Asiknya menjadi programmer!

(Selamat datang, mahasiswa-mahasiswi baru S1 ilmu komputer)

Kenapa?

- Bangun sesuatu dari nol

  (seperti menulis buku dan mengarang lagu)

- Dimana, hasil akhirnya dapat digunakan oleh banyak orang

  (pada **miliaran*** perangkat: komputer, ponsel, televisi, *home entertainment*, mobil, pesawat, kulkas, dan lainnya)

- Dan, Anda dapat memulainya sendirian

  (apabila Anda sungguh-sungguh; dan bersama tim nantinya)

* Jumlah pengguna Internet saja telah mencapai 3,2 Milyar pada tahun 2015, menurut sebuah laporan dari International Telecommunication Union (http://www.itu.int/net/pressoffice/press_releases/2015/17.aspx)

Lalu?

- Ketahui apa yang ingin Anda bangun

  (dari yang mendasar seperti sistem operasi / sistem database; yang menjadi bagian strategis dari bisnis seperti ERP; yang menghibur seperti permainan komputer; yang menyentuh banyak orang seperti layanan berbasis Internet; Anda tentukan sendiri!)

- Persiapkan diri Anda :)

Mari kita mulai!

- Tentukan platform*

  (misal: dimana software tersebut akan berjalan)

- Rancang aplikasi anda

  (kadang tidak terpikirkan ketika baru memulai)

- Pilih teknologi

  (suatu saat, Anda mungkin menemukan teknologi baru)

- Jangan lupakan teori

  (alias: kuliah tidak hanya membuang waktu)

* Cakupannya bisa luas. Dalam batasan tertentu, ini bisa berupa sistem operasi, framework pengembangan aplikasi, dan lainnya

Konkritnya?

- Mari mulai dari program kecil

  (sebagai contoh, aplikasi kasir / penjualan ritel)

- Asumsikan aplikasi akan dijalankan di desktop (Windows, Linux, macOS)

  (bisa semuanya, bisa satu saja → Anda tentukan)

- Rancangan aplikasi? Teori pendukung? Apa itu?
- Pilih bahasa pemrograman. Pilih IDE yang menawarkan GUI builder.

  (Anda mungkin menghasilkan sebuah program tanpa menulis sebaris kode pun!)

Bukan hanya ‘Hello World’

- Aplikasi Anda akan bekerja dengan sistem lain:
  - Database
  - Perangkat keras (langsung atau tidak) misal printer
  - Sistem eksternal (misal lewat Internet)
- Anda dapat memanfaatkan pustaka bawaan, ataupun pustaka dari pihak lain

  (DRY vs WET)
- Memahami bidangnya akan lebih baik

  (Alias: Anda tahu betul pekerjaan di kasir dan yang terkait; apabila Anda membangun game, Anda mungkin seorang gamer)

Bebas bug?

- Program ‘Hello World’ satu baris mungkin bebas dari bug

  (dari satu penelitian, bisa 6 sampai 16 bug per 1000 baris; penelitian lain 2 sampai 75 bug per 1000 baris)*

- Lakukanlah pengujian secara menyeluruh

  (suatu hari, Anda mungkin tertarik dengan Test-Driven Development)

- Software Anda bisa merugikan atau membahayakan pihak lain

  (toko ritel mungkin dirugikan secara finansial; tapi bagaimana kalau Anda membangun aplikasi untuk sistem transportasi atau kesehatan?)

* Tanenbaum, A. S., Herder, J. N., & Bos, H. Can We Make Operating Systems Reliable and Secure?.

Bekerja dalam tim

- Anda akan butuh software Source Code Management

  (misal untuk mengelola revisi; atau ketika beberapa programmer bekerja pada fungsionalitas yang sama)

- Tidak hanya *menulis kode*
- Anda mungkin bekerja lintas bahasa pemrograman, lintas sistem operasi, lintas arsitektur komputer, dan lainnya

  (bacalah juga tentang microservices)

Lisensi

- Ketika didistribusikan, Anda menentukan bagaimana software Anda dilisensikan
  - Proprietary
  - Free/Open Source Software
  - (dan aneka bentuk lisensi lainnya)
- Software bisa tidak didistribusikan

  (sebagai contoh, Anda membangun layanan berbasis Internet, dan aplikasi diakses lewat web browser)

Bahasa pemrograman

- Masa-masa komputer elektronik pertama (1940-an)
  - Programmer memandang waktu komputer lebih berharga → bahasa mesin
  - Program membesar → bahasa-bahasa assembly
  - Agar tidak ditulis ulang untuk setiap arsitektur dan dekat dengan bahasa manusia → bahasa pemrograman high level
- Mengutip Wikipedia: sejak 1950-an, lebih kurang 50 bahasa pemrograman baru lahir setiap dekade

Bahasa pemrograman: Popularitas

- Menurut TIOBE, tiga bahasa pemrograman terpopuler berikut tidak pernah berubah posisi selama lebih 15 tahun (sejak 2001): Java, C, C++
- Indeks bahasa pemrograman (cuplikan 1 sampai 10; pada saat slide ini dibuat):

1	Java
2	C
3	C++
4	C#
5	Python
6	PHP
7	JavaScript
8	Visual Basic .NET
9	Perl
10	Ruby

Sumber: TIOBE Programming Community Index, https://www.tiobe.com/tiobe-index/ (diakses pada 16-Sep-2017)

Bahasa pemrograman: Pilih yang mana?

- TIOBE Programming Community Index adalah salah satu indikator saja. Terdapat yang lainnya.

  (atau, Anda bisa saja tidak peduli akan seberapa populer suatu bahasa)

- (1) Apabila bergabung ke tim tertentu, atau membangun aplikasi di platform spesifik → Anda mungkin tidak punya banyak pilihan
- (2) Apabila punya banyak pilihan:
  - Pilih yang paling disukai/dikuasai, paling bisa dipelajari bersama, paling cocok untuk kebutuhan tertentu, dan lainnya
  - Popularitas bukanlah penentu*. Bagaimana aplikasi dirancang akan lebih penting.

* Carilah informasi, dengan bahasa apa (sebagian besar) suatu aplikasi populer dikembangkan. Bagaimana aplikasi tersebut dirancang, atau seberapa kompleks aplikasi tersebut, Anda mungkin menemukan bahwa subsistem/komponen tertentu ditulis dengan bahasa pemrograman yang belum pernah Anda dengar.

Bahasa pemrograman: Mempelajari bahasa ‘low level’?

- Bahasa pemrograman populer tertentu, seperti C, dapat dikategorikan low level (pada konteks tertentu, misal dibandingkan dengan Python)
- Keuntungan mempelajari bahasa yang lebih low level (misal C):
  - Anda akan lebih paham bagaimana sistem operasi/komputer bekerja (seperti contoh: layout memori)
  - Bahasa tertentu, seperti C, adalah *architecture-independent*
- Kerugiannya:
  - Lebih banyak hal harus dikerjakan sendiri

    (tentu saja, Anda bisa mempergunakan pustaka yang sudah ada)
  - Umumnya, dibutuhkan lebih banyak langkah (dibandingkan bahasa yang lebih high level) untuk mencapai sesuatu

Anda dapat membangun aplikasi web dengan C. Anda mungkin bisa membangun device driver dengan Python. Tapi bahasa tertentu mungkin lebih cocok, dan Anda mungkin ingin lebih konsentrasi pada produk yang dihasilkan.

Tentang teori

- Suatu hari, Anda mungkin menemukan teknologi baru
  - Butuh pemahaman yang baik, termasuk dari sisi teori
- Ingin yang mudah dan cepat jadi dalam membangun software? Barangkali ke depannya akan lebih merepotkan.

Maintainability

- Anda mungkin menulis program sampai puluhan tahun kemudian

  (walau bukan selalu berarti: software yang ditulis akan dapat berjalan puluhan tahun kemudian apa adanya, tapi Anda mungkin peduli dengan software maintainability)

Terima kasih