



Case Study: The Development of C

CS345 - Programming Languages

Dr. Greg Lavender
Department of Computer Sciences
The University of Texas at Austin



C Language: motivation

- designed as a systems implementation language
 - for the development of an operating system, compiler, & utilities
- created by a small group of software engineers at Bell Labs in early 1970s
 - Ken Thompson & Dennis Richie
- born out of frustration with "big" OS projects and large complex languages
 - Multics, PL/I, Algol 68
 - Unix (Unics) instead of Multics
 - originally designed for small memory machines (PDP-7, PDP-11)
 - later ported to mainframes & made "portable" to other machines
 - C + Unix turned out to be perfect for small memory machines such as mini-computers (e.g., DEC Vax), early Unix-based workstations (e.g., Sun), and personal computers (Apple and PC)
 - Apple Lisa (1985) and the Mac II (1988) ran Apple versions of Unix using Motorola 68010 processors. Today, MacOS X is based on the Mach microkernel and a BSD Unix variant

C Language: influences

- indirect: Fortran, Algol, PL/I, Algol68
- direct: BCPL, B, NB (new B)
 - B is C without types
 - B is derived from BCPL
 - BCPL invented at MIT by Martin Richards
 - B "... is BCPL squeezed into 8K bytes of memory..."
"... and filtered through [Ken] Thompson's brain."
- BCPL, B and C all differ in syntax
 - B & C do not allow nested procedure scopes, BCPL does
 - all support separate compilation and allow "include" files
 - B is "simpler" than BCPL because of small memory machine
 - very compact syntax required so compiler could work in one pass

9/10/07 14:58

CS345 - Programming Languages

3

C Language: syntax

- BCPL used '=' for assignment (like Algol and Pascal)
 - B dropped the '=' and just used the '='
 - how many times have you made the following mistake in C/C++:
 - if (a = b) then ... else ...
 - B uses /*...*/ for comments, BCPL uses // which C++ adopted
 - /*...*/ comes from PL/I
 - note that C++ re-introduced //, C99 standard allows // comments
 - Bjarne Stroustrup, inventor of C++, used BCPL during his PhD research
 - B declarations begin with 'auto' or 'static'. C introduced type specifiers, but auto is optional and assumed in C for local variables
 - static int x = 0;
 - int f() { auto int y; register int x ... }
 - Auto and register are no longer used in C/C++, but are still present as keywords in those languages. Java dropped both, along with "goto"

9/10/07 14:58

CS345 - Programming Languages

4

C Language: syntax

- generalized assignment operations
 - BCPL/B/early C: $x += y$;
 - C: $x += y$;
- pre/post increment operators: $++$, $--$
 - appeared in B as a more compact syntax for parsing
 - $x++$: $x = x + 1$;
 - $y--$: $y = y - 1$;
 - Question: what is the meaning of $++x++$ in C/C++
 - What is the meaning of $x = ++x + x++$ in C/C++ ?

9/10/07 14:58

CS345 - Programming Languages

5

C Language: semantics

- B kept much of BCPL semantics
 - equivalence of pointers and arrays, pointer arithmetic
 - BCPL: let $V = \text{vec } 10$
 - $V!i$ to index to i^{th} element
 - B/C: $\text{auto } V[10]$
 - $*(V+i)$ or $V[i]$ to index to i^{th} element
 - strings
 - BCPL: first byte contains the length
 - B: last byte contains a special "end of string" character (i.e., null)
 - C strings are a null terminated sequence of bytes referenced either by a pointer-to-char or an array variable $s[]$.
 - $\text{char* } s = \text{"hello, world"}$ /* implicit null byte '\0' terminates the string */
 - $\text{char } s[] = \{\text{'h','e','l','l','o',' ',' ','w','o','r','l','d','\0'}\}$

9/10/07 14:58

CS345 - Programming Languages

6

C Language: pragmatics

- B deficiencies motivated an improved compiler
 - B generated "threaded-code" (like byte code), not native-code
 - B used packed strings causing inefficient string manipulation
 - overhead in dealing with pointers
- "New B" (NB) invented by Dennis Richie to be more efficient
 - introduced int, char & their array types, and pointer types
 - compiled to native code
- C fully equated typed arrays with typed pointers
 - `int a[10]; ... x = a[i];` means `x = (&a+i*sizeof(int))` where `&a` is pointer to the first memory cell of the array

9/10/07 14:58

CS345 - Programming Languages

7

C Language: types

- C primarily improved on B by adding type declarations & type rules
 - syntax influenced by Algol 68
 - `int i, *pi, **ppi;`
 - `int f(), *f(), *(*pf)();`
 - `int *api[10], (*pai)[10];`
 - also structs and unions
- What do the following function type declarations mean in C/C++?
 - `int *f(); int **f(); int *(*pf)(); int (*pf)(int);`

9/10/07 14:58

CS345 - Programming Languages

8

C Language: standardization

- C definition complete in 1973
 - new features added from 1973-1980
 - unsigned, long, union, enums
 - porting of C compiler to other machines
- K&R C book published 1978
 - de facto language standard for 10 years
- Explosion of C compilers in early 1980s
 - for mainframes, mini-computers, workstations and PCs
- ANSI C standardization 1989
 - key additions:
 - function prototypes as in C++
 - double sin(double) instead of double sin()
 - const and volatile keywords introduced
- ISO 9899:1999 also known as "C99"
 - Added inline functions, C++-like decls, bools, variable arrays & more

9/10/07 14:58

CS345 - Programming Languages

9

C Language: evolution

- variants & descendants
 - Concurrent C, Objective C, C*, C++
 - Java, C#
- used as a portable assembly language
 - early C++ and Modula-3 and Eiffel translated to C
 - Called source-to-source translation, then compiled
- C compilers generate very efficient code today
 - for all kinds of devices
 - at last count, GNU gcc generated native code for 70 different instruction set architectures

9/10/07 14:58

CS345 - Programming Languages

10