۷ ترفند مفید برای تسلط بر یک زبان برنامه نویسی جدید

برنامه نویسی دشوار است و آنهایی که خلاف این را می گویند کسانی اند که سال ها تجربه کدنویسی در چنته دارند. طبیعی است که حین یادگیری کمی مضطرب شوید. موارد بسیاری برای یادگیری وجود دارد و احتمالا آموخته ها را به همان سرعت یادگیری فراموش می کنید. بنابرین نگران این مسئله نشوید.

این مساله در عین معمول بودن، آزار دهنده هم خواهد بود. صادقانه بگوییم در صورت نداشتن ذهنیت و رویکرد درست، فراگیری کدنویسی بسیار استرس زا و پریشان کننده خواهد بود.

شما به دنبال فراگیری زبان، کتابخانه یا فریم ورک در اسرع وقت هستید، اینطور نیست؟ این مساله قابل درک است و خوشبختانه توصیه های مفیدی وجود دارد که در حفظ بهتر اطلاعات سنگین و فرار برنامه نویسی به شما کمک می کند.

دور یادگیری شتاب زده را خط بکشید

هیچ کس دوست ندارد این جمله را بشنود اما یادگیری عجولانه، بدترین راه برای یادگیری برنامه نویسی است. اگر شما هم مثل من باشید حفظ کردن شب امتحانی در مدرسه و دانشگاه نقشه فرارتان بوده و تنها راه مطالعه ای که می دانید همین است.

اما از اشتباهات من درس بگیرید: **هرچه بیشتر به دنبال یادگیری سریع و یکباره باشید، کمتر به خاطر می سپارید**. این مساله درباره اغلب موضوعات دانش محور به ویژه برنامه نویسی صدق می کند. برای اثبات این موضوع باید به مطالعه ای اشاره کنیم که در سال ۲۰۰۸ و توسط دانشگاه کالیفرنیا صورت گرفته است:

کارایی دانشجویان در صورت استراحت بین جلسات مطالعه نسبت به زمانی که سعی در حفظ یکباره مطالب دارند، بالاتر است.

دلیل این مساله احتمالا به اثر موقعیت سریالی برمی گردد که توسط روانشناس آلمانی به نام Hermann Ebbinghaus کشف شد و بر اساس آن مغز هنگام مواجهه با چند آیتم که در یک سری قرار دارند، مورد اول و آخر را بهتر به یاد می آورد.

به عبارت دیگر در یک جلسه مطالعه احتمال به خاطر آوردن اطلاعاتی که در اوایل و اواخر جلسه یاد گرفته اید بیشتر از مواردی است که در اواسط آن آموخته اید.

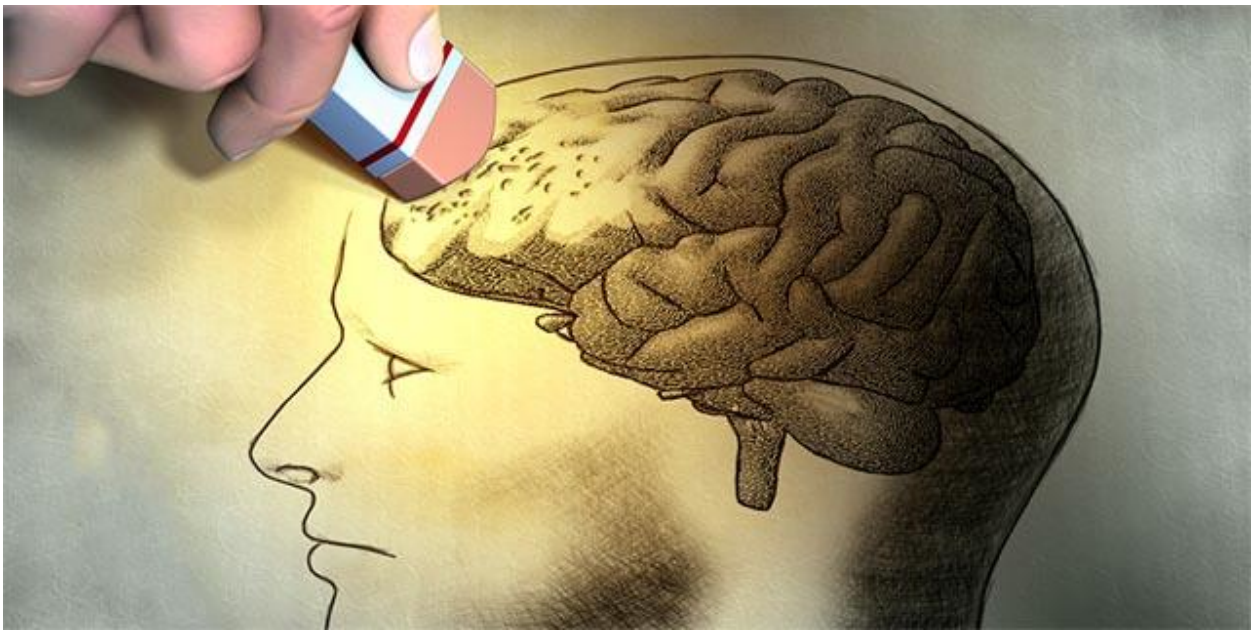
بنابرین باید ابتدا و انتهای جلسه مطالعاتی را تا حد ممکن افزایش دهید و بهترین راه برای این کار مطالعه دروس در جلسات کوتاه تر و اجتناب از مطالعه بدون توقف و استراحت است. از این رو مرتبه بعدی که به سراغ فراگیری کدنویسی رفتید در هر بار تنها یک موضوع را مطالعه کرده و بین آنها به خودتان استراحت دهید. (فقط دقت کنید که زمان های استراحت را بیش از حد کش ندهید).

مرور، مرور، مرور

مدرکی دیگر برای اینکه یادگیری عجولانه نقطه مقابل به خاطر سپردن آنها در بلند مدت است این است که حافظه با گذشت زمان کمرنگ می شود. این ماسله همیشه صادق نیست و شما هم خاطراتی از کودکی دارید که هیچوقت از بین نمی‌روند، ولی این مساله تنها برای خاطراتی است که با عواطف همراه هستند.

بحث‌های بسیاری در رابطه با اینکه آیا حافظه به خاطر گذشت زمان از بین می‌رود (نظریه زوال) یا توسط خاطرات جدید جایگزین می‌شود (نظریه تداخل)، در جریان است. فارغ از اینکه کدام یکی از این تئوری‌ها صحیح باشد، نتیجه این است که پس از مدتی آموخته ها محو می‌شوند.

اهمیت مرور کردن در اینجا مشخص می‌شود.



به این مساله در قالب پیاده روی در جنگل آموخته ها بنگرید. هر زمان که بخواهید به یک آموخته دسترسی داشته باشید، برای پیدا کردن آن باید یک مسیر عصبی را دنبال کنید. با هربار دنبال کردن آموخته مسیر آن کمی مشخص تر می‌شود، درست مانند بیراهه ای که با گذر مکرر عابران از آن، به صورت طبیعی شکل می گیرد. اگر از این مسیر استفاده نکنید، با گذر زمان محو شده و جایی در جنگل گم می‌شود.

فارغ از این بحث روانشناسی، نکته اصلی این است: وقتی پای برنامه نویسی وسط می‌آید، یادگیری یک موضوع برای یک یا دو بار کافی نیست. باید هر کدام از این موضوعات را ده ها و حتی صدها بار مرور کنید. با هر مطالعه یک مبحث بهتر در ذهن شما شکل می گیرد. اگر به یادگیری شتابزده عادت داشته باشید، این کار برایتان بسیار دشوار است، با این حال وقتی شروع به مرور منظم مباحث کنید، از یادآوری سریع آنها شگفت زده خواهید شد.

از چند منبع مختلف استفاده کنید

سخت ترین بخش های برنامه نویسی (حداقل برای برنامه نویس های تازه کار)، گستردگی بالای جزئیات و نکات ظریفی می‌باشد که برای شروع باید به آنها مسلط شد. تا زمانی که تسلط شما بر این نکات به حد کافی نرسد، در سردرگمی دائمی به سر خواهید برد.

بسته به زبانی که در حال یادگیری آن هستید، باید صدها سینتکس و دستور مختلف را به خاطر بسپارید (برای مثال کلمات کلیدی، سمی کولن ها و فضاهای خالی). برخی زبان‌ها سخت تر بوده و برخی دیگر ساده تر هستند. تعدادی از این زبان ها به طور کلی دستورات خودشان را دارند که در موارد دیگر کاربرد ندارند. اگر تجربه‌ی قبلی در کدنویسی نداشته باشید، این موارد می‌توانند گیج کننده باشند.

علاوه بر این اطلاعات مفهومی دیگری را که در میان زبان ها مشترک هستند را هم باید مد نظر قرار دهیم. یادگیری موضوعاتی نظیر برنامه نویسی شی گرا، سیستم های الگوی معماری entity-component و الگوهای مشاهده گر در بار اول برنامه نویسی می‌تواند کار طاقت فرسایی باشد.

فرض کنید شخصی یک تصویر از مجسمه ای به شما نشان داده است. شاید این تصویر کمک کند یک دید کلی نسبت به مجسمه پیدا کنید، اما جزئیات آن را درک نمی کنید. یک تصویر از نمای باز جزئیات کافی را ندارد و تصویر زوم شده هم دید کلی را از بین می برد. با این حال وجود تصاویری دیگر از نماهای مختلف، اطلاعات کاملی از بافت، اندازه و جزئیات از زوایای جلو، کنار یا بالا را در اختیار شما قرار می دهد.

فراگیری برنامه نویسی به صورت عجیبی قراردادی است. شاید همه از منبع A به عنوان بهترین روش برای یادگیری زبان X یاد کنند ولی برای شما کارایی نداشته باشد. شاید همه از منبع B متنفر باشند اما در همان نگاه او برای شما فوق العاده به نظر برسد. ممکن است شخصی دیگر با منابع A و B به مشکل بخورد ولی از منبع C بهره کافی ببرد.

به همین خاطر باید از انواع منابع استفاده کنید، چرا که نحوه یادگیری در هر فرد متفاوت است. اگر با یک موضوع خاص مشکلی دارید، به دنبال یک منابع دیگر باشید، شاید منبع جدید با شما تناسب بیشتری داشته باشد.

حتی اگر فکر می‌کنید موضوع خاصی را به خوبی درک کرده اید، باز هم ممکن است موارد بیشتری برای یادگیری در آن وجود داشته باشد. حتی ممکن است توضیحات فردی دیگر به ماندگاری بیشتر مفهوم در ذهنتان کمک کند. بنابرین بهتر است تا جای ممکن از منابع مختلف استفاده کنید.

توجه داشته باشید که بازی ها ممکن است منابعی فوق العاده مفید باشند.

همزمان با یادگیری مفاهیم آنها را تدریس کنید

در برنامه نویسی مفهوم زیبایی به نام اشکال زدایی اردک پلاستیکی وجود دارد. در این روش زمانی که کد به درستی کار نمی کند، برنامه نویس کد نوشته شده را خط به خط برای اردک پلاستیکی توضیح می‌دهد. این تکنیک زمانی استفاده می‌شود که بخشی از کد مشکل دارد، ولی دلیل مشخصی برای آن وجود ندارد.

نکته عجیب این است که اکثر برنامه نویس ها حین توضیح کد برای اردک متوجه خطا در منابع منطقی کد می‌شوند. بیان شفاهی کدها باعث فعال شدن بخش متفاوتی از مغز شده و شما را وادار به مشاهده مشکل از زاویه متفاوتی می کند.

این روش در یادگیری مطالب جدید نیز به شما کمک می کند. شاید جمله قصار زیر ا که اغلب به آلبرت انیشتین نسبت داده می شود، شنیده باشید:

اگر نمی توانید مساله ای را به سادگی توضیح دهید، پس آن را به خوبی درک نکرده اید.

جدا از برخی زمینه ها که با دانش نظری پیشرفته سرو کار دارند، این جمله در موارد دیگر صدق می کند. هرچه درکتان از یک موضوع بیشتر باشد، بهتر می‌توانید آن را به گونه ای توضیح دهید که شخصی بدون دانش قبلی هم متوجه آن شود.

خلاف این قضیه نیز صادق است. زمانی که سعی در تدریس یک موضوع می‌کنید، با مفاهیمی برخورد می کنید که نمی‌توانید به خوبی آن‌ها را توضیح دهید. این روش نه تنها در پیدا کردن نقاط ضعف علمی به شما کمک می کند بلکه با پیدا کردن توضیح مناسب آن، مفهوم را در ذهنتان پایدارتر می کنید. این روش یادگیری از طریق تدریس نام داشته و اساسا مدل دیگری از اشکال زدایی اردک پلاستیکی است.

البته منظور من این نیست که باید واقعا مطالب را به دیگران تدریس کنید، بلکه سعی کنید هر موضوع جدیدی که فرا می گیرد برای اردک (یا یک دوست نامرئی) هم تشریح کنید. شاید این کار ابتدا کمی احمقانه به نظر برسد اما ارزش آن را زمانی می فهمید که به حفظ مطالب در بلند مدت منجر می شود.

مهارت نتیجه تمرین آگاهانه است

مفهوم استعداد به خودی خود ارزشی ندارد. هیچ کسی به طور مادرزادی یک ویولنیست، کشتی‌گیر یا برنامه نویس کلاس جهانی نیست. قطعا برخی افراد به رشته خاصی تمایل بیشتری دارند، ولی استعداد بدون تجربه بی فایده است. به همین ترتیب سخت کوشی نیز همیشه از استعداد ارزش بیشتری دارد.

البته سخت کوشی صرف هم گاهی اوقات نتیجه ای در پی ندارد. مالکوم گلادول، مبدع قانون ۱۰،۰۰۰ ساعت می‌گوید برای تسلط بر یک موضوع خاص باید ۱۰،۰۰۰ ساعت زمان صرف آن کنید. شاید این جمله درست باشد اما خیلی از مردم حرف وی را اشتباه برداشت می‌کنند.

به طور خلاصه تعهد ۱۰،۰۰۰ ساعته اصلا تضمینی برای تسلط نیست. به قول معروف مهارت نتیجه تمرین نیست بلکه نتیجه تمرین ماهرانه است. **بنابرین تمرین باید آگاهانه صورت بگیرد.** تسلط بر یک موضوع تنها با تمرین آگاهانه و طی ۱۰،۰۰۰ ساعت به دست می آید.

چگونگی تمرین از اینکه چقدر تمرین کنید به مراتب مهمتر است

با مطالعه، مشاهده فیلم های آموزشی، گوش دادن به فایل های صوتی می توانید برنامه نویسی را تمرین کنید. شاید به عنوان یک برنامه نویس تازه کار وسوسه شوید که بدون استفاده از دانش‌های تئوری به طور عملی، از یک خودآموز به خودآموز بعدی رفته و موضوعات را یکی پس از دیگری شروع کنید. با این حال در برابر این وسوسه مقاومت کنید.

درک یک مثال خاص با کنار هم قرار دادن آن از پایه، کاملا متفاوت است. اگر می‌خواهید مراحل یادگیری را سریعتر طی کنید، باید حاضر باشید که به جای منفعل بودن و تماشاگر ماندن، فعالیت عملی بیشتری داشته باشید. در نهایت این تمرین فعال است که اهمیت خواهد داشت.

آزمایش پروژه های شخصی

در دوران مدرسه تکلیف برای من بدترین بخش آموزش بود. در واقع تکلیف شبیه شگردی استادانه برای از بین بردن سرگرمی و مشغول کردن دانش آموزان بود که در واقع همین طور هم هست. ولی الان که به عقب نگاه می‌کنیم، اهمیت تکالیف مشخص می شود. آنها من را مجبور می کردند از دانش جدیدی که به دست آورده بودم به طور عملی استفاده کنم.

اگر شما هم در دوره ها و کلاس های برنامه نویسی ثبت نام کرده اید، کارایی تکالیف را دست کم نگیرید. با جدی گرفتن تکالیف این شانس را دارید که آموخته ها را به حافظه بلند مدت بسپارید.

با این حال در بسیاری موارد انجام تکالیف کفایت نمی کند (اگر به عنوان یک خودآموز و بدون مربی در حال فراگیری کدنویی هستید، احتمالا هیچ تکلیفی در کار نیست).

راه حل این مساله ایجاد چند پروژه جانبی و موقتی است. بدین منظور باید چندین برنامه را که با علایق شما سازگاری دارند، به عنوان پروژه اجرا کنید. اگر در برنامه نویسی تازه کار هستید، می‌توانید یک بازی دوز یا دارزن طراحی کنید. اگر تجربه دارید و می‌خواهید یک فریمورک جدید را بیاموزید، یک اپلیکیشن موبایل یا بازی وبی ساده طراحی کنید. هر چیزی که با علایق شما سازگار باشد را می توانید پیاده سازی کنید.

این روش در واقع یک تیر و دو نشان است:

اولا، توجه شما را جلب خواهد کرد. تحقیقات نشان داده‌اند که دانش‌آموزان هنگام دنبال کردن موضوعات مورد علاقه شان، بهتر یاد می‌گیرند. این همان چیزی است که پروژه شخصی ارائه می کند. شما در این حالت هدفی دارید که واقعا می‌خواهید به آن برسید، در نتیجه احتمال بیشتری دارد که اطلاعات فراگرفته شده را بهتر به خاطر بسپارید.

ثانیا، هیچ فشاری برای رسیدن به هدف روی شما نیست. اگرچه کسب موفقیت خوب است اما رسمی نبودن آن باعث می‌شود خلاقیت بیشتری به خرج داده و تجربه بیشتری کسب کنید. درست است که باید با موضوع سر و کله بزنید، ولی این کار بیشتر شبیه به بازی لگو خواهد بود تا انجام تکلیف. از اینرو لذت بخش بوده و استرسی به دنبال ندارد.

آرامش خود را حفظ کرده و همه چیز را نشانه گذاری کنید

واقعیت این است که هیچ برنامه نویسی تمام آموخته هایش را به یاد ندارد. حتی اگر مدت زمان زیادی را صرف کار با یک فریم ورک یا کتابخانه کرده باشید، بعید نیست که تمام توابع و متغیرهای آن را به یاد بیاورید. در واقع، سعی در حفظ کردن همه چیز ممکن است به قیمت اتلاف زمان و همه تلاش ها تمام شود. وجود مراجع بی دلیل نیست، چرا وقتی می توان برای جستجوی یک موضوع به این منابع مراجعه کرد، همه چیز را حفظ کنیم؟

سوال این است که در چه مواقعی نیاز به حفظ کردن داریم و چه موقع سراغ مراجع برویم؟

وقتی پای مسائل مفهومی در میان است، همیشه سعی کنید تا حد توان آن‌ها را حفظ کنید. منظور من این است که درک بخش تئوری حتی بدون توانایی در تبدیل آن به کد کفایت می کند (البته باید مفهوم تئوری را آنقدر خوب درک کرده باشید که بتوانید آن را به دیگران تدریس کنید).

در موارد دیگر مثل نام توابع خاص، لیست پارامترها و حتی پیش فرض های زبان الزامی برای حفظ کردن وجود نداشته و همینکه آن‌ها را یادداشت کنید کفایت می‌کند. بعضی مواقع آنقدر به یک مساله مرجع می دهید که در نهایت به حافظه تان منتقل می شود. اگر این اتفاق بیفتد خوب است، در غیر این صورت هم مشکلی نیست.

شخصا صدها بوکمارک اینترنتی بهAPI ها، راهنماها و خودآموزهای مختلف دارم. اگر به دنبال پیاده سازی الگوریتم مسیریابی خاصی باشم، به این راهنماها مراجعه می کنم.این مساله به درک مفاهیم زیربنایی کمک می کند اما چندان به دنبال جزییات غیر ضروری آنها نمی روم.

جمع بندی

اگر لازم باشد یک میلیون بار دیگر هم تکرار می کنم که برنامه نویسی سخت است و طبیعی است که در آن با چالش مواجه شوید. من در طول یک دهه گذشته به عنوان سرگرمی برنامه نویسی کرده ام و حتی حال هم گاهی اوقات در صورت مواجه با مفاهیم جدید دست و پای خودم را گم می کنم.

اگر برخی چیزها را به خاطر نمی آورید، خودتان را سرزنش نکنید. امیدواریم که توصیه های بالا در این زمینه به شما کمک کند