Bab 1
Teknologi Informasi dan Java

Bab ini membahas tentang gambaran umum dunia teknologi informasi dengan maksud untuk memberikan pemahaman tentang posisi Java dalam teknologi informasi. Dengan memahami bab ini, anda bisa mengetahui berbagai komponen teknologi informasi secara umum dan kemudian memahami posisi dan ruang lingkup pekerjaan anda jika anda memutuskan untuk mendalami Java.

1.1 Memahami Teknologi Informasi

1.1.1 Pengertian Teknologi Informasi

Teknologi informasi sering disingkat dengan TI dan merupakan penyerapan istilah dari bahasa Inggris ***Information Technology (IT)***. Dalam praktik dan berbagai artikel serta buku, banyak istilah lainnya yang merujuk ke penyebutan TI, misalnya:

- Telematika (merujuk pada konvergensi antara telekomunikasi, media, dan informatika).
- TIK (Teknologi Informasi dan Komunikasi), berasal dari istilah ***ICT (Information and Communication Technology)***, sinonim dengan TI tetapi lebih menekankan pada telekomunikasi.

Berbagai istilah tersebut sebenarnya menunjukkan adanya kerancuan pemahaman. Istilah IT/TI/TIK/ICT masih bisa dipahami dalam kerangka industri, tetapi penggunaan istilah **Informatika** (dari bahasa Inggris: ***Informatics***) seharusnya hanya merujuk pada suatu disiplin keilmuan seperti halnya ***Mathematics*** – ***Matematika***, ***Statistics*** – ***Statistika***, dan lain-lain. Istilah **Informatika** lebih sesuai digunakan dalam kerangka proses pendidikan.

Buku ini akan menggunakan istilah teknologi informasi dan akan menggunakan TI jika akan disingkat. Penggunaan istilah TI akan merujuk ke hal yang sama seperti yang dimaksud oleh berbagai jargon tersebut di atas. Meskipun demikian, perlu diketahui bahwa tidak ada satu badan atau organisasi di dunia yang mempunyai "hak" untuk membuat standar istilah sehingga diferensiasi istilah ini akan tetap ditemui.

Definisi dari ITIL (*Information Technology Infrastructure Library*) yang dikembangkan oleh *Office of government Commerce* adalah sebagai berikut[1]:

> *The use of technology for the storage, communication or processing of information. The technology typically includes computers, telecommunications, Applications and other software. The information may include Business data, voice, images, video, etc. Information Technology is often used to support Business Processes through IT Services.*

Terjemahan bebas dalam bahasa Indonesia adalah sebagai berikut:

> Penggunaan teknologi untuk penyimpanan, komunikasi, atau pemrosesan informasi. Teknologi tersebut pada umumnya meliputi teknologi komputer, telekomunikasi, aplikasi, dan berbagai peranti lunak lainnya. Informasi mencakup data bisnis, suara, citra, video, dan lain-lain. Teknologi Informasi sering digunakan untuk mendukung proses bisnis melalui layanan TI.

1.1.2 Komponen Teknologi Informasi

Dari pengertian di atas, kita akan bergerak ke bagian yang lebih spesifik. TI pada

1 ITIL® V3 Glossary v01, 30 May 2007 - http://www.itil-officialsite.com/InternationalActivities/TranslatedGlossaries.asp

dasarnya terdiri atas berbagai komponen:

1. **Peranti keras komputer**, meliputi berbagai peranti yang saat ini bisa dikategorikan sebagai komputer: PC, *workstation*, NetBook, Laptop, dan lain-lain.
2. **Peranti telekomunikasi dan jaringan**, meliputi berbagai peranti yang membentuk infrastruktur telekomunikasi dan jaringan seperti *router, switch, wireless access point,* kabel, server, satelit, dan berbagai teknologi konektivitas jaringan lainnya.
3. **Peranti lunak**, yaitu peranti yang diinstalasi atau di-*embed* dalam peranti keras dan berbagai peranti telekomunikasi dan jaringan lainnya sehingga memungkinkan peranti-peranti tersebut dapat diprogram dan bekerja.

Peranti lunak (diterjemahkan dari kata *software* dan seringkali juga disebut sebagai *program*) biasanya bisa dikategorikan menjadi 3 bagian:

1. **Sistem operasi**, yaitu peranti lunak yang menjadi perantara antara peranti keras dan peranti lunak lainnya. Fungsi utama dari sistem operasi adalah pengelolaan berbagai sumber daya komputasi serta menyediakan abstraksi dalam bentuk API (*Application Programming Interface*) yang bisa digunakan oleh peranti lunak lainnya untuk mengendalikan kerja dari peranti keras. Contoh dari sistem operasi adalah Linux, FreeBSD, NetBSD, OpenBSD, MacOSX, Minix, Windows, dan lain-lain.
2. **Aplikasi**, yaitu peranti lunak yang digunakan untuk membantu manusia menyelesaikan pekerjaan dalam bidang tertentu, misalnya OpenOffice (http://www.openoffice.org) digunakan untuk keperluan aplikasi perkantoran dan bisa digunakan untuk menyelesaikan berbagai masalah komputasi perkantoran (pengolah kata, *spreadsheet*, dan presentasi).
3. **Utilitas**, yaitu peranti lunak yang menyediakan fasilitas-fasilitas untuk memperbaiki kinerja dari komputer, misalnya utilitas antivirus, utilitas pengatur partisi harddisk, dan lain-lain.
4. **Peranti pengembangan (*software development tools*)**, yaitu peranti yang digunakan untuk membuat peranti lunak (termasuk diantaranya untuk membuat sistem operasi, aplikasi, utilitas, maupun peranti pengembangan). Contoh dari kategori ini adalah JDK (Java Development Kit), GNU C++, Python, Ruby, Nasm (Netwide Assembler), Microsoft Visual Studio, Anjuta, KDevelop, dan lain-lain.

1.2 Peranti Pengembangan, Bahasa Pemrograman, Interpreter dan Kompiler

Jika membicarakan bahasa pemrograman (komputer) dan keterkaitannya dengan industri software, maka berarti kita bicara tentang spesifikasi bahasa pemrograman serta implementasinya. Penyebutan "bahasa pemrograman" sebenarnya hanya mengacu pada spesifikasi, yaitu aturan sintaksis serta tata bahasa dan kelengkapan bahasa pemrograman tersebut. Tentu saja hal itu tidak akan lengkap tanpa "implementasi". Implementasi dari spesifikasi bahasa pemrograman tersebut terletak pada peranti lunak ***interpreter / kompiler***. Jadi, jika kita bicara tentang **"Java"**, berarti sebenarnya kita membicarakan tentang spesifikasi bahasa pemrograman Java dan implementasi kompiler Java (**JDK, OpenJDK,** dan lain-lain).[2]

Fungsi dari interpreter / kompiler adalah menerjemahkan instruksi yang ditulis sesuai dengan spesifikasi bahasa pemrograman tertentu ke dalam bentuk target yang bisa dijalankan oleh mesin komputer. Bentuk hasil akhirnya bisa berupa berkas *executable* atau berkas yang bisa dijalankan secara langsung oleh sistem operasi, atau format tertentu (biasanya disebut sebagai *object code*) yang masih memerlukan mesin virtual untuk

2 Ada juga kompiler / interpreter yang namanya sama dengan spesifikasi bahasa pemrogramannya, misalnya Python, Ruby

menjalankannya.

Peranti pengembangan yang lengkap terdiri atas banyak bagian, meskipun secara minimal bisa dikatakan bahwa peranti pengembangan adalah kompiler / interpreter serta pustaka (*library*) standar. Kompiler / interpreter biasanya berkaitan dengan sintaksis standar sesuai spesifikasi bahasa pemrograman, misalnya untuk **if … else**, **while … do**, pengaturan variabel, dan lain-lain. Pustaka standar merupakan konstruksi program siap pakai yang langsung bisa digunakan, misalnya:

1. Java mempunyai API (*Application Programming Interface*) yang berisi pustaka standar untuk membangun aplikasi GUI dengan Swing, Kriptografi, operasi String, operasi Matematika, koneksi basis data dengan JavaDB, dan lain-lain.
2. Python mempunyai kelas siap pakai yang bisa digunakan untuk XMLRPC, JSON, dan lain-lain.

Pustaka standar ini yang membedakan "fasilitas" yang dimiliki oleh setiap peranti pengembangan. Semakin lengkap pustaka standar ini, semakin mudah kita menggunakan peranti pengembangan tersebut. Jika tidak terdapat pada pustaka standar, biasanya pemrogram harus mencari dari pihak ketiga, misalnya untuk koneksi Java menggunakan JDBC ke basis data PostgreSQL pemrogram harus menggunakan API yang bisa diperoleh di http://jdbc.postgresql.org karena tidak disediakan di pustaka standarnya.

1.3 Kategori Bahasa Pemrograman

Pada dasarnya pemrogram menggunakan bahasa pemrograman untuk menuliskan sintaksis dan kemudian menggunakan kompiler / interpreter untuk menterjemahkan ke bentuk bahasa yang dipahami mesin dan bisa dijalankan oleh mesin. Dengan demikian, kita bisa memahami bahwa mesin komputer itu sendiri mempunyai "bahasa". Ada banyak cara manusia untuk memprogram mesin supaya tidak langsung berurusan dengan bahasa mesin yang rumit. Pada bagian ini kita akan melihat kategorisasi bahasa pemrograman berdasarkan kedekatannya dengan bahasa mesin.

1.3.1 Bahasa Mesin

Instruksi bahasa mesin adalah satu-satunya instruksi yang dipahami oleh komputer. Instruksi ini merupakan suatu pola bit yang hanya terdiri atas angka 0 dan 1 (*binary digit*). Setiap perintah mempunyai suatu pola tersendiri. Bagian dari bahasa mesin yang menyebutkan operasi yang harus dilaksanakan disebut sebagai *opcode (**op**eration **code**)*. Bahasa ini berbeda-beda untuk setiap prosesor, sehingga untuk mendalami bahasa ini, berarti harus mendalami tipe prosesor tertentu dan tidak bisa diaplikasikan ke tipe prosesor lain. Berikut ini adalah contoh bahasa mesin untuk melompat ke alamat memory 1024[3]:

```
[  op  |        target address        ]
    2                 1024                 decimal
 000010 00000 00000 00000 10000 000000    binary
```

1.3.2 Bahasa Aras Rendah (Assembly)

Bahasa assembly merupakan penyederhanaan dari bahasa mesin dengan cara mengubah pola bit angka 0 dan 1 tersebut menjadi suatu kode yang bisa dibaca. Kode tersebut dikenal dengan istilah *mnemonic*. Suatu peranti lunak yang digunakan untuk mengubah instruksi bahasa assembly (mnemonic) ke dalam *opcode* serta mengatur alokasi

3 http://en.wikipedia.org/wiki/Machine_code

memory dari nama-nama simbolis dan entitas lainnya adalah **assembler**. Untuk mengubah ke bentuk yang siap dijalankan oleh mesin, biasanya dibutuhkan **linker**. Bahasa ini juga sangat tergantung pada mesin dan platform sistem operasi. Di Linux, bisa digunakan **Nasm**, **GNU As** sebagai assembler dan **GNU ld** sebagai linker. Di Windows, bisa digunakan **TASM** dan **MASM** untuk keperluan ini. Berikut adalah contoh dari bahasa assembly:

```
…
…
MOV AL, 61h
…
…
```

Instruksi di atas digunakan untuk mengisi register AL dengan nilai hexadecimal 61. Jika diterjemahkan ke bahasa mesin, mnemonic MOV di atas akan menjadi "B0 61" dalam hexadecimal atau 10110000 01100001 dalam biner atau bahasa mesin.

Untuk bisa memprogram dalam bahasa mesin dan assembly, pemrogram perlu membaca manual prosesor yang bersangkutan. Sebagai contoh, jika menggunakan prosesor dari Intel, instruksi-instruksi di atas bisa diperoleh di website dari Intel di URL http://www.intel.com/products/processor/manuals/

1.3.3 Bahasa Tingkat Menengah

Bahasa tingkat menengah (*middle level language*) adalah bahasa yang memungkinkan untuk membuat aplikasi di level pemakai maupun untuk melakukan pemrograman yang mengakses sistem dan sumber daya komputer melalui perantara sistem operasi. Bahasa ini memang dikatakan bahasa tingkat menengah karena mempunyai kemampuan baik untuk mengakses sistem secara langsung (biasanya melalui *inline assembly* atau langsung menuliskan *mnemonic* di kode sumber) dan di sisi lainnya bahasa ini mempunyai tingkat kemudahan yang mendekati bahasa tingkat tinggi (menggunakan istilah bahasa manusia dalam pemberian perintahnya). Bahasa C adalah bahasa yang masuk dalam kategori ini. Berikut ini adalah contoh dari Bahasa C menggunakan kompiler GCC yang menyertakan bahasa assembly[4]:

```
#include <stdio.h>

int main(void) {
        int foo=10,bar=15;

        __asm__ __volatile__ ("addl     %%ebxx,%%eax"
                : "=eax"(foo)            // ouput
                : "eax"(foo), "ebx"(bar)// input
                : "eax"                  // modify
        );
        printf("foo+bar=%d\n", foo);
        return 0;
}
```

4 Diambil dari maual / *info page* dari GCC

1.3.4 Bahasa Tingkat Tinggi

Bahasa tingkat tinggi adalah bahasa pemrograman yang mempunyai sintaks mirip dengan bahasa manusia dan mempunyai tingkat abstraksi penyelesaian masalah yang jauh dari bahasa mesin ataupun assembly. Bahasa ini biasanya bisa dikategorikan ke dalam:

1. *General purpose high level programming language*, yaitu bahasa pemrograman dengan tujuan penggunaan umum atau bisa digunakan untuk membuat berbagai jenis aplikasi. Contoh dari bahasa pemrograman ini antara lain adalah Java, Pascal, Python, Ruby, dan lain-lain.
2. *Special purpose high level programming language*, yaitu bahasa pemrograman dengan tujuan khusus dan spesifik untuk menyelesaikan masalah dalam suatu bidang tertentu, misalnya Prolog untuk pemrograman logika atau kecerdasan buatan, PHP untuk aplikasi web (meskipun ada yang non web, yaitu PHP-GTK, tetapi penggunaan utamanya untuk web),

1.4 Paradigma Pemrograman

Paradigma pemrograman berkaitan dengan cara pandang dalam menyelesaikan masalah dengan bahasa pemrograman. Cara pandang ini akan mengakibatkan berbagai perbedaan antar paradigma dalam hal abstraksi dan berbagai konsep yang merepresentasikan elemen dari suatu program (seperti obyek, fungsi, variabel, dan lain-lain) serta berbagai langkah atau proses komputasi (penugasan, evaluasi ekspresi, alur kendali, dan lain-lain). Bagian ini akan menguraikan sedikit dari beberapa paradigma tersebut.

1.4.1 Pemrograman Prosedural

Pemrograman prosedural sering diasosiasikan dengan pemrograman terstruktur dan mengacu pada cara menyelesaikan masalah dengan membuat langkah-langkah terstruktur (disebut algoritma) dan kemudian mengimplementasikan langkah-langkah tersebut dalam berbagai perintah dan prosedur yang akan dipanggil sesuai dengan urutan eksekusi program. Setiap prosedur tersebut berisi rangkaian perintah dalam bahasa pemrograman yang bersangkutan. Biasanya rangkaian langkah-langkah tersebut dijabarkan dalam suatu *pseudo code*. Sebagai contoh, untuk menghitung biaya depresiasi tiap periode dilakukan dengan melihat pada rumus “Biaya Depresiasi = (Kos Aset – Nilai Residu)/Umur Ekonomis” dan diwujudkan dalam algoritma berikut:

1. Mulai
2. Masukkan / *input* kos aset
3. Masukkan / *input* nilai residu
4. Masukkan / *input* umur ekonomis
5. Hitung biaya depresiasi = (kos aset – nilai residu) / umur ekonomis
6. Tampilkan hasil perhitungan biaya depresiasi ke layar

Setelah itu, setiap bagian dan langkah di atas diterjemahkan ke dalam kode bahasa pemrograman. Contoh-contoh bahasa pemrograman yang bisa digunakan untuk mengimplementasikan algoritma di atas antara lain adalah Pascal, Python, BASIC, dan lain-lain. Berikut ini adalah hasil penerjemahan ke dalam bahasa pemrograman Python untuk algortima di atas:

```
kos_aset = float(raw_input("Masukkan kos aset = "))
nilai_residu =float(raw_input("Masukkan nilai residu = "))
umur_ekonomis = float(raw_input("Umur ekonomis = "))
biaya_depresiasi = (kos_aset - nilai_residu) / umur_ekonomis
print "Biaya depresiasi setiap periode = " + str(biaya_depresiasi)
```

1.4.2 Pemrograman Berorientasi Obyek

Pemrograman berorientasi obyek (diterjemahkan dari "*Object-Oriented Programming*") adalah paradigma pemrograman yang berusaha untuk menyelesaikan masalah pemrograman dengan cara membuat abstraksi dari berbagai obyek yang ada dalam suatu masalah kemudian mendefinisikan interaksi antar obyek dalam menyelesaikan masalah tersebut. Setiap obyek merupakan suatu manifestasi dari kelas yang merupakan cetak biru dari obyek yang bersangkutan. Suatu kelas terdiri atas berbagai atribut / karakteristik umum dan berbagai perilaku / *behaviour* / *method* dari kelas tersebut. Kedudukan antara kelas dengan obyek adalah kedudukan cetak biru serta *instance* dari cetak biru tersebut. Sebagai contoh, terdapat kelas manusia dan Bambang adalah manusia, untuk contoh tersebut kita bisa mendefinisikan kelas manusia dan Bambang adalah salah satu *instance* dari kelas manusia tersebut. Beberapa bahasa pemrograman yang mendukung PBO sebagai paradigma utamanya adalah Java, C++, C#, Ruby, Smalltalk, dan lain-lain. Paradigma ini yang akan menjadi pembahasan utama dari buku ini.

1.4.3 Pemrograman Fungsional

Pemrograman fungsional (diterjemahkan dari "*Functional Programming*") adalah paradigma pemrograman yang berusaha menyelesaikan masalah pemrograman dengan melalui fungsi (*function*) matematis dan menghindari *mutable data*. Jadi, tidak seperti paradigma pemrograman prosedural yang mengutamakan berbagai tipe data dan kemudian membuat prosedur atau fungsi untuk memanipulasi data tersebut. Paradigma ini memang cenderung lebih "matematis" dan akan lebih mudah dipahami jika pemrogram yang belajar paradigma ini memahami matematika. Paradigma ini berasal dari suatu sistem formal yang disebut dengan *Lambda Calculus*. *Spreadsheet* adalah salah satu contoh dari pola pikir fungsional ini, terutama dengan melihat pada data yang terdapat pada *spreadsheet* serta berbagai elemen rumus-rumus di berbagai sel di *spreadsheet*. Contoh bahasa pemrograman yang mempunyai paradigma pemrograman fungsional sebagai paradigma utamanya adalah Haskell, Erlang, OCaml, Lisp, Scheme, Scala, Clojure, F#, dan lain-lain.

1.4.4 Beberapa Pandangan Lain dalam Paradigma Pemrograman

Perlu diketahui bahwa pembagian menjadi beberapa paradigma ini merupakan pembagian yang bersifat subyektif. Selain paradigma-paradigma di atas, masih ada beberapa lagi yang sering dianggap sebagai paradigma tersendiri.

Pemrograman Imperatif

Pemrograman imperatif sering kali dianggap sama dengan pemrograman prosedural, meskipun demikian, ada juga yang menyatakan lain. Perbedaan dengan pemrograman prosedural terutama terletak pada penggunaan prosedur (atau sub program atau *subroutines* atau *functions*) yang banyak untuk memperbaiki keterbacaan (*readability*) dan kemudahan pemeliharaan program (*maintainability*). Model yang lebih tegas lagi dengan menggunakan aturan penggunaan variabel lokal di level prosedur sebenarnya juga merupakan bagian dari pemrograman imperatif tetapi lebih sering dimasukkan dalam

pemrograman terstruktur (*structured programming*). Beberapa bahasa pemrograman yang mendukung paradigma ini sama dengan bahasa pemrograman yang mendukung paradigma pemrograman prosedural hanya teknik pemrogramannya yang berbeda.

Pemrograman Deklaratif

Berbeda dengan pemrograman imperatif yang secara eksplisit cenderung untuk menetapkan algoritma untuk menyelesaikan suatu masalah pemrograman, pemrograman deklaratif cenderung untuk mengekspresikan logika komputasi tanpa mendefinisikan alur kendali program. Pemrograman logika menggunakan Prolog merupakan salah satu contoh paradigma pemrograman deklaratif. Paradigma pemrogaman ini cenderung lebih sesuai dipelajari jika ranah aplikasi yang dibangun melibatkan matematika logika. Contoh dari bahasa pemrograman yang mempunyai paradigma utama pemrograman deklaratif antara lain adalah Prolog,

Pemrograman *Event-Driven*

Pemrograman *event-driven* adalah suatu paradigma pemrograman yang mengatur aliran program melalui berbagai *event* / kejadian. Paradigma ini kadang kala juga disebut dengan paradigma pemrograman visual. Pemrogram biasanya membuat form dan meletakkan berbagai komponen (menu, *pushbutton*, *radio button*, *text area*, dan lain-lain) ke dalam form tersebut kemudian mengisikan rangkaian perintah yang dijalankan jika terdapat kejadian yang berkaitan dengan komponen-komponen tersebut (misalnya menu dipilih, *push button* di klik, form di tutup, dan lain-lain). Contoh peranti pengembangan yang masuk dalam kategori ini antara lain adalah Borland Delphi (dengan bahasa Pascal), QT Designer (dengan bahasa C++), Swing di NetBeans (dengan bahasa Java), dan lain-lain.

1.4.5 Kondisi Saat Ini

Pada dasarnya, saat ini jarang ada peranti pengembangan yang hanya menggunakan satu paradigma saja. Saat ini kebanyakan peranti pengembangan merupakan peranti pengembangan yang *multiparadigm* atau menganut lebih dari satu paradigma meskipun biasanya ada satu paradigma yang dijadikan dasar dari peranti pengembangan tersebut. Beberapa contoh dari *multiparadigm* tersebut adalah sebagai berikut:

1. Python, mempunyai dasar paradigma pemrograman berorientasi obyek, tetapi juga mendukung prosedural maupun fungsional.
2. OCaml, mempunyai dasar paradigma pemrograman fungsional, tetapi juga mendukung pemrograman berorientasi obyek.
3. Java, mempunyai dasar paradigma pemrograman berorientasi obyek, tetapi juga mendukung pemrograman *event-driven*.

1.5 Posisi Java dalam Dunia Pemrograman

Java sebenarnya bisa dimasukkan dalam kategori bahasa pemrograman tingkat tinggi dan secara umum termasuk dalam kelompok paradigma pemrograman berorientasi obyek. Peranti pengembangan Java merupakan bagian dari peranti pengembangan yang digunakan untuk membangun berbagai aplikasi maupun berbagai kategori peranti lunak lain (misalnya sistem operasi dengan Jnode). Dengan demikian, jika tujuan anda adalah membangun aplikasi, maka peranti pengembangan Java ini akan menjadi wilayah pekerjaan anda.

Perlu diketahui, di Internet terdapat beberapa situs web yang memang dibangun untuk mengetahui posisi berbagai peranti pengembangan di dunia. Salah satu dari situs web tersebut antara lain adalah indeks menurut Tiobe. Tiobe bisa diakses melalu URL

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. Berikut ini adalah posisi bulan Oktober 2010:

Position Oct 2010	Position Oct 2009	Delta in Position	Programming Language	Ratings Oct 2010	Delta Oct 2009	Status
1	1	=	Java	18.166%	-0.48%	A
2	2	=	C	17.177%	+0.33%	A
3	4	↑	C++	9.802%	-0.08%	A
4	3	↓	PHP	8.323%	-2.03%	A
5	5	=	(Visual) Basic	5.650%	-3.04%	A
6	6	=	C#	4.963%	+0.55%	A
7	7	=	Python	4.860%	+0.96%	A
8	12	↑↑↑↑	Objective-C	3.706%	+2.54%	A
9	8	↓	Perl	2.310%	-1.45%	A
10	10	=	Ruby	1.941%	-0.51%	A
11	9	↓↓	JavaScript	1.659%	-1.37%	A
12	11	↓	Delphi	1.558%	-0.58%	A
13	17	↑↑↑↑	Lisp	1.084%	+0.48%	A-
14	24	↑↑↑↑↑↑↑↑↑↑	Transact-SQL	0.820%	+0.42%	A-
15	15	=	Pascal	0.771%	+0.10%	A-
16	18	↑↑	RPG (OS/400)	0.708%	+0.12%	A-
17	29	↑↑↑↑↑↑↑↑↑↑↑↑	Ada	0.704%	+0.40%	A--
18	14	↓↓↓↓	SAS	0.664%	-0.14%	B
19	19	=	MATLAB	0.627%	+0.05%	B
20	-	↑↑↑↑↑↑↑↑↑↑	Go	0.626%	+0.63%	B