Jurnal Computech & Bisnis, Vol. 8, No. 1, Juni 2014, 35-42
ISSN 2442-4943

EVOLUSI BAHASA PEMROGRAMAN
(Evolution of Programming Language)

Muhammad Taufiq Pratama
Departemen Teknik Komputer dan Informatika – Politeknik Negeri Bandung
Jln. Gegerkalong Hilir Ds. Ciwaruga Bandung 40551
Email: muhammad.taufiq.tif414@polban.ac.id

Abstract

Programming language, such a technology, always change from time to time. Evolution of programming language can't be released from its history. There have been always revision of a programming language, or any new programming language each decade, to fulfill human's need which is getting complex. Thus, the quantity of revisions or new languages occured can't be counted as easily anymore, hence reconstructing its history wouldn't be an easy thing to do as well. Resoluting this case, the Conference on the History of Programming Languages (HOPL) were held twice in USA in 1978 and 1993. On the first conference, the conference committee decides thirteen languages which fulfill criteria such as: having been in use for at least 10 years, had significant influence, and were still in use. On the second conference, the chosen programming languages were presented by the author of each chosen programming language to be documented in detail, which the result becomes an internationally approved history base of programming language. Yet of course the documented programming languages range are only those which occurred ten years before the second conference were held, hence the programming languages which occurred after then weren't documented by HOPL.

Keywords: *hopl; selection; conference*

Abstrak

Bahasa pemrograman, layaknya sebuah teknologi, selalu berubah dari waktu ke waktu. Evolusi dari bahasa pemrograman tidak terlepas dari sejarah bahasa pemrograman tersebut. Selalu terdapat revisi dari sebuah bahasa pemrograman, atau munculnya bahasa pemrograman baru setiap dekade, untuk memenuhi kebutuhan manusia yang semakin kompleks. Akibatnya, jumlah revisi maupun bahasa baru yang muncul tidak lagi dapat dihitung secara mudah, sehingga merekonstruksi sejarahnya menjadi hal yang tidak mudah pula. Mengatasi hal ini, pada diadakanlah Conference on the History of Programming Languages (HOPL) sebanyak dua kali di Amerika Serikat pada tahun 1978 dan 1993. Pada konferensi pertama, komite konferensi tersebut menentukan tiga belas bahasa pemrograman yang memenuhi kriteria berupa: telah digunakan selama setidaknya sepuluh tahun sebelum diadakannya konferensi, memiliki pengaruh yang signifikan terhadap perkembangan bahasa pemrograman, dan bahasanya masih digunakan hingga saat itu. Pada konferensi kedua, bahasa pemrograman yang terpilih dipresentasikan oleh para pengembang dari bahasa pemrograman terpilih untuk selanjutnya didokumentasikan secara rinci, yang hasilnya kemudian menjadi dasar sejarah bahasa pemrograman yang diakui secara internasional. Namun tentu saja lingkup bahasa pemrograman yang terdokumentasi hanya bahasa pemrograman yang

muncul sepuluh tahun sebelum konferensi kedua diadakan, sehingga bahasa pemrograman yang muncul setelahnya tidak dicantumkan dalam dokumentasi HOPL.

Kata kunci: hopl; seleksi; konferensi

PENDAHULUAN

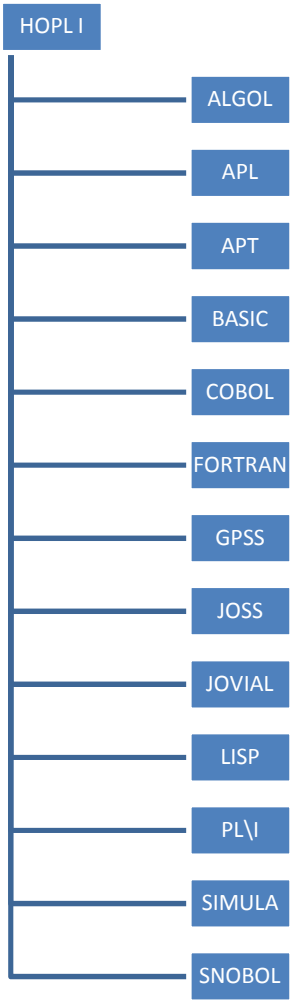
Seiring dengan perkembangan teknologi komputer, bahasa yang digunakan untuk merakit sistem dalam komputer pun berkembang setiap tahunnya. Bahasa pemrograman yang dikenal pada saat ini pada dasarnya berasal dari bahasa mesin yang dimodifikasi sintaksnya, sehingga bisa lebih mudah dipahami manusia. Bahasa pemrograman tingkat tinggi yang dikenal saat ini dapat dipahami oleh komputer dengan terlebih dahulu mengubah perintah-perintah yang diberikan pemrogram melalui *compiler* ataupun *interpreter*. Proses tersebut tentu tidak ditemukan begitu saja, pasti terdapat sejarah dari bahasa pemrograman sehingga dapat berevolusi menjadi bahasa tingkat tinggi yang kita kenal saat ini.

Sejarah bahasa pemrograman tingkat tinggi dimulai dengan munculnya pseudocode pada tahun 1949 yang mulai menggantikan pemrograman menggunakan bahasa mesin yang dirasa terlalu rumit. Dengan ditemukannya sistem penulisan yang mendekati bahasa manusia tersebut, mulailah muncul bahasa-bahasa tingkat tinggi pertama seperti FORTRAN (1957), LISP (1960), ALGOL (1958), COBOL (1960), dan BASIC (1964). Bahasa-bahasa tersebut pada dasarnya diciptakan atas semangat yang sama, yaitu bahasa yang digunakan diharapkan tidak hanya bisa mendekati bahasa manusia, namun juga bisa dimengerti oleh komputer dengan terlebih dahulu melalui proses *compiling*. Orang pertama yang menemukan *compiler* (penerjemah bahasa pemrograman tingkat tinggi ke bahasa mesin) adalah Grace Murray Hopper [Wexelbat, 1981]. Beliau pula yang menemukan istilah *debugging* dalam mengatasi *glitch* pada bahasa pemrograman.

Bahasa pemrograman yang telah ada saat itu terus berkembang dan bahasa-bahasa baru terus bermunculan. Perkembangan bahasa pemrograman yang pesat ini menarik perhatian lembaga ACM *Special Interest Group on Programming Languages* (SIGPLAN) dalam mengadakan konferensi untuk mendokumentasikan sejarah bahasa pemrograman hingga titik waktu tersebut, yang dinamakan *Conference on the History of Programming Languages* (HOPL). Konferensi dilaksanakan sebanyak dua kali. Konferensi pertama (HOPL) pada tahun 1978 memilih tiga belas bahasa pemrograman yang memenuhi kriteria sebagai berikut:

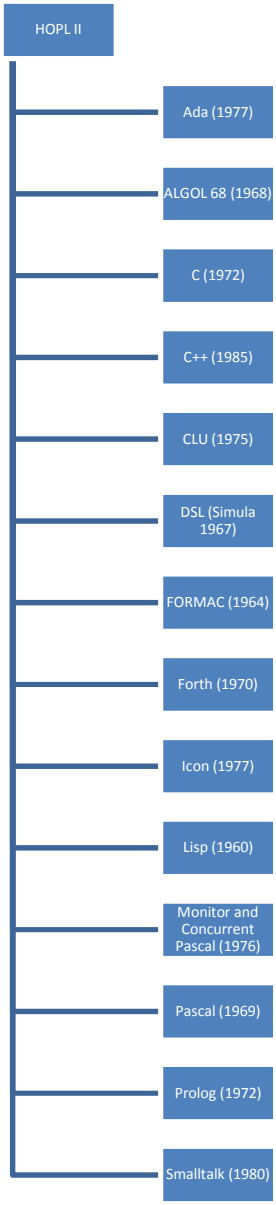
1. Telah digunakan selama setidaknya sepuluh tahun
2. Memiliki pengaruh yang signifikan terhadap bidang *programming*
3. Masih digunakan hingga saat konferensi diadakan

Dari ketiga kriteria tersebut, bahasa-bahasa pemrograman yang terpilih pada seleksi pertama adalah:



Gambar 1 Bahasa Pemrograman yang Terpilih dalam HOPL I

Konferensi kedua (HOPL-II) diadakan pada tahun 1993 untuk mematangkan konsep sejarah bahasa pemrograman yang bersifat kronologis. Terdapat perubahan yang cukup signifikan terhadap bahasa-bahasa pemrograman yang terpilih pada konferensi ini, mengingat celah waktu antara konferensi yang pertama dengan yang kedua cukup jauh, dan bahasa pemrograman terus mengalami perkembangan selama jangka waktu tersebut. Bahasa-bahasa pemrograman yang terpilih pada seleksi kedua adalah:



Gambar 2 Bahasa Pemrograman yang Terpilih dalam HOPL II

Dapat dilihat bahwa ada beberapa bahasa pemrograman yang mungkin seharusnya ada, namun tidak terdapat dalam HOPL-II. Misalnya, FORTRAN dan COBOL yang dipilih pada konferensi pertama, tidak terpilih dalam konferensi kedua. Hal ini dikarenakan bahasa tersebut masih mengalami perubahan yang signifikan pada saat diadakannya HOPL-II. Misalnya, bahasa FORTRAN baru disempurnakan pada tahun 1990 dengan dirilisnya versi FORTRAN 90, sedangkan bahasa yang

dipilih adalah bahasa yang telah relatif stabil dan telah digunakan sepuluh tahun sebelum konferensi kedua diadakan.

Bahasa yang Tercantum dalam HOPL-II

Secara singkat, berikut ini deskripsi dari bahasa pemrograman yang dipresentasikan dalam konferensi kedua (HOPL-II).

1. ALGOL 68

   Bahasa ini merupakan kelanjutan dari ALGOL 60, yang sebelumnya menjadi bahasa pertama yang tidak bergantung pada mesin (*machine independent language*). Bahasa ini merupakan bahasa yang sangat berpengaruh terhadap bahasa pemrograman berikutnya, terutama Pascal, C, dan Ada (yang secara tidak langsung berpengaruh pula terhadap bahasa-bahasa modern yang dipengaruhi oleh ketiga bahasa tersebut). Kontribusi dari bahasa pemrograman ini diantaranya adalah diterapkannya *user-defined data structures*, tipe referensi, dan array dinamik (*flex array*). Bahasa ini dibangun dengan harapan bidang penggunaannya akan lebih luas dengan banyaknya sintaks dan semantik yang disediakan. Namun sayangnya hal ini membuat ALGOL 68 lebih sulit untuk digunakan dibanding dengan versi sebelumnya, ALGOL 60, sehingga kalah bersaing dengan C dan Pascal yang relatif lebih umum digunakan karena pemakaiannya lebih mudah.

2. Pascal

   Bahasa ini dibangun pada tahun 1968-1969 oleh Niklaus Wirth yang keluar dari tim ALGOL 68. Pascal pada masanya merupakan bahasa prosedural yang sangat berpengaruh. Bahkan, Pascal masih digunakan dalam latihan pemrograman struktural dan struktur data hingga saat ini. Pascal dibangun karena kesadaran Niklaus Wirth terhadap mahasiswanya yang sulit beradaptasi dengan teknik pemrograman terstruktur dengan menggunakan bahasa yang ada (FORTRAN dan ALGOL). Bahasa ini didesain secara sederhana dan efisien, berbeda dengan ALGOL 68 yang juga dibangun pada tahun yang kurang lebih sama. Meskipun sederhana, Pascal pernah digunakan dalam pengembangan sistem operasi Macintosh dan aplikasi Skype.

3. Concurrent Pascal

   Bahasa ini dibangun oleh Per Brinch Hansen secara terpisah dari Pascal yang dibuat oleh Niklaus Wirth. Perbedaan yang signifikan dari Concurrent Pascal (atau sering disebut PASCAL-FC) adalah sistemnya yang bersifat konkuren, berbeda dengan Pascal yang bersifat sekuensial. Hal ini memungkinkan komputer untuk mengeksekusi beberapa perintah "sekaligus". Bahasa ini ditujukan untuk diaplikasikan dalam sistem operasi ataupun sistem monitor secara *real-time* dalam beberapa komputer yang bersifat *shared memory*. Pada dasarnya sintaks dan semantik yang digunakan sama dengan Pascal yang bersifat sequensial, hanya saja fitur-fitur seperti *variant records*, *goto statement*, *procedures as parameters*, *pointer types*, *packed arrays*, dan *file types* dalam Pascal yang bersifat sekuensial ditiadakan untuk membuat bahasa ini lebih sederhana.

4. Ada

   Bahasa ini sebenarnya dibangun oleh tim yang dipimpin Jean Ichbiah untuk United States Department of Defense (DoD) pada tahun 1977 hingga 1983 untuk menggantikan ratusan bahasa pemrograman yang sebelumnya digunakan oleh DoD. Nama bahasa pemrograman yang dipengaruhi oleh sintaks ALGOL 68

ini diambil dari orang yang dianggap sebagai pemrogram komputer pertama, Ada Lovelace [Fuegi dan Francis, 2003]. Pengembangan bahasa ini melibatkan ratusan orang, biaya besar, dan waktu pengembangan yang relatif lama. Kompiler yang pertama dibuat pada tahun 1977 sangatlah rumit. Kompiler yang dapat digunakan secara luas baru dirilis 5 tahun kemudian setelah desain bahasanya telah disempurnakan. Pada awalnya bahasa Ada difokuskan untuk diaplikasikan pada sistem *embedded and real-time*. Namun seiring perkembangannya Ada sekarang bisa digunakan untuk pemrograman berorientasi objek (OOP). Fitur yang pekat dalam bahasa ini adalah *strong typing*, mekanisme modular menggunakan *package*, *exception handling*, dan *runtime checking*.

5. Lisp

   Bahasa ini dibangun oleh John McCarthy pada tahun 1958 dan desainnya dipublikasikan pada tahun 1960, menjadikannya salah satu bahasa pemrograman tertua. Beberapa sumber mengatakan bahwa Lisp merupakan salah satu bahasa pemrograman terbaik karena strukturnya yang memudahkan pemrogram untuk memperluas bahasa yang digunakan ataupun mengimplementasikan dialek baru. Fitur yang dimiliki Lisp membuatnya digunakan secara luas dalam bidang penelitian kecerdasan buatan (AI). Kontribusi dari bahasa ini diantaranya penggunaan *tree data structures (functional programming)*, *automatic storage management*, *dynamic typing*, *conditionals*, *high-order functions*, *recursion*, dan *self-hosting compiler*. Namun kontribusi yang paling dominan adalah sebagai pionir dalam penggunaan fungsi dalam pemrograman.

6. Prolog

   Bahasa pemrograman ini pada awalnya tidak ditujukan untuk menjadi sebuah bahasa pemrograman, melainkan untuk memroses bahasa natural [Clocksin, 2003]. Prolog sendiri dibangun pada tahun 1972 oleh Colmerauer bersama dengan Philippe Roussel. Seiring perkembangannya, Prolog menjadi bahasa pemrograman pengolah logika yang berasosiasi dengan kecerdasan buatan dan komputasi linguistik. Hingga saat ini Prolog masih digunakan dalam pembuatan sistem pakar, pembuktian teori, dan pemrosesan bahasa natural karena kemampuannya dalam pemrosesan logika. Dalam Prolog, logika program diekspresikan dengan relasi dan pengolahan *query*.

7. Discrete Simulation Language

   Bahasa yang termasuk dalam kategori ini adalah bahasa yang memiliki model operasi dari sistem yang sifatnya diskrit. Dengan kata lain, setiap *event* dalam suatu waktu menandakan perubahan *state* dari suatu sistem, lalu simulasi hanya dilakukan pada *event* tertentu. Berbeda dengan *continous simulation* yang mendeteksi perubahan-perubahan yang terjadi pada sistem setiap waktunya. Karena *discrete simulation* melakukan simulasi lebih sedikit dibanding dengan *continous simulation*, maka *discrete simulation* akan bekerja lebih cepat dalam melakukan pemodelan. Contoh yang paling populer adalah bahasa Simula. Bahasa ini dibangun oleh Ole- Johan Dahl dan Kristen Nygaard sekitar tahun 1967. Simula dapat dikatakan sebagai bahasa pemrograman berorientasi objek (OOP) yang pertama karena fiturnya yang disebut *coroutine* (semacam subprogram) yang diimplementasi dalam struktur yang disebut *class*.

Struktur inilah yang nantinya menjadi acuan pemrograman berorientasi objek untuk bahasa-bahasa yang muncul berikutnya seperti C++, Java, dan C#.

8. FORMAC

   Bahasa ini dibangun oleh Jean E. Sammet bersama IBM pada tahun 1964. FORMAC merupakan bahasa pertama yang membuat simbol-simbol matematika dapat dikomputasikan secara praktis dan signifikan dalam pemrograman. FORMAC memiliki fitur untuk melakukan komputasi, manipulasi, dan penggunaan ekspresi simbolik dalam persoalan aljabar. Sistem inilah nantinya akan diadaptasi oleh FORTRAN IV dan membuat bahasa tersebut menjadi lebih sempurna. FORMAC menjadi sebuah ekstensi dari FORTRAN dan memiliki tipe data tersendiri, yaitu "FORMAC *variable*." Nama tersebut digunakan untuk membedakan operasi aljabar dengan operasi matematika biasa.

9. CLU

   Bahasa ini dibuat oleh Barbara Liskov dan murid-muridnya di MIT sekitar tahun 1974—975. CLU adalah bahasa pertama yang mengimplementasikan *constructor* untuk *abstract data types* (ADT), yang menjadi salah satu kunci dari pemrograman berorientasi objek (OOP). Namun terdapat beberapa kunci utama OOP yang tidak terdapat dalam CLU, sehingga CLU disebut sebagai bahasa "*object-based*", bukan sebagai "*object-oriented*." Konsep *class* dari Simula digunakan kembali dalam bahasa ini, sehingga CLU lebih mendekati bahasa OOP dibanding dengan Simula.

10. Smalltalk

    Bahasa ini dibangun dan dikembangkan di Xerox Palo Alto Research Center (Xerox PARC) oleh Alan Kay, kemudian dilanjutkan oleh Adele Goldberg. Smalltalk merupakan bahasa pertama yang mengimplementasikan seluruh kunci dari konsep OOP berupa data *abstraction*, *inheritance*, dan *dynamic type binding*. Disamping kelengkapan konsep OOP, Smalltalk merupakan pionir dari konsep *What You See is What You Get* (WYSIWYG). Dengan kata lain, Smalltalk adalah bahasa pemrograman yang berhasil mengimplementasikan *Graphical User Interface* (GUI). Konsep ini merupakan pintu menuju bahasa pemrograman generasi selanjutnya, yaitu pemrograman berbasis visual. *Built-in debugging* dan *object inspection tools* yang dimiliki oleh *environment* Smalltalk menjadi standar dari *Integrated Development Environment* yang ada saat ini.

11. Icon

    Bahasa ini dibangun oleh Ralph Griswold pada tahun 1977. Fitur utama dari Icon adalah pengolahan *string* dan pola tekstual. Icon adalah hasil pengembangan dari bahasa sebelumnya, yaitu SNOBOL yang juga merupakan bahasa pengolah *string*. Pada awalnya Icon tidak bersifat *object-oriented*, namun pada perkembangannya dibuatlah ekstensi yang mendukung sifat tersebut pada tahun 1996, yang dinamakan Idol, sehingga nantinya Icon berubah nama menjadi Unicon. Perbedaan yang cukup signifikan antara Icon dengan bahasa lain pada masanya adalah penggunaan konsep *goal-directed execution*, yaitu pengembalian ekspresi *success* atau *failure* dalam operasi logika, dibanding mengembalikan nilai boolean seperti bahasa umumnya. Icon memiliki fitur *string scanning* untuk mempermudah dalam membandingkan *string* dengan menghindari hal-hal mendetail yang tidak perlu dalam menganalisis *string*.

12. Forth

    Bahasa ini dibangun oleh Charles H. Moore pada sekitar tahun 1970. Berbeda dengan bahasa pemrograman berskala besar lainnya yang pengembangannya dibantu oleh sebuah perusahaan besar tertentu, pengembangan Forth hanya dilakukan oleh satu orang, yaitu pembuatnya sendiri. Namun dalam perkembangannya, bahasa ini menarik perhatian pemrogram dalam menciptakan *tools* untuk menyelesaikan masalah dalam sebuah aplikasi. Forth bersifat *stack-based*, yaitu sistem yang bergantung kepada model *stack machine* dalam melakukan *passing parameter*. Forth yang sifatnya portabel menjadi populer pada tahun 1980 karena cocok digunakan dalam mikrokomputer pada masanya.

13. C

    Bahasa ini dibangun oleh Dennis Ritchie dari AT&T Bell Labs pada tahun 1972. Bahasa C khusus dibuat untuk pemrograman secara umum dan menjadi bahasa yang paling luas digunakan di seluruh dunia [TIOBE, 2009]. Kompiler C dapat digunakan hampir di seluruh arsitektur komputer dan sistem operasi. Desain bahasa C menyediakan kompiler yang memberikan akses level rendah ke memori sehingga hasil kompilasi relatif lebih efisien untuk dipindahkan ke dalam bahasa mesin. Kemampuan bahasa C tersebut memunngkinkan terciptanya sistem operasi baru seperti UNIX. Bahkan Microsoft kemudian mengadopsi bahasa C untuk membuat sistem operasi Windows pada tahun 80-an [Suarga, 2012].

14. C++

    Bahasa ini dibangun oleh Bjarne Stroustrup di AT&T Bell Labs pada tahun 1985. Pada dasarnya, C++ bisa dikatakan sebagai pengembangan dari bahasa C yang diberi fitur OOP yang terdapat dalam SIMULA. Akibatnya, C++ menjadi bahasa yang besar dan kompleks karena mendukung baik *procedural programming* maupun *object-oriented programming*. Latar belakang dibangunnya bahasa ini adalah menciptakan program berorientasi objek yang dapat digunakan dalam pembuatan aplikasi berskala besar, berbeda dengan bahasa SIMULA yang hanya mampu mendukung pembuatan aplikasi berbasis OOP berskala kecil. Disamping itu, C++ didesain agar tidak hanya mudah digunakan seperti SIMULA, tetapi juga dapat mengungguli bahasa C dalam ketepatan koreksi logika program, *run-time speed*, dan kompresi ukuran kode yang dihasilkan.

KESIMPULAN

Konsep evolusi dari bahasa pemrograman dapat digambarkan sebagai proses saling mempengaruhi dan dipengaruhi antara satu bahasa pemrograman dengan bahasa pemrograman lainnya. Sebuah bahasa memelopori sebuah fitur yang akan memengaruhi paradigma bahasa pemrograman yang muncul selanjutnya, menghhasilkan kelebihan dan kekurangan dari setiap bahasa pemrograman yang ada saat ini. Di satu sisi, kita sebagai pembuat program yang bergantung kepada bahasa pemrograman yang ada selayaknya bersikap bijak dalam menentukan bahasa pemrograman yang digunakan. Kebutuhan yang berbeda dapat dieksekusi dengan fitur yang berbeda, sehingga memutuskan *initial state* dan *final state* merupakan langkah penting dalam menentukan bahasa pemrograman yang akan digunakan.

REFERENSI

Bergin, Thomas J. dan Richard G. Gibson. 1996. *History of Programming Languages*. Addison-Wesley.

Clocksin, William F. dan Christopher S. Mellish. 2003. *Programming in Prolog*. New York: Springer-Verlag.

Fuegi, J. dan J. Francis, "*Lovelace & Babbage and the creation of the 1843 'notes'*." Annals of the History of Computing 25 #4 (October–December 2003), hlm: 16-26.

Ierusalimschy, R.; De Figueiredo, L. H.; Celes, W. 2007. "*Proceedings of the third ACM SIGPLAN conference on History of programming languages - HOPL III*" http://www.lua.org/doc/hopl.pdf , hlm: 2–26.

McCarthy, John. 1979. *History of Lisp*. http://www.formal.stanford.edu/jmc/history/lisp/node3.html

TIOBE Programming Community. 2009. "*TIOBE Programming Community Index*",http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Wexelblat, Richard L. 1981. *History of Programming Languages*. New York: Academic Press.