A New Computing Paradigm

C.D. Jones

While Object Oriented Programming provides us with a black box to work with, Physically Oriented Programming (POP) provides us with a physical black box to work with in space to manipulate.

An example is stream modeling and using it to solve deadlock/livelock problems.

flow

k

Stream a

m

Stream c

l

l

Stream b

So, k is an object in stream a’s flow.

L is an object in stream b’s flow

M is an object in stream m’s low

Our goal, in this case, is to have individual streams that represent portions of code, traditionally objects or in the case of multiple streams to act like individual threads per stream. K,m, and l represent the progress of the application. The unique thing about this example of Physically Oriented Programming is that each stream has its own set of swimmers. A stream isn’t just a set of instructions, it also has these objects K,M, and L which are objecfts in themselves that are modified by the operations of the stream.

Each flow indicates an object or series objects and origin and direction like a vector in mathematics.

New Language - cross between Haskell, LISP, and ALGOL Family

Charter 1 - NO more excessive nulls and null exceptions

Charter 2 - A more elegant form for try, catch, finally forms

Charter 3 - Keep Java’s format for implementation and class definition in one file

Charter 4 - Keep Java’s root “Object” class

Charter 5 - Improve and make easier “double arrays” or and other methods of declaring arrays. Should be able to define any type of array in one line of code

Charter 6 - Include iterators

Charter 7 - Include garbage collection

Charter 8 - Build in OO paradigms

Charter 9 - Run like a script (Python, LISP,etc.) but compiles intermediate code like Java

Charter 10 - Clearer error messages when an error is thrown

Charter 11 - For convenience purposes, you may mark in a function definition what functions call it.

COMMON FEATURES IN A PROG LANGUAGE

Every programming language has core features. These features have evolved over a period, depending on the purpose the languages were created for and the market they targeted. All programming languages have common core set of common features. Implementation of these core set of features varies from language to language. The history of the language will give us an idea of the market the languages were intended for.

Here is a list of the common features:

A place for storing data. Arrays are advanced storing data facility. Also known as data structures.

Rules for writing programs in that programming language

Control statements – which are building blocks for logic implementation.

Most programming languages of today support OOPs. So, constructs to implement like features Class declaration, objects, inheritance, polymorphism and constructors are included.

Every language has operators. Operators are used execute mathematical operations.

All languages include facility to write programs, functions and procedures. Incidentally, this is the place where you write your programs.

Functions return values after execution, whereas procedures simply execute programs.

All programs include facility to write libraries. Libraries are themselves programs, which can be used in other programs.

All languages support exception handling. This feature is helpful to identify errors and generate appropriate messages.

All languages include built in functionalities, provided as classes and functions. These classes help to write better programs.

All languages include a compiler and memory handling features. These are implemented in different ways by the person (s) who have developed the language.

New Language that has pointers, but aren't messy

“Entity Bean” style pool

Don’t have to worry about shallow or deep copy

String equality is possible

In method definition, can use ret keyword to declare more than one return value

sample definition:

public int HelloWorld(String input, input2) : ret (int output1, int output2)

{

return (1,2);

}

Implements access modifiers like java and c# of both classes and methods

implements interface and abstract class like java

public class Main implements MyInterface extends MyClass

MAKE RECURSION EASIER (INCLUDING TAIL END)

Web services like combination between modules

Easily nest types within types

Easy to use multidimensional arrays

Better null pointer handling

Automatic string conversion for output/input purposes (like Java toString)

Doubly having primitives and classes of primitives like Java

Garbage Collection is a plus

HOW ABOUT STRING TYPE AS THE BASE?? (instead of list)

Object->String->Float->Integer

Object->String->Char Sequence->Char

**What is the single worst feature in any programming language?**

Special consideration should be given to ones that are not needed in any language (hiding/not hiding stuff does not count)

Want Answers**49**

**22 ANSWERS**

[](http://www.quora.com/Tikhon-Jelvis)

**[Tikhon Jelvis](http://www.quora.com/Tikhon-Jelvis)**, studied programming languages and did... (more)

152 upvotes by [Rob St. Amant](http://www.quora.com/Rob-St-Amant), Anonymous, [Dylan Hirshkowitz](http://www.quora.com/Dylan-Hirshkowitz), (more)

Intercal's COME FROM is a good candidate: it's the inverse of a GOTO. It takes the execution state from a specified line to here. Of course, Intercal is a joke language, so it's not entirely fair to look at features from there.

Intercal is actually a great case study of bad language design. It's bad on purpose which makes it funny rather than tragic. On this note, Google's [INTERCAL Style Guide](http://cadie.googlecode.com/svn/trunk/INTERCAL-style-guide.html) is a great read.

From real languages, it's really hard to pick a *single* worst feature. There are just so many to choose from :P.

**Nulls** everywhere would be a good candidate thanks to its pervasiveness—almost every single language has them. Some have nulls everywhere except for a blessed set of primitive types, which is even worse (Go, Java). It's Tony Hoare's famous "billion dollar mistake"—although, given how much the tech industry has grown since he said it, without fixing the problem, he was **significantly underestimating its cost**.

So-called "**weak typing**" à la JavaScript and PHP is a bigger problem but also less common. Consider JavaScript:

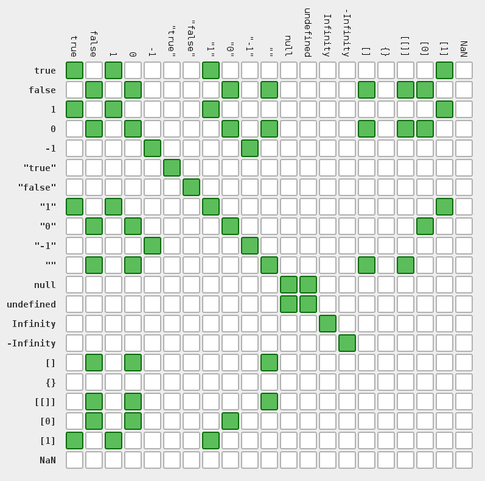
|  |  |
| --- | --- |
| 1  2 | if ("0") { log("a") } else { log("b") } // "a"  if ("0" == true) { log("a") } else { log("b") } // "b" |

Or how about:

|  |  |
| --- | --- |
| 1  2  3 | "0" == false // true  "0" ==[] // true  [] == false // false |

You get the idea: it's absolutely unpredictable. And it doesn't really gain you any meaningful expressiveness.

Just take a look at the JS equality comparison table:



If that doesn't scream "wanton complexity", I don't know what does!

(Taken from [JS Comparison Table](http://dorey.github.io/JavaScript-Equality-Table/), which is worth a look.)

To make life more exciting, PHP also does implicit conversions, but in a different way. In JS:

|  |  |
| --- | --- |
| 1  2 | 1 + "1"  11 |

in PHP:

|  |  |
| --- | --- |
| 1  2 | 1 + "1"  2 |

.

Yeah.

[Written 4 Apr, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Tikhon-Jelvis). 9,015 views.

Upvote**152**

Downvote

Comments**8+**

More Answers Below. **Related Questions**

* [Haskell (programming language): What are generalized algebraic datatypes (GADTs)?](http://www.quora.com/Haskell-programming-language/What-are-generalized-algebraic-datatypes-GADTs)
* [C# (programming language): How can a programmer limit the number of times a particular software feature can been used?](http://www.quora.com/C-programming-language-2/How-can-a-programmer-limit-the-number-of-times-a-particular-software-feature-can-been-used)
* [Programming Languages: What is syntactic sugar?](http://www.quora.com/Programming-Languages/What-is-syntactic-sugar)
* [Programming Languages: Why is C++14 better than C++11? What extra features are added there?](http://www.quora.com/Programming-Languages/Why-is-C++14-better-than-C++11-What-extra-features-are-added-there)
* [C++ (programming language): What are the features of C++ which most programmers don't know?](http://www.quora.com/C++-programming-language/What-are-the-features-of-C++-which-most-programmers-dont-know)

[](http://www.quora.com/Richard-H-Schwartz)

**[Richard H. Schwartz](http://www.quora.com/Richard-H-Schwartz)**, Experienced messaging, collaboration,... (more)

37 upvotes by [Travis Hance](http://www.quora.com/Travis-Hance), [Brian Bi](http://www.quora.com/Brian-Bi), [Ryan Landay](http://www.quora.com/Ryan-Landay), (more)

I have to admit, I've done some warped things in a variety of langauges, but I never did anything quite as warped as using the ALTER statement in COBOL.

Where X and Y are labels, execution of the statement ALTER X TO PROCEED TO Y caused any subsequent branch to label X in a COBOL program to instead go to Y.  The existence of this feature meant that the target of every GO TO was potentially dynamic, with no syntactic clue local to either the branching-from or branching-to lines of code! You had to read the whole thing to know for sure that someone wasn't messing with your flow of control. (More like messing with your mind, IMHO!)

SQL NULL.  Even though you have to support if you're writing a modern database, it introduces a ton of complexity in the engine and a whole lot of semantic badness.

In my last project, we dropped NULL storage support and made our embedded database engine smaller by almost 1/3, because we were able to get rid of "if something is NULL, do something weird, else do something normal" that had to be scattered up and down the code.

We still had NULLs as something that could be fetched so we could support outerjoins, but we didn't allow them to be stored in the database or used with comparisons.  Few users complained once they figured out how to live without them.

GOSUB from the BASIC language of my childhood.  More recent BASICs may have got rid of it, but GOSUB is like a procedure call without the ability to pass arguments.  All variables were global, and all parameter passing was through global variables.  It was *horrible*.

Global variables

when reading code that uses lots of global variables, often there are tons of WTFs "from where this value comes from/where this is set". This problem grows exponentially with number of team members.

Don't use global variables.

MORE VERBOSE ERROR MESSAGES

first, plenty of languages use „error codes" which are so notoriously unpleasant to check, and so easy to omit [and even in exception-aware languages, libraries which keep using error codes are fairly frequent];

- then, even where there are exceptions, they usually are poor (poorly documented and designed classes, lack of information about possible exceptions, limited information available from exception objects, convoluted semantics of corner cases like exception during exception-caused object destruction etc etc)

Unreseerved keywords in C# (and related languages)

|  |  |
| --- | --- |
|  | assertEquality(addFloatingPointNumbers(new FloatingPointSuccessor(new FloatingPointSuccessor(new FloatingPointZero()))), FloatingPointSuccessor(new FloatingPointZero()), new FloatingPointSuccessor(new FloatingPointSuccessor(new FloatingPointSuccessor(new FloatingPointZero())))) |

|  |
| --- |
| ... uh, no. numeric literals are necessary for legibility. arithmetic operators avoid typos and bugs (if \_ / \_ meant actual division, that is!).  1 |

2 + 1 == 3

in Java for example, people don't take this crap for numbers, they take some crap for strings, and they take a lot of crap for every other data-structure, like the ubiquitous lists and dicts.

1

2

3

List<Integer> list = new ArrayList();

list.add(1);

list.add(2);

SHOULD FORCE NAMING CONVENTION

NULL-TERMINATED STRING <- problem

ByRef function arguments.  It shouldn't be possible to mutate state in the caller's context.  Functions should only return values in the normal manner, via the return result.

**SHARE QUESTION**

[Twitter](https://www.twitter.com/intent/tweet?related=quora&text=Question%20on%20%40Quora%3A%20What%20is%20the%20single%20worst%20feature%20in%20any%20programming%20language%3F&url=http%3A//www.quora.com/What-is-the-single-worst-feature-in-any-programming-language)

Facebook

Google+

**RELATED TOPICS**

[Science, Engineering, and Technology](http://www.quora.com/Science-Engineering-and-Technology-1)

[Technology](http://www.quora.com/Technology)

[Software and Applications](http://www.quora.com/Software-and-Applications)

[Computer Programming](http://www.quora.com/Computer-Programming)

[Programming Languages](http://www.quora.com/Programming-Languages)

**What is the single worst feature in any programming language?**

Special consideration should be given to ones that are not needed in any language (hiding/not hiding stuff does not count)

Want Answers**49**

**22 ANSWERS**

[](http://www.quora.com/Tikhon-Jelvis)

**[Tikhon Jelvis](http://www.quora.com/Tikhon-Jelvis)**, studied programming languages and did... (more)

152 upvotes by [Rob St. Amant](http://www.quora.com/Rob-St-Amant), Anonymous, [Dylan Hirshkowitz](http://www.quora.com/Dylan-Hirshkowitz), (more)

Intercal's COME FROM is a good candidate: it's the inverse of a GOTO. It takes the execution state from a specified line to here. Of course, Intercal is a joke language, so it's not entirely fair to look at features from there.

Intercal is actually a great case study of bad language design. It's bad on purpose which makes it funny rather than tragic. On this note, Google's [INTERCAL Style Guide](http://cadie.googlecode.com/svn/trunk/INTERCAL-style-guide.html) is a great read.

From real languages, it's really hard to pick a *single* worst feature. There are just so many to choose from :P.

**Nulls** everywhere would be a good candidate thanks to its pervasiveness—almost every single language has them. Some have nulls everywhere except for a blessed set of primitive types, which is even worse (Go, Java). It's Tony Hoare's famous "billion dollar mistake"—although, given how much the tech industry has grown since he said it, without fixing the problem, he was **significantly underestimating its cost**.

So-called "**weak typing**" à la JavaScript and PHP is a bigger problem but also less common. Consider JavaScript:

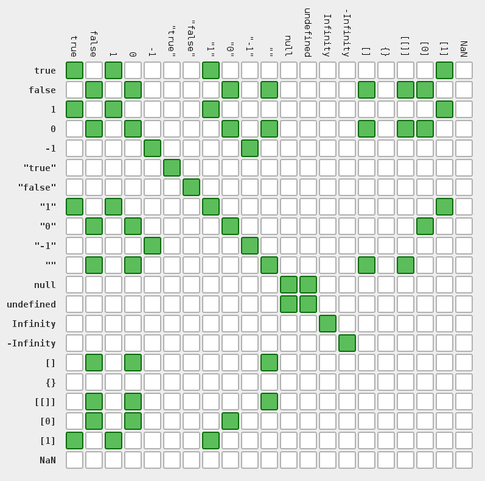
|  |  |
| --- | --- |
| 1  2 | if ("0") { log("a") } else { log("b") } // "a"  if ("0" == true) { log("a") } else { log("b") } // "b" |

Or how about:

|  |  |
| --- | --- |
| 1  2  3 | "0" == false // true  "0" ==[] // true  [] == false // false |

You get the idea: it's absolutely unpredictable. And it doesn't really gain you any meaningful expressiveness.

Just take a look at the JS equality comparison table:



If that doesn't scream "wanton complexity", I don't know what does!

(Taken from [JS Comparison Table](http://dorey.github.io/JavaScript-Equality-Table/), which is worth a look.)

To make life more exciting, PHP also does implicit conversions, but in a different way. In JS:

|  |  |
| --- | --- |
| 1  2 | 1 + "1"  11 |

in PHP:

|  |  |
| --- | --- |
| 1  2 | 1 + "1"  2 |

.

Yeah.

[Written 4 Apr, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Tikhon-Jelvis). 9,015 views.

Upvote**152**

Downvote

Comments**8+**

More Answers Below. **Related Questions**

* [Haskell (programming language): What are generalized algebraic datatypes (GADTs)?](http://www.quora.com/Haskell-programming-language/What-are-generalized-algebraic-datatypes-GADTs)
* [C# (programming language): How can a programmer limit the number of times a particular software feature can been used?](http://www.quora.com/C-programming-language-2/How-can-a-programmer-limit-the-number-of-times-a-particular-software-feature-can-been-used)
* [Programming Languages: What is syntactic sugar?](http://www.quora.com/Programming-Languages/What-is-syntactic-sugar)
* [Programming Languages: Why is C++14 better than C++11? What extra features are added there?](http://www.quora.com/Programming-Languages/Why-is-C++14-better-than-C++11-What-extra-features-are-added-there)
* [C++ (programming language): What are the features of C++ which most programmers don't know?](http://www.quora.com/C++-programming-language/What-are-the-features-of-C++-which-most-programmers-dont-know)

[](http://www.quora.com/Richard-H-Schwartz)

**[Richard H. Schwartz](http://www.quora.com/Richard-H-Schwartz)**, Experienced messaging, collaboration,... (more)

37 upvotes by [Travis Hance](http://www.quora.com/Travis-Hance), [Brian Bi](http://www.quora.com/Brian-Bi), [Ryan Landay](http://www.quora.com/Ryan-Landay), (more)

I have to admit, I've done some warped things in a variety of langauges, but I never did anything quite as warped as using the ALTER statement in COBOL.

Where X and Y are labels, execution of the statement ALTER X TO PROCEED TO Y caused any subsequent branch to label X in a COBOL program to instead go to Y.  The existence of this feature meant that the target of every GO TO was potentially dynamic, with no syntactic clue local to either the branching-from or branching-to lines of code! You had to read the whole thing to know for sure that someone wasn't messing with your flow of control. (More like messing with your mind, IMHO!)

Fortunately, most programmers had the good sense to avoid using ALTER, but not everyone.

[Written 6 May, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Richard-H-Schwartz). 1,738 views.

Upvote**37**

Downvote

Comments**3+**

[](http://www.quora.com/Greg-Kemnitz)

**[Greg Kemnitz](http://www.quora.com/Greg-Kemnitz)**, wrote the innards of "the world's sma... (more)

12 upvotes by [Tikhon Jelvis](http://www.quora.com/Tikhon-Jelvis), [Mostafa Hany Gomaa](http://www.quora.com/Mostafa-Hany-Gomaa), [Rohit Shinde](http://www.quora.com/Rohit-Shinde-2), (more)

SQL NULL.  Even though you have to support if you're writing a modern database, it introduces a ton of complexity in the engine and a whole lot of semantic badness.

In my last project, we dropped NULL storage support and made our embedded database engine smaller by almost 1/3, because we were able to get rid of "if something is NULL, do something weird, else do something normal" that had to be scattered up and down the code.

We still had NULLs as something that could be fetched so we could support outerjoins, but we didn't allow them to be stored in the database or used with comparisons.  Few users complained once they figured out how to live without them.

[Written 30 Aug, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Greg-Kemnitz). 914 views.

Upvote**12**

Downvote

Comment

[](http://www.quora.com/Paul-Bone-1)

**[Paul Bone](http://www.quora.com/Paul-Bone-1)**, Software Engineer, Specialising in pr... (more)

20 upvotes by [Greg Kemnitz](http://www.quora.com/Greg-Kemnitz), [Peter Schachte](http://www.quora.com/Peter-Schachte), [Toby Thain](http://www.quora.com/Toby-Thain), (more)

Tony Hoare (a very notable computer scientist)  invented the NULL reference (NULL pointer) in 1965.  He later described this as his Billion Dollar Mistake.  Non-nullable references, such as in Ada or in languages with discriminated union type systems prevent the programmer from assuming data will be available when either that assumption is incorrect or the data was not provided, both causing software to crash and placing extra responsibility on the programmer.

The worst thing about this mistake is that many many languages have copied it, and we're \_still\_ making this mistake in new languages and it continues to cost us time effort and money.

[http://lambda-the-ultimate.org/n...](http://lambda-the-ultimate.org/node/3186)

[Written 9 Oct, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Paul-Bone-1). 2,892 views.

Upvote**20**

Downvote

Comments**3+**

[](http://www.quora.com/Peter-Schachte)

**[Peter Schachte](http://www.quora.com/Peter-Schachte)**, I teach functional, logic, and object... (more)

6 upvotes by [Tikhon Jelvis](http://www.quora.com/Tikhon-Jelvis), Quora User, [Johan Bjäreholt](http://www.quora.com/Johan-Bj%C3%A4reholt), (more)

GOSUB from the BASIC language of my childhood.  More recent BASICs may have got rid of it, but GOSUB is like a procedure call without the ability to pass arguments.  All variables were global, and all parameter passing was through global variables.  It was *horrible*.

[Written 9 Sep, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Peter-Schachte). 477 views.

Upvote**6**

Downvote

Comment**1**

[](http://www.quora.com/Tae-Lim-Kook)

**[Tae Lim Kook](http://www.quora.com/Tae-Lim-Kook)**

20 upvotes by [Travis Hance](http://www.quora.com/Travis-Hance), [Oszkár Józsa](http://www.quora.com/Oszk%C3%A1r-J%C3%B3zsa), [Bogdan Taranu](http://www.quora.com/Bogdan-Taranu-1), (more)

One-based indexing.

And TI-89 style fucked lists.

[Updated 18 Oct, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Tae-Lim-Kook). 766 views.

Upvote**20**

Downvote

Comments**4+**

[](http://www.quora.com/Max-Lewin-1)

**[Max Lewin](http://www.quora.com/Max-Lewin-1)**, Computerologist

5 upvotes by [Pavel Vitsoft Šrubař](http://www.quora.com/Pavel-Vitsoft-%C5%A0ruba%C5%99), [Oldrich Svec](http://www.quora.com/Oldrich-Svec), [Peter Schachte](http://www.quora.com/Peter-Schachte), (more)

I've never directly experienced the resulting "badness", but a teacher of mine had some pretty hilarious stories about it: In Fortran 77 variables starting with the letters 'i' through 'n'  would be implicitly typed to integers, while all other variables would be implicitly typed to reals. Not that you couldn't override with explicit typing, and indeed that quickly became an important best practice in Fortran to avoid triggering this "feature", but it nonetheless could lead to some real insidious bugs if you didn't know about it.

[Written 5 Apr, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Max-Lewin-1). 872 views.

Upvote**5**

Downvote

Comments**3**

[](http://www.quora.com/Igor-Hlina)

**[Igor Hlina](http://www.quora.com/Igor-Hlina)**, Web applications designer

5 upvotes by [Peter Schachte](http://www.quora.com/Peter-Schachte), [Johan Bjäreholt](http://www.quora.com/Johan-Bj%C3%A4reholt), [Erik Bjäreholt](http://www.quora.com/Erik-Bj%C3%A4reholt), (more)

Global variables

when reading code that uses lots of global variables, often there are tons of WTFs "from where this value comes from/where this is set". This problem grows exponentially with number of team members.

Don't use global variables.

[Written 27 Apr, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Igor-Hlina). 466 views.

Upvote**5**

Downvote

Comments**2**

[](http://www.quora.com/Marco-Mariani)

**[Marco Mariani](http://www.quora.com/Marco-Mariani)**

6 upvotes by [Richard H. Schwartz](http://www.quora.com/Richard-H-Schwartz), [Sunmit Girme](http://www.quora.com/Sunmit-Girme), [Asdrúbal Iván Suárez Rivera](http://www.quora.com/Asdr%C3%BAbal-Iv%C3%A1n-Su%C3%A1rez-Rivera), (more)

My favourite from JavaScript:

> new Date('2013-01-01').getMonth()

> 0

not because it's the worst feature, but because I can't fathom any possible rationale behind it :-)

[Written 16 May, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Marco-Mariani). 406 views.

Upvote**6**

Downvote

Comments**1+**

[](http://www.quora.com/Nana-Ghartey)

**[Nana Ghartey](http://www.quora.com/Nana-Ghartey)**, Geek with Swag

11 upvotes by [Aryeh Friedman](http://www.quora.com/Aryeh-Friedman), [Joran Beasley](http://www.quora.com/Joran-Beasley), Quora User, (more)

Python, c++, c#,java allowing the use of Unicode to define  variables,classes & functions Eg.for (int λ = 0; λ < 20; λ++)  אלף+=λ; return אלף;

[Written 4 Apr, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Nana-Ghartey). 968 views.

Upvote**11**

Downvote

Comments**2+**

[](http://www.quora.com/Aryeh-Friedman)

**[Aryeh Friedman](http://www.quora.com/Aryeh-Friedman)**, LeadDeveloper, PetiteCloud/ThinStorm

3 upvotes by [Travis Hance](http://www.quora.com/Travis-Hance), [William Chargin](http://www.quora.com/William-Chargin), and [Eric Bowersox](http://www.quora.com/Eric-Bowersox).

Unreseerved keywords in C# (and related languages)

[Written 4 Apr, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Aryeh-Friedman). 929 views.

Upvote**3**

Downvote

Comment

[](http://www.quora.com/Marcin-Kasperski-1)

**[Marcin Kasperski](http://www.quora.com/Marcin-Kasperski-1)**

3 upvotes by [Tikhon Jelvis](http://www.quora.com/Tikhon-Jelvis), [Ricardo Kagawa](http://www.quora.com/Ricardo-Kagawa), and Quora User.

There are plenty, and my first instinct always depends on what I recently did (too-strict typing in case of C++, lack of signatures in Perl, UnicodeDecodeError in Python, …). But if we are too look wider, I’d name **error handling:**

- first, plenty of languages use „error codes" which are so notoriously unpleasant to check, and so easy to omit [and even in exception-aware languages, libraries which keep using error codes are fairly frequent];

- then, even where there are exceptions, they usually are poor (poorly documented and designed classes, lack of information about possible exceptions, limited information available from exception objects, convoluted semantics of corner cases like exception during exception-caused object destruction etc etc)

[Written 5 Apr, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Marcin-Kasperski-1). 708 views.

Upvote**3**

Downvote

Comment

[](http://www.quora.com/Andy-Dent)

**[Andy Dent](http://www.quora.com/Andy-Dent)**, CTO of Touchgr.am, used C++, Objectiv... (more)

1 upvote by [Jeff Darcy](http://www.quora.com/Jeff-Darcy).

TCL upvar and uplevel which allows a function to set a variable or evaluate code in the stack context N levels above it in the calling stack.

 see [Practical Introduction to Upvar and Uplevel](http://wiki.tcl.tk/4227)

I first encountered this concept in the AppMaker GUI Code Generator V2 scripting language, which was otherwise a clean OO language. I wondered where the usually-sane and pragmatic author got the idea. Then a few years ago I started work on a CAD product which has its entire UI defined in and scripted by TCL.

You wouldn't believe some of the code people can write in TCL using such features.

[Written 28 Apr, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Andy-Dent). 389 views.

Upvote**1**

Downvote

Comment**1**

[](http://www.quora.com/Sam-Boosalis)

**[Sam Boosalis](http://www.quora.com/Sam-Boosalis)**, wanderer, witness

5 upvotes by [Tikhon Jelvis](http://www.quora.com/Tikhon-Jelvis), [Johnny Ho](http://www.quora.com/Johnny-Ho), [Vijay Chidambaram](http://www.quora.com/Vijay-Chidambaram), (more)

i'll add contempt for syntax (it's not worse than the semantics of weak types or no lambdas or no , but it hasn't been said). it's a feature because there are no operators to overload, thus easier to read, right?

|  |  |
| --- | --- |
| 1 | assertEquality(addFloatingPointNumbers(new FloatingPointSuccessor(new FloatingPointSuccessor(new FloatingPointZero()))), FloatingPointSuccessor(new FloatingPointZero()), new FloatingPointSuccessor(new FloatingPointSuccessor(new FloatingPointSuccessor(new FloatingPointZero())))) |

... uh, no. numeric literals are necessary for legibility. arithmetic operators avoid typos and bugs (if \_ / \_ meant actual division, that is!).

|  |  |
| --- | --- |
| 1 | 2 + 1 == 3 |

in Java for example, people don't take this crap for numbers, they take some crap for strings, and they take a lot of crap for every other data-structure, like the ubiquitous lists and dicts.

|  |  |
| --- | --- |
| 1  2  3 | List<Integer> list = new ArrayList();  list.add(1);  list.add(2); |

there are good static fluent libraries (like Guava), but operators would be better.

|  |  |
| --- | --- |
| 1  2 | list += 3  list ++ [4, 5] |

and not just binary infix, the binary "out"fix (?) brackets with commas are the right way to write lists, because they're the easiest way to read lists.

if operators are interfaces or typeclasses or something (not C++), used between alphanumerics (not APL), and well-documented, it's almost always easier for me read expressions of alphanumeric functions when mixed with symbolic operators.

[Written 31 May, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Sam-Boosalis). 446 views.

Upvote**5**

Downvote

Comment

[](http://www.quora.com/Alexandru-Ifrim)

**[Alexandru Ifrim](http://www.quora.com/Alexandru-Ifrim)**, Independent software developer

1 upvote by [Rui Valente Maia](http://www.quora.com/Rui-Valente-Maia).

One that comes to mind is case-insensitivity in method names (PHP for instance).

Also on PHP, the use undeclared/implicit class members - can make development/debugging a pain if you mistype something.

[Written 14 May, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Alexandru-Ifrim). 328 views.

Upvote**1**

Downvote

Comments**1+**

[](http://www.quora.com/Todd-A-Gibson)

**[Todd A Gibson](http://www.quora.com/Todd-A-Gibson)**

2 upvotes by [Brent Collins](http://www.quora.com/Brent-Collins-5) and [Shubham Agrawal](http://www.quora.com/Shubham-Agrawal-131).

From C (and C++), the null-terminated string. Here's a fascinating article on "The most expensive one-byte mistake"  [http://queue.acm.org/detail.cfm?...](http://queue.acm.org/detail.cfm?id=2010365)

[Written 23 Oct, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Todd-A-Gibson). 176 views.

Upvote**2**

Downvote

Comment

[](http://www.quora.com/Phil-Jones)

**[Phil Jones](http://www.quora.com/Phil-Jones)**, Still figuring it out

3 upvotes by [Will Newton](http://www.quora.com/Will-Newton-1), [Tikhon Jelvis](http://www.quora.com/Tikhon-Jelvis), and [Tae Lim Kook](http://www.quora.com/Tae-Lim-Kook).

See [Phil Jones' answer to What are the advantages of dynamic scoping?](http://www.quora.com/What-are-the-advantages-of-dynamic-scoping/answer/Phil-Jones)

[Written 4 Apr, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Phil-Jones). 908 views.

Upvote**3**

Downvote

Comment

[](http://www.quora.com/Mario-T-Lanza)

**[Mario T. Lanza](http://www.quora.com/Mario-T-Lanza)**, Less is more

In [VB.NET](http://vb.net/), ByRef function arguments.  It shouldn't be possible to mutate state in the caller's context.  Functions should only return values in the normal manner, via the return result.

[Written 12 Oct, 2014](http://www.quora.com/What-is-the-single-worst-feature-in-any-programming-language/answer/Mario-T-Lanza). 226 views.

Upvote

Downvote

Comment

[](http://www.quora.com/Daia-Heinrich)

**[Daia Heinrich](http://www.quora.com/Daia-Heinrich)**, C monkey (C++ monkey)

1 upvote by [Amir Sagiv](http://www.quora.com/Amir-Sagiv).

The string handling in C++. The basic way of string handling is just plain simple array of characters, where no equals or assignment operator is defined. Means that instead of string1 == string2 you have to write strcmp(string1, string2) == 0 and instead of string1 = "abc" is it strcpy(string1, "abc").

Then you have an STL string type which avoids this inconvenients, but, in Windows Visual C++, the UI works with another string type, MFC CString, so you have to convert.

Another "fun" thing is that CString in Unicode is UTF16 and STL Unicode string is UTF8 and there is no way to work with plain UTF8.

**10. Python**

What it is: A high level, all purpose programming language that prides itself on its readability. It’s often used as a scripting language, though it can also be compiled.

Biggest complaint: Indentation is used to specify block structures rather than brackets or braces. Also, heavy use of colons and underscores and module/variable name collision.

How To Get Rid: Don’t work at Google, Yahoo or NASA.

**8. JavaScript**

What it is: An interpreted language, originally developed by Netscape, used primarily as a client-side scripting language on web pages. It’s also been implemented for server-side web scripting and as an embedded scripting language.

Biggest complaints: Case sensitivity, different implementations across browsers, lack of debugging capabilities (though Firebug solves that) and odd inheritance rules.

How To Get Rid: Don’t work as a web developer.

**7. Tcl**

What it is: Developed as an embedded command language, the Tool Control Language has evolved into a general purpose scripting language used for things such as web applications, network administration and test automation.

Biggest complaints: The syntax is almost too simple, it lacks pointers so there’s no way to pass by reference, arrays are stored as strings, it has poor list semantics and confusing variable scoping.

How To Get Rid: Don’t work for Cisco, AOL or CNET or anyplace using AOLserver or the OpenACS platform.

**5. C++**

What it is: An intermediate-level language created as an extension of C which supports, among other enhancements, object oriented programming. It remains one of the most popular languages, used in a wide variety of systems and applications.

Biggest complaints: Too big of a feature set, manual memory management, slow compilation speed and the fact that it allows programmers to switch between object oriented and procedural code in the same program.

How To Get Rid: Don’t work for Adobe, Google or the gaming industry, in general.

**4. PHP**

What it is: An interpreted language most often used for server-side scripting to generate HTML pages dynamically. It can also be used as a stand alone scripting language on many operating systems.

Biggest complaints: Inconsistent naming conventions for its many functions, security holes, no native support for Unicode, plus it often gets mixed in with presentation code (e.g., HTML, CSS).

**3. Java**

What it is: An object-oriented language originally created for interactive television and one of the most popular programming languages in use today. Java code gets compiled into bytecode, which is then interpreted by a platform-specific Java Virtual Machine, meaning Java programs are “Write Once, Run Anywhere.”

Biggest complaints: The syntax is too verbose, it’s slow, it’s not easy to pass functions, the API’s are over-engineered and lots of other languages can do what it does, but more efficiently.

|  |
| --- |
| Objective-C.  The annotations are confusing, using brackets to call methods still does not compute in my brain, and what is worse is that all of the library functions from C are called using the standard operators in C, -> and ., and it seems like the only company that is driving this language is Apple.  I admit I have only used the language when programming for the iPhone (and looking into programming for OS X), but it feels as if C++ were merely forked, adding in annotations and forcing the implementation and the header files to be separate would make much more sense. |

Second place would be Scheme, which had exactly one variable type: the linked list. Trying to learn how to think in it is like trying to learn how to think like a polar bear if you're an American-born Martian. It's totally alien to everything else.

BEST LANGUAGE FEATURES

10. Expression Bodied Methods

How many times have you had to write a method just for one line of code? Now, with C# 6 you can simply create an expression bodied member with only the expression and without the curly braces or explicit returns.

class Employee

{

// Method with only the expression

public static int

CalculateMonthlyPay(int dailyWage)

=> dailyWage \* 30;

}

9. ?—Conditional Access Operator

In earlier versions of the C# language, you always had to write the explicit if condition NULL checks before using an object or its property, as shown below.

* **Post a comment**
* [**Email Article**](http://www.developer.com/e-mail/net/csharp/top-10-c-6.0-language-features.html)
* [**Print Article**](http://www.developer.com/print/net/csharp/top-10-c-6.0-language-features.html)
* **Share Articles**

private void GetMiddleName(Employee employee)

{

string employeeMiddleName = "N/A";

if (employee != null && employee.EmployeeProfile

!= null)

employeeMiddleName =

employee.EmployeeProfile.MiddleName;

}

The same can be converted into a one-liner by using the Conditional Access Operator in C# 6.

private void GetMiddleName(Employee employee)

{

string employeeMiddleName =

employee?.EmployeeProfile?.MiddleName ?? "N/A";

}

Notice the default value provided on the same line of code.

8. Auto-Property Initializers

**Related Articles**

* [Nintex Workflow with SharePoint 2013](http://www.developer.com/net/nintex-workflow-with-sharepoint-2013.html)
* [What's New in Windows 10 for Developers](http://www.developer.com/net/whats-new-in-windows-10-for-developers.html)
* [Becoming a SharePoint Hero Has Never Been Easier](http://www.developer.com/net/becoming-a-sharepoint-hero-has-never-been-easier.html)
* [Managing Projects Backlog with Visual Studio Online](http://www.developer.com/net/managing-projects-backlog-with-visual-studio-online.html)

With the Auto-Property initialization feature, the developer can initialize properties without using a private set or the need for a local variable. Following is the sample source code.

class PeopleManager

{

public List<string> Roles { get; } =

new List<string>() { "Employee", "Managerial"};

7. Primary Constructor

Primary Constructor is a feature in which you are allowed to pass the constructor parameters at the class declaration level instead of writing a separate constructor. The scope of the primary constructor parameters values is class level and will be available only at the time of class initialization. It comes to good use when it is used with the Auto-Property initializers.

// Primary constructor

class Basket(string item, int price)

{

// Using primary constructor parameter values

// to do auto property initialization.

public string Item { get; } = item;

public int Price { get; } = price;

}

OUT Parameter Declaration During Method Call

This is one of my favorites because I was feeling something not good about the separate declaration of the OUT parameter before the method call. This feature allows you to declare the OUT parameter during the method call, as shown below.

public bool ConvertToIntegerAndCheckForGreaterThan10

(string value)

{

if (int.TryParse(value, out int convertedValue)

&& convertedValue > 10)

{

return true;

}

return false;

}

Note that the same out parameter is used in the consequent IF condition expression.

Await in the Catch Block

This is an important non-syntactic enhancement that will be available in C# 6. The await keyword can be called inside the catch and finally blocks. This opens up the way to perform an async exception handling or fallback process in case an exception happened during an async process call.

public async void Process()

{

try

{

Processor processor = new Processor();

await processor.ProccessAsync();

}

catch (Exception exception)

{

ExceptionLogger logger = new ExceptionLogger();

// Catch operation also can be aync now!!

await logger.HandleExceptionAsync(exception);

}

}

Exceptions can be filtered in the catch blocks with ease and cleanly with C# 6. Following is a sample source code where the intention is to handle all Exceptions except the SqlException type.

public async void Process()

{

try

{

DataProcessor processor = ne

}

// Catches and handles only non sql exceptions

catch (Exception exception) if(exception.GetType()

!= typeof(SqlException))

{

ExceptionLogger logger = new ExceptionLogger();

logger.HandleException(exception);

}

}

This feature is something to make your code less cluttered and will reduce duplications. As with the namespaces, you can include a static class in the using statement similar to a namespace.

using System;

// A static class inclusion

using System.Console;

namespace CSharp6Demo

{

class Program

{

static void Main(string[] args)

{

WriteLine("Console. is not required

as it is included in the usings!");

}

}

}

static void Main(string[] args)

{

string name = "Robert";

string car = "Audi";

WriteLine("\{name}'s favourite car is

{car}!");

}

+ is a concatenation operator in JavaScript

*The head scratcher*: The + operator is overloaded in JavaScript, being an addition operator for numbers and a concatenation operator for strings. If one operand is a string, JavaScript converts the other variable to a string and concatenation occurs, so that *‘1' + 1* yields *11*.

*The reason*: This is ultimately due to JavaScript’s loose typing. Python, for example, also uses + for string concatenation but, being a strongly typed language, it will raise an error if one tries to add a string and an integer.



<img alt="Picture of a woman showing two thumbs up" src="http://images.techhive.com/images/idge/imported/imageapi/2014/10/08/18/slide\_perl\_module\_true-620x465-100505011-gallery.idge.jpg" itemprop="image" />

[See larger image](http://images.techhive.com/images/idge/imported/imageapi/2014/10/08/18/slide_perl_module_true-620x465-100505011-orig.jpg)

*Image courtesy* [*flickr/anthony kelly*](https://www.flickr.com/photos/62337512@N00/3958637561)

Perl modules must return TRUE

*The head scratcher*: Perl modules almost always end with the statement *1;*and if they don’t, or, rather, if the last statement doesn’t [return a TRUE value](http://returnvalues.useperl.at/), an error is raised.

*The reason*: Perl modules can contain initialization code as well as subroutines. After the file is loaded, [Perl checks that any such code executes successfully](http://perldoc.perl.org/functions/require.html) by looking for a return value of TRUE. Even if there is no initialization code, Perl still expects the final statement to return TRUE or it raises an exception.



<img alt="Picture of a pointer dog pointing" src="http://images.techhive.com/images/idge/imported/imageapi/2014/10/08/18/slide\_c\_pointer-620x465-100505016-gallery.idge.jpg" itemprop="image" />

[See larger image](http://images.techhive.com/images/idge/imported/imageapi/2014/10/08/18/slide_c_pointer-620x465-100505016-orig.jpg)

*Image courtesy* [*flickr/Blue*♦*Gum*](https://www.flickr.com/photos/jorunngro/4686710592)

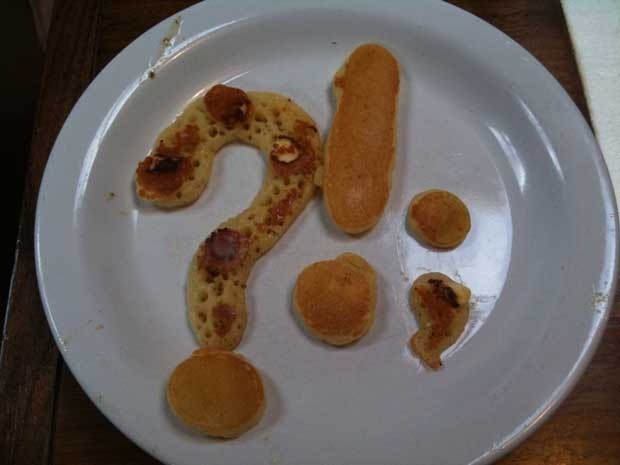
Array indexing in C behaves like pointer arithmetic

*The head scratcher*: In addition to referencing element *i* of array *a* as *a[i]*, C also allows you to [reference the same element as *i[a]*](http://pw1.netcom.com/~tjensen/ptr/ch4x.htm).

*The reason*: In C, arrays act like pointers to blocks of memory, so *a[i]* = *\*(a + i)* = *\*(i + a)* = *i[a]*.

*Quotes*: “... it is invaluable if you want to compete in the obfuscated C contest....” [Confusion](http://stackoverflow.com/questions/1995113/strangest-language-feature/1995278#comment1937772_1995156)

“I don't see it as a feature - so much as exposing the core of what C is about. Its all about pointers and getting to the memory directly with as little indirection as possible. Kind of beautiful, really.”  [Michael Neale](http://stackoverflow.com/questions/1995113/strangest-language-feature/1995278#comment3634860_1995156)



<img alt="Picture of punctuation marks made of pancakes" src="http://images.techhive.com/images/idge/imported/imageapi/2014/10/08/18/slide\_js\_semicolon-620x465-100505018-gallery.idge.jpg" itemprop="image" />

[See larger image](http://images.techhive.com/images/idge/imported/imageapi/2014/10/08/18/slide_js_semicolon-620x465-100505018-orig.jpg)

*Image courtesy* [*flickr/Nate Angell*](https://www.flickr.com/photos/ixmati/4571280627)

JavaScripts automatic semicolon insertion

*The head scratcher*: JavaScript makes the use of semicolons to end certain statements optional by [automatically inserting them](http://bclary.com/2004/11/07/#a-7.9.1) where it thinks they belong, such as after line breaks. This often leads to errors or baffling behavior with no exceptions raised.

*The reason*: Semicolon insertion was meant as a convenience, to make JavaScript's C-like syntax easier for new developers.

*Quotes*: “You always run into problems when you design language features around the assumption that your users will mostly be idiots.” [Rob Van Dam](http://stackoverflow.com/questions/1995113/strangest-language-feature/1995278#comment1924259_2003277)

“My advice: Figure out where the semicolons go, put them in the right place. You’ll be much better off.” [Doug Crockford](http://youtu.be/hQVTIJBZook?t=13m1s)

“Semicolon insertion is one of the m

Javas autoboxing with Integer caching

*The head scratcher*: Java will automatically convert primitives types to objects ([autoboxing](http://docs.oracle.com/javase/specs/jls/se7/html/jls-5.html#jls-5.1.7)), such as int to an Integer object. It will also, by default, cache Integer objects for values from -128 and 127. This can lead to [unexpected behavior](http://vanillajava.blogspot.com/2012/01/surprising-results-of-autoboxing.html) when using == to compare autoboxed Integers with the same value (TRUE from -128 and 127; FALSE otherwise).

*The reason*: Autoboxing reduces the amount of code that developers need to write, while Integer caching improves performance.

*Quotes*: “Thats the result of premature optimization.” [Joschua](http://stackoverflow.com/questions/1995113/strangest-language-feature#comment3822538_2001861)

“This isn't a common mistake, but it is a good reason to use the native Java types for numbers, booleans, etc…” [Ravi Wallau](http://stackoverflow.com/questions/18850930/java-integer-auto-auto-boxing#comment27813387_18850962)

“So glad I'm a C# programmer.” [Will](http://stackoverflow.com/questions/1995113/strangest-language-feature#comment1926765_2001861)

GOOD STUFF:

1. The **ability to pass functions in JavaScript** as arguments to other functions - makes things a lot easier. Functions are just objects with the additional ability of being invoked. You can write/design really beautiful generic templates/callback mechanisms as a result. I miss this in languages like Java.
2. **Prototypal inheritance in JavaScript** - you create new objects by inheriting from other objects. Almost like copying all properties of the current object and tacking on more as you need. Pretty powerful but not well understood/embraced concept of inheritance. See: [Page on stackoverflow.com](http://stackoverflow.com/a/16872315/609074) for a beautiful elucidation of concept.
3. **Pointers in C/C++**. Some folks hate them but man, they are too good to pass up. You can screw up your system if you don't understand them, but if you do understand them, it screws with your mind, in a good way. You really understand how to use disparate locations of memory to structure correlated data in a meaningful way i.e., stacks/queues. Languages like Java, make you understand this at a high level.
4. **Prefix operators/functions** in functional languages like LISP and its dialects. That is, you write (myFunction arg1 arg2 arg3) in LISP instead of the infix/proper notation like myFunction(arg1, arg2, arg3) in imperative languages - the benefits? Phenomenal! You have variable argument lists in a snap. Almost anything can be written as a function followed by 'n' number of arguments. Functions can easily be defined in terms of other functions. You basically have to think of applying the function 'recursively' to every argument. If it operates on a pair, then the return value of one execution along with the next argument and so on. This is just a crude overview. This lets you almost create everything with very very very minimal syntax. Quite powerful, mind bending and brilliant.
5. **Unions in C:** I was always fascinated by them and they made sense with memory limitations and still do with embedded systems. I could have the following code:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 | 1. typedef union{ 2. struct    { 3. uint8\_t a; 4. uint8\_t b; 5. uint8\_t c; 6. uint8\_t d; 7. }; 8. uint32\_t num; 9. } my32BitInteger; |  |  |

1. I could either access the whole integer or just its individual bits (endianness will bite you in the rear if you don't handle it properly :) Another advantage is you can use it to store a bunch of different objects if you know you'll only be accessing one at given time. Saves lot of memory and you can keep destructing them at will i.e., reusing the same space!
2. **Operator overloading in C++:** I seriously miss this feature in Java. The '+' is the only thing that's overloaded for strings, but can't be configured. It pains to write code like matrix.setMatrix(matrix.getMatrix().addMatrix(someMatrix)) instead of just matrix = matrix + someMatrix. That is self addition to another element of the same type just leads to kludgey code! I miss it in Java :'(
3. **Cons/Pairs in LISP:** Calling (cons `a `b) in LISP creates a Pair of two elements. You can individually access each element using car and cdr respectively. Now, you can define almost any complicated data structure with just this pair!! Each element of a pair could be a pair or every *cdr* could hold another pair and so on! If this is confusing, think of an equivalent in Java:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 1 2. 2 3. 3 4. 4 | 1. public class Pair{ 2. Object car; 3. Object cdr; 4. } |  |  |

1. You can then create arbitrary data structures using this! Simply brilliant! You also have macros and the like but I'm not so familiar with them so will leave that to the experts to answer.
2. **TSL-like instructions in Assembly:** Now this may not be feature of the language per se, but more of the hardware, but you may assume Assembly is the language of the hardware. The TSL (Test-and-set) instruction guarantees atomicity to enable "safe" access of data in case of concurrency allowing systems built on "top of it" correct concurrency mechanisms! [Test-and-set](http://en.wikipedia.org/wiki/Test-and-set)
3. There are of course many more in other languages. I've given my 2 cents on the ones I like the most in the languages I've played around the most.
5. There are plenty of extremely valuable advanced/modern features I appreciate and use - abstract types and inheritance for polymorphism, parameterized types for generic programming and covariance/contravariance, lambda closures for higher-order functions, lazy evaluation etc. But the single best feature for me is a very simple construct I fell in love with 3 decades ago and am still in love with - RECURSION. I am often surprised at how much people shy away from recursion especially when there is an opportunity to write tail-recursive code. Lastly, many of the advanced features I mentioned above when combined with recursion produce some of the most elegant code.

If your computer program is your party list, than variables are the sticky notes you use to keep track of all the data you've collected. Without your sticky notes, you'd be at a loss as to how to move forward with planning. Without [using variables](http://ruby.about.com/od/tutorials/ss/usevariables.htm), your Ruby program isn't able to keep track of its data either.

**Ruby Language Feature Number Two:** [**Arrays**](http://ruby.about.com/od/rubyfeatures/a/arrayhash.htm)

Programs often have to manage collections of variables. Take for example, a program that manages your party information. You're bound to have party guests, each of whom must be stored in a variable and a list of which can be stored together in an array variable. That way you can access each guest via the [array](http://ruby.about.com/od/af/g/array.htm).

**Ruby Language Feature Number Three:** [**Hashes**](http://ruby.about.com/od/rubyfeatures/a/hashes.htm)

Now imagine your party is a potluck event and each guest is required to bring a dish or other item. You'll still need a list of guests, which can be managed as an array, but you'll also need a way to keep track of what types of food are being brought. Additionally, you'll probably want to know who is bringing what.

Ads

[Online Ruby Classes](http://www.googleadservices.com/pagead/aclk?sa=L&ai=CmUpajyhdVZfPMNDf0AGK7oK4BMb-xKQGtra3mfwBsMeTgwgQBCCf_dggKAVgyYaMjNCk3BGgAdr__OUDyAEBqAMBqgSHAU_QuNMsjhOaOjpg8y8iTeDZcv_ZHA9Brk3LacSqNmIe6dPiYwanH2me_KJmX6i451koAJJzriU-hcS8oeT2FLOwnwoWbkIuAZuLve6iZsJlB2eKlqM5H-SiKQSrqKiWZ_vRH_X17t2bDkc3H99R5yQOoP4l0s0bgYAz-KHu32E_q1nz0-ohK4gGAYAHjoCDGqgHpr4b2AcB&num=3&cid=5GicgXf6PWnuCjDEvsO6Lyp_&sig=AOD64_0xJUobhencfMwwOgicKoNXmcryEA&client=ca-aj-about-premium&adurl=http://www.pluralsight.com/%3Fcreative%3D67632054678%26keyword%3D%26matchtype%3D%26network%3Dd%26device%3Dc)

[www.pluralsight.com/Free-Trial](http://www.googleadservices.com/pagead/aclk?sa=L&ai=CmUpajyhdVZfPMNDf0AGK7oK4BMb-xKQGtra3mfwBsMeTgwgQBCCf_dggKAVgyYaMjNCk3BGgAdr__OUDyAEBqAMBqgSHAU_QuNMsjhOaOjpg8y8iTeDZcv_ZHA9Brk3LacSqNmIe6dPiYwanH2me_KJmX6i451koAJJzriU-hcS8oeT2FLOwnwoWbkIuAZuLve6iZsJlB2eKlqM5H-SiKQSrqKiWZ_vRH_X17t2bDkc3H99R5yQOoP4l0s0bgYAz-KHu32E_q1nz0-ohK4gGAYAHjoCDGqgHpr4b2AcB&num=3&cid=5GicgXf6PWnuCjDEvsO6Lyp_&sig=AOD64_0xJUobhencfMwwOgicKoNXmcryEA&client=ca-aj-about-premium&adurl=http://www.pluralsight.com/%3Fcreative%3D67632054678%26keyword%3D%26matchtype%3D%26network%3Dd%26device%3Dc)

Unlimited Access. 3500+ Courses. Free 10 Day Trial. Sign Up Now!

[Medicare Health Plans](http://www.googleadservices.com/pagead/aclk?sa=L&ai=C_SFzjyhdVZfPMNDf0AGK7oK4BKWgnrcE_Z7iq-ABxbrJ914QBSCf_dggKAVgyYaMjNCk3BGgAePTyP0DyAEBqAMBqgSHAU_Q6IEGjhWaOjpg8y8iTeDZcv_ZHA9Brk3LacSqNmIe6dPiYwanH2me_KJmX6i451koAJJzriU-hcS8oeT2FLOwnwoWbkIuAZuLve6iZsJlB2eKlqM5H-SmKQS4y4KmZ_vRH_X17t2bDkc3H99R5yQOoP4l0s0bgYAz-KHu32E_q1nzqIFfM4gGAYAHhay3AqgHpr4b2AcB&num=5&cid=5GicgXf6PWnuCjDEvsO6Lyp_&sig=AOD64_1v9YH2Afk0d8xBcFbSmBMWyiK1Ig&client=ca-aj-about-premium&adurl=http://www.MedicareSolutions.com/original.aspx%3Ftemp1%3D1%26refid%3Drem)

[www.medicaresolutions.com](http://www.googleadservices.com/pagead/aclk?sa=L&ai=C_SFzjyhdVZfPMNDf0AGK7oK4BKWgnrcE_Z7iq-ABxbrJ914QBSCf_dggKAVgyYaMjNCk3BGgAePTyP0DyAEBqAMBqgSHAU_Q6IEGjhWaOjpg8y8iTeDZcv_ZHA9Brk3LacSqNmIe6dPiYwanH2me_KJmX6i451koAJJzriU-hcS8oeT2FLOwnwoWbkIuAZuLve6iZsJlB2eKlqM5H-SmKQS4y4KmZ_vRH_X17t2bDkc3H99R5yQOoP4l0s0bgYAz-KHu32E_q1nzqIFfM4gGAYAHhay3AqgHpr4b2AcB&num=5&cid=5GicgXf6PWnuCjDEvsO6Lyp_&sig=AOD64_1v9YH2Afk0d8xBcFbSmBMWyiK1Ig&client=ca-aj-about-premium&adurl=http://www.MedicareSolutions.com/original.aspx%3Ftemp1%3D1%26refid%3Drem)

Compare Medicare Insurance Plans. View All Plans & Prices Online.

* [Ruby on Rails](http://ruby.about.com/z/js/o.htm?k=ruby%20on%20rails&SUName=ruby&d=Ruby%20on%20Rails&r=http%3A%2F%2Fruby.about.com%2Fod%2Fbeginningruby%2Fa%2F5features.htm)
* [Ruby](http://ruby.about.com/z/js/o.htm?k=ruby&SUName=ruby&d=Ruby&r=http%3A%2F%2Fruby.about.com%2Fod%2Fbeginningruby%2Fa%2F5features.htm)
* [Programming](http://ruby.about.com/z/js/o.htm?k=programming&SUName=ruby&d=Programming&r=http%3A%2F%2Fruby.about.com%2Fod%2Fbeginningruby%2Fa%2F5features.htm)
* [What Programming Language](http://ruby.about.com/z/js/o.htm?k=what%20programming%20language&SUName=ruby&d=What%20Programming%20Language&r=http%3A%2F%2Fruby.about.com%2Fod%2Fbeginningruby%2Fa%2F5features.htm)
* [Ruby Web Development](http://ruby.about.com/z/js/o.htm?k=ruby%20web%20development&SUName=ruby&d=Ruby%20Web%20Development&r=http%3A%2F%2Fruby.about.com%2Fod%2Fbeginningruby%2Fa%2F5features.htm)

Using a hash to store [key/value](http://ruby.about.com/od/gl/g/keyvalue.htm) pairs allows you to store that information. Say Susan is bringing chocolate cake to your party. The "key" will be "Susan" and the variable stored and accessed by the key (the value) will be "chocolate cake."

**Ruby Language Feature Number Four:** [**Loops**](http://ruby.about.com/od/rubyfeatures/a/loops.htm)

Computer programs often have to perform actions more than just once. For example, if your program prints all new information added to your pot luck list, it will need to print more than just a single time. This can be accomplished using a construct known as a [loop](http://ruby.about.com/od/rubyfeatures/a/loops.htm).

**Ruby Language Feature Number Five:** [**Blocks**](http://ruby.about.com/od/beginningruby/a/blocks.htm)

Blocks aren't anything new, especially to functional programmers. In the Ruby language, the use of these nameless [methods](http://ruby.about.com/od/mr/g/method.htm) is widespread and the block is one of the language's key features. They're often used to abstract the more tedious and repetitive loops into a more user-friendly forms.