اگه تا حالا خواستی برنامه‌نویسی یاد بگیری ولی وسط راه گیر کردی و با خودت گفتی: "من که دارم کُد می‌زنم، چرا هیچی درست کار نمی‌کنه؟"، شاید یه چیزی کم بوده:**تفکر الگوریتمی**. این اون مهارتیه که قبل از اینکه انگشتتو بزاری رو کیبورد، باید توی مغزت فعالش کنی.

یاد گرفتن یه زبان برنامه‌نویسی مثل یاد گرفتن دستور زبان یه زبانه. ولی اینکه بدونی چی می‌خوای بگی، چه جوری بچینی‌ش تا طرف مقابل (کامپیوتر) بفهمه چی می‌خوای، داستانش فرق داره. اون جوری فکر کردن اسمش میشه "تفکر الگوریتمی".

بریم با هم دقیق تر بفهمیم چخبره!!

**تفکر الگوریتمی**

تفکر الگوریتمی یعنی بتونی یه مسئله رو بشکنی به تیکه‌های کوچیک و براش یه راه‌حل گام‌به‌گام پیدا کنی. درست مثل وقتی که می‌خوای صبحونه درست کنی: اول نون میاری، بعد پنیر می‌ذاری، بعد گردو و در نهایت نوش جان. این یه الگوریتمه، بدون اینکه بفهمی داری ازش استفاده می‌کنی.

توی دنیای برنامه‌نویسی، همین کارو باید برای مسائل پیچیده‌تر انجام بدی. مثلاً می‌خوای یه بازی بسازی که توش کاربر باید سکه جمع کنه. باید**قدم‌به‌قدم** فکر کنی که سکه کجا باشه، چطوری کاربر سکه رو بگیره، امتیاز چجوری زیاد بشه و الی آخر. بدون این نوع فکر کردن، کُد نوشتن مثل دویدن تو تاریکیه.

**اهمیتش؟**

تا اینجا فکر کنم متوجه اهمیش شدید ولی خب:

ببین رفیق، کامپیوتر یه موجود بی‌احساسه! نه احساس داره، نه فکر، نه خلاقیت. هر چیزی که بهش می‌گی، همونو دقیقاً و بدون خلاقیت اجرا می‌کنه. پس اگه بخوای ازش یه کاری بخوای، باید خیلی دقیق و مرحله‌به‌مرحله براش توضیح بدی.**نمی‌تونی بگی: "برو یه چایی بیار!"** باید بگی: 1. برو سمت آشپزخونه 2. کتری رو بردار 3. آب بریز 4. بذار رو گاز 5. منتظر بمون جوش بیاد 6. چای خشک بریز تو قوری و... تا آخر!

تفکر الگوریتمی باعث میشه ذهنت منظم بشه. هرچی مسأله بزرگ‌تر باشه، بیشتر بهت کمک می‌کنه که آشفته نشی و بدونی از کجا باید شروع کنی. برنامه‌نویسی بدون الگوریتم مثل اینه که بخوای یه خونه بسازی، ولی ندونی اول باید پی بریزی یا دیوار بچینی.

#### اگه نباشه چی میشه؟

اگه تفکر الگوریتمی نداشته باشی، هر بار که می‌خوای کدی بنویسی، مثل اینه که داری با چشمای بسته پازل حل می‌کنی. یه چیزی می‌نویسی، یه جای دیگه خراب میشه. آخرشم خسته می‌شی و فکر می‌کنی برنامه‌نویسی به درد تو نمی‌خوره. در حالی که اصل قضیه، طرز فکرته نه هوشت یا استعدادت.

خیلی‌ها میان سراغ برنامه‌نویسی، یه زبان یاد می‌گیرن، چندتا دستور و حلقه می‌زنن، ولی چون بلد نیستن الگوریتمی فکر کنن، به یه جایی می‌رسن که نمی‌دونن چی‌کار باید بکنن. نتیجه‌اش؟ ناامیدی، فرار از کدنویسی و احساس شکست. در حالی که با یه کم تمرین روی تفکر الگوریتمی، می‌تونستن راحت‌تر پیش برن.

خب حالا که اهمیتشو فهمیدید بریم یکم بیشتر ازش بگیم.

#### مراحل تفکر الگوریتمی

**1. درک دقیق مسئله :** قبل از هر کاری باید بفهمی چی ازت خواسته شده. مثلاً اگه گفتن یه ماشین حساب بساز، منظوره فقط جمع و تفریق ساده‌ست یا قراره توان و جذر هم بزنه؟ اینجا باید سوالای درست بپرسی و کامل موضوع رو بفهمی. نصف مشکلات از همینجاست که یکی دقیق نمی‌دونه چی می‌خواد بنویسه!

**2. تحلیل و شکستن مسئله به مراحل کوچیک‌تر :** بعد که فهمیدی چی می‌خوای، باید ببینی چطوری می‌شه حلش کرد. مثلاً برای جمع دو عدد، بهش فکر نمی‌کنی چون ساده‌ست. ولی وقتی یه پروژه پیچیده باشه، باید به تیکه‌های کوچیک‌تر تقسیمش کنی. بهش می‌گن Divide and Conquer یعنی "بشکاف و غلبه کن"!

**3. طراحی الگوریتم (نوشتن مراحل حل روی کاغذ):** تو این مرحله، مسئله رو باید بیاری رو کاغذ یا ذهن، که هر مرحله چی‌کار کنه.کُد نه ! فقط مراحل منطقی. مثلاً: ورودی بگیر ، بررسی کن عدد مثبته یا منفی ، اگر مثبته، فلان کار ، اگر منفی، بهمان کار و الی آخر...

**4. تبدیل الگوریتم به کد (پیاده‌سازی) :** حالا که همه‌چی شفاف شد، تازه وقتشه بری سراغ کدنویسی. اینجاست که یه زبان مثل پایتون یا جاوا یا هرچی که بلدی، میاد وسط. چون حالا دیگه دقیقاً می‌دونی قراره چی بنویسی. به جای اینکه وسط کد نویسی فکر کنی "خب حالا بعدش چی؟"، همه‌چی آماده‌ست.

**5. آزمایش و اصلاح (Debug & Test):** کدی که نوشتی رو باید تست کنی. شاید یه جاییش اشتباه باشه. باید ورودی‌های مختلف بدی ببینی درست جواب می‌ده یا نه. اگه نداد، برگردی به مرحله الگوریتم و ببینی کجا پیچیدی!