I. Introduction

Vasili Slapik
vasili_slapik@epam.com

10 февраля 2014 г.

C main steps

- 1969 - 1973: developed by Dennis Ritchie at AT&T Bell Labs.

C main steps

- 1969 - 1973: developed by Dennis Ritchie at AT&T Bell Labs.
- 1978: Brian Kernighan and Dennis Ritchie published the first edition of The C Programming Language. (K&R)

C main steps

- 1969 - 1973: developed by Dennis Ritchie at AT&T Bell Labs.
- 1978: Brian Kernighan and Dennis Ritchie published the first edition of The C Programming Language. (K&R)
- 1989: the C standard ratified as ANSI X3.159-1989 "Programming Language C". (ANSI C or C89)

C main steps

- 1969 - 1973: developed by Dennis Ritchie at AT&T Bell Labs.
- 1978: Brian Kernighan and Dennis Ritchie published the first edition of The C Programming Language. (K&R)
- 1989: the C standard ratified as ANSI X3.159-1989 "Programming Language C". (ANSI C or C89)
- 1990: approved by the ISO as an international standard (C90)

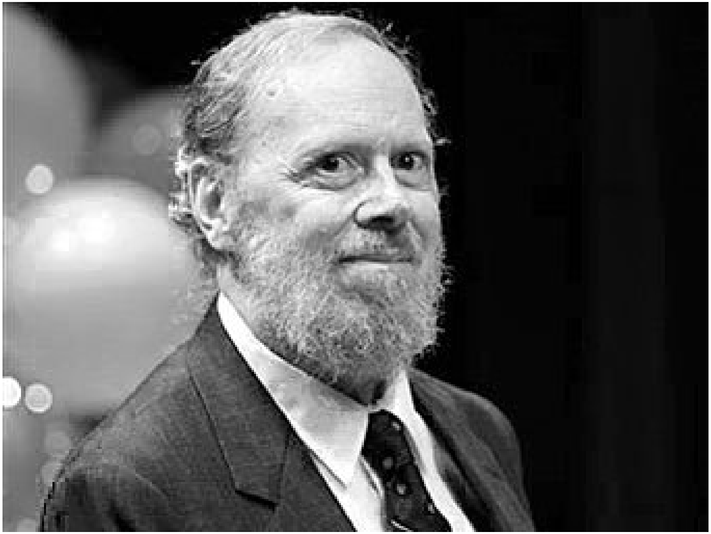
C main steps

- 1969 - 1973: developed by Dennis Ritchie at AT&T Bell Labs.
- 1978: Brian Kernighan and Dennis Ritchie published the first edition of The C Programming Language. (K&R)
- 1989: the C standard ratified as ANSI X3.159-1989 "Programming Language C". (ANSI C or C89)
- 1990: approved by the ISO as an international standard (C90)
- 1999: revised standart by ISO (C99)

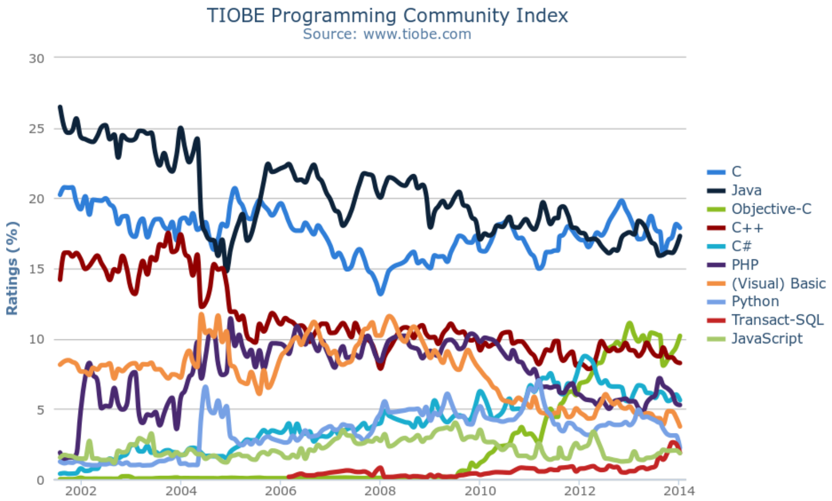
C main steps

- 1969 - 1973: developed by Dennis Ritchie at AT&T Bell Labs.
- 1978: Brian Kernighan and Dennis Ritchie published the first edition of The C Programming Language. (K&R)
- 1989: the C standard ratified as ANSI X3.159-1989 "Programming Language C". (ANSI C or C89)
- 1990: approved by the ISO as an international standard (C90)
- 1999: revised standart by ISO (C99)
- 2011: the current version approved by ISO (C11)

Dennis Ritchie, 1941 - 2011



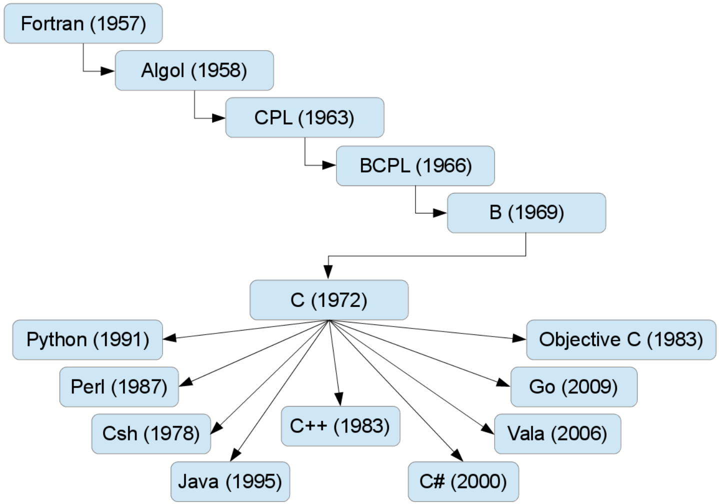
Position in TIOBE index



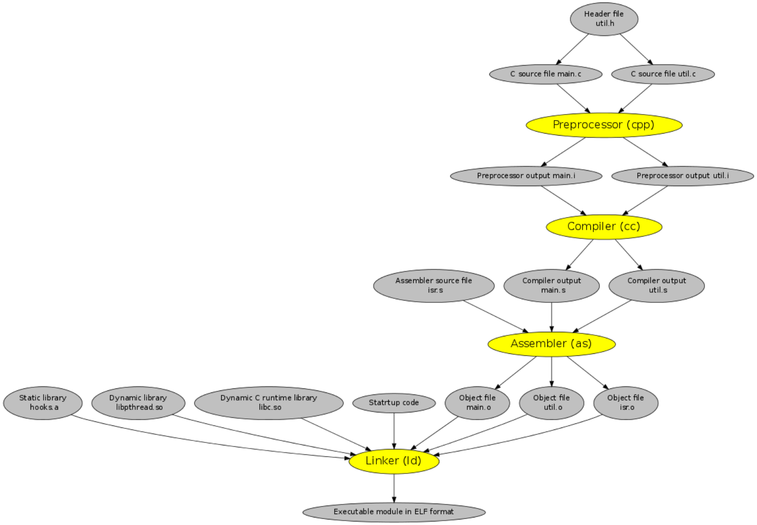
Spirit of C

- Trust the programmer.
- Don't prevent the programmer from doing what needs to be done.
- Keep the language small and simple.
- Provide only one way to do an operation.
- Make it fast, even if it is not guaranteed to be portable.

Languages based on C



Creation of executable module



GCC most frequent options

```
gcc [-c|-S|-E] [-std=standard]
    [-g] [-pg] [-Olevel]
    [-Wwarn...] [-Wpedantic]
    [-Idir...] [-Ldir...]
    [-Dmacro[=defn]...] [-Umacro]
    [-foption...] [-mmachine-option...]
    [-o outfile] [@file] infile...
```

GCC most frequent options

`-c`	Compile or assemble the source files, but do not link; the output is object files.
`-S`	Compile only; output assembly code.
`-E`	Pre-process only; output pre-processed code.
`-std=standart`	Determine the language standard; could be but not limited to `c89`, `c99`, `c11`, `gnu89` (default).
`-g`	Produce debug information.
`-pg`	Generate extra code to write profile information suitable for the analysis program gprof.
`-Olevel`	Optimization level from `0` (default) to `3`; also can be `-Ofast`, `-Os`.
`-Wwarn`	Enable `warn` warning; frequently used are: `all`, `pedantic`, `error`, `format`.
`-llibrary`	Links to a standard library; use -lm for maths library (libm.so).
`-Dmacro`	Define a macro, one can also use `-Dmacro=val`.
`-Umacro`	Undefine any previous definition of name, either built-in or provided with a `-D` option.
`-Idir`	Add the directory dir to the head of the list of directories to be searched for header files.
`-Ldir`	Add directory `dir` to the list of directories to be searched for `-l`.
`@file`	Read command-line options from file.
`-save-temps`	Save intermediate files in the current directory.

Hello world!

```c
/* compile with -save-temps to see intermediate files */
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hello, world!\n");
    return 0;
}
```

Undefined, unspecified, implementation-defined behaviour

The terms unspecified behavior, undefined behavior, and implementation-defined behavior are used to categorize the result of writing programs whose properties the Standard does not, or cannot, completely describe.

Undefined, unspecified, implementation-defined behaviour

Unspecified behaviour:

behavior where the standard provides two or more possibilities and imposes no further requirements on which is chosen in any instance.

```c
#include <stdio.h>

int sum(int a, int b)
{
    return a + b;
}

int main(void)
{
    int n = sum(printf("Hello "), printf("world!"));
    printf("\n%d\n", n);

    return 0;
}
```

Undefined, unspecified, implementation-defined behaviour

Implementation defined behavour:

unspecified behavior where each implementation documents how the choice is made.

```c
#include <stdio.h>

int main(void)
{
    printf("%zd\n", sizeof(int));
    printf("%zd\n", sizeof(long int));
    printf("%zd\n", sizeof(long long int));

    return 0;
}
```

Undefined, unspecified, implementation-defined behaviour

Undefined behaviour:

behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which the standard imposes no requirements.

```c
#include <limits.h>
#include <stdio.h>

int main(void)
{
    printf("%u\n", (INT_MAX + 1) > 0);

    return 0;
}
```