Member-only story

Are all Programming Languages in English?

Michele Riva · Follow
Published in ITNEXT
5 min read · May 10, 2021

Listen Share More

How fun would it be to write code in your favorite language? But is there any programming language that uses non-English keywords? Let's find out!

After some time spent searching for the right programming language for your project, you may have thought: “Hey, but is there a programming language with non-English keywords?”.

No? Ok, maybe I’m the only one wondering these strange things while doing researches for tools and languages for my projects!

But the question itself seems very interesting, so let’s find out.

A bit of History

When talking about programming (and programming languages in general), English seems to be a kind of lingua franca for writing computer programs. The reason is pretty obvious: modern programming languages are proud offsprings of other programming languages mainly developed in the United States in the mid of 20th century, so it seems natural that those languages evolved, maintaining the current language as a source vocables to be used as keywords.

A nice fact about programming languages: the first high-level programming was the Plankalkül. It has been developed by Konrad Zuse, a German engineer, between 1942 and 1945. Here it is a “Hello World” Program in Plankalkül:

```
R1.1(V0[:sig]) => R0
R1.2(V0[:m x sig]) => R0
0 => i | m + 1 => j
[W [ i < j -> [ R1.1(V0[i: m x sig]) => R0 | i + 1 => i ] ] ]
END
R1.3() => R0
'H';'e';'l';'l';'o';',';' ','w';'o';'r';'l';'d';'!' => Z0[: m x sig] R1.2(Z0) => R0
END
```

plankalkül hosted with ❤ by GitHub view raw

(code source: Twitter)

Starting from the 1950s, a lot of notable programming languages began to come to life. In 1955, in fact, FLOW-MATIC appeared for the first time. Its syntax may be pretty familiar:

```
(0)  INPUT INVENTORY FILE-A PRICE FILE-B ; OUTPUT PRICED-INV FILE-C UNPRICED-INV
    FILE-D ; HSP D .
(1)  COMPARE PRODUCT-NO (A) WITH PRODUCT-NO (B) ; IF GREATER GO TO OPERATION 10 ;
    IF EQUAL GO TO OPERATION 5 ; OTHERWISE GO TO OPERATION 2 .
(2)  TRANSFER A TO D .
(3)  WRITE-ITEM D .
(4)  JUMP TO OPERATION 8 .
(5)  TRANSFER A TO C .
(6)  MOVE UNIT-PRICE (B) TO UNIT-PRICE (C) .
(7)  WRITE-ITEM C .
(8)  READ-ITEM A ; IF END OF DATA GO TO OPERATION 14 .
(9)  JUMP TO OPERATION 1 .
(10)  READ-ITEM B ; IF END OF DATA GO TO OPERATION 12 .
(11)  JUMP TO OPERATION 1 .
(12)  SET OPERATION 9 TO GO TO OPERATION 2 .
(13)  JUMP TO OPERATION 2 .
(14)  TEST PRODUCT-NO (B) AGAINST ; IF EQUAL GO TO OPERATION 16 ;
    OTHERWISE GO TO OPERATION 15 .
(15)  REWIND B .
(16)  CLOSE-OUT FILES C ; D .
(17)  STOP . (END)
```

FLOW-MATIC hosted with ❤️ by GitHub view raw

(code source: Wikipedia)

yes, you guessed it! FLOW-MATIC is an ancestor of COBOL. As you can see, the English language is strictly part of the programming language itself (just like in COBOL).

The next big (I would say huge) programming language from those years was Fortran, developed in 1957 by IBM. Just like FLOW-MATIC, its keywords are written in English (`IF` , `GO TO` , `PROGRAM` , and so on).

In 1958, John McCarthy developed LISP, one of the oldest programming languages still in use today (together with Fortran). Just like the other programming languages of the time, it was developed with English keywords (`list` , `quote` , `if` , `or` , and so on).

The same year, ALGOL 58 came to life, and like all the previously mentioned programming languages, it used English keywords for building programs.

During these years, many other programming languages appeared on the scene: COBOL (1959), APL (1962), SIMULA (1962), CPL (1963), BASIC (1964), and many

others.

Why is that so important? Because these programming languages were the base for the modern languages that we love to use every day. In fact, more modern languages are an evolution of the programming languages born in the 1950s. CPL (with the B Programming Language) led to C. C led to C++, C#, PHP, Java, and many others. LISP led to Clojure, Racket, Scheme, and many other LISP dialects still widely used today. ALGOL 58 led to Pascal, Simula (the first Object-Oriented language), and so on.

So, with the evolution of programming languages, the English language has been adopted as a convention for writing computer programs. This doesn't mean that there weren't other programming languages (maybe in French, Spanish, or other languages); they didn't become as popular and widely adopted as the English-based programming languages.

Non-American Languages

We've just taken a look at the early days of programming languages. Most of them were developed in the US, but that doesn't mean that more modern languages are coming from there.

In fact, we can say that Object Oriented Programming has been invented in Norway by Ole-Johan Dahl and Kristen Nygaard while working with Simula in the early 1960s.

ML, the ancestor of Standard ML, OCaml, F#, ReasonML, and Coq, developed in France in 1973. ML (which has roots in LISP) improved the development of functional language drastically.

Pascal has been designed by Niklaus Wirth, a Swiss computer engineer in the 1970's.

The list of non-American programming languages continues with Python (Netherlands), Ruby (Japan), OCaml (France), Elixir and Lua (Brazil), and so on.

By the way, in the Online Historical Encyclopaedia of Programming Languages book, we can see that in 8900+ known programming languages, circa 2400 of them has been developed in the United States, ~600 in the UK, 160 in Canada, and 75 in Australia, all countries with the same mother language.

Symbolic Programming Languages

Symbolic Programming Languages use characters and symbols to represent concepts (such as loops, operations, conditionals, and so on) instead of commonly used keywords.

Most of the time, these languages can be considered as non-English-programming languages.

An example of a popular symbolic programming language is APL (do you remember? We mentioned it before!):

This is how a “Hello World” looks like in APL:

```
1  'Hello, world'
```

hello_world.apl hosted with ❤ by GitHub view raw

Ok, maybe that’s a trivial example; let’s see something more complex:

```
1  (~R∈R∘.×R)/R←1↓⍳R
```

primes.apl hosted with ❤ by GitHub view raw

the program above finds all the prime numbers from `1` to `R`. As you can see, no English keywords have been used for expressing a non-trivial problem.

Alonzo Church’s Lambda Calculus is another awesome example of how we can implement symbolic programming languages without using English keywords at all. In fact, Lamda Calculus emphasizes the use of anonymous functions, meaning that you’d be able to build even complex programs just using anonymous functions. An example, an implementation of multiplication of two numbers abstraction:

```
1  (λxyz.x(yz))
```

abstraction hosted with ❤ by GitHub view raw

Its application:

```
1  (λxyz.x(yz))22
```

application hosted with ❤ by GitHub view raw

Its reduction:

```
1   (λz.2(2z))
```

reduction hosted with ❤️ by GitHub view raw

The lambda expressions above can be translated in every language that supports anonymous functions, making it irrelevant which language you're speaking!

Many esoterical programming languages take advantage of symbolic programming for their syntax. The most famous example is Brainfuck:

```
1   +[-[<<[+[--->]-[<<<]]]>>>-]>-.---.>..>.<<<<-.<+.>>>>>.>.<<.<-.
```

hello_world.bf hosted with ❤️ by GitHub view raw

The code above prints "hello world" to the console. Other well known esoteric and symbolic programming languages are (with their "hello world" programs):

- Choon `AGb-A#A#+A+%A#DF-AC#`
- Elang `"" .H .e .l .l .o ., ._ .w .o .r .l .d .! ()`
- Integ `](104)](101)](108)](108)](111)](44)](32)](119)](111)](114)](108)](100)](10)`
- Ouroboros `S"Hello, World!"1( ewSoL!(`

And many, many others.

Non-English-Based Programming Languages

Now, let's come back to the original question: are programming languages all in English? The answer is: no. As we've just seen, the most used programming languages in the real world are based on the English language, but that doesn't mean that other languages can't be used for coding.

In fact, there are a wide number of programming languages that use non-English keywords.

Linotte

Linotte is an awesome example of a non-English-based programming language. It has been a developer for using french keywords, and its “Hello world” program looks like this:

```
BonjourLeMonde:
début
affiche "Bonjour le monde!"
```

linotte hosted with ❤ by GitHub view raw

SAKO

One example could be the SAKO (System Automatycznego Kodowania Operacji) programming language, which uses polish as for its keywords:

```
K) PROGRAM DRUKUJE NAPIS HELLO WORLD
   LINIA
   TEKST:
   HELLO WORLD
   KONIEC
```

SAKO hosted with ❤ by GitHub view raw

some times SAKO is described as “the Polish Fortran.”

Rapira

Rapira is another awesome example of non-english programming languages. It uses Russian keywords:

```
ПРОЦ СТАРТ()
    ВЫВОД: 'Привет, мир!'
КОН ПРОЦ
```

rapira hosted with ❤ by GitHub view raw

it has been developed for educational purposes in the Soviet Union and has been highly influenced by Algol.

Himawari

A famous Japanese programming language called ひまわり (Himawari) is another great example of Japanese coding:

```
1  「Hello, World!」と、表示。 ’母艦（メインフォーム）に表示。
2  「Hello, World!」と、言う。 ’ダイアログボックスで表示。
```

himawari hosted with ❤ by GitHub view raw

Qalb

Another interesting programming language is قلب (Qalb), as known as the Arabic LISP:

```
1  (قول "مرحبا يا عالم")
```

qalb hosted with ❤ by GitHub view raw

EPL (Easy Programming Language)

Chinese engineers developed 易语言 (Easy Programming Language, as known as EPL):

```
1  公开 类 启动类
2  {
3    公开 静态 启动()
4    {
5      控制台.输出("你好，世界！");
6    }
7  }
```

EPL hosted with ❤ by GitHub view raw

Conclusion

There are thousands of programming languages out there, and no matter where you come from, programming itself is like a universal language for both humans and computers.