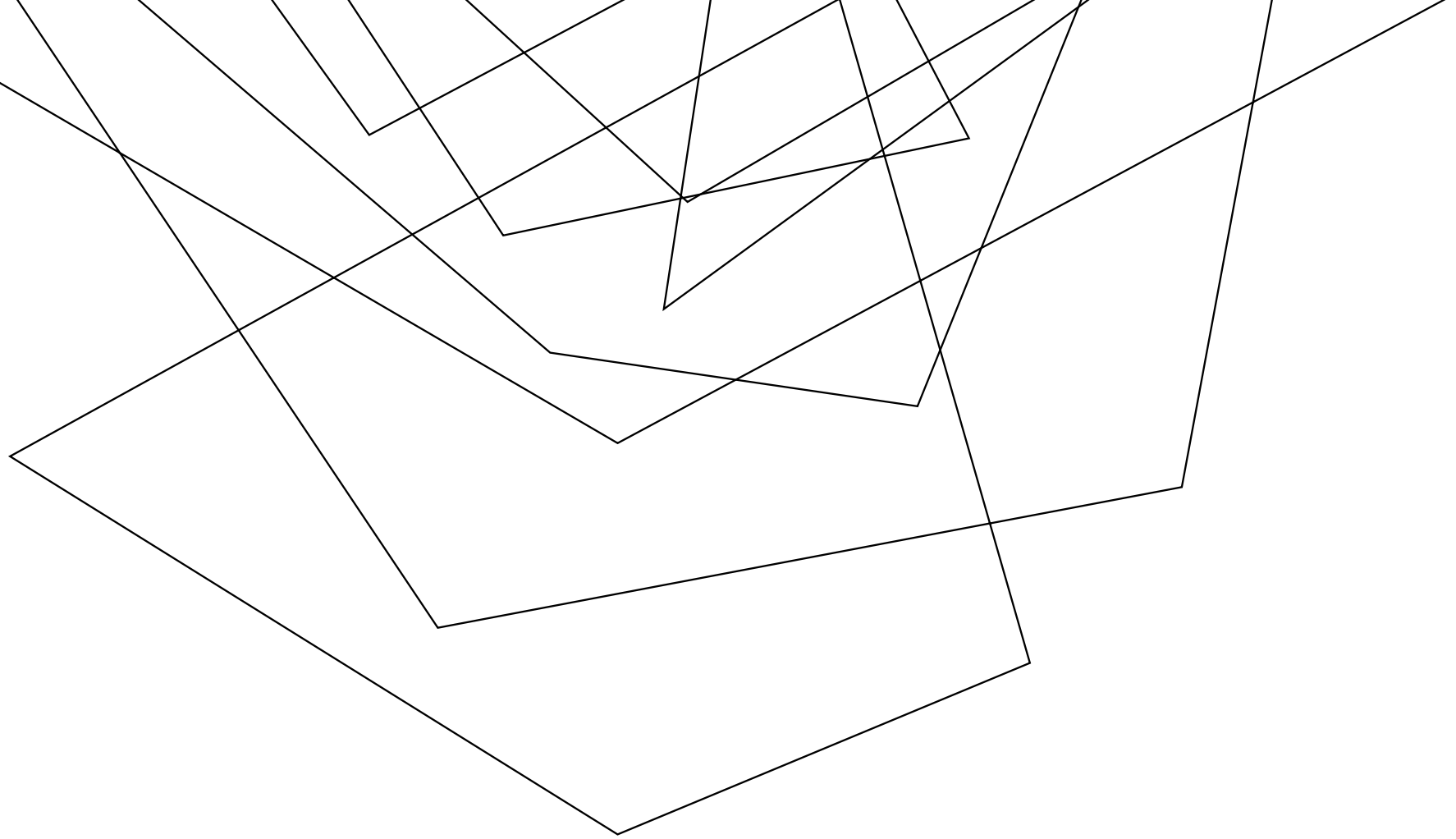


1- PROGRAMLAMA DİLLERİ VE C

İÇERİK

- Programlama dili nedir?
- Programlama dillerinde sözdizim ve anlam nedir?
- Tip sistemi nedir?
- Programlama dillerinin tarihi nasıl şekillenmiştir?
- Programlama dilleri nasıl sınıflandırılır?
- Programlama paradigmaları nelerdir?

PROGRAMLAMA DİLLERİ

Bir programlama dili, çeşitli türde makine kodu çıktıları üreten bir dizi metinden oluşan biçimsel bir dildir. Programlama dilleri bir tür bilgisayar dilidir ve bilgisayar programlamada algoritmaları uygulamak için kullanılır.

https://gowithcode.com/top-programming-languages

PROGRAMLAMA DİLLERİNİN TARİHİ

İlkel programlama: Mantık devreleri değiştirilerek veya fiziksel kontrol sistemleri ayarlanarak programlama yapılırdı. Örnek: Colossus (1943–1945)

Birinci Nesil Programlama Dilleri: Bir zaman sonra, programlar, programcının her talimatı donanımın doğrudan yürütebileceği sayısal bir kod biçimde tanımladığı makine dilinde yazılabildi. Makine dilleri daha sonra birinci nesil programlama dilleri (1GL) olarak adlandırıldı.

İkinci Nesil Programlama Dilleri: Bir sonraki adım, belirli bir bilgisayarın komut seti mimarisine hâlâ sıkı sıkıya bağlı olan ikinci nesil programlama dillerinin (2GL) veya çevirici (**assembly**) dillerinin geliştirilmesiydi. Bunlar, programı insan tarafından çok daha okunabilir hale getirmeye hizmet etti ve programcıyı sıkıcı ve hataya açık adres hesaplamalarından kurtardı.

PROGRAMLAMA DİLLERİNİN TARİHİ

Üçüncü Nesil Programlama Dilleri: İlk yüksek seviyeli programlama dilleridir. Bir bilgisayar için tasarlanan ilk üst düzey programlama dili, 1943 ve 1945 yılları arasında Konrad Zuse tarafından Alman Z3 için geliştirilen Plankalkül'dü.

1954'te **FORTRAN**, IBM'de **John Backus** tarafından icat edildi. Yalnızca kağıt üzerindeki bir tasarımın aksine, işlevsel bir uygulamaya sahip, yaygın olarak kullanılan ilk yüksek seviyeli genel amaçlı programlama diliydi. Hâlâ yüksek performanslı bilgi işlem için popüler bir dildir ve dünyanın en hızlı süper bilgisayarlarını kıyaslayan ve sıralayan programlar için kullanılır.

PROGRAMLAMA DİLLERİNİN TARİHİ

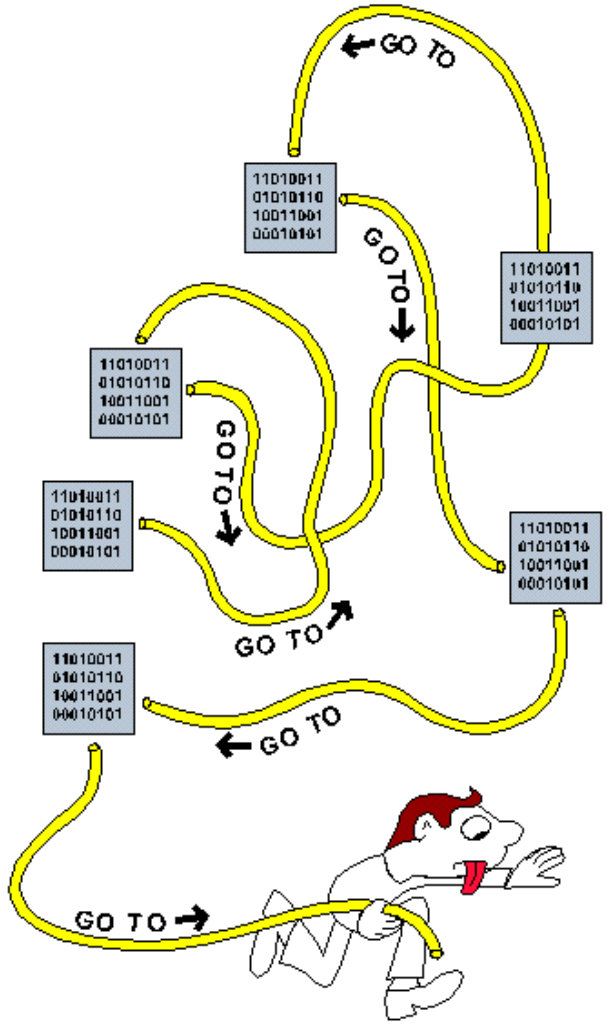
1960’lar ve 1970’lerde, şu anda kullanımda olan başlıca dil paradigmaları geliştirildi:

- **APL**, dizi programlamayı tanıttı ve işlevsel programlamayı etkiledi.
- **ALGOL**, hem yapılandırılmış prosedürel programlamayı hem de dil belirtimi disiplinini iyileştirdi.
- 1958'de uygulanan **Lisp**, dinamik olarak yazılan ilk işlevsel programlama diliydi.
- 1960'larda **Simula**, nesne yönelimli programlamayı desteklemek için tasarlanmış ilk dildi; 1970'lerin ortalarında, **Smalltalk** ilk "tamamen" nesne yönelimli dil olarak onu izledi.
- **C**, 1969 ve 1973 yılları arasında Unix işletim sistemi için bir sistem programlama dili olarak geliştirildi ve popülerliğini koruyor.
- 1972 yılında tasarlanan **Prolog**, ilk mantıksal programlama diliydi.
- 1978'de **ML (meta language)**, Lisp'in üzerine polimorfik bir tip sistemi kurarak statik olarak yazılan fonksiyonel programlama dillerine öncülük etti.

PROGRAMLAMA DİLLERİNİN TARİHİ

1960'lar ve 1970'ler ayrıca **yapılandırılmış programlama**nın esası ve programlama dillerinin onu desteklemek için tasarlanıp tasarlanmaması gerektiği konusunda önemli tartışmalara sahne oldu.

Edsger Dijkstra, *Communications of the ACM*'de yayınlanan 1968 tarihli ünlü bir mektupta, *Goto* ifadelerinin tüm "yüksek seviyeli" programlama dillerinden çıkarılması gerektiğini savundu.



https://www.pcmag.com/encyclopedia/term/spaghetti-code

PROGRAMLAMA DİLLERİNİN TARİHİ

1980'ler görece konsolidasyon (sağlamlaştırma) yıllarıydı.

C++ nesne yönelimli paradigma ile sistem programlamayı birleştirdi.

Amerika Birleşik Devletleri hükümeti, Pascal'dan türetilen ve savunma müteahhitleri tarafından kullanılması amaçlanan bir sistem programlama dili olan **Ada**'yı standartlaştırdı.

Japonya'da ve başka yerlerde, mantık programlama yapılarını birleştiren **beşinci nesil** adı verilen dilleri araştırmak için büyük meblağlar harcandı.

İşlevsel diller topluluğu, ML ve Lisp'i standart hale getirmek için harekete geçti. Yeni paradigmalar icat etmek yerine, tüm bu hareketler önceki on yıllarda icat edilen fikirleri ayrıntılandırdı.

PROGRAMLAMA DİLLERİNİN TARİHİ

1980'lerde büyük ölçekli sistemleri programlamak için, modüllerin veya büyük ölçekli organizasyonel kod birimlerinin kullanımına eğilim arttı.

Modula-2, Ada ve ML'nin tümü, 1980'lerde, genellikle genel programlama yapılarına bağlı olan dikkate değer **modül sistemleri** geliştirdi.

1990'ların ortalarında internetin hızlı büyümesi yeni diller için fırsatlar yarattı. İlk olarak 1987'de piyasaya sürülen bir Unix komut dosyası oluşturma aracı olan **Perl**, dinamik web sitelerinde yaygınlaştı.

Java, sunucu taraflı programlama için kullanılmaya başlandı ve byte kodlu sanal makineler, "Bir kez yaz, her yerde çalıştır" vaadiyle ticari ortamlarda yeniden popüler hale geldi. Bu gelişmeler temelde yeni değildi; daha ziyade, mevcut birçok dilin ve paradigmanın iyileştirmeleriydi.

PROGRAMLAMA DİLLERİNİN TARİHİ

Programlama dilinin gelişimi hem endüstride hem de akademik araştırmalar düzleminde devam etmektedir. Güncel yönergeler arasında güvenlik ve güvenilirlik doğrulaması, yeni modülerlik türleri ve Microsoft'un LINQ'su gibi veritabanı entegrasyonu yer alıyor.

Dördüncü nesil programlama dilleri (4GL), dahili bilgisayar donanımı ayrıntılarının 3GL'lerden daha yüksek düzeyde soyutlanmasını sağlamayı amaçlayan bilgisayar programlama dilleridir.

Beşinci nesil programlama dilleri (5GL), bir programcı tarafından yazılmış bir algoritma kullanmak yerine, programa verilen kısıtlamaları kullanarak problemleri çözmeye dayanan programlama dilleridir.

PROGRAMLAMA DİLLERİ

Tüm programlama dilleri, verilerin tanımı ve bunlara uygulanan işlemler veya dönüşümler için bazı ilkel yapı taşlarına sahiptir (iki sayının eklenmesi veya bir koleksiyondan bir öğenin seçilmesi gibi).

Bu ilkel yapı taşları, sırasıyla yapılarını ve anlamlarını tanımlayan **sözdizimsel** ve **anlamsal** kurallarla tanımlanır.

SÖZDİZİM (SYNTAX)

Bir programlama dilinin yüzey formu, **sözdizim** (syntax) olarak bilinir. Çoğu programlama dili tamamen metinseldir; yazılı doğal dillere çok benzer şekilde sözcükler, sayılar ve noktalama işaretleri içeren metin dizileri kullanırlar. Öte yandan, bir programı belirtmek için semboller arasındaki görsel ilişkileri kullanan, doğası gereği daha grafiksel olan bazı programlama dilleri vardır.

Bir dilin sözdizimi, sözdizimsel olarak doğru bir program oluşturan olası **sembol kombinasyonları**nı tanımlar. Bir sembol kombinasyonuna verilen anlam, semantik tarafından ele alınır.

SÖZDİZİM (SYNTAX)

Programlama dili sözdizimi genellikle **düzenli ifadeler**in (sözcüksel yapı için) ve **Backus–Naur form**unun (dilbilgisel yapı için) bir kombinasyonu kullanılarak tanımlanır.

Aşağıda, Lisp'e dayalı basit bir dilbilgisi verilmiştir:

```
expression ::= atom | list
atom       ::= number | symbol
number     ::= [+-]?['0'-'9']+
symbol     ::= ['A'-'Z''a'-'z'].*
list       ::= '(' expression* ')'
```

ANLAMBİLİM (SEMANTICS)

Anlambilim terimi, dillerin biçimlerinin (sözdizimlerinin) aksine anlamlarını ifade eder.

Derlenmiş diller için, **statik anlambilim**, esas olarak, derleme zamanında kontrol edilebilen anlamsal kuralları içerir. Java ve C# gibi daha yeni programlama dilleri, statik anlambilimlerinin bir parçası olarak bir veri akışı analizi biçimi olan kesin atama analizine sahiptir.

Bir dilin **dinamik anlambilimi** (yürütme semantiği olarak da bilinir), bir dilin çeşitli yapılarının nasıl ve ne zaman bir program davranışı üretmesi gerektiğini tanımlar. Yürütme semantiğini tanımlamanın birçok yolu vardır. Doğal dil genellikle uygulamada yaygın olarak kullanılan dillerin yürütme anlamlarını belirtmek için kullanılır.

TİP SİSTEMİ

Bir **tip sistemi**, bir programlama dilinin değerleri ve ifadeleri nasıl sınıflandırdığını, bu tipleri nasıl değiştirebileceğini ve nasıl etkileşime girdiklerini tanımlar.

Bir tip sisteminin amacı, belirli yanlış işlemleri tespit ederek o dilde yazılmış programlarda belirli bir doğruluk düzeyini yakalamak ve genele uygulamaktır.

TİP SİSTEMİ

Her işlemin belirtimi, işlemin uygulanabileceği veri tiplerini tanımlıyorsa, bu dil tiplidir.

Örneğin, "tırnak işaretleri arasındaki bu metin" ile temsil edilen veriler bir String'dir ve birçok programlama dilinde bir sayıyı bir String'e bölmenin bir anlamı yoktur ve yürütülmeyecektir.

Geçersiz işlem, program derlendiğinde ("statik" tip kontrolü) tespit edilebilir ve derleyici tarafından bir derleme hata mesajı ile reddedilir veya program çalışırken tespit edilebilir ("dinamik" tip kontrolü). Birçok dil, istisna işleyicisi adı verilen bir işlevin bu istisnayı işlemesine izin verir ve örneğin, sonuç olarak her zaman "-1" döndürür.

TİP SİSTEMİ

Statik tip sisteminde, tüm ifadelerin tipleri program yürütülmeden önce, genellikle derleme zamanında belirlenir. Örneğin, 1 ve (2+2) tamsayı ifadeleridir; String bekleyen bir fonksiyona geçilemezler veya tarihleri tutmak için tanımlanmış bir değişkende saklanamazlar.

Dinamik tip sisteminde, işlemlerin tip güvenliği çalışma zamanında belirlenir; başka bir deyişle, tipler metinsel ifadeler yerine çalışma zamanı değerleriyle ilişkilendirilir.

Dinamik tipli diller, programcının ifadelere tip tanımlayıcıları yazmasını gerektirmez. Bu durum, tek bir değişkenin, programın yürütülmesinde farklı noktalarda farklı tipteki değerlere atıfta bulunmasına izin verebilir. Ancak, bir kod parçası gerçekten yürütülene kadar tip hataları otomatik olarak algılanamaz, bu da potansiyel olarak hata ayıklamayı daha da zorlaştırır.

Lisp, **Smalltalk**, **Perl**, **Python**, **JavaScript** ve **Ruby**, dinamik tipli dillerin örnekleridir.

STANDART KÜTÜPHANE SİSTEMİ

Çoğu programlama dili, geleneksel olarak dilin tüm uygulamaları tarafından kullanılabilir hale getirilen ilişkili bir çekirdek kütüphaneye sahiptir. Çekirdek kütüphane yaygın olarak kullanılan algoritmalar, veri yapıları ve girdi - çıktı mekanizmaları için tanımları içerir.

SINIFLANDIRMA

Programlama dilleri için kapsamlı bir sınıflandırma şeması yoktur. Belirli bir programlama dilinin genellikle **tek bir ata dili bulunmaz**. Diller genellikle, o sırada dolaşımda olan birkaç önceki dilin öğelerini yeni fikirlerle birleştirerek ortaya çıkar. Bir dilde ortaya çıkan fikirler, ilgili diller ailesine yayılacak ve daha sonra birdenbire farklı bir ailede ortaya çıkacaktır.

Programlama dillerini sınıflandırmak, diller birden fazla eksen boyunca gruplanabildikleri için **karmaşık** bir iştir. Örneğin, Java hem nesne yönelimli bir dildir (çünkü nesne yönelimli organizasyonu teşvik eder) hem de eşzamanlı bir dildir (çünkü birden çok iş parçacığını paralel olarak çalıştırmak için yerleşik yapılar içerir). Python, nesne yönelimli bir betik dilidir.

Genel hatlarıyla, programlama dilleri için, alana özgü olanlardan ayrılan genel amaçlı dillerle birlikte, **programlama paradigmaları**na ve amaçlanan **kullanım alanı**na göre bir sınıflandırma yapılabilir.

PROGRAMLAMA PARADİGMALARI

Programlama paradigması, bilgisayar programlarının yapısını ve öğelerini oluşturma tarzıdır.

Başlıca iki temel programlama paradigmasından söz edilebilir:

- buyurucu (imperative)
- bildirimsel (declarative)

Programlama Paradigmaları

- Buyurucu
  - Yordamsal
  - Nesne Yönelimli
- Bildirimsel
  - İşlevsel
  - Mantıksal

PROGRAMLAMA PARADİGMALARI

Buyurucu (Imperative) Programlama

- Bir programın durumunu değiştiren ifadeleri kullanan bir programlama paradigmasıdır.
- Doğal dillerdeki **emir kipi**nin komutları ifade etmesi gibi bilgisayarın gerçekleştirmesi için komutlardan oluşur.
- Bir programın **nasıl çalıştığını** açıklamaya odaklanır.

PROGRAMLAMA PARADİGMALARI

Yordamsal (Procedural) Programlama

- Programın bir veya daha fazla prosedürden (alt rutinler veya işlevler olarak da adlandırılır) oluşturulduğu bir tür buyurucu programlamadır.
- Durum değişikliklerinin yordamlarla (prosedür) yerelleştirildiği veya açık argümanlar ve prosedürlerden geri dönüşlerle sınırlandırıldığı yoğun prosedürel programlama, bir yapılandırılmış programlama şeklidir.
- Bir programcı, çoğu zaman, prosedürlerin adlarına, argümanlarına ve dönüş türlerine (ve ilgili yorumlara) bakarak, belirli bir prosedürün ne yapması gerektiğini, mutlaka sonuca nasıl ulaştığının ayrıntılarına bakmadan söyleyebilir.

Prosedürel programlama dilleri; Fortran, ALGOL, COBOL, PL/I ve BASIC, Pascal ve C'dir.

PROGRAMLAMA PARADİGMALARI

Nesne Yönelimli Programlama (Object-Oriented Programming) (OOP)

- “Nesneler" kavramına dayanan, veri ve kod içerebilen bir programlama paradigmasıdır.
- Nesnelerin özellikleri/veri alanları ve nesneler üzerinde gerçekleştirilebilen işlemlerden (method) oluşur.
- Nesneler kendi prosedürleri aracılığıyla kendi veri alanlarına erişebilir ve sıklıkla bunları değiştirebilir (nesnelerin “bu”~this veya “kendi”~self kavramı vardır).
- Programlar, birbirleriyle etkileşime giren nesnelerden oluşturularak tasarlanır.

Önemli nesne yönelimli diller: Java, C++, C#, Python, R, PHP, Visual Basic.NET, JavaScript, Ruby, Perl, SIMSCRIPT, Object Pascal, Objective-C, Dart, Swift, Scala, Kotlin, Common Lisp, MATLAB ve Smalltalk.

PROGRAMLAMA PARADİGMALARI

Bildirimsel (Declarative) Programlama

- Hesaplama mantığını kontrol akışını tanımlamadan ifade eder.
- Bu stili uygulayan birçok dil, programı komutlarla ifade etmek yerine, programın **neyi başarması** gerektiğini tanımlayarak yan etkileri en aza indirmeye veya ortadan kaldırmaya çalışır.
- Genellikle programları biçimsel bir mantığın teorileri olarak ve hesaplamaları bu mantık uzayındaki çıkarımlar olarak kabul eder.
- Paralel programlar yazmayı büyük ölçüde basitleştirebilir.

Yaygın bildirim dilleri, veritabanı sorgulama dillerini (örneğin, SQL, XQuery), düzenli ifadeleri, mantıksal programlamayı, işlevsel programlamayı ve konfigürasyon yönetim sistemlerini içerir.

PROGRAMLAMA PARADİGMALARI

İşlevsel (Functional) Programlama

- Programların işlevleri uygulayarak ve bir araya getirerek oluşturulduğu bir programlama paradigmasıdır.
- Programın çalışma durumunu güncelleyen bir dizi buyurucu ifade yerine, değerleri diğer değerlere **eşleyen** ifade ağaçları bulunur.
- İşlevler birinci sınıf vatandaşlar olarak kabul edilir, yani diğer herhangi bir veri türünün yapabileceği gibi adlara (yerel tanımlayıcılar dahil), argüman olarak iletilebilir ve diğer işlevlerden döndürülebilirler.

Common Lisp, Scheme, Clojure, Wolfram Language, Racket, Erlang, Elixir, OCaml, Haskell ve F# dahil olmak üzere birçok işlevsel dil bugün endüstri ve eğitimde kullanım görmektedir.

PROGRAMLAMA PARADİGMALARI

Mantıksal (Logic) Programlama

- Büyük ölçüde **biçimsel mantığa** dayanan bir programlama paradigmasıdır.
- Mantıksal programlama dilinde yazılmış herhangi bir program, bazı problem alanları hakkında gerçekleri ve kuralları ifade eden, mantıksal biçimde bir dizi cümledir.
- Bu dillerin tümünde kurallar tümceler şeklinde yazılır:
  - fallible(X) :- human(X)
  - human(socrates)

Başlıca mantık programlama dili aileleri arasında Prolog, answer set programming (ASP) ve Datalog bulunur.

KULLANIM ALANINA GÖRE SINIFLANDIRMA

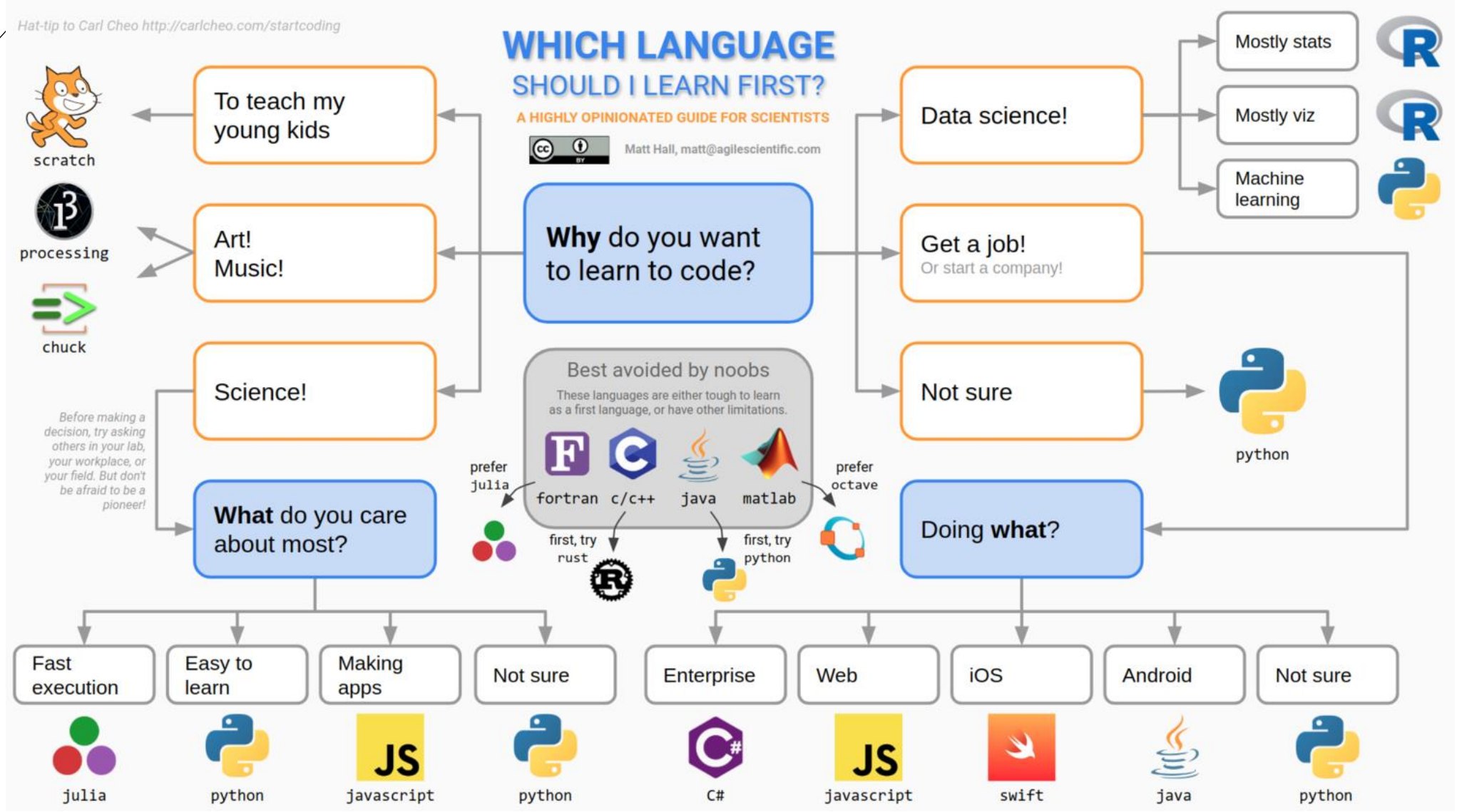
Genel amaçlı programlama dilleri:

C, C++, C#, Clojure, Crystal, Dart, Elixir, Erlang, F#, Go, Harbour, Haskell, Java, JavaScript, Julia, Kotlin, Lua, Modula-2, Oberon, Objective-C, Perl, PHP, Pike, PL/I, Python, Racket, Ruby, Rust, Scala, Swift, Tcl, Visual Basic, Visual Basic .NET, Zig, ...

Alana özgü programlama dilleri:

HTML (web), Logo (çizim), Verilog ve VHDL (donanım tanımlama dilleri), MATLAB ve GNU Octave (matris programlama), Mathematica, Maple ve Maxima (sembolik matematik), SQL (ilişkisel veritabanı sorguları için), YACC (dilbilgisi ve ayrıştırıcı üretmek için), düzenli ifadeler (regular expressions), Csound (ses ve müzik sentezi), ...

HANGİ DİLİ SEÇMELİSİNİZ?



https://agilescientific.com/blog/2021/4/8/which-programming-language-should-you-learn-first

2022 YAZILIM GELİŞTİRİCİ ANKETİ

https://survey.stackoverflow.co/2022/

C DİLİ

C programlama dili ilk olarak **Dennis Ritchie** tarafından 1969 ve 1973 yılları arasında ABD'deki AT&T Bell Laboratuarlarında geliştirilmiştir.

B, C'nin öncü diliydi. B, Ken Thompson tarafından Bell Laboratuarlarında oluşturuldu ve **UNIX**'in ilk sürümlerinde kullanılan yorumlanmış bir dildi. Zamanla, B dilini geliştirdiler ve derlenmiş bir programlama dili olan mantıksal halefi C'yi yarattılar.

C programlama dilinin gelişimi, UNIX işletim sisteminin gelişimiyle yakından bağlantılıydı. Unix çekirdeği, C'de uygulanacak ilk işletim sistemi çekirdeklerinden biriydi. UNIX'in taşınabilirliği ve mimari tarafsızlığı, hem C'nin hem de UNIX'in ilk başarısının ana nedeniydi.

C DİLİ

1978'de Dennis Ritchie ve Brian Kernighan, The C Programming Language'ın ilk baskısını yayınladılar.

Amerikan Ulusal Standartlar Enstitüsü, 1983 yılında C dilini standartlaştırma çalışmalarına başladı ve 1989'da standardı tamamladı.

C programlama dilinin standardizasyonu 1983'te Amerikan Ulusal Standartlar Enstitüsü (ANSI) tarafından başladı ve 1989'da tamamlandı.

1990 yılında, **ANSI C** standardı Uluslararası Standardizasyon Örgütü tarafından kabul edildi.

C DİLİ

```
#include “stdio.h”
main() {
        int x, y, z;
        x=5;
        y=6
        z=x*y;
        printf(“\nz=%d”,z)
}
```

C DİLİ

Ayrılmış Sözcükler (Reserved Words)

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

C DİLİ

Değişkenlere İsim Verme Kuralları

- İsimler bir harf ya da _ sembolü ile başlayabilir; bunların dışındaki karakterlerle başlayamaz.
- Değişken isimleri içinde harfler, rakamlar ve _ sembolü bulunabilir. Özel karakterler (?,+,* vb.) kullanılamaz.
- C dili büyük-küçük harf ayrımı yapan (case sensitive) bir dildir. C dili için Abx değişkeni ile abx değişkeni farklı değişkenler olarak kabul edilir.
- Değişken isimlerinin uzunlukları ANSI standardına göre en az 31 karakter uzunluğa kadar olabilmelidir. Bunun anlamı şudur: C programcısı istenilen uzunlukta bir değişken ismi tanımlayabilir fakat ANSI C standardına uyan bir C derleyicisi, en kötü durumda bunun ilk 31 karakterini dikkate alacaktır.

Program İçindeki Açıklama Satırları

Bir C program içinde, gereken her yerde, programın akışı ya da deyimleri ile ilişkili açıklamalarda bulunulabilir. Bu tür ifadeler derleyici tarafından dikkate alınmaz; sadece listeyi inceleyen programcılar açısından yararlıdır. Bu tür açıklama ifadeleri /* */ sembolleri içine alınır.

KAYNAKLAR

- https://en.wikipedia.org/wiki/Programming_language
- C ile Programlama, Mithat Uysal, Nirvana Yayınları, 4. Baskı, 2012.