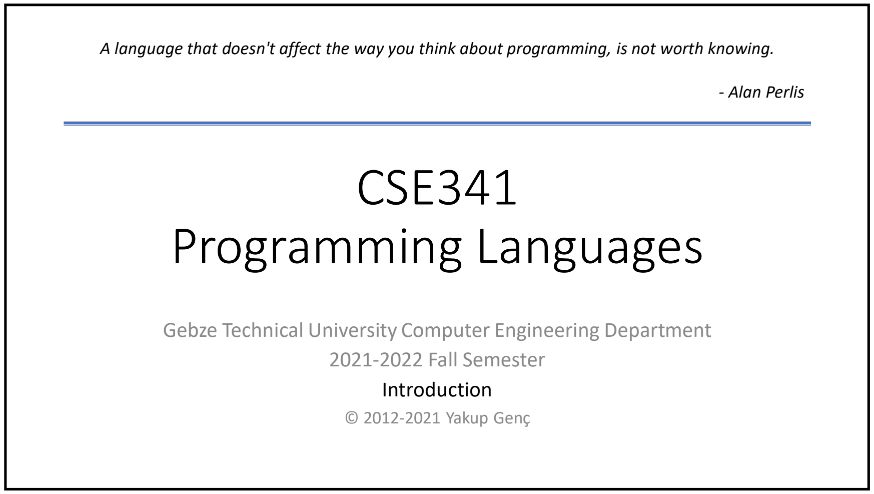


A language that doesn't affect the way you think about programming, is not worth knowing.

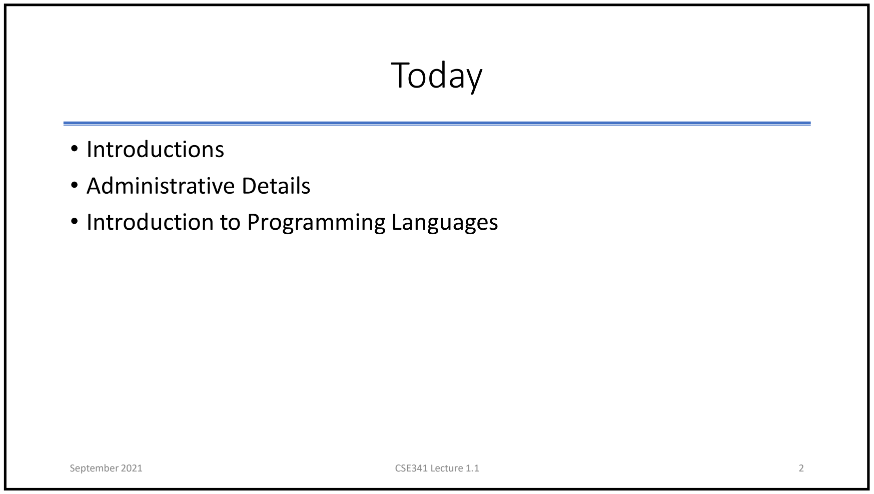
- Alan Perlis

CSE341
Programming Languages

Gebze Technical University Computer Engineering Department
2021-2022 Fall Semester

Introduction

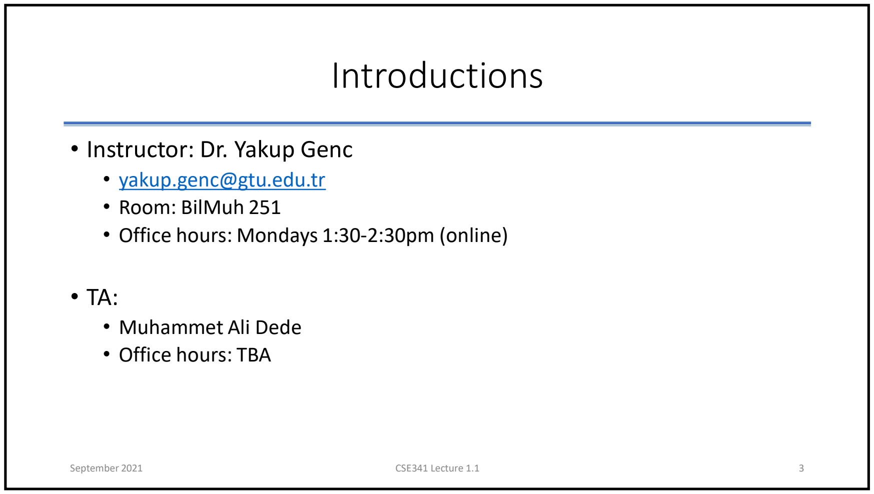
1



Today

- Introductions
- Administrative Details
- Introduction to Programming Languages

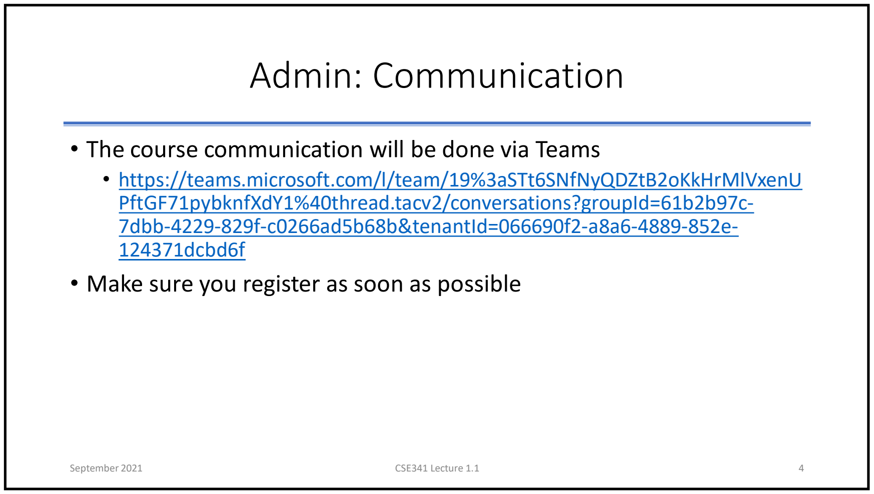
2



Introductions

- Instructor: Dr. Yakup Genc
  - yakup.genc@gtu.edu.tr
  - Room: BilMuh 251
  - Office hours: Mondays 1:30-2:30pm (online)
- TA:
  - Muhammet Ali Dede
  - Office hours: TBA

3



Admin: Communication

- The course communication will be done via Teams
  - https://teams.microsoft.com/l/team/19%3aSTt6SNfNyQDZtB2oKkHrMlVxenUPftGF71pybknfXdY1%40thread.tacv2/conversations?groupId=61b2b97c-7dbb-4229-829f-c0266ad5b68b&tenantId=066690f2-a8a6-4889-852e-124371dcbd6f
- Make sure you register as soon as possible

4

Admin: Prerequisites

- You should know/have
  - Data structures
  - Good working knowledge of C/C++ & Java

Admin: Attendance etc.

- Policy:
  - Attendance is mandatory (at least %70)
  - Failure to meet requirement will result in a fail
  - No late admittance

Admin: Grading

- Grading (tentative)
  - %40 – Homework and projects
    - Average less than “40 out of 100” will directly result in “FF”
  - %25 – Midterm
    - Less than “30 out of 100” will directly result in “FF”
  - %35 – Final
    - Less that “35 out of 100” will directly result in “FF”
- Rules
  - Zero tolerance on cheating

Homework and Projects

- Homework (each to be graded out of 100)
  - 2x Written Analysis
  - 3x Lexer/Parsers in C++
  - 3x Common Lisp Programming
  - 1x Prolog Programming
- You will learn “Common Lisp”
- Rules
  - Hand in on the due date – No late submission will be accepted
  - No cheating allowed
    - Discuss among yourselves but do your own work
    - First offence will result in -100 for that homework, second -200, and third and more will directly result in “FF”

Reading Material

- Book:
  - Concepts of Programming Languages by R. W. Sebesta, Eleventh Edition
  - Chapters will be assigned for reading
- Other reading materials will be provided as needed

Programming Languages

Programming Languages

What's a programming language?

A programming language is an artificial language designed to communicate instructions to a machine, particularly a computer [Wikipedia].

A language is a "conceptual universe" providing a framework for problem-solving and useful concepts and programming methods [Perlis].

How many programming languages are there?

Thousands!

Which one to use?

We'll hopefully have a good idea after completing this course.

Top Programming Languages – IEEE Spectrum

IEEE Top Programming Languages: Design, Methods, and Data Sources

The *IEEE Spectrum* Top Programming Languages app synthesizes 12 metrics from 10 sources to arrive at an overall ranking of language popularity. The sources cover contexts that include social chatter, open-source code production, and job postings.

https://spectrum.ieee.org/top-programming-languages/

Top Programming Languages – IEEE Spectrum

What is Tracked

Starting from a list of over 300 programming languages gathered from GitHub, we looked at the volume of results found on Google when we searched for each one using the template “X programming” where “X” is the name of the language. We filtered out languages that had a very low number of search results and then went through the remaining entries by hand to narrow them down to the most interesting. We labeled each language according to whether or not it finds significant use in one or more of the following categories: Web, mobile, enterprise/desktop, or embedded environments.

Our final set of 52 languages includes names familiar to most computer users, such as Java, stalwarts like Cobol and Fortran, and languages that thrive in niches, like Haskell. We gauged the popularity of each using 11 metrics across 8 sources in the following ways:

Top Programming Languages – IEEE Spectrum

Google Search

We measured the number of hits for each language by using Google’s API to search for the template “X programming.” This number indicates the volume of online information resources about each programming language. We took the measurement in June 2019, so it represents a snapshot of the Web at that particular moment in time. This measurement technique is also used by the oft-cited TIOBE rankings.

Google Trends

We measured the index of each language as reported by Google Trends using the template “X programming” in June 2019. This number indicates the demand for information about the particular language, because Google Trends measures how often people search for the given term. As it measures searching activity rather than information availability, Google Trends can be an early cue to up-and-coming languages. Our methodology here is similar to that of the Popularity of Programming Language (PYPL) ranking.

Top Programming Languages – IEEE Spectrum

- **Twitter**

  We measured the number of hits on Twitter for the template “X programming” for the 12 months ending June 2019 using the Twitter Search API. This number indicates the amount of chatter on social media for the language and reflects the sharing of online resources like news articles or books, as well as physical social activities such as hackathons.

- **GitHub**

  GitHub is a site where programmers can collaboratively store repositories of code. Using the GitHub API and GitHub tags, we measured two things for the 12 months ending June 2019: (1) the number of new repositories created for each language, and (2) the number of active repositories for each language, where “active” means that someone has edited the code in a particular repository. The number of new repositories measures fresh activity around the language, whereas the number of active repositories measures the ongoing interest in developing each language.

- **Stack Overflow**

  Stack Overflow is a popular site where programmers can ask questions about coding. We measured the number of questions posted that mention each language for the 12 months ending June 2019. Each question is tagged with the languages under discussion, and these tags are used to tabulate our measurements using the Stack Exchange API.

- **Reddit**

  Reddit is a news and information site where users post links and comments. On Reddit we measured the number of posts mentioning each of the languages, using the template “X programming” from June 2018 to June 2019 across any subreddit on the site. We collected data using the Reddit API.

Top Programming Languages – IEEE Spectrum

- **Hacker News**

  Hacker News is a news and information site where users post comments and links to news about technology. We measured the number of posts that mentioned each of the languages using the template “X programming” for the 12 months ending June 2019. Just like those used by the websites Topsy, Stack Overflow, and Reddit, this metric also captures social activity and information sharing around the various languages. We used the Algolia Search API.

- **CareerBuilder**

  We measured the demand for different programming languages on the CareerBuilder job site. We measure the number of fresh job openings (those that are less than 30 days old) on the U.S. site that mention the language. Because some of the languages we track could be ambiguous in plain text—such as D, Go, J, Processing, and R—we use strict matching of the form “X programming” for these languages. For other languages we use a search string composed of “X AND programming,” which allows us to capture a broader range of relevant postings. We collected data in June 2019 using the CareerBuilder API.

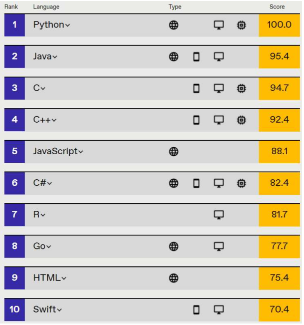
- **IEEE Job Site**

  We measured the demand for different programming languages in job postings on the IEEE Job Site. Because some of the languages we track could be ambiguous in plain text—such as D, Go, J, Processing, and R—we use strict matching of the form “X programming” for these languages. For other languages we use a search string composed of “X AND programming,” which allows us to capture a broader range of relevant postings. Because no externally exposed API exists for the IEEE Job Site, data was extracted using an internal custom-query tool in June 2019.

- **IEEE Xplore Digital Library**

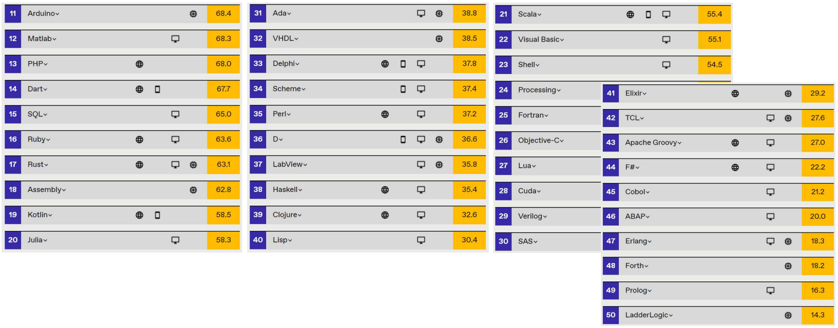
  IEEE maintains a digital library with over 3.6 million conference and journal articles covering a range of scientific and engineering disciplines. We measured the number of articles that mention each of the languages in the template “X programming” for the years 2018 and 2019. This metric captures the prevalence of the different programming languages as used and referenced in scholarship. We collected data using the IEEE Xplore API.

Top 10



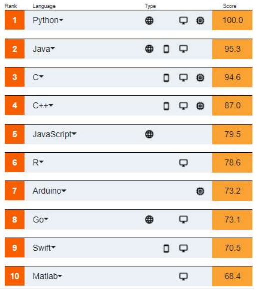
Rank	Language	Type	Score
1	Python	🌐 🖥 ⚙	100.0
2	Java	🌐 📱 🖥	95.4
3	C	📱 🖥 ⚙	94.7
4	C++	📱 🖥 ⚙	92.4
5	JavaScript	🌐	88.1
6	C#	🌐 📱 🖥 ⚙	82.4
7	R	🖥	81.7
8	Go	🌐 🖥	77.7
9	HTML	🌐	75.4
10	Swift	📱 🖥	70.4

Top 10



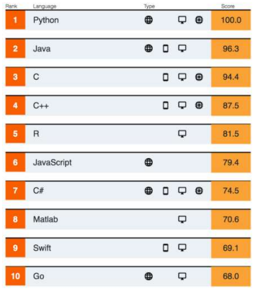
11	Arduino	⚙	68.4
12	Matlab	🖥	68.3
13	PHP	🌐	68.0
14	Dart	🌐 📱	67.7
15	SQL	🖥	65.0
16	Ruby	🌐 🖥	63.6
17	Rust	🌐 🖥 ⚙	63.1
18	Assembly	⚙	62.8
19	Kotlin	🌐 📱	58.5
20	Julia	🖥	58.3
31	Ada	🖥 ⚙	38.8
32	VHDL	⚙	38.5
33	Delphi	🌐 📱 🖥	37.8
34	Scheme	📱 🖥	37.4
35	Perl	🌐 🖥	37.2
36	D	📱 🖥 ⚙	36.6
37	LabView	🖥 ⚙	35.8
38	Haskell	🌐 🖥	35.4
39	Clojure	🌐 🖥	32.6
40	Lisp	🖥	30.4
21	Scala	🌐 📱 🖥	55.4
22	Visual Basic	🖥	55.1
23	Shell	🖥	54.5
24	Processing		
25	Fortran		
26	Objective-C		
27	Lua		
28	Cuda		
29	Verilog		
30	SAS		
41	Elixir	🌐 ⚙	29.2
42	TCL	🖥 ⚙	27.6
43	Apache Groovy	🌐 🖥	27.0
44	F#	🌐 🖥	22.2
45	Cobol	🖥	21.2
46	ABAP	🖥	20.0
47	Erlang	🖥 ⚙	18.3
48	Forth	⚙	18.2
49	Prolog	🖥	16.3
50	LadderLogic	⚙	14.3

Top Programming Languages – IEEE Spectrum



Rank	Language	Type	Score
1	Python	🌐 🖥 ⚙	100.0
2	Java	🌐 📱 🖥	95.3
3	C	📱 🖥 ⚙	94.6
4	C++	📱 🖥 ⚙	87.0
5	JavaScript	🌐	79.5
6	R	🖥	78.6
7	Arduino	⚙	73.2
8	Go	🌐 🖥	73.1
9	Swift	📱 🖥	70.5
10	Matlab	🖥	68.4

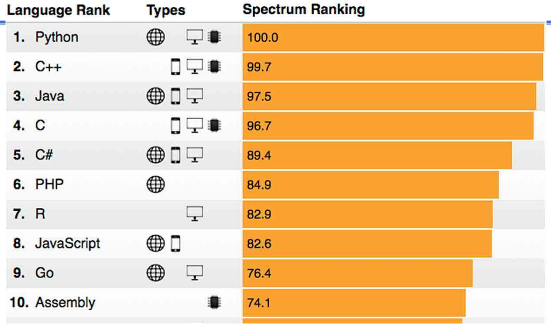
September 2020



Rank	Language	Type	Score
1	Python	🌐 🖥 ⚙	100.0
2	Java	🌐 📱 🖥	96.3
3	C	📱 🖥 ⚙	94.4
4	C++	📱 🖥 ⚙	87.5
5	R	🖥	81.5
6	JavaScript	🌐	79.4
7	C#	🌐 📱 🖥 ⚙	74.5
8	Matlab	🖥	70.6
9	Swift	📱 🖥	69.1
10	Go	🌐 🖥	68.0

September 2019

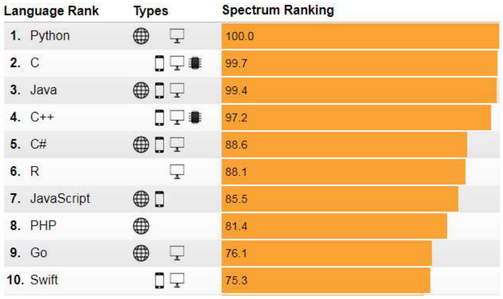
Top Programming Languages – IEEE Spectrum



Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥 ⚙	100.0
2. C++	📱 🖥 ⚙	99.7
3. Java	🌐 📱 🖥	97.5
4. C	📱 🖥 ⚙	96.7
5. C#	🌐 📱 🖥	89.4
6. PHP	🌐	84.9
7. R	🖥	82.9
8. JavaScript	🌐 📱	82.6
9. Go	🌐 🖥	76.4
10. Assembly	⚙	74.1

September 2018

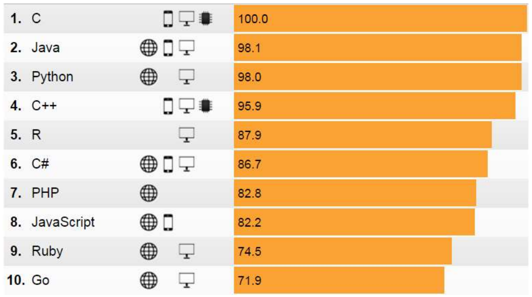
Top Programming Languages – IEEE Spectrum



Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥	100.0
2. C	📱 🖥 ▣	99.7
3. Java	🌐 📱 🖥	99.4
4. C++	📱 🖥 ▣	97.2
5. C#	🌐 📱 🖥	88.6
6. R	🖥	88.1
7. JavaScript	🌐 📱	85.5
8. PHP	🌐	81.4
9. Go	🌐 🖥	76.1
10. Swift	📱 🖥	75.3

September 2017

Top Programming Languages – IEEE Spectrum



1. C	📱 🖥 ▣	100.0
2. Java	🌐 📱 🖥	98.1
3. Python	🌐 🖥	98.0
4. C++	📱 🖥 ▣	95.9
5. R	🖥	87.9
6. C#	🌐 📱 🖥	86.7
7. PHP	🌐	82.8
8. JavaScript	🌐 📱	82.2
9. Ruby	🌐 🖥	74.5
10. Go	🌐 🖥	71.9

September 2016

Top Programming Languages – TIOBE

TIOBE Index for September 2020

September Headline: Programming Language C++ is doing very well

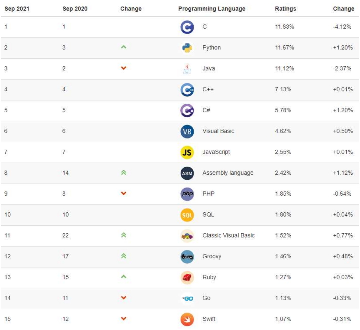
Back in 2003, the programming language C++ was a real winner. It peaked at 17.53% in August 2003, being close to the number #2 position and becoming winner of the programming language award of 2003. From then on C++ went downhill. After 2005 it didn't hit the 10% anymore and in 2017 it scored an all time low of 4.55%. But if compared to last year, C++ is now the fastest growing language of the pack (+1.48%). I think that the new C++20 standard might be one of the main causes for this. Especially because of the new modules feature that is going to replace the dreadful include mechanism. C++ beats other languages with a positive trend such as R (+1.33%) and C# (+1.18%). On the other hand, Java is in real trouble with a loss of -3.18% in comparison to last year. - Paul Jansen, CEO TIOBE Software

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the best programming language or the language in which most lines of code have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

https://www.tiobe.com/tiobe-index/

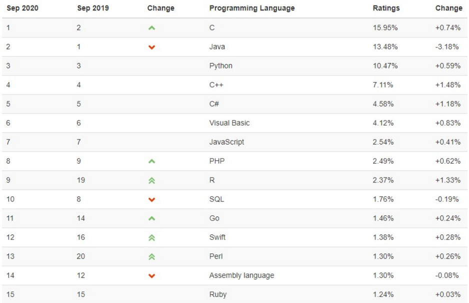
Top Programming Languages – TIOBE



Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		C	11.83%	-4.12%
2	3	^	Python	11.67%	+1.20%
3	2	⌄	Java	11.12%	-2.37%
4	4		C++	7.13%	+0.01%
5	5		C#	5.78%	+1.20%
6	6		Visual Basic	4.62%	+0.50%
7	7		JavaScript	2.55%	+0.01%
8	14	⇈	Assembly language	2.42%	+1.12%
9	8	⌄	PHP	1.85%	-0.64%
10	10		SQL	1.80%	+0.04%
11	22	⇈	Classic Visual Basic	1.52%	+0.77%
12	17	⇈	Groovy	1.46%	+0.48%
13	15	^	Ruby	1.27%	+0.03%
14	11	⌄	Go	1.13%	-0.33%
15	12	⌄	Swift	1.07%	-0.31%

TIOBE Programming Community Index for September 2021

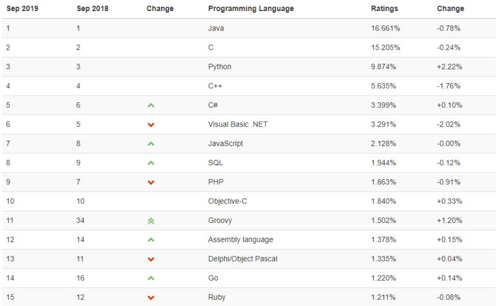
Top Programming Languages – TIOBE



Sep 2020	Sep 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	15.95%	+0.74%
2	1	↓	Java	13.48%	-3.18%
3	3		Python	10.47%	+0.59%
4	4		C++	7.11%	+1.48%
5	5		C#	4.58%	+1.18%
6	6		Visual Basic	4.12%	+0.83%
7	7		JavaScript	2.54%	+0.41%
8	9	↑	PHP	2.49%	+0.62%
9	19	⇈	R	2.37%	+1.33%
10	8	↓	SQL	1.76%	-0.19%
11	14	↑	Go	1.46%	+0.24%
12	16	⇈	Swift	1.38%	+0.28%
13	20	⇈	Perl	1.30%	+0.26%
14	12	↓	Assembly language	1.30%	-0.08%
15	15		Ruby	1.24%	+0.03%

TIOBE Programming Community Index for September 2020

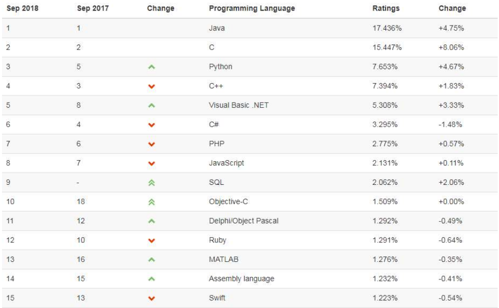
Top Programming Languages – TIOBE



Sep 2019	Sep 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.661%	-0.78%
2	2		C	15.205%	-0.24%
3	3		Python	9.874%	+2.22%
4	4		C++	5.635%	-1.76%
5	6	↑	C#	3.399%	+0.10%
6	5	↓	Visual Basic .NET	3.291%	-2.02%
7	8	↑	JavaScript	2.128%	-0.00%
8	9	↑	SQL	1.944%	-0.12%
9	7	↓	PHP	1.863%	-0.91%
10	10		Objective-C	1.840%	+0.33%
11	34	⇈	Groovy	1.502%	+1.20%
12	14	↑	Assembly language	1.378%	+0.15%
13	11	↓	Delphi/Object Pascal	1.335%	+0.04%
14	16	↑	Go	1.220%	+0.14%
15	12	↓	Ruby	1.211%	-0.08%

TIOBE Programming Community Index for September 2019

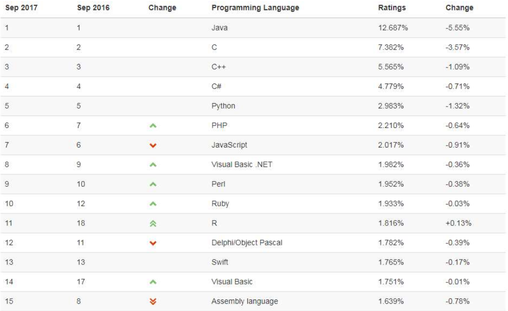
Top Programming Languages – TIOBE



Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	↑	Python	7.653%	+4.67%
4	3	↓	C++	7.394%	+1.83%
5	8	↑	Visual Basic .NET	5.308%	+3.33%
6	4	↓	C#	3.295%	-1.48%
7	6	↓	PHP	2.775%	+0.57%
8	7	↓	JavaScript	2.131%	+0.11%
9	-	⇈	SQL	2.062%	+2.06%
10	18	⇈	Objective-C	1.509%	+0.00%
11	12	↑	Delphi/Object Pascal	1.292%	-0.49%
12	10	↓	Ruby	1.291%	-0.64%
13	16	↑	MATLAB	1.276%	-0.35%
14	15	↑	Assembly language	1.232%	-0.41%
15	13	↓	Swift	1.223%	-0.54%

TIOBE Programming Community Index for September 2018

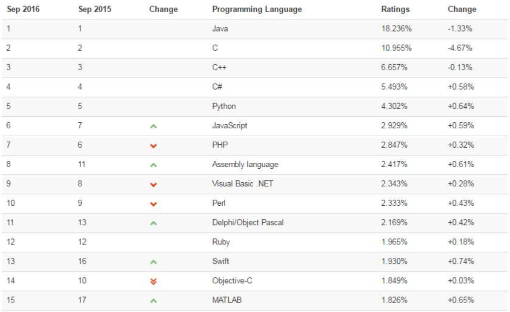
Top Programming Languages – TIOBE



Sep 2017	Sep 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.687%	-5.55%
2	2		C	7.382%	-3.57%
3	3		C++	5.565%	-1.09%
4	4		C#	4.779%	-0.71%
5	5		Python	2.983%	-1.32%
6	7	↑	PHP	2.210%	-0.64%
7	6	↓	JavaScript	2.017%	-0.91%
8	9	↑	Visual Basic .NET	1.982%	-0.36%
9	10	↑	Perl	1.952%	-0.38%
10	12	↑	Ruby	1.933%	-0.03%
11	18	⇈	R	1.816%	+0.13%
12	11	↓	Delphi/Object Pascal	1.782%	-0.39%
13	13		Swift	1.765%	-0.17%
14	17	↑	Visual Basic	1.751%	-0.01%
15	8	⇊	Assembly language	1.639%	-0.78%

TIOBE Programming Community Index for September 2017

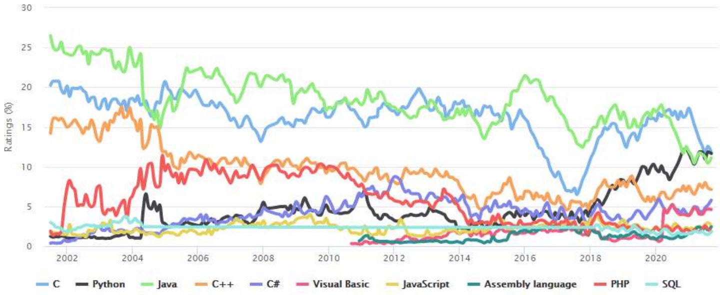
Top Programming Languages – TIOBE



Sep 2016	Sep 2015	Change	Programming Language	Ratings	Change
1	1		Java	18.236%	-1.33%
2	2		C	10.955%	-4.67%
3	3		C++	6.657%	-0.13%
4	4		C#	5.493%	+0.58%
5	5		Python	4.302%	+0.64%
6	7	▲	JavaScript	2.929%	+0.59%
7	6	▼	PHP	2.847%	+0.32%
8	11	▲	Assembly language	2.417%	+0.61%
9	8	▼	Visual Basic .NET	2.343%	+0.28%
10	9	▼	Perl	2.333%	+0.43%
11	13	▲	Delphi/Object Pascal	2.169%	+0.42%
12	12		Ruby	1.965%	+0.18%
13	16	▲	Swift	1.930%	+0.74%
14	10	⏬	Objective-C	1.849%	+0.03%
15	17	▲	MATLAB	1.826%	+0.65%

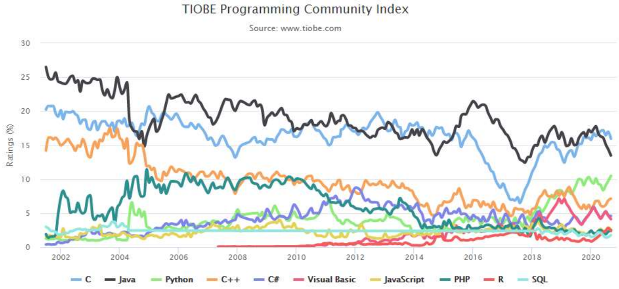
TIOBE Programming Community Index for September 2016

Top Programming Languages – TIOBE



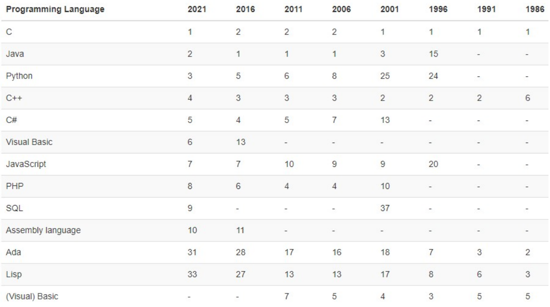
TIOBE Programming Community Index for September 2021

Top Programming Languages – TIOBE



TIOBE Programming Community Index for September 2020

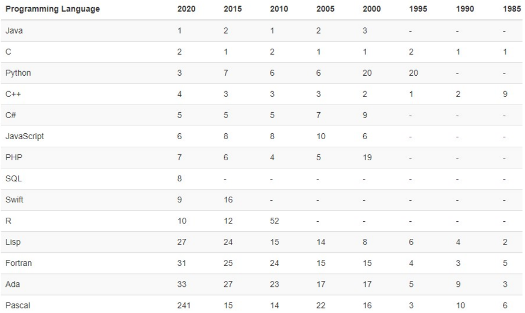
Top Programming Languages – TIOBE



Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Java	2	1	1	1	3	15	-	-
Python	3	5	6	8	25	24	-	-
C++	4	3	3	3	2	2	2	6
C#	5	4	5	7	13	-	-	-
Visual Basic	6	13	-	-	-	-	-	-
JavaScript	7	7	10	9	9	20	-	-
PHP	8	6	4	4	10	-	-	-
SQL	9	-	-	-	37	-	-	-
Assembly language	10	11	-	-	-	-	-	-
Ada	31	28	17	16	18	7	3	2
Lisp	33	27	13	13	17	8	6	3
(Visual) Basic	-	-	7	5	4	3	5	5

TIOBE Programming Community Index for September 2021

Top Programming Languages – TIOBE

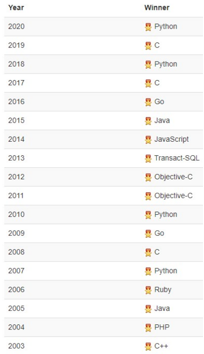


Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	20	20	-	-
C++	4	3	3	3	2	1	2	9
C#	5	5	5	7	9	-	-	-
JavaScript	6	8	8	10	6	-	-	-
PHP	7	6	4	5	19	-	-	-
SQL	8	-	-	-	-	-	-	-
Swift	9	16	-	-	-	-	-	-
R	10	12	52	-	-	-	-	-
Lisp	27	24	15	14	8	6	4	2
Fortran	31	25	24	15	15	4	3	5
Ada	33	27	23	17	17	5	9	3
Pascal	241	15	14	22	16	3	10	6

TIOBE Programming Community Index for September 2020

33

Top Programming Languages – TIOBE

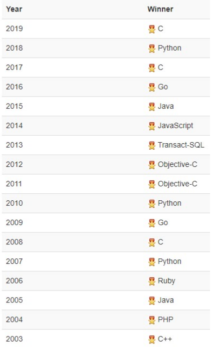


Year	Winner
2020	Python
2019	C
2018	Python
2017	C
2016	Go
2015	Java
2014	JavaScript
2013	Transact-SQL
2012	Objective-C
2011	Objective-C
2010	Python
2009	Go
2008	C
2007	Python
2006	Ruby
2005	Java
2004	PHP
2003	C++

TIOBE Programming Community Index for September 2021

34

Top Programming Languages – TIOBE

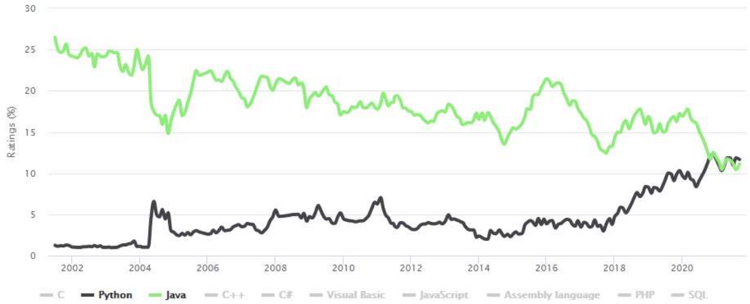


Year	Winner
2019	C
2018	Python
2017	C
2016	Go
2015	Java
2014	JavaScript
2013	Transact-SQL
2012	Objective-C
2011	Objective-C
2010	Python
2009	Go
2008	C
2007	Python
2006	Ruby
2005	Java
2004	PHP
2003	C++

TIOBE Programming Community Index for September 2020

35

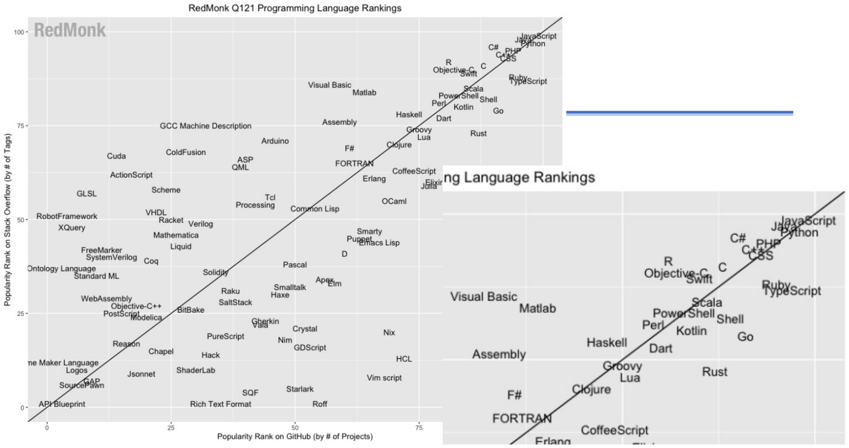
Java or Python

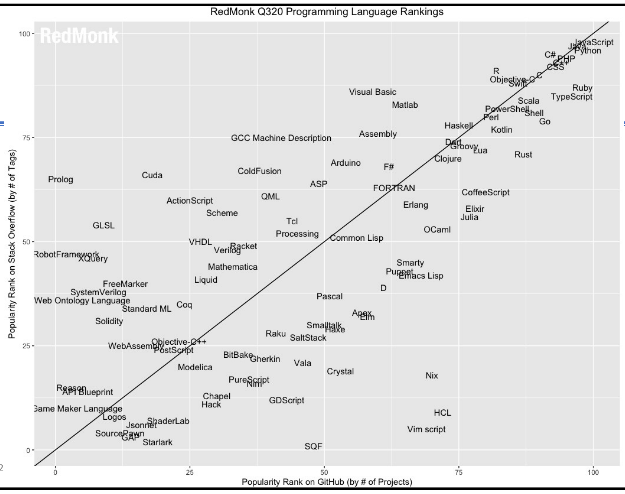


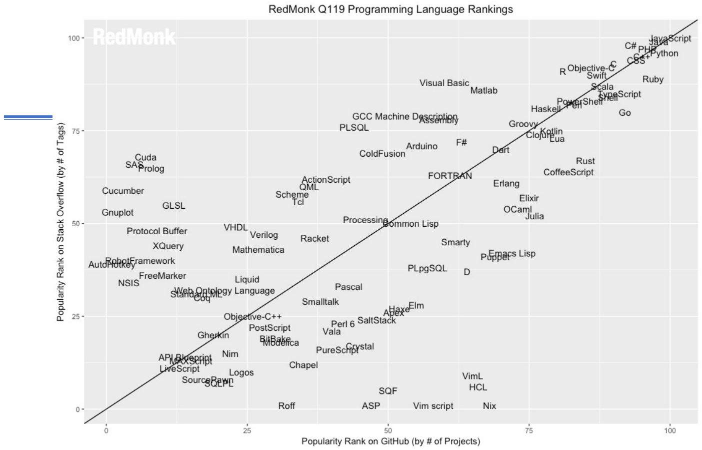
36

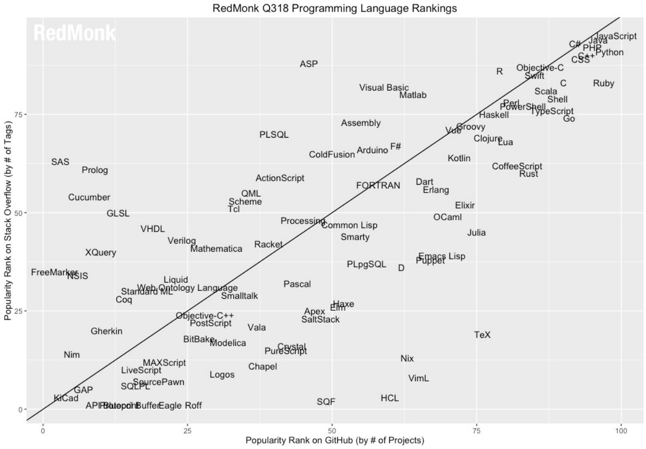
Programming Languages – Popularity

https://redmonk.com/sogrady/2021/03/01/language-rankings-1-21/


RedMonk Q121 Programming Language Rankings


RedMonk Q320 Programming Language Rankings


RedMonk Q119 Programming Language Rankings



Programming Languages – Popularity

1	JavaScript
2	Python
3	Java
4	PHP
5	C#
5	C++
5	CSS
8	TypeScript
9	Ruby
10	C
11	Swift
12	R
13	Objective-C
14	Shell
14	Scala
16	Go
17	PowerShell
18	Kotlin
19	Rust
19	Perl

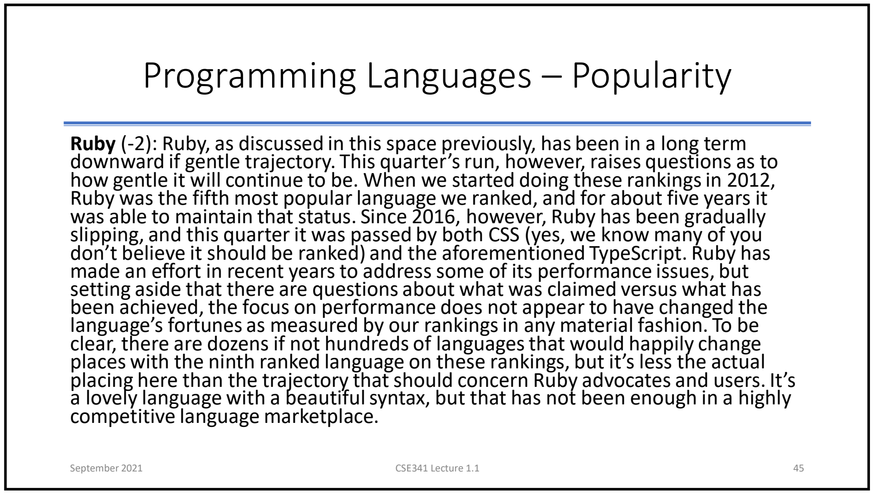
In a contrast to our last run that was fairly static within the Top 20 – not unusual for accretive metrics – this quarter's run featured quite a bit of change and movement. Fully half of the Top 20 experienced a degree of movement, which is very unusual. It's difficult to attribute this definitively to any higher level macro trends, but the data is consistent with an industry that picked the pace back up in the last two quarters of the year after the initial chaos of lockdowns and so on gave way to livable if extremely suboptimal new routines.

Programming Languages – Popularity

Dart (3): Less than three years ago Dart was languishing in the thirties, having shown minimal traction by the proxies for developer interest and activity that we use. Two years after the introduction of the Flutter framework, however, Dart is up another three spots to sit just outside of our Top 20 at 21. This jump comes two quarters after Dart had seemingly stalled – along with Kotlin – raising questions of whether it had peaked. This quarter's run suggests that the answer to that question is no. It seems clear that Flutter has had a material impact on the language's popularity, and clearly its ability to compile to the most popular programming in the world is likewise not hurting it. While it's extremely difficult to merely get to the #21 spot on our rankings – as Rust, among others, can attest – with this quarter's resumption of its upwards trajectory, we can turn our attention back to watching whether Dart can break into the Top 20, and if so what it might displace along the way.

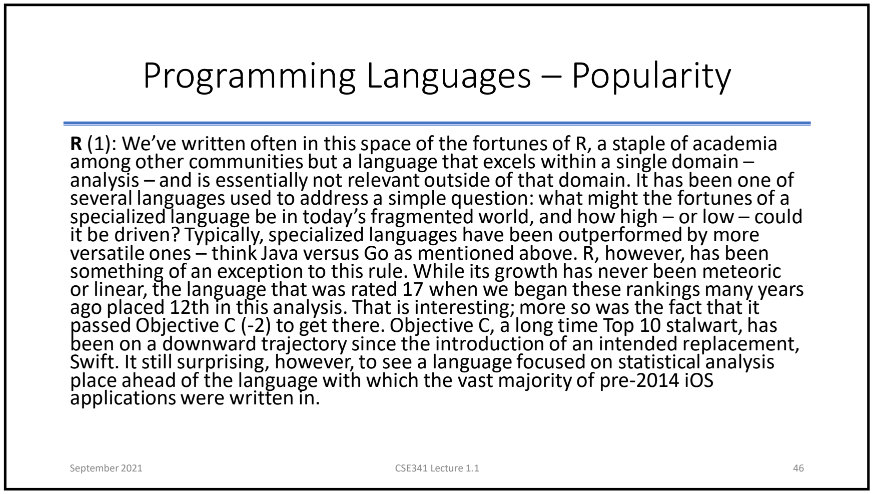
Programming Languages – Popularity

TypeScript (1): Speaking of JavaScript's performance, TypeScript's ascent up our rankings continues. This is impressive on its own rights; the only language in recent memory to penetrate the Top 10 was Swift, but that was for a single quarter and it quickly bounced back out and has remained relatively static since 2018 in 11th place. The initial question facing TypeScript was whether it would be able to hold on. The more appropriate question now is what the language's ultimate ceiling might be. TypeScript moved up for the sixth of its latest eight quarterly rankings, and its popularity is evident when one looks around the industry. Just as interesting as the growth, however, is the language at whose expense the growth comes from.



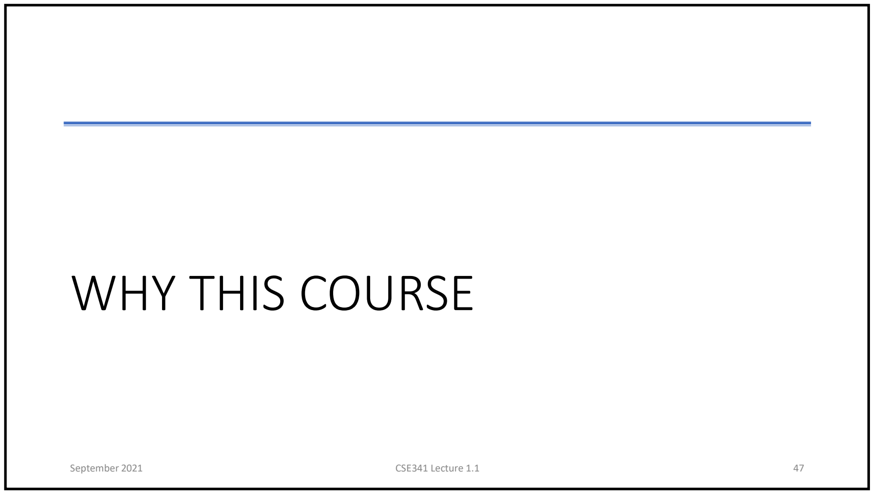
Programming Languages – Popularity

Ruby (-2): Ruby, as discussed in this space previously, has been in a long term downward if gentle trajectory. This quarter's run, however, raises questions as to how gentle it will continue to be. When we started doing these rankings in 2012, Ruby was the fifth most popular language we ranked, and for about five years it was able to maintain that status. Since 2016, however, Ruby has been gradually slipping, and this quarter it was passed by both CSS (yes, we know many of you don't believe it should be ranked) and the aforementioned TypeScript. Ruby has made an effort in recent years to address some of its performance issues, but setting aside that there are questions about what was claimed versus what has been achieved, the focus on performance does not appear to have changed the language's fortunes as measured by our rankings in any material fashion. To be clear, there are dozens if not hundreds of languages that would happily change places with the ninth ranked language on these rankings, but it's less the actual placing here than the trajectory that should concern Ruby advocates and users. It's a lovely language with a beautiful syntax, but that has not been enough in a highly competitive language marketplace.

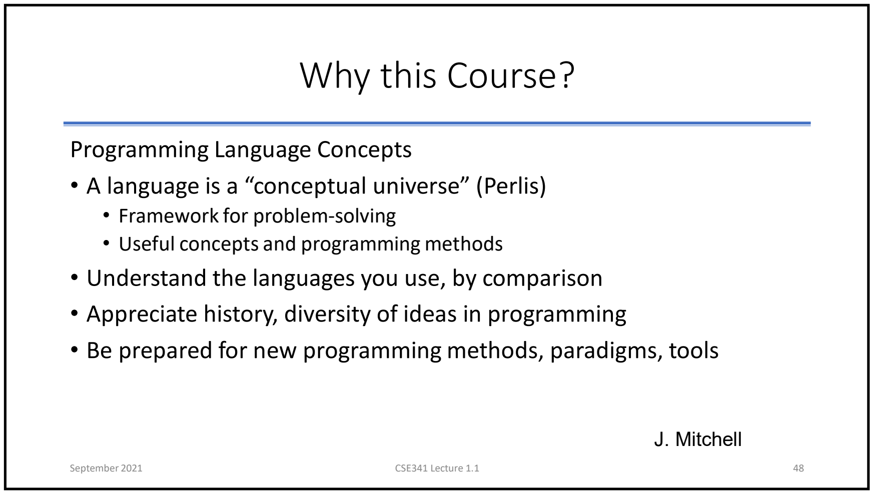


Programming Languages – Popularity

R (1): We've written often in this space of the fortunes of R, a staple of academia among other communities but a language that excels within a single domain – analysis – and is essentially not relevant outside of that domain. It has been one of several languages used to address a simple question: what might the fortunes of a specialized language be in today's fragmented world, and how high – or low – could it be driven? Typically, specialized languages have been outperformed by more versatile ones – think Java versus Go as mentioned above. R, however, has been something of an exception to this rule. While its growth has never been meteoric or linear, the language that was rated 17 when we began these rankings many years ago placed 12th in this analysis. That is interesting; more so was the fact that it passed Objective C (-2) to get there. Objective C, a long time Top 10 stalwart, has been on a downward trajectory since the introduction of an intended replacement, Swift. It still surprising, however, to see a language focused on statistical analysis place ahead of the language with which the vast majority of pre-2014 iOS applications were written in.



WHY THIS COURSE



Why this Course?

Programming Language Concepts

- A language is a "conceptual universe" (Perlis)
  - Framework for problem-solving
  - Useful concepts and programming methods
- Understand the languages you use, by comparison
- Appreciate history, diversity of ideas in programming
- Be prepared for new programming methods, paradigms, tools

J. Mitchell

Why this Course?

Critical thought

- Identify properties of language, not just syntax or sales pitch
- Language and implementation

Every convenience has its cost

- Recognize the cost of presenting an abstract view of machine
- Understand trade-offs in programming language design

J. Mitchell

Trends

Commercial trend over past 5+ years

- Increasing use of type-safe languages: Java, C#, …
- Scripting languages, other languages for web applications

Teaching trends

- Python replaced Java replaced C as most common intro language
    - Less emphasis on how data, control represented in machine
- Objective C

J. Mitchell

Trends

Research and development trends

- Modularity
    - Java, C++: standardization of new module features
- Program analysis
    - Automated error detection, programming env, compilation
- Isolation and security
    - Sandboxing, language-based security, …
- Web 2.0
    - Increasing client-side functionality, mashup isolation problems

J. Mitchell

Example

What is D?

D is the culmination of *decades of experience implementing compilers* for many diverse languages and has a unique set of features:

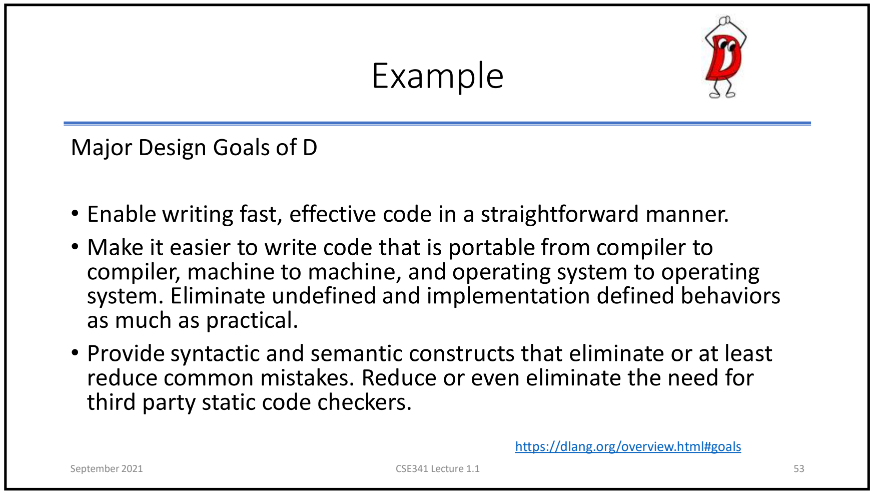
- *high level* constructs for great modeling power
- *high performance*, compiled language
- static typing
- direct interface to the operating system API's and hardware
- blazingly fast compile-times
- memory-safe subset (SafeD)
- *maintainable*, *easy to understand* code
- gradual learning curve (C-like syntax, similar to Java and others)
- compatible with C application binary interface
- limited compatibility with C++ application binary interface
- multi-paradigm (imperative, structured, object oriented, generic, functional programming purity, and even assembly)
- built-in error detection (contracts, unittests)
- ... and many more features.

```
import std.stdio;
import std.algorithm;
import std.range;

void main()
{
   // Let's get going!
   writeln("Hello World!");

   // Take three arrays, and without allocating any new
   // memory, sort across all the arrays in-place
   int[] arr1 = [4, 9, 7];
   int[] arr2 = [5, 2, 1, 10];
   int[] arr3 = [6, 8, 3];
   sort(chain(arr1, arr2, arr3));
   writeln("%s\n%s\n%s\n", arr1, arr2, arr3);
}
```

https://tour.dlang.org/



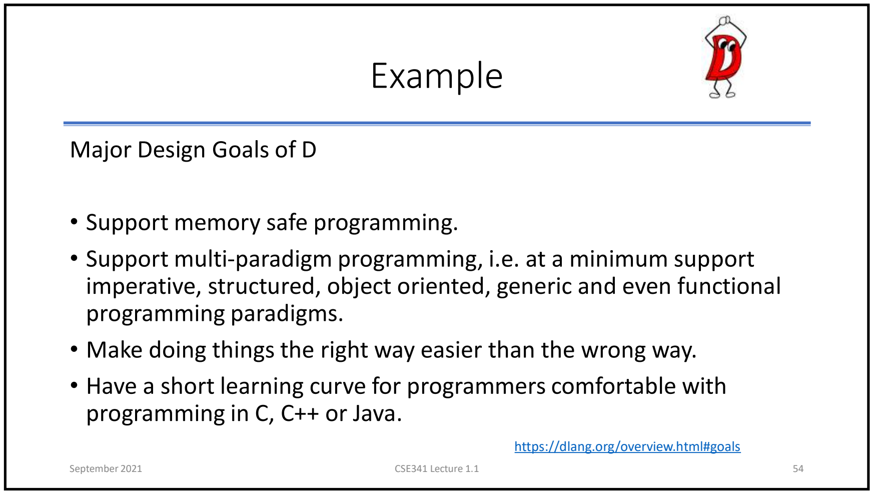
Example

Major Design Goals of D

- Enable writing fast, effective code in a straightforward manner.
- Make it easier to write code that is portable from compiler to compiler, machine to machine, and operating system to operating system. Eliminate undefined and implementation defined behaviors as much as practical.
- Provide syntactic and semantic constructs that eliminate or at least reduce common mistakes. Reduce or even eliminate the need for third party static code checkers.

https://dlang.org/overview.html#goals

53



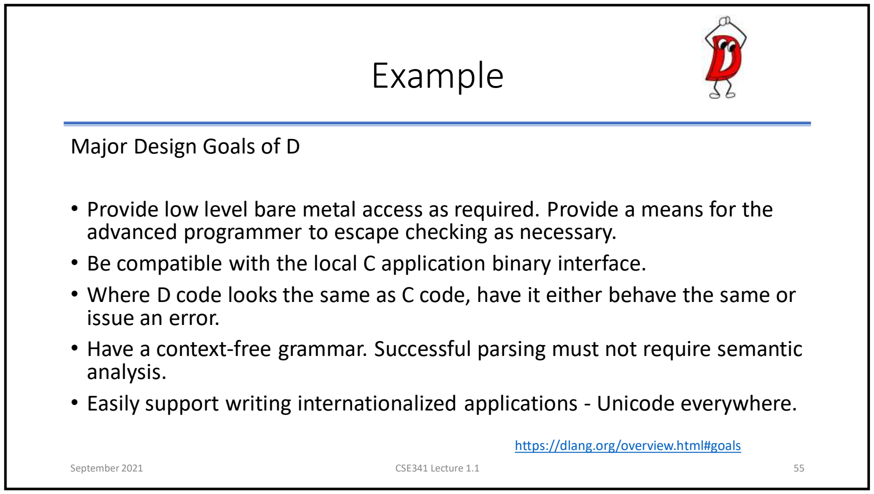
Example

Major Design Goals of D

- Support memory safe programming.
- Support multi-paradigm programming, i.e. at a minimum support imperative, structured, object oriented, generic and even functional programming paradigms.
- Make doing things the right way easier than the wrong way.
- Have a short learning curve for programmers comfortable with programming in C, C++ or Java.

https://dlang.org/overview.html#goals

54



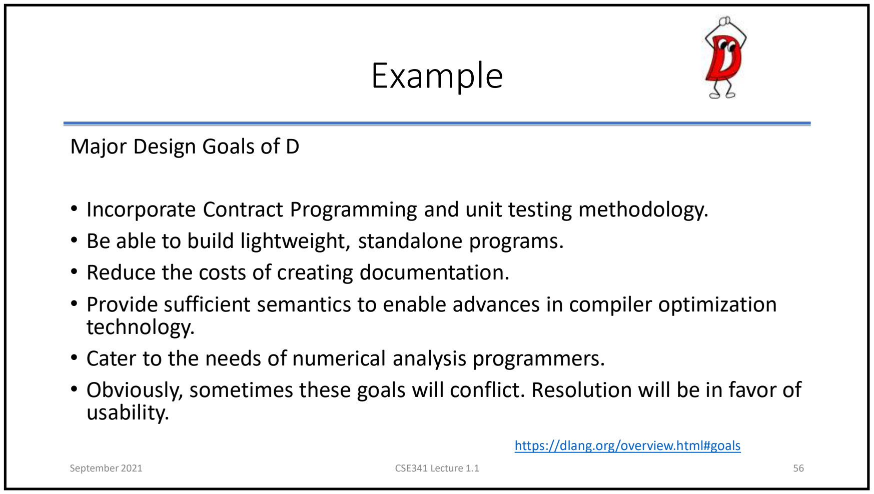
Example

Major Design Goals of D

- Provide low level bare metal access as required. Provide a means for the advanced programmer to escape checking as necessary.
- Be compatible with the local C application binary interface.
- Where D code looks the same as C code, have it either behave the same or issue an error.
- Have a context-free grammar. Successful parsing must not require semantic analysis.
- Easily support writing internationalized applications - Unicode everywhere.

https://dlang.org/overview.html#goals

55



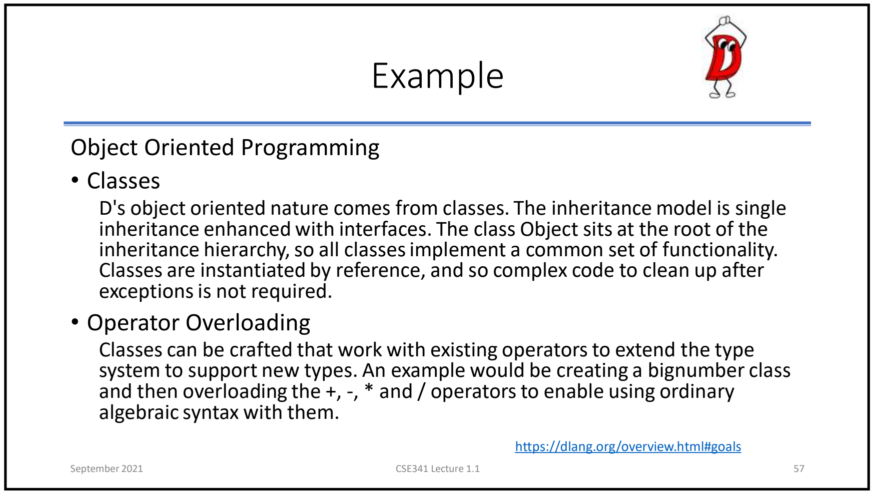
Example

Major Design Goals of D

- Incorporate Contract Programming and unit testing methodology.
- Be able to build lightweight, standalone programs.
- Reduce the costs of creating documentation.
- Provide sufficient semantics to enable advances in compiler optimization technology.
- Cater to the needs of numerical analysis programmers.
- Obviously, sometimes these goals will conflict. Resolution will be in favor of usability.

https://dlang.org/overview.html#goals

56



Example

Object Oriented Programming

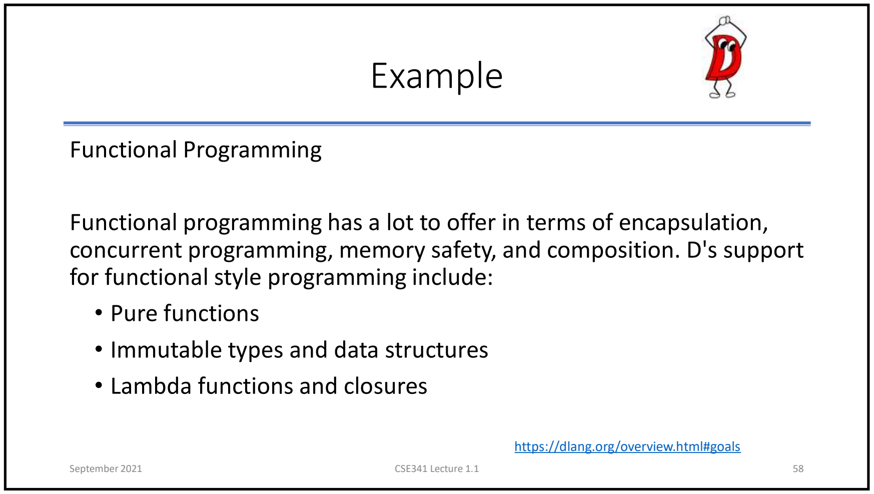
- Classes

  D's object oriented nature comes from classes. The inheritance model is single inheritance enhanced with interfaces. The class Object sits at the root of the inheritance hierarchy, so all classes implement a common set of functionality. Classes are instantiated by reference, and so complex code to clean up after exceptions is not required.

- Operator Overloading

  Classes can be crafted that work with existing operators to extend the type system to support new types. An example would be creating a bignumber class and then overloading the +, -, * and / operators to enable using ordinary algebraic syntax with them.

https://dlang.org/overview.html#goals



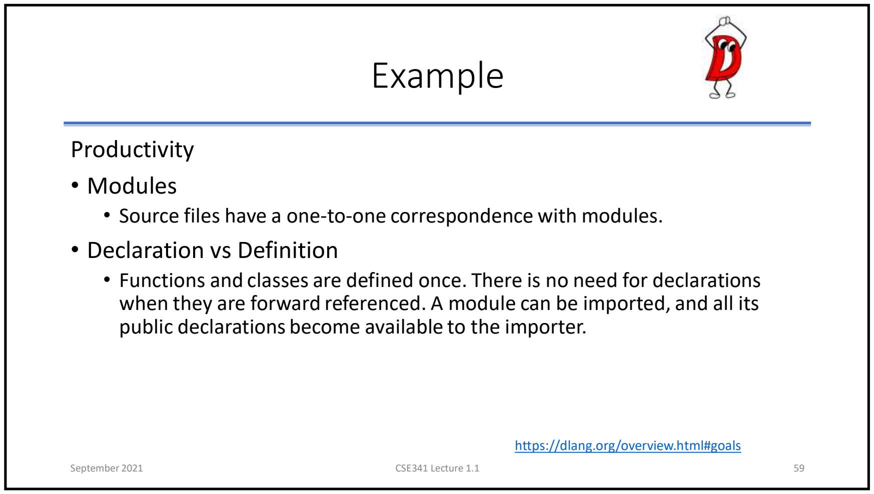
Example

Functional Programming

Functional programming has a lot to offer in terms of encapsulation, concurrent programming, memory safety, and composition. D's support for functional style programming include:

- Pure functions
- Immutable types and data structures
- Lambda functions and closures

https://dlang.org/overview.html#goals

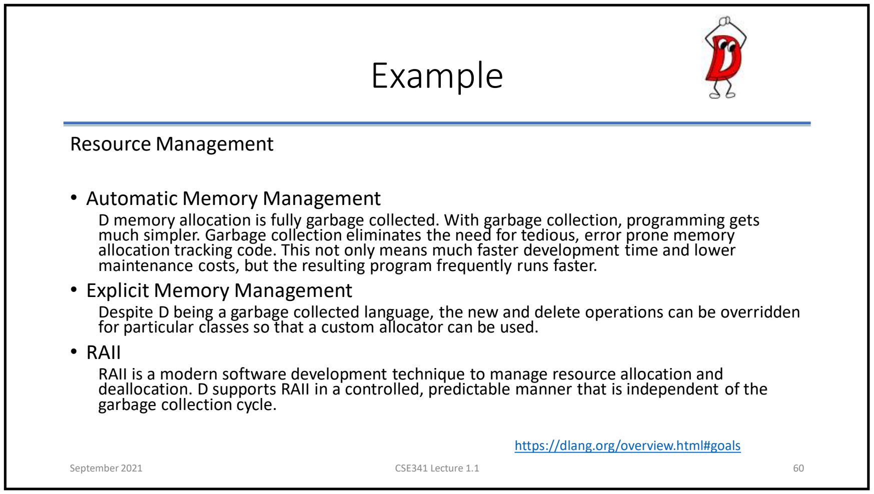


Example

Productivity

- Modules
  - Source files have a one-to-one correspondence with modules.
- Declaration vs Definition
  - Functions and classes are defined once. There is no need for declarations when they are forward referenced. A module can be imported, and all its public declarations become available to the importer.

https://dlang.org/overview.html#goals



Example

Resource Management

- Automatic Memory Management

  D memory allocation is fully garbage collected. With garbage collection, programming gets much simpler. Garbage collection eliminates the need for tedious, error prone memory allocation tracking code. This not only means much faster development time and lower maintenance costs, but the resulting program frequently runs faster.

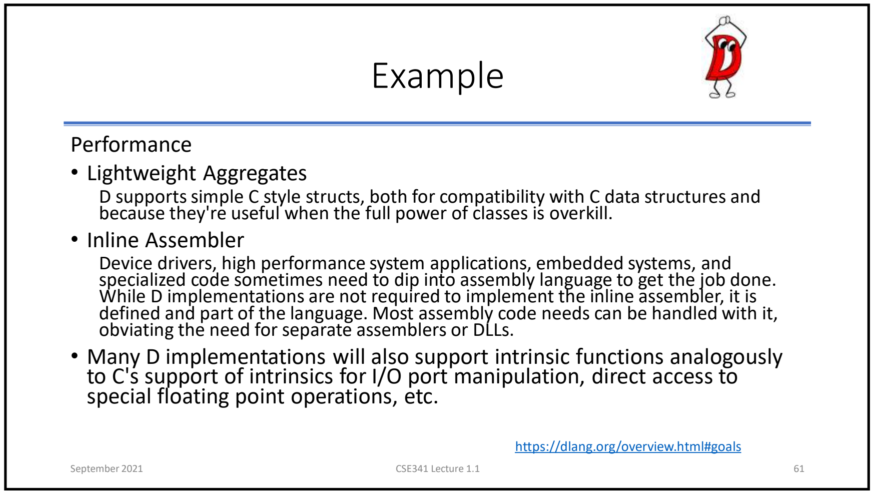
- Explicit Memory Management

  Despite D being a garbage collected language, the new and delete operations can be overridden for particular classes so that a custom allocator can be used.

- RAII

  RAII is a modern software development technique to manage resource allocation and deallocation. D supports RAII in a controlled, predictable manner that is independent of the garbage collection cycle.

https://dlang.org/overview.html#goals



Example

Performance

- Lightweight Aggregates

  D supports simple C style structs, both for compatibility with C data structures and because they're useful when the full power of classes is overkill.

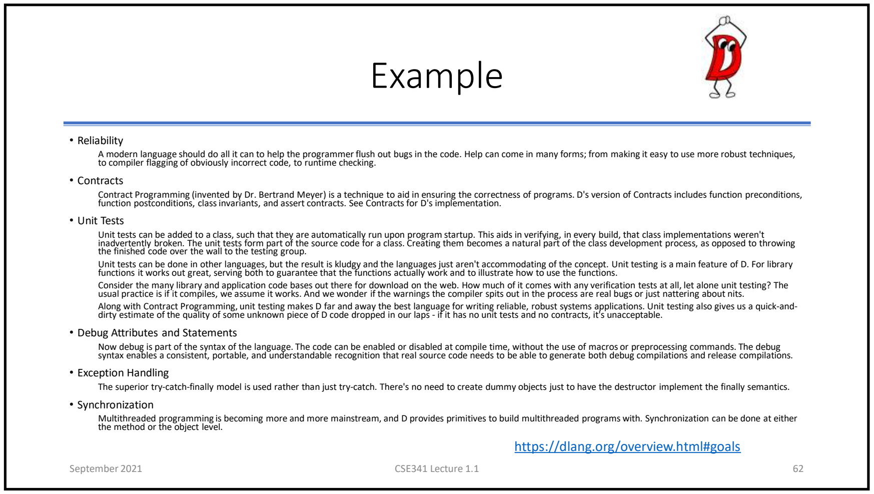
- Inline Assembler

  Device drivers, high performance system applications, embedded systems, and specialized code sometimes need to dip into assembly language to get the job done. While D implementations are not required to implement the inline assembler, it is defined and part of the language. Most assembly code needs can be handled with it, obviating the need for separate assemblers or DLLs.

- Many D implementations will also support intrinsic functions analogously to C's support of intrinsics for I/O port manipulation, direct access to special floating point operations, etc.

https://dlang.org/overview.html#goals

61



Example

- Reliability

  A modern language should do all it can to help the programmer flush out bugs in the code. Help can come in many forms; from making it easy to use more robust techniques, to compiler flagging of obviously incorrect code, to runtime checking.

- Contracts

  Contract Programming (invented by Dr. Bertrand Meyer) is a technique to aid in ensuring the correctness of programs. D's version of Contracts includes function preconditions, function postconditions, class invariants, and assert contracts. See Contracts for D's implementation.

- Unit Tests

  Unit tests can be added to a class, such that they are automatically run upon program startup. This aids in verifying, in every build, that class implementations weren't inadvertently broken. The unit tests form part of the source code for a class. Creating them becomes a natural part of the class development process, as opposed to throwing the finished code over the wall to the testing group.

  Unit tests can be done in other languages, but the result is kludgy and the languages just aren't accommodating of the concept. Unit testing is a main feature of D. For library functions it works out great, serving both to guarantee that the functions actually work and to illustrate how to use the functions.

  Consider the many library and application code bases out there for download on the web. How much of it comes with any verification tests at all, let alone unit testing? The usual practice is if it compiles, we assume it works. And we wonder if the warnings the compiler spits out in the process are real bugs or just nattering about nits.

  Along with Contract Programming, unit testing makes D far and away the best language for writing reliable, robust systems applications. Unit testing also gives us a quick-and-dirty estimate of the quality of some unknown piece of D code dropped in our laps - if it has no unit tests and no contracts, it's unacceptable.

- Debug Attributes and Statements

  Now debug is part of the syntax of the language. The code can be enabled or disabled at compile time, without the use of macros or preprocessing commands. The debug syntax enables a consistent, portable, and understandable recognition that real source code needs to be able to generate both debug compilations and release compilations.

- Exception Handling

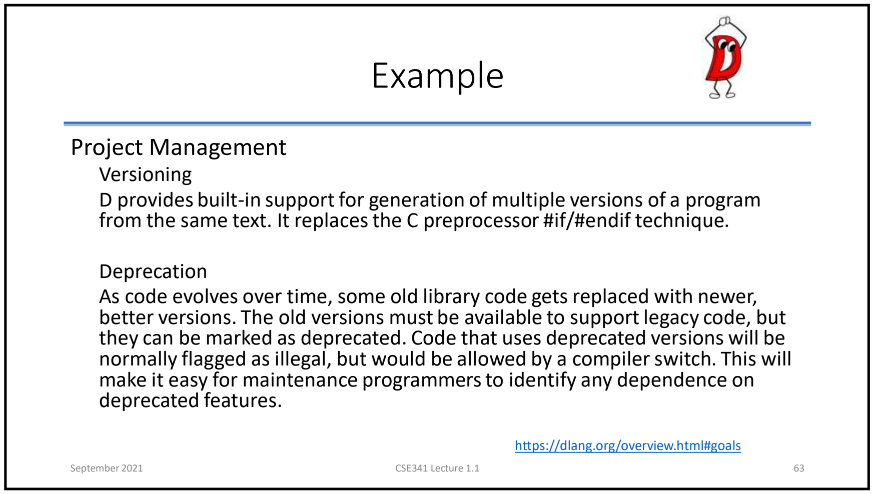
  The superior try-catch-finally model is used rather than just try-catch. There's no need to create dummy objects just to have the destructor implement the finally semantics.

- Synchronization

  Multithreaded programming is becoming more and more mainstream, and D provides primitives to build multithreaded programs with. Synchronization can be done at either the method or the object level.

https://dlang.org/overview.html#goals

62



Example

Project Management

Versioning

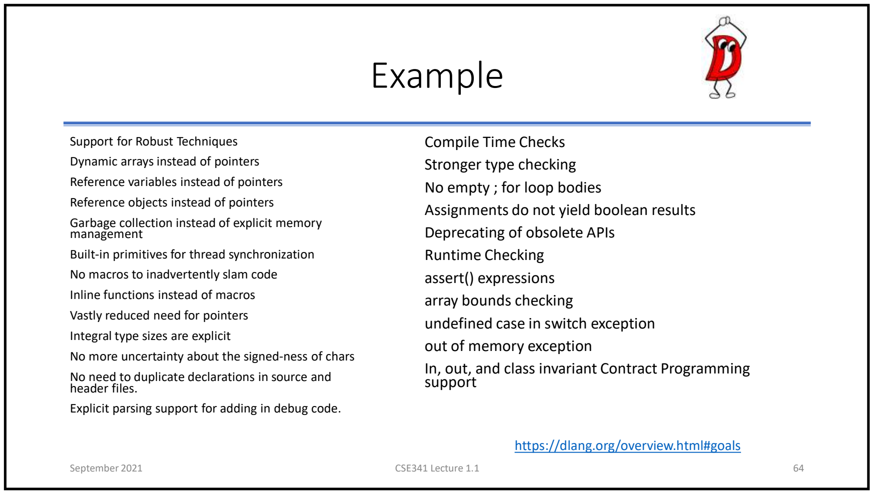
D provides built-in support for generation of multiple versions of a program from the same text. It replaces the C preprocessor #if/#endif technique.

Deprecation

As code evolves over time, some old library code gets replaced with newer, better versions. The old versions must be available to support legacy code, but they can be marked as deprecated. Code that uses deprecated versions will be normally flagged as illegal, but would be allowed by a compiler switch. This will make it easy for maintenance programmers to identify any dependence on deprecated features.

https://dlang.org/overview.html#goals

63



Example

Support for Robust Techniques

Dynamic arrays instead of pointers

Reference variables instead of pointers

Reference objects instead of pointers

Garbage collection instead of explicit memory management

Built-in primitives for thread synchronization

No macros to inadvertently slam code

Inline functions instead of macros

Vastly reduced need for pointers

Integral type sizes are explicit

No more uncertainty about the signed-ness of chars

No need to duplicate declarations in source and header files.

Explicit parsing support for adding in debug code.

Compile Time Checks

Stronger type checking

No empty ; for loop bodies

Assignments do not yield boolean results

Deprecating of obsolete APIs

Runtime Checking

assert() expressions

array bounds checking

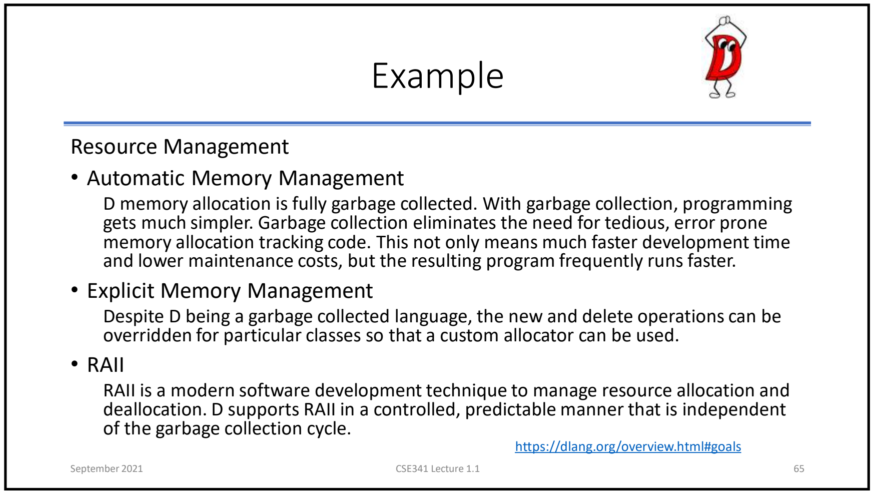
undefined case in switch exception

out of memory exception

In, out, and class invariant Contract Programming support

https://dlang.org/overview.html#goals

64



Example

Resource Management

- Automatic Memory Management

  D memory allocation is fully garbage collected. With garbage collection, programming gets much simpler. Garbage collection eliminates the need for tedious, error prone memory allocation tracking code. This not only means much faster development time and lower maintenance costs, but the resulting program frequently runs faster.

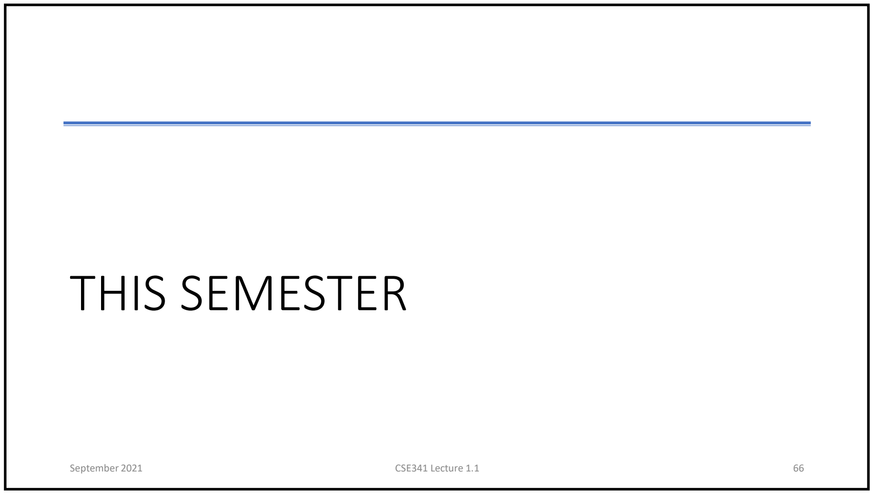
- Explicit Memory Management

  Despite D being a garbage collected language, the new and delete operations can be overridden for particular classes so that a custom allocator can be used.

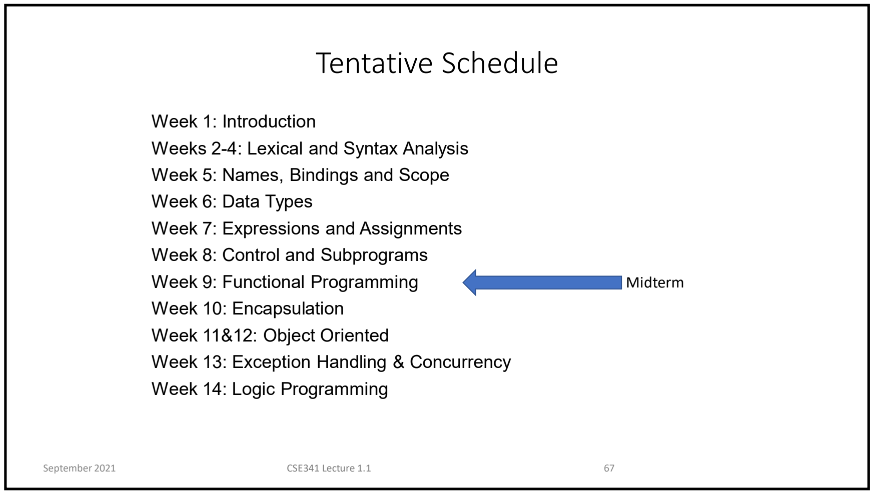
- RAII

  RAII is a modern software development technique to manage resource allocation and deallocation. D supports RAII in a controlled, predictable manner that is independent of the garbage collection cycle.

https://dlang.org/overview.html#goals



THIS SEMESTER



Tentative Schedule

Week 1: Introduction
Weeks 2-4: Lexical and Syntax Analysis
Week 5: Names, Bindings and Scope
Week 6: Data Types
Week 7: Expressions and Assignments
Week 8: Control and Subprograms
Week 9: Functional Programming ← Midterm
Week 10: Encapsulation
Week 11&12: Object Oriented
Week 13: Exception Handling & Concurrency
Week 14: Logic Programming

Thank you for listening!