The Programming Language Landscape

Stephen A. Edwards

Columbia University

Fall 2014

The Diversity of Programming Languages

`http://www.99-bottles-of-beer.net` has programs in over 1,500 different programming languages and variations to generate the lyrics to the song “99 Bottles of Beer.”

99 Bottles of Beer

99 bottles of beer on the wall, 99 bottles of beer.
Take one down and pass it around, 98 bottles of beer on the wall.

98 bottles of beer on the wall, 98 bottles of beer.
Take one down and pass it around, 97 bottles of beer on the wall.

⋮

2 bottles of beer on the wall, 2 bottles of beer.
Take one down and pass it around, 1 bottle of beer on the wall.

1 bottle of beer on the wall, 1 bottle of beer.
Take one down and pass it around, no more bottles of beer on the wall.

No more bottles of beer on the wall, no more bottles of beer.
Go to the store and buy some more, 99 bottles of beer on the wall.

Java

```
class Bottles {
  public static void main(String args[]) {
    String s = "s";
    for (int beers=99; beers>-1;) {
      System.out.print(beers+" bottle"+s+" of beer on the wall, ");
      System.out.println(beers + " bottle" + s + " of beer, ");
      if (beers==0) {
        System.out.print("Go to the store, buy some more, ");
        System.out.println("99 bottles of beer on the wall.\n");
        System.exit(0);
      } else
        System.out.print("Take one down, pass it around, ");
      s = (--beers == 1)?"":"s";
      System.out.println(beers+" bottle"+s+" of beer on the wall.\n");
    }
  }
}
```

Sean Russell,
`http://www.99-bottles-of-beer.net/language-java-4.html`

Java

```
class Bottles {
  public static void main(St
    String s = "s";
    for (int beers=99; beers
      System.out.print(beers
      System.out.println(bee
      if (beers==0) {
        System.out.print("Go
        System.out.println("
        System.exit(0);
      } else
        System.out.print("Ta
      s = (--beers == 1)?"":
      System.out.println(bee                                          ;
    }
  }
}
```

Gosling et al., Sun, 1991

Imperative, object-oriented, threaded

Based on C++, C, Algol, etc.

Statically typed

Automatic garbage collection

Architecturally neutral

Defined on a virtual machine (Java Bytecode)

Sean Russell,
http://www.99-bottles-of-beer.net/language-java-4.html

C

```
#define MAXBEER 99
void chug(int beers);

int main()
{
  int beers;
  for(beers = MAXBEER; beers; chug(beers--)) ;
  puts("\nTime to buy more beer!\n");
  return 0;
}

void chug(int beers)
{
  char howmany[8], *s;
  s = beers != 1 ? "s" : "";
  printf("%d bottle%s of beer on the wall,\n", beers, s);
  printf("%d bottle%s of beeeeer . . . ,\n", beers, s);
  printf("Take one down, pass it around,\n");
  if (--beers) sprintf(howmany, "%d", beers);
  else strcpy(howmany, "No more");
  s = beers != 1 ? "s" : "";
  printf("%s bottle%s of beer on the wall.\n", howmany, s);
}
```

Bill Wein, `http://www.99-bottles-of-beer.net/language-c-116.html`

C

```
#define MAXBEER 99
void chug(int beers);

int main()
{
  int beers;
  for(beers = MAXBEER; beers
  puts("\nTime to buy more b
  return 0;
}

void chug(int beers)
{
  char howmany[8], *s;
  s = beers != 1 ? "s" : "";
  printf("%d bottle%s of bee
  printf("%d bottle%s of bee
  printf("Take one down, pas
  if (--beers) sprintf(howma
  else strcpy(howmany, "No m
  s = beers != 1 ? "s" : "";
  printf("%s bottle%s of bee
}
```

Dennis Ritchie, Bell Labs, 1969

Procedural, imperative

Based on Algol, BCPL

Statically typed; liberal conversion policies

Harmonizes with processor architecture

For systems programming: unsafe by design

Remains language of choice for operating systems

Bill Wein, `http://www.99-bottles-of-beer.net/language-c-116.html`

FORTRAN

```
      program ninetyninebottles
      integer bottles
      bottles = 99
 1    format (I2, A)
 2    format (A)
 3    format (I2, A, /)
 4    format (A, /)
 10   write (*,1) bottles, ' bottles of beer on the wall,'
      write (*,1) bottles, ' bottles of beer.'
      write (*,2) 'Take one down, pass it around...'
      if (bottles - 1 .gt. 1) then
         write (*,3) bottles - 1, ' bottles of beer on the wall.'
      else
         write (*,3) bottles - 1, ' bottle of beer on the wall.'
      end if
      bottles = bottles - 1
      if (bottles - 1) 30, 20, 10
*     Last verse
 20   write (*,1) bottles, ' bottle of beer on the wall,'
      write (*,1) bottles, ' bottle of beer.'
      write (*,2) 'Take one down, pass it around...'
      write (*,4) 'No bottles of beer on the wall.'
 30   stop
      end
```

FORTRAN

```
      program ninetyninebott
      integer bottles
      bottles = 99
 1    format (I2, A)
 2    format (A)
 3    format (I2, A, /)
 4    format (A, /)
 10   write (*,1) bottles, '
      write (*,1) bottles, '
      write (*,2) 'Take one
      if (bottles - 1 .gt. 1
         write (*,3) bottles
      else
         write (*,3) bottles
      end if
      bottles = bottles - 1
      if (bottles - 1) 30, 2
*     Last verse
 20   write (*,1) bottles, '
      write (*,1) bottles, '
      write (*,2) 'Take one
      write (*,4) 'No bottle
 30   stop
      end
```

Backus, IBM, 1956

Imperative language for science and engineering

First compiled language

Fixed format lines (for punch cards)

Arithmetic expressions, If, Do, and Goto statements

Scalar (number) and array types

Limited string support

Still common in high-performance computing

Inspired most modern languages, especially BASIC

AWK

```
BEGIN {
   for(i = 99; i >= 0; i--) {
      print ubottle(i), "on the wall,", lbottle(i) "."
      print action(i), lbottle(inext(i)), "on the wall."
      print
   }
}
function ubottle(n) {
   return sprintf("%s bottle%s of beer", n?n:"No more", n-1?"s":"")
}
function lbottle(n) {
   return sprintf("%s bottle%s of beer", n?n:"no more", n-1?"s":"")
}
function action(n) {
   return sprintf("%s", n ? "Take one down and pass it around," : \
                            "Go to the store and buy some more,")
}
function inext(n) {
   return n ? n - 1 : 99
}
```

OsamuAoki,
`http://www.99-bottles-of-beer.net/language-awk-1623.html`

AWK

```
BEGIN {
   for(i = 99; i >= 0; i--) {
      print ubottle(i), "on the wall,", lbottle(i) "."
      print action(i), lbottle(inext(i)), "on the wall."
      print
   }
}
function ubottle(n) {
   return sprintf("%s bottle
}
function lbottle(n) {
   return sprintf("%s bottle
}
function action(n) {
   return sprintf("%s", n ?

}
function inext(n) {
   return n ? n - 1 : 99
}
```

Aho, Weinberger, and Kernighan, Bell Labs, 1977

Interpreted domain-specific scripting language for text processing

Pattern-action statements matched against input lines

C-inspired syntax

Automatic garbage collection

OsamuAoki,
http://www.99-bottles-of-beer.net/language-awk-1623.html

AWK (bottled version)

```
      BEGIN{
      split( \
      "no mo"\
      "rexxN"\
      "o mor"\
      "exsxx"\
      "Take "\
     "one dow"\
    "n and pas"\
   "s it around"\
  ", xGo to the "\
 "store and buy s"\
 "ome more, x bot"\
 "tlex of beerx o"\
 "n the wall" , s,\
 "x"); for( i=99 ;\
 i>=0; i--){ s[0]=\
 s[2] = i ; print \
 s[2 + !(i) ] s[8]\
 s[4+ !(i-1)] s[9]\
 s[10]", " s[!(i)]\
 s[8] s[4+ !(i-1)]\
 s[9]".";i?s[0]--:\
 s[0] = 99; print \
 s[6+!i]s[!(s[0])]\
 s[8] s[4 +!(i-2)]\
 s[9]s[10] ".\n";}}
```

Wilhelm Weske,
`http://www.99-bottles-of-beer.net/language-awk-1910.html`

Python

```
for quant in range(99, 0, -1):
   if quant > 1:
      print quant, "bottles of beer on the wall,", \
            quant, "bottles of beer."
      if quant > 2:
         suffix = str(quant - 1) + " bottles of beer on the wall."
      else:
         suffix = "1 bottle of beer on the wall."
   elif quant == 1:
      print "1 bottle of beer on the wall, 1 bottle of beer."
      suffix = "no more beer on the wall!"
   print "Take one down, pass it around,", suffix
   print ""
```

Gerold Penz,
`http://www.99-bottles-of-beer.net/language-python-808.html`

Python

```
for quant in range(99, 0, -1
   if quant > 1:
      print quant, "bottles
            quant, "bottles
      if quant > 2:
         suffix = str(quant
      else:
         suffix = "1 bottle
   elif quant == 1:
      print "1 bottle of bee
      suffix = "no more beer
   print "Take one down, pas
   print ""
```

Guido van Rossum, 1989

Object-oriented, imperative

General-purpose scripting language

Indentation indicates grouping

Dynamically typed

Automatic garbage collection

Gerold Penz,
`http://www.99-bottles-of-beer.net/language-python-808.html`

APL

```
⍝ APL (A Programming Language)

⍝ Program written by JT. Taylor, www.jttaylor.net

T1←98↑[1]ØΦ1 99ρι99

T4←ØΦ1 98ρι98

T1,(98 30ρ’ BOTTLES OF BEER ON THE WALL, ‘),T1,
(98 47ρ’BOTTLES OF BEER, TAKE ONE DOWN, PASS IT
AROUND,’),T4,(98 28ρ’BOTTLES OF BEER ON THE
WALL ,’)

‘1 BOTTLE OF BEER ON THE WALL, 1 BOTTLE OF BEER,
TAKE IT DOWN, PASS IT AROUND, NO BOTTLES OF BEER
ON THE WALL.’
```

http://www.99-bottles-of-beer.net/language-apl-715.html

APL

```
⍝ APL (A Programming Lar

⍝ Program written by JT.

T1←98↑[1]Ø⌽1 99⍴⍳99

T4←Ø⌽1 98⍴⍳98

T1,(98 30⍴' BOTTLES OF B
(98 47⍴'BOTTLES OF BEER,
AROUND,'),T4,(98 28⍴'BOT
WALL ,')

'1 BOTTLE OF BEER ON THE
TAKE IT DOWN, PASS IT AR
ON THE WALL.'
```

- Iverson, IBM, 1960
- Imperative, matrix-centric
- E.g., perform an operation on each element of a vector
- Uses own specialized character set
- Concise, effectively cryptic
- Primarily symbols instead of words
- Dynamically typed
- Odd left-to-right evaluation policy
- Useful for statistics, other matrix-oriented applications

http://www.99-bottles-of-beer.net/language-apl-715.html

FORTH

```
: .bottles ( n -- n-1 )
   dup 1 = IF   ." One bottle of beer on the wall," CR
                ." One bottle of beer," CR
                ." Take it down,"
   ELSE  dup . ." bottles of beer on the wall," CR
         dup . ." bottles of beer," CR
         ." Take one down,"
   THEN
   CR
   ." Pass it around," CR
   1-
   ?dup IF  dup 1 = IF  ." One bottle of beer on the wall;"
            ELSE  dup . ." bottles of beer on the wall;"
            THEN
        ELSE  ." No more bottles of beer on the wall."
   THEN
   CR
;
: nbottles ( n -- )
  BEGIN  .bottles  ?dup NOT UNTIL ;

99 nbottles
```

Dan Reish,
http://www.99-bottles-of-beer.net/language-forth-263.html

FORTH

```
: .bottles ( n -- n-1 )
   dup 1 = IF  ." One bottle
               ." One bottle
               ." Take it do
   ELSE  dup . ." bottles of
         dup . ." bottles of
         ." Take one down,"
   THEN
   CR
   ." Pass it around," CR
   1-
   ?dup IF  dup 1 = IF  ." O
            ELSE  dup . ." b
            THEN
        ELSE  ." No more bot
   THEN
   CR
;
: nbottles ( n -- )
  BEGIN  .bottles  ?dup NOT

99 nbottles
```

Moore, NRAO, 1973

Stack-based imperative language

Trivial, RPN-inspired grammar

Easily becomes cryptic

Untyped

Low-level, very lightweight

Highly extensible: easy to make programs compile themselves

Used in some firmware boot systems (Apple, IBM, Sun)

Inspired the PostScript language for laser printers

Dan Reish,
`http://www.99-bottles-of-beer.net/language-forth-263.html`

The Whitespace Language

Edwin Brady and Chris Morris, April 1st, 2003

Imperative, stack-based language

Space, Tab, and Line Feed characters only

Number literals in binary: Space=0, Tab=1, LF=end

Less-than-programmer-friendly syntax; reduces toner consumption

Andrew Kemp, `http://compsoc.dur.ac.uk/whitespace/`

Prolog

```
bottles :-
    bottles(99).

bottles(1) :-
    write('1 bottle of beer on the wall, 1 bottle of beer,'), nl,
    write('Take one down, and pass it around,'), nl,
    write('Now they are all gone.'), nl,!.
bottles(X) :-
    write(X), write(' bottles of beer on the wall,'), nl,
    write(X), write(' bottles of beer,'), nl,
    write('Take one down and pass it around,'), nl,
    NX is X - 1,
    write(NX), write(' bottles of beer on the wall.'), nl, nl,
    bottles(NX).
```

Remko Trocon et al.,
`http://www.99-bottles-of-beer.net/language-prolog-965.html`

Prolog

```
bottles :-
    bottles(99).

bottles(1) :-
    write('1 bottle of beer
    write('Take one down, ar
    write('Now they are all
bottles(X) :-
    write(X), write(' bottle
    write(X), write(' bottle
    write('Take one down and
    NX is X - 1,
    write(NX), write(' bottl
    bottles(NX).
```

Alain Colmerauer et al., 1972

Logic programming language

Programs are relations: facts and rules

Program execution consists of trying to satisfy queries

Designed for natural language processing, expert systems, and theorem proving

Remko Trocon et al.,
`http://www.99-bottles-of-beer.net/language-prolog-965.html`

SQL

```
SELECT
  CASE (bottlecount)
    WHEN 0 THEN 'No more bottle of beer on the wall, no more bottles o
                'Go to the store and buy some more, 99 bottles of beer
    WHEN 1 THEN '1 bottle of beer on the wall, 1 bottle of beer. ' ||
                'Take one down and pass it around, no more bottles of
    WHEN 2 THEN '2 bottles of beer on the wall, 2 bottles of beer. ' |
                'Take one down and pass it around, 1 bottle of beer on
    ELSE
      rtrim (cast((BottleCount) as char(2))) || ' bottles of beer on t
      rtrim (cast((BottleCount) as char(2))) || ' bottles of beer. ' |
      'Take one down and pass it around, ' ||
      rtrim (cast((BottleCount)-1 as char(2))) || ' bottles of beer on
  END
FROM
(
  SELECT avalue * 10 + bvalue as bottlecount
  FROM
    (VALUES (9), (8), (7), (6), (5), (4), (3), (2), (1), (0)) a(avalue
    (VALUES (9), (8), (7), (6), (5), (4), (3), (2), (1), (0)) b(bvalue
) as valuelist;
```

Kent Olsen,
http://www.99-bottles-of-beer.net/language-sql-967.html

SQL

```
SELECT
  CASE (bottlecount)
    WHEN 0 THEN 'No more bottle of beer on the wall, no more bottles o
                'Go to the store and buy some more, 99 bottles of beer
    WHEN 1 THEN '1 bottle of beer on the wall, 1 bottle of beer. ' ||
                'Take one down and pass it around, no more bottles of
    WHEN 2 THEN '2 bottles of beer on the wall, 2 bottles of beer. ' |
                'Take one do                                          n
    ELSE
      rtrim (cast((BottleCou                                          t
      rtrim (cast((BottleCou                                          |
      'Take one down and pas
      rtrim (cast((BottleCou                                          n
  END
FROM
(
  SELECT avalue * 10 + bvalu
  FROM
    (VALUES (9), (8), (7), (                                          e
    (VALUES (9), (8), (7), (                                          e
) as valuelist;
```

Chamberlin and Boyce, IBM, 1974

Declarative language for databases

Semantics based on the relational model

Queries on tables: select with predicates, joining, aggregating

Database query optimization: declaration to procedure

Kent Olsen,
`http://www.99-bottles-of-beer.net/language-sql-967.html`

LISP

```
(defun bottles-of-bier (n)
  (case n
   (0
    '(No more bottles of beer on the wall no more bottles of beer.
          Go to the store and buy some more 99 bottles of beer on the w
   (1
    `(1 bottle of beer on the wall 1 bottle of beer.
         Take one down and pass it around no more bottles of beer on th
         ,@(bottles-of-bier 0)))
   (2
    `(2 bottles of beer on the wall 2 bottles of beer.
        Take one down and pass it around 1 bottle of beer on the wall.
        ,@(bottles-of-bier 1)))
   (t
    `(,n bottles of beer on the wall ,n bottles of beer.
          Take one down and pass it around
          ,(1- n) bottles of beer on the wall.
          ,@(bottles-of-bier (1- n))))))
```

jimka, `http://www.99-bottles-of-beer.net/language-lisp-1465.html`

LISP

```
(defun bottles-of-bier (n)
  (case n
   (0
    '(No more bottles of bee
         Go to the store and
   (1
    `(1 bottle of beer on th
        Take one down and pa
        ,@(bottles-of-bier 0
   (2
    `(2 bottles of beer on t
       Take one down and pas
       ,@(bottles-of-bier 1)
   (t
    `(,n bottles of beer on
         Take one down and p
         ,(1- n) bottles of
         ,@(bottles-of-bier
```

McCarthy, MIT, 1958

Functional: recursive, list-focused functions

Semantics from Church's Lambda Calculus

Simple, heavily parenthesized S-expression syntax

Dynamically typed

Automatic garbage collection

Originally for AI applications

Dialects: Scheme and Common Lisp

jimka, `http://www.99-bottles-of-beer.net/language-lisp-1465.html`

Haskell

```
bottles :: Int -> String
bottles n
   | n == 0 = "no more bottles"
   | n == 1 = "1 bottle"
   | n >  1 = show n ++ " bottles"

verse :: Int -> String
verse n
   | n == 0 = "No more bottles of beer on the wall, "
              ++ "no more bottles of beer.\n"
              ++ "Go to the store and buy some more, "
              ++ "99 bottles of beer on the wall."
   | n > 0  = bottles n ++ " of beer on the wall, "
              ++ bottles n
              ++ " of beer.\n"
              ++ "Take one down and pass it around, "
              ++ bottles (n-1) ++ " of beer on the wall.\n"

main      = mapM (putStrLn . verse) [99,98..0]
```

Simon Johansson,

http://www.99-bottles-of-beer.net/language-haskell-1613.html

Haskell

```
bottles :: Int -> String
bottles n
   | n == 0 = "no more bottle
   | n == 1 = "1 bottle"
   | n >  1 = show n ++ " bot

verse :: Int -> String
verse n
   | n == 0 = "No more bottle
            ++ "no more bot
            ++ "Go to the s
            ++ "99 bottles
   | n > 0  = bottles n ++ "
            ++ bottles n
            ++ " of beer.\n
            ++ "Take one do
            ++ bottles (n-1

main      = mapM (putStrLn .
```

Peyton Jones et al., 1990

Functional

Pure: no side-effects

Lazy: computation only on demand; infinite data structures

Statically typed; types inferred

Algebraic data types, pattern matching, lists, strings

Great for compilers, domain-specific languages, type system research

Related to ML, OCaml

Simon Johansson,

`http://www.99-bottles-of-beer.net/language-haskell-1613.html`