Bilgisayar Programcılığının Tarihi ve Yazılımın Geleceği

Bu makalede, bilgisayar programcılığının tarihine ve yazılımın geleceğine dair önemli noktalara değinelim.

Bu makaleyi Spotify'da sesli olarak dinlemek için podcast'ine bu linkten ulaşabilirsiniz.

1.1 Bilgisayar Programcılığının Tarihi

Bilgisayar programcılığı, sohbetin başlarında bahsedildiği gibi, **punch kartları** olarak adlandırılan delikli kartlarla program yazıldığı "cilalı taş devri" veya "taş devri" olarak tanımlanan dönemlere kadar uzanıyor. Bu kartlara delikler delerek program yazılıyordu ve bir program yüzlerce karttan oluşabiliyordu. Programlar büyük makinelerde çalıştırılıyordu ve makine bir seferde bir kişinin işini yapıyordu. Hata yapma şansı yüksekti ve bir hata olursa makine kartı atıyor, program çalışmıyordu. Sonuçlar genellikle yazıcıdan bir kağıt olarak alınıyordu, ekran arayüzü yoktu.

Daha sonra **80'li yıllara** gelindiğinde bilgisayarlar daha yaygınlaştı. Bu dönemde **assembly** lisanını bilmek, bilgisayarcı sayılmak için önemliydi çünkü makineler çok daha limitliydi ve programcının işlemciyi, hafızayı ve yönetimi çok iyi anlaması gerekiyordu. Basic gibi diller de vardı ama assembly gibi donanıma yakın diller daha ön plandaydı. Bu dönemde kaynaklar limitliydi ve çoğu İngilizceydi. Bilgiye ulaşmak zordu; ilgilenen kişiler birbirini buluyor ve bilgi akışı çok yavaş ve organik ilerliyordu. Bilgisayarcılık o zamanlar henüz gözde bir iş değildi; hatta bazı büyükler tarafından sadece "oyun gibi bir şey" olarak görülebiliyordu.

90'lı yıllar ve sonrası bilgisayarların daha da yaygınlaşması ve web'in ortaya çıkmasıyla programlama büyük bir değişim geçirdi. **Python gibi daha üst seviye programlama dilleri** sayesinde, ekrana "print barış" yazdığınızda bunun çalışması için assembly kodunun arkasını bilmeniz gerekmiyordu. Yazılım, sadece temel işlevi yerine getiren bir kod yığını olmaktan çıktı; kullanıcı arayüzü, kullanıcı yönetimi, şifreleme gibi birçok ek bileşeni içermesi gerekiyordu. Yapılan bir araştırmaya göre, modern yazılımların çok büyük bir kısmının (%90'lara varan oranın) **"kalıp kod"** yani her programda olması gereken temel yapılar olduğu, sadece küçük bir kısmının (%10 civarı) ise programa özgü, orijinal kod olduğu düşünülüyor. Bu kalıp kodları yazmak zaman alıyordu ve programcılar Stack Overflow gibi platformlardan hazır çözümler bularak bu kısımları oluşturuyorlardı.

1.2 Yapay Zeka ve Yazılımın Geleceği

Yapay zeka (AI), bu kalıp kod yazma sürecini otomatikleştirerek yazılım geliştirmede önemli bir rol oynamaya başladı. AI, programcıların Stack Overflow'dan elle bulup eklediği kalıp kodları hızlı bir şekilde üretebiliyor. Bu, **yazılım geliştirme sürecinde ciddi bir verimlilik artışı** potansiyeli taşıyor. Programcılar, artık kodun büyük bir kısmını (bu

%90'lık kalıp kısmı) AI'a bırakıp, asıl problemin çözümüne odaklanabilirler (%10'luk orijinal kısım).

Ancak, yapay zeka tarafından üretilen kodun kalitesi önemli bir soru işareti. Sohbetin bir noktasında belirtildiği gibi, bildiğimiz bir konuda AI'a bir şey sorduğumuzda hataları rahatlıkla görebiliyoruz; AI **"destekli atıyor"** diyebiliriz. Dolayısıyla, yapay zeka araçlarını kullanarak yazılım geliştiren bir programcının, **çıkan çıktıyı anlama, değerlendirme ve gerektiğinde düzeltme yeteneğine** sahip olması gerekiyor. Bu yeni dünyada programcı, adeta bir **"usta başı"** gibi, AI'ı bir **"çırak"** gibi kullanarak, ortaya çıkan işi kontrol etmeli. AI, doğru şekilde yönlendirilmezse tehlikeli bir araç olabilir çünkü ne yapmak istediğinizi anlattığınız gibi, **neleri yapmamasını da anlatmanız** gerekebilir. Şu anki yapay zeka modelleri, aynı girdiyle bile bazen farklı sonuçlar üretebilir, bu da bambaşka bir programlama paradigması yaratır.

Yapay zeka, aynı zamanda **programlamacılığı demokratize etme** potansiyeline sahip. Daha önce bir programcıya ihtiyaç duyan basit problemler veya kişisel ihtiyaçlar için, programlama bilgisi olmayan kişiler bile AI'a ne istediklerini anlatarak bir yazılım ortaya çıkarabilirler. Bu, bilgisayar erişimini daha geniş kitlelere açar.

Bu bağlamda, **güçlü iletişim becerileri** yeni dünyada hayati önem taşıyor. Yapay zeka araçlarına etkili bir şekilde talimat vermek, yani derdinizi doğru bir şekilde anlatabilmek, başarılı olmanın olmazsa olmazı olacak. Ayrıca, geliştirilen uygulamanın nihai kullanıcının ihtiyaçlarına hizmet etmesi için **yaratıcılık ve kritik düşünme yeteneği** de çok önemli. Yapay zeka otomasyon sağlasa da, neyin yapılacağını tanımlamak, potansiyel sorunları öngörmek ve çıkan çıktıyı insan kullanımına uygun hale getirmek hala insanın yaratıcılığına ve yargısına bağlı.

Gelecekte, **AI'ın kendini en iyi ifade edebileceği "AI-first" programlama lisanları veya işletim sistemleri** ortaya çıkabilir. Eğer bir programlama lisanını en çok AI kullanacaksa, o lisan AI'a göre şekillenebilir. Bu durumda, programcıların bu yeni lisanları anlaması veya AI tarafından üretilen kodda düzeltmeler yapabilmesi gerekebilir.

Google gibi büyük şirketlerde bile yapay zeka araçları halihazırda kullanılıyor. Özellikle **çok büyük ve karmaşık kod tabanlarını anlamak, yeni yazılımcıların adaptasyonunu hızlandırmak, kod incelemelerini yapmak ve genel verimliliği artırmak** için AI destekli araçlardan faydalanılıyor.

1.3 Genç Yazılımcılara Tavsiyeler

Yapay zekanın programcıların işini tamamen alacağına dair endişeler dile getirilse de, sohbette AI'ın bilgisayar dalından gelen bir ürün olduğu ve bu dalı yok etmesinin beklenmediği ifade edildi. Hatta, **AI'ın bilgisayarcılığa olan ihtiyacı artırabileceği** düşünülüyor. AI, programlama yapamayan kitleleri bu alana çekecek ve yazılacak program sayısı artacak. Ancak bu programların hepsi her problemi çözemeyecek veya belirli performans gereksinimlerini karşılayamayacak. Bu tür özel veya optimize alanlarda hala

deneyimli programcılara ihtiyaç olacak.

Dolayısıyla, bilgisayar programcılığına ilgi duyan genç arkadaşlara bu alana girmeleri tavsiye edilebilir. İşin şekli değişecek; belki artık kalıp kod yazmak yerine, AI'ın yetersiz kaldığı veya optimal olmadığı alanlara odaklanacaklar. Programlama, düşüncelere şekil verilen yaratıcı bir dal olarak tanımlandı ve yaratıcı dalların sonunun hızlı gelmesi beklenmiyor.

Özetle, bilgisayar programcılığı punch kartlarından günümüzün karmaşık yazılımlarına evrildi ve AI ile birlikte yeni bir dönüşüm yaşıyor. AI, kalıp kod otomasyonuyla verimliliği artırsa da, programcıların rolü tamamen ortadan kalkmıyor; aksine, **usta başı rolünde, AI'ı yöneten, çıkan çıktıyı değerlendiren, güçlü iletişim becerilerine sahip ve yaratıcı bireylere** dönüşüyorlar.