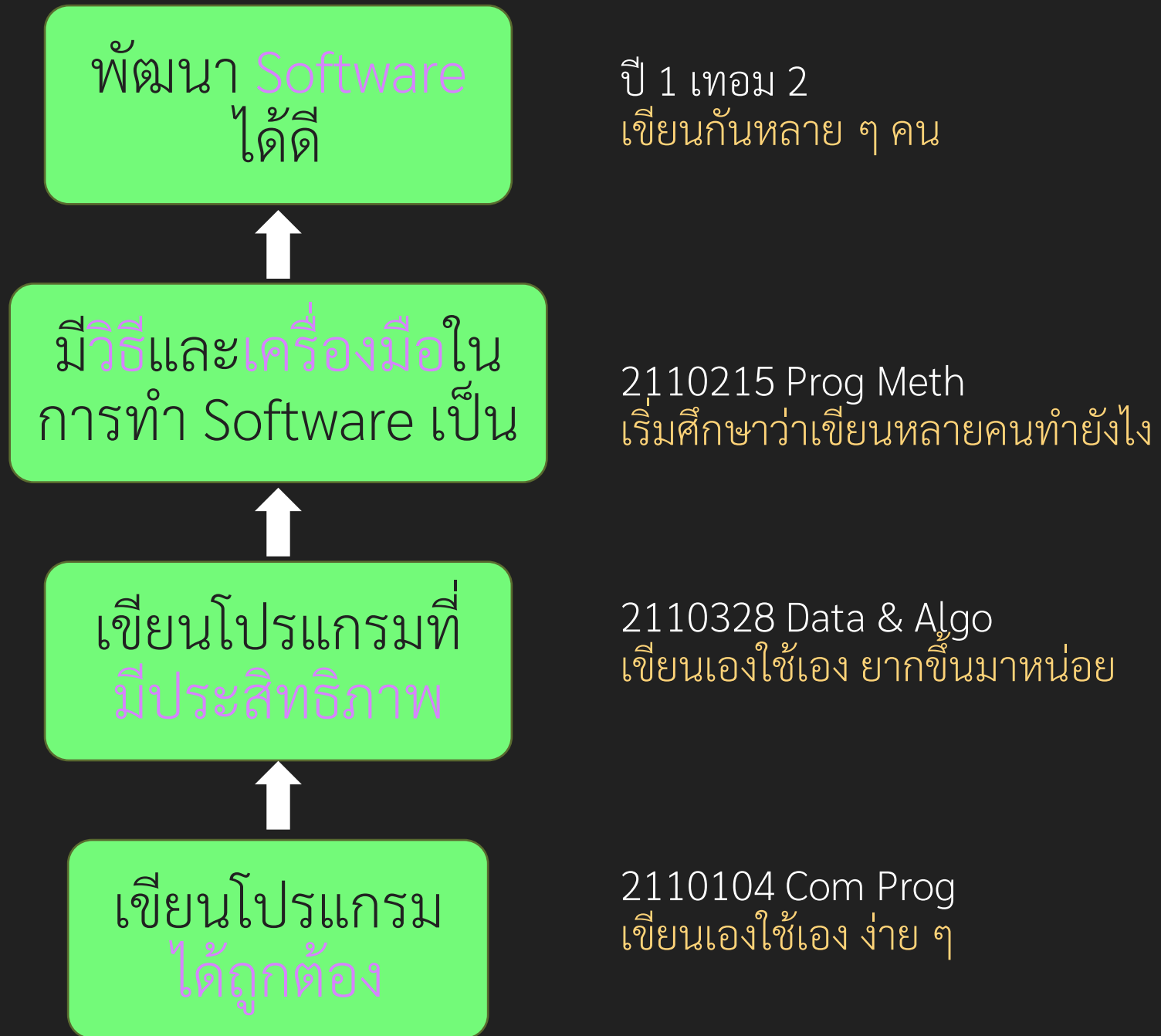


Introduction to C++

2110104 Computer Programming

เป้าหมาย



เป้าหมาย

- อ่านโปรแกรมเป็น
 - เข้าใจพื้นฐาน และ หลักการทำงาน
- เขียนโปรแกรมเป็น
 - เข้าใจโจทย์
 - เขียน program ที่ทำงานตามที่โจทย์อยากให้ทำ
 - รู้ว่าต้องทำอะไร

ทำอะไร?

- ทำความเข้าใจ หลักการพื้นฐาน ในการเขียนโปรแกรม
- ศึกษา ภาษา C++ (บางส่วน เอาเฉพาะที่จำเป็น)
 - เข้าใจ syntax
 - ฝึกอ่านโปรแกรม
 - ฝึกเขียนโปรแกรม
- ทำโจทย์ปัญหาต่าง ๆ
 - ส่งตรวจ

ทำไมต้อง C++

- เป้าของวิชาไม่ได้เน้นที่ภาษา
 - แต่ก็ต้องมีสัก 1 ภาษาที่เราเลือกใช้เป็นสื่อกลาง
- ทำไม C++
 - ทำให้เข้าใจพื้นฐานการเขียนโปรแกรมได้ดี
 - หลายภาษาที่ใช้ในปัจจุบันมีความคล้ายคลึงกับ C++ เช่น Java, C, C#, JS
 - ตลาดงานกว้าง

Jul 2023	Programming Language		Ratings
1		Python	13.42%
2		C	11.56%
3		C++	10.80%
4		Java	10.50%
5		C#	6.87%
6		JavaScript	3.11%

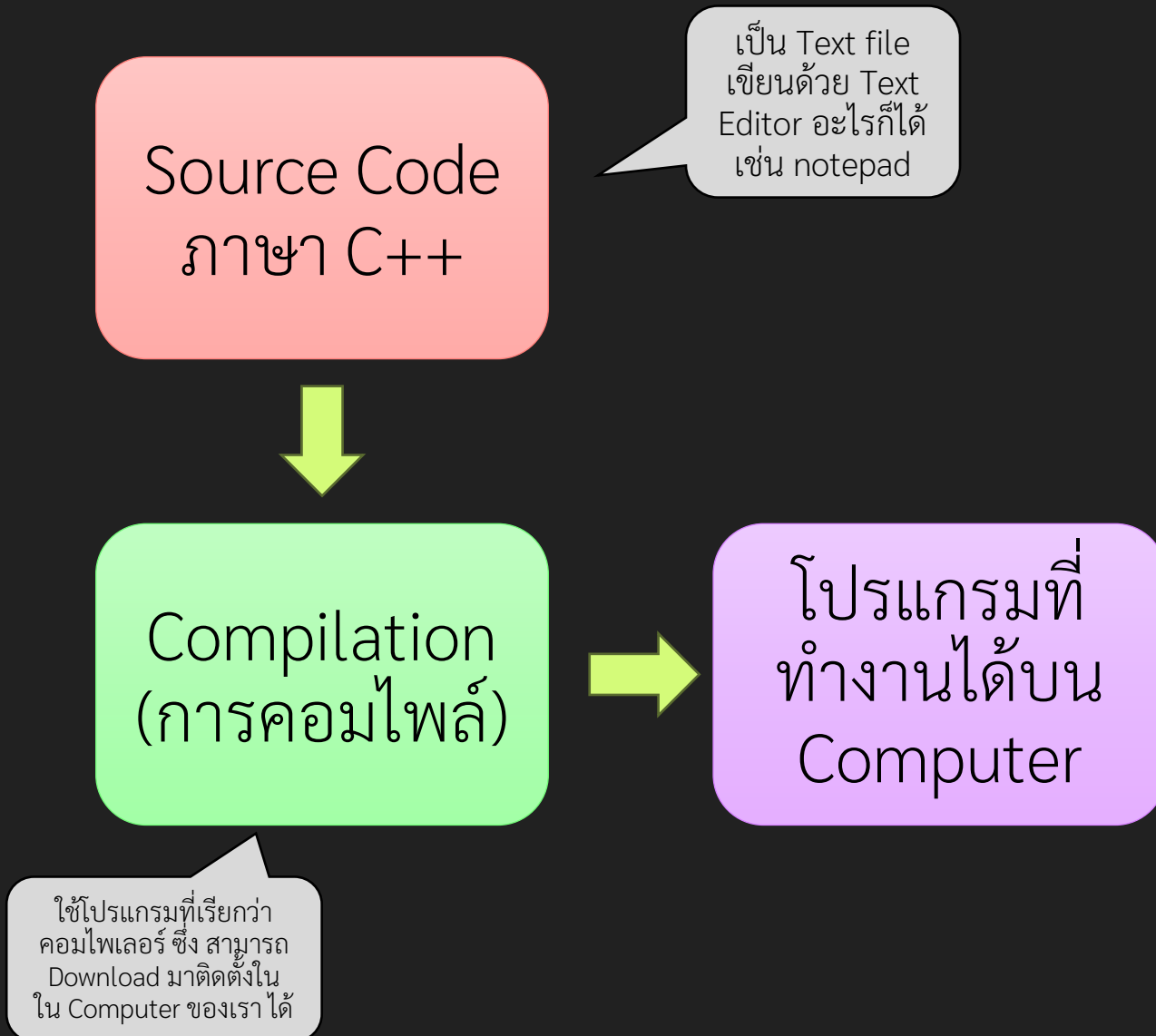
<https://www.tiobe.com/tiobe-index/>

ภาษา C++

พื้นฐาน

- เป็นภาษาแบบ Imperative แปลว่า ตัวโปรแกรมจะสั่งให้คอมพิวเตอร์ทำงานต่าง ๆ ตามที่กำหนด
 - เราต้องเข้าใจคำสั่งเหล่านั้นว่าสั่งแล้วเกิดอะไรขึ้นกับคอมพิวเตอร์
 - เราต้องสามารถสร้างลำดับของคำสั่งที่สั่งแล้วคอมพิวเตอร์ทำงานตามที่เราต้องการ
- โปรแกรมภาษา C++ จะประกอบด้วย คำสั่ง (statement) ต่าง ๆ ที่สั่งการให้คอมพิวเตอร์ทำงาน
 - เราจะมีรู้จักคำสั่งต่าง ๆ เหล่านั้น และ รูปแบบ (ไวยากรณ์, syntax) ในการใช้งานคำสั่งต่าง ๆ กัน

ขั้นตอนการทำงาน



- C++ จะใช้วิธี **Compile** ในการทำงาน
- มีโปรแกรม **Compiler** ของภาษา C++ ซึ่งทำหน้าที่แปลง Source Code (โปรแกรมที่เขียนด้วยภาษา C++) เป็น App สำหรับคอมพิวเตอร์
 - คอมพิวเตอร์ระบบต่าง ๆ จะใช้ Compiler แตกต่างกัน (compiler ของ Windows ก็แตกต่างจาก Compiler ของ iPhone และแตกต่างจาก Android)
- วิชานี้ใช้ **VS Code** เป็นโปรแกรมหลักในการทำงาน
 - VS Code ทำให้เราใช้งาน Compiler ได้ง่ายขึ้น

ตัวอย่าง Hello World

- เป็น App แบบ Console
 - app ที่รับข้อมูลและแสดงผลเป็นข้อความตัวอักษร
 - สะดวกในการพัฒนา เพราะ การรับส่งข้อมูลง่าย
- ตัวอย่างโปรแกรมที่นิยมใช้กันในการเริ่มเขียนโปรแกรมใด ๆ ก็คือ Hello World ซึ่งทำงานโดยแสดงข้อความออกทางหน้าจอ

```
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
}
```

โครงสร้างโปรแกรมภาษา C++

output

```
Hello, World!  
Hello, again  
Hello, third time
```

```
#include <iostream>
```

```
int main() {
```

```
    std::cout << "Hello, World!" << std::endl;
```

```
    std::cout << "Hello, again" << std::endl;
```

```
    std::cout << "Hello, third time" << std::endl;
```

```
}
```

ส่วน main

- โปรแกรมภาษา C++ จะต้องมีส่วนที่เรียกว่า `main` อยู่เสมอ โดยมีหน้าที่ระบุจุดเริ่มต้นของการทำงาน
 - โปรแกรมจะเริ่มทำงานตามคำสั่งที่อยู่ในส่วน `main` ไปตามลำดับที่เขียน
 - `main` จะเริ่มตั้งแต่เครื่องหมาย `{` ที่ตามหลังคำว่า `main()` และจบที่เครื่องหมาย `}`
 - `main` สามารถเขียนได้หลายแบบ แต่ในวิชานี้ ให้ใช้แบบด้านขวามือนี้ไว้ก่อน
- ตอนนี้น่าจะเดาได้แล้วว่า `std::cout << XXX << std::endl;` เป็นการแสดง XXX ออกทางหน้าจอแล้วขึ้นบรรทัดใหม่
- เครื่องหมาย `;` จะใช้เพื่อระบุว่าจบคำสั่งแล้ว ต้องมีหลังทุกคำสั่ง (โปรแกรมข้างบนมี 3 คำสั่ง)

```
int main() {  
  
}
```

การแสดงผลออกทางหน้าจอ

- การแสดงผลจะใช้คำสั่ง `std::cout`
- ตามด้วยเครื่องหมาย `<<` แล้วตามด้วยสิ่งที่ต้องการจะแสดงผล เช่น
 - `std::cout << "Hello!";` เป็นการแสดงผลคำว่า Hello!
 - `std::cout << 1;` เป็นการแสดงผลค่า 1
 - `std::cout << 3+5;` เป็นการแสดงผลค่า 8
 - `std::cout << std::endl;` เป็นการแสดงผลการขึ้นบรรทัดใหม่
- สามารถแสดงต่อกันได้
 - `std::cout << 1 << " " << 2 << " " << 5+7-4*2 << std::endl;`
 - เดาให้หน่อยว่าข้างบนนี้แสดงอะไรออกทางหน้าจอ?

#include

- คำสั่งต่าง ๆ ในภาษา C++ มีอยู่เยอะแยะมากมาย
- เราสามารถสร้างคำสั่งใหม่ ๆ ได้ด้วย (ไว้คุยกันบทถัด ๆ ไป)
- เพื่อให้เป็นระเบียบ คำสั่งต่าง ๆ จะถูกรวมไว้ใน **library** (คลังคำสั่ง)
 - อยากใช้คำสั่งไหน ต้องดูว่าคำสั่งนั้นอยู่ในคลังไหน และ เราต้องระบุว่าต้องการใช้คลังคำสั่งนั้นก่อนใช้งาน
 - ระบุได้ด้วยคำสั่ง **#include <ชื่อ library>**
- **std::cout** เป็นคำสั่งที่อยู่ใน library ชื่อ **iostream**

```
#include <iostream>    การ include library

int main() {
    std::cout << "Hello, World!" << std::endl;
}
```

namespace

- เนื่องจากคำสั่งมีเยอะแยะ และเราเองก็สามารถจะสร้างคำสั่งใหม่ ๆ ขึ้นมาได้ ดังนั้น ชื่อคำสั่ง อาจจะซ้ำกัน
- แก้ปัญหาโดยทำให้ คำสั่งต่าง ๆ จะมีการกำหนด **ชื่อเต็ม** ไว้ไว้เพื่อกันความสับสน โดยชื่อเต็ม ๆ จะใช้สิ่งที่เรียกว่า **namespace** มาประกอบ
- เวลาเรียกใช้งานคำสั่งใด ๆ ก็จะใช้ชื่อเต็ม ๆ ในรูปแบบ **namespace::คำสั่ง**
 - คำสั่งใน library หลักของภาษา C++ จะใช้ namespace เป็น **std**
- ดังนั้น **std::cout** จริง ๆ แล้วคือคำสั่ง **cout** ที่มี namespace เป็น **std**
- หากเราสร้างคำสั่ง **cout** ขึ้นมาเอง เราก็สามารถแยก **cout** ของเรากับ **cout** ที่มีอยู่แล้วได้
 - หากคำสั่งคือ **cout** ก็จะหมายถึง **cout** ที่เราสร้าง
 - หากคำสั่งคือ **std::cout** ก็จะหมายถึง **cout** ที่เป็นของ **iostream**

การใช้ using namespace

- เพื่อความสะดวก เนื่องจากเราเรียกใช้แต่คำสั่งที่มีอยู่แล้วของ C++ (ซึ่งมี namespace เป็น std) จะให้พิมพ์ std:: ทุกครั้งก็ดูเหนื่อย
- C++ มีคำสั่ง using namespace XXX; เพื่อระบุว่า คำสั่งใด ๆ ที่ใช้ ให้ลองใช้โดยถือว่าเป็นเหมือนกับมี namespace ชื่อ XXX นำหน้า
- ไม่ควรใช้ในการทำงานจริง!!!! (ในวิชานี้ใช้ได้)

```
#include <iostream>

using namespace std;

int main() {
    cout << 1;
    cout << 5;
    cout << " hello";
    cout << endl;
}
```

```
#include <iostream>

int main() {
    std::cout << 1;
    std::cout << 5;
    std::cout << " hello";
    std::cout << std::endl;
}
```

สรุปหน่วย

main เป็นตัวระบุว่า
จุดเริ่มต้นของคำสั่ง
เราอยู่ตรงไหน
(ตั้งแต่ { จนถึง })

บอกว่ายากใช้ library
ชื่อ iostream (เพราะ
มีคำสั่ง cout ให้ใช้)

```
#include <iostream>
```

```
int main() {  
    std::cout << "Hello, World!" << std::endl;  
}
```

เครื่องหมาย ;
ระบุว่าจบคำสั่งแล้ว

คำสั่ง cout ตามด้วย <<
เอาไว้แสดงผลออกทาง
หน้าจอ (ชื่อเต็มคือ
std::cout)

คำสั่งที่เอาไว้บอก
ว่าขึ้นบรรทัดใหม่

ทิ้งท้ายก่อนไปต่อ

- C++ เป็นเครื่องมือที่ใช้กันแพร่หลาย และใช้กันมานาน มีความซับซ้อนสูงมาก เราไม่จำเป็นต้องเข้าใจมันทั้งหมด ในตอนแรก ให้ทำความเข้าใจเฉพาะส่วนที่จำเป็นพอ
- C++ มีการกำหนดเป็นมาตรฐาน ISO ไว้ และมีหลายรุ่น รุ่นปัจจุบันคือ C++20 (ออกปี 2020)
 - รุ่นก่อนหน้าคือ C++98, C++03, C++11, C++14, C++17
 - เป็นแบบ Backward Compatibility กล่าวคือ ของที่สร้างด้วย C++ รุ่นเก่า สามารถใช้ในรุ่นใหม่ได้ (แต่ของที่สร้างตามมาตรฐานใหม่ อาจจะใช้ไม่ได้กับรุ่นเก่า)
 - เราจะยึด C++11 เป็นหลัก
- อยากอ่านมาตรฐานภาษา C++ ???
 - คู่มือภาษา C++ <https://cplusplus.com/reference/>
 - เอกสารอ้างอิง <https://en.cppreference.com/w/>

อย่าเพิ่งไปอ่าน
เลย ยาว เยอะ
ซับซ้อน