had been prepping for a year and a half. I didn't just fit in — I stormed in.

I started wondering: why me? What made the difference?

The first thing I remembered — six months before the olympiad, I deleted all short-form videos from my life. No reels, no TikToks, no YouTube Shorts. It cleared up my brain. Later I wrote about this in one of the earlier chapters — you've already read it.

After that olympiad, I was added to a private group of all the 8th-grade programmers in Kazakhstan. There I made friends, and two months later went to camp with them. I guess I became known as "that guy who came out of nowhere and won

gold." Before the olympiad, nobody knew me. Then — boom, podium.

And you know what? I really did sell everything for that gold. There was no going back. Or so it felt.

The next year felt like Groundhog Day. The dark green Codeforces calendar became my god. I ditched everything: no walks, no parties, no unnecessary movements. Just code, until my brain screamed, "Stop, I can't take it anymore."

I had a limit — around six hours of coding a day. After that, it turned to mush. That's when I'd shut down and switch to something primitive. Chess became my escape. In two months, I played a thousand games, raising my rating from zero to 1100. Maybe that's what toughened my mind, helped me think three moves ahead — in both code and life.

So, day by day, no breaks, no pity for myself, I chased one goal: gold at the European Junior Olympiad. To me, it was more than a medal. Becoming part of the national team of Kazakhstan — it sounded epic. It was my dream, my challenge, my ultimatum.

And I did it. I tore that gold out with my teeth. But…

Over time, I realized I'd gone too deep into the forest. I looked around — and saw only programmers. People who knew nothing beyond sport, ratings, and templates. And I realized: I'd become one of them.

I had no hobbies left, no social life. Take away my laptop — and poof, I basically didn't exist. Life was warm, but empty. Home-laptop-home. Every day the same.

Yes, I was improving. Yes, everyone else just watched. I was proud to be "not like the others" — but eventually, it stopped feeling like a win and started feeling like a cage. I had no one to really talk to, no one to discuss anything beyond code and ratings.

That's when I started asking: "What do I actually want from life?"

The goal I'd put above all else had been achieved. New olympiads? No spark. The competition in Kazakhstan was insane. More gold? Why, if I was already burned out?

Then came the final realization: I was starting to lose my sense of purpose. Just running scripts I'd written for myself a year ago. Inside — emptiness.

Then, on September 1, 2024, I woke up different. A new chapter began. What it meant — I didn’t know. But one thing was clear: I didn’t want to live in a tunnel anymore.

From the moment I said to myself, "Enough. Time to change," everything flipped.

I started searching. Hobbies, friends, people who think differently. Anything to break the cycle of "solve — win — burn out." First thing I tried — mountains.

And you know what? They hit me right in the soul. Not just enjoyable — addictive.

In the mountains, you’re alone with yourself. No chaos, no screens, no schedules. Just you, nature, and a trail leading upward. On my first hike, I saw wild horses, bulls, waterfalls, and pine forests straight out of a fantasy movie. And I realized: I’d been alive all this time, but not really living.

I didn’t even recover — next Sunday, I was already climbing another peak. Then another. And another.

Every week — a new summit. The mountains became both my drug and my cure. And between hikes — running. Through

pain, sore legs, because by the end of September there was a quarter-marathon, and I couldn't afford to flop.

In just a month, that same kid who once couldn't run 2 kilometers at snail pace:

- climbed 4 peaks,
- ran 10 kilometers in 1:12,
- and finally learned how to breathe again.

And here's the wild part:

I barely coded for a month. Zero contests, zero problems, maybe a few thoughts on runs. But when I came back — boom. 2197 rating. A dream that seemed far away just a month earlier. No overwork, no 6-hour marathons. Just — a new me.

I fixed my health. Now I had to bring people back into my life.

Since September 1, I decided: no more being a ghost. I started talking. To everyone. No exceptions. I tried to be lively, fun, different. And — it worked. My class stopped seeing me as the guy always behind a laptop, and started seeing me as someone worth talking to.

In two months, I met dozens of new people, leveled up not just in my head, but in life. Then came new hobbies: snowboarding, skiing, even drawing.

And for the first time in a long time, I looked at my life and thought: "Damn, it’s beautiful."

In late February, I started preparing for university applications. And suddenly realized — my experience was fading. Thoughts, insights, diagrams, experiments, reflections — everything I’d built over this two-year marathon — was starting to disappear.

And that’s why I sat down to write this book. To capture it. To preserve it. So that you, **dear reader**, could take it all not through two years of pain and a thousand problems, but — in a single evening.

This is my final gift to you. My experience. My story. My observations. Treasure them. And keep moving forward.

My Final Words to <u>You</u>

So, you've read it all? Absorbed the stuff about motivation, burnout, and mindset?

Good. Now forget that you're "just a beginner."

You're armed now. You've got everything you need to walk into a contest, grab a problem, and tear it apart. Want top ranks on Codeforces? Go for it. IOI/ICPC/Google/Meta? Aim higher.

There will be mistakes. There will be panic. That's fine — you didn't come to kindergarten, you came to sportprog.

Here, survival isn't about being smart — it's about being stubborn. Here, winners aren't

geniuses — they're grinders who get back up after every WA.

So close this book, open your IDE — and show the world who's boss now.

GL & HF, fighter. You're ready. See you at the top.

If this book helped you, write to me, I will be pleased.

Author:

Anri Zhan

16 y.o.

Telegram: Wandering_Ssoul

Codeforces: soullless

Email - @anrizhan555@gmail.com

Date: 27.03.2025