B (programming language)

B is a programming language developed at Bell Labs circa 1969. It is the work of Ken Thompson with Dennis Ritchie.

B was derived from BCPL, and its name may be a contraction of BCPL. Thompson's coworker Dennis Ritchie speculated that the name might be based on Bon, an earlier, but unrelated, programming language that Thompson designed for use on Multics.[note 1]

B was designed for recursive, non-numeric, machine-independent applications, such as system and language software.[3] It was a typeless language, with the only data type being the underlying machine's natural memory word format, whatever that might be. Depending on the context, the word was treated either as an integer or a memory address.

As machines with ASCII processing became common, notably the DEC PDP-11 that arrived at Bell, support for character data stuffed in memory words became important. The typeless nature of the language was seen as a disadvantage, which led Thompson and Ritchie to develop an expanded version of the language supporting new internal and user-defined types, which became the C programming language.

B	
Designed by	Ken Thompson
Developer	Ken Thompson, Dennis Ritchie
First appeared	1969[1]
Typing discipline	typeless (everything is a word)
Filename extensions	.b
Influenced by	
BCPL, PL/I, TMG	
Influenced	
C	

Contents

History

Circa 1969, Ken Thompson[2] and later Dennis Ritchie[3] developed B basing it mainly on the BCPL language Thompson used in the Multics project. B was essentially the BCPL system stripped of any component Thompson felt he could do without in order to make it fit within the memory capacity of the minicomputers of the time. The BCPL to B transition also included changes made to suit Thompson's preferences (mostly along the lines of reducing the number of non-whitespace characters in a typical program).[2] Much of the typical ALGOL-like syntax of BCPL was rather heavily changed in this process. The assignment operator `:=` changed to `=` and the equality operator `=` was replaced by `==`.

Thompson added "two-address assignment operators" using `x =+ y` syntax to add y to x (in C the operator is written `+=`). This syntax came from Douglas McIlroy's implementation of TMG, in which B's compiler was first implemented (and it came to TMG from ALGOL 68's `x +:= y` syntax).[2][4] Thompson went further by inventing the increment and decrement operators (`++` and `--`). Their prefix or postfix position determines whether the value is taken before or after alteration of the operand. This innovation was not in the earliest versions of B. According to Dennis Ritchie, people often assumed that they were created for the auto-increment and auto-decrement address modes of the DEC PDP-11, but this is historically impossible as the machine didn't exist when B was first developed.[2]

B is typeless, or more precisely has one data type: the computer word. Most operators (e.g. `+`, `-`, `*`, `/`) treated this as an integer, but others treated it as a memory address to be dereferenced. In many other ways it looked a lot like an early version of C. There are a few library functions, including some that vaguely resemble functions from the standard I/O library in C.[3]

Early implementations were for the DEC PDP-7 and PDP-11 minicomputers using early Unix, and Honeywell GE 645[5] 36-bit mainframes running the operating system GCOS. The earliest PDP-7 implementations compiled to threaded code, and Ritchie wrote a compiler using TMG which produced machine code.[6][7][8] In 1970 a PDP-11 was acquired and threaded code was used for the port; an assembler, `dc` , and the B language itself were written in B to bootstrap the computer. An early version of yacc was produced with this PDP-11 configuration. Ritchie took over maintenance during this period.[2][8]

The typeless nature of B made sense on the Honeywell, PDP-7 and many older computers, but was a problem on the PDP-11 because it was difficult to elegantly access the character data type that the PDP-11 and most modern computers fully support. Starting in 1971 Ritchie made changes to the language while converting its compiler to produce machine code, most notably adding data typing for variables. During 1971 and 1972 B evolved into "New B" (NB) and then C.[2]

B is almost extinct, having been superseded by the C language.[9] However, it continues to see use on GCOS mainframes (as of 2014)[10] and on certain embedded systems (as of 2000) for a variety of reasons: limited hardware in small systems, extensive libraries, tooling, licensing cost issues, and simply being good enough for the job.[9] The highly influential AberMUD was originally written in B.

Examples

The following examples are from the *Users' Reference to B* by Ken Thompson:[3]

```
/* The following function will print a non-negative number, n, to
   the base b, where 2<=b<=10.  This routine uses the fact that
   in the ASCII character set, the digits 0 to 9 have sequential
   code values.  */

printn(n, b) {
        extrn putchar;
        auto a;
        /* Wikipedia note: auto declares a variable with automatic
           storage (lifetime is function scope), not "automatic typing"
           as in C++. */

        if (a = n / b)        /* assignment, not test for equality */
                printn(a, b); /* recursive */
        putchar(n % b + '0');
}
```

```
/* The following program will calculate the constant e-2 to about
   4000 decimal digits, and print it 50 characters to the line in
   groups of 5 characters.  The method is simple output conversion
   of the expansion
```

```
     1/2! + 1/3! + ... = .111....
   where the bases of the digits are 2, 3, 4, . . . */

main() {
    extrn putchar, n, v;
    auto i, c, col, a;

    i = col = 0;
    while(i<n)
        v[i++] = 1;
    while(col<2*n) {
        a = n+1 ;
        c = i = 0;
        while (i<n) {
            c =+ v[i] *10;
            v[i++]  = c%a;
            c =/ a--;
        }

        putchar(c+'0');
        if(!(++col%5))
            putchar(col%50?' ': '*n');
    }
    putchar('*n*n');
}
v[2000];
n 2000;
```

See also

Computer programming portal

Notes

1. "Its name most probably represents a contraction of BCPL, though an alternate theory holds that it derives from Bon [Thompson 69], an unrelated language created by Thompson during the Multics days. Bon in turn was named either after his wife Bonnie or (according to an encyclopedia quotation in its manual), after a religion whose rituals involve the murmuring of magic formulas."[2]

References

1. "B - computer programming language" (http://www.britannica.com/EBchecked/topic/1663863/B).
2. Ritchie, Dennis M. (March 1993). "The Development of the C Language" (http://www.bell-labs.com/usr/dmr/www/chist.html). *ACM SIGPLAN Notices*. **28** (3): 201–208. doi:10.1145/155360.155580 (https://doi.org/10.1145%2F155360.155580).
3. Thompson, Ken (7 January 1972). "Users' Reference to B" (https://web.archive.org/web/20150317033259/https://www.bell-labs.com/usr/dmr/www/kbman.pdf) (PDF). Bell Laboratories. Archived from the original (https://www.bell-labs.com/usr/dmr/www/kbman.pdf) (PDF) on 17 March 2015. Retrieved 21 March 2014.
4. Michael S. Mahoney (18 August 1989). "Interview with M.D. McIlroy" (https://www.princeton.edu/~hos/mike/transcripts/mcilroy.htm). *Princeton.edu*. Murray Hill.
5. Ritchie, Dennis M. (1984). "The Evolution of the Unix Time-sharing System" (https://web.archive.org/web/20150611114353/https://www.bell-labs.com/usr/dmr/www/hist.html). *AT&T Bell Laboratories Technical Journal*. **63** (6 Part 2): 1577–1593. Archived from the original (https://www.bell-labs.com/usr/dmr/www/hist.html) on 11 June 2015.
6. "TMG" (http://www.multicians.org/tmg.html). multicians.org.

7. Ritchie, Dennis M. "The Development of the C Language" (https://web.archive.org/web/20150611114355/https://www.bell-labs.com/usr/dmr/www/chist.html). Bell Labs/Lucent Technologies. Archived from the original (https://www.bell-labs.com/usr/dmr/www/chist.html) on 11 June 2015.
8. McIlroy, M. D. (1987). *A Research Unix reader: annotated excerpts from the Programmer's Manual, 1971–1986* (http://www.cs.dartmouth.edu/~doug/reader.pdf) (PDF) (Technical report). CSTR. Bell Labs. 139.
9. Johnson and Kernighan. "THE PROGRAMMING LANGUAGE B" (https://web.archive.org/web/20150611114355/https://www.bell-labs.com/usr/dmr/www/bintro.html). Bell Laboratories. Archived from the original (https://www.bell-labs.com/usr/dmr/www/bintro.html) on 11 June 2015. Retrieved 21 March 2014.
10. "Thinkage UW Tools Package" (http://www.thinkage.ca/english/gcos/product-uwtools.shtml). Thinkage, Ltd. Retrieved 26 March 2014.

External links

- *Manual page for b(1) from Unix First Edition (http://man.cat-v.org/unix-1st/1/b)*
- *The Development of the C Language (https://www.bell-labs.com/usr/dmr/www/chist.html)*, Dennis M. Ritchie. Puts B in the context of BCPL and C.
- *Users' Reference to B (https://www.bell-labs.com/usr/dmr/www/kbman.html)*, Ken Thompson. Describes the PDP-11 version.
- *The Programming Language B (https://www.bell-labs.com/usr/dmr/www/bintro.html)*, S. C. Johnson & B. W. Kernighan, Technical Report CS TR 8, Bell Labs (January 1973). The GCOS version on Honeywell equipment.
- *B Language Reference Manual (http://www.thinkage.ca/english/gcos/expl/b/index.html)*, Thinkage Ltd. The production version of the language as used on GCOS, including language and runtime library.