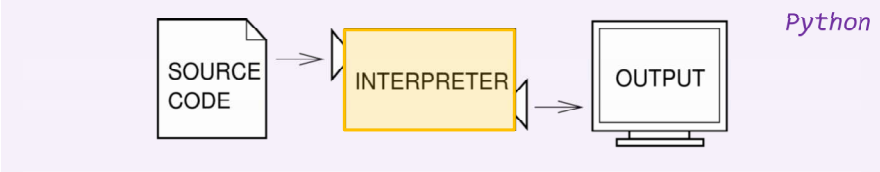
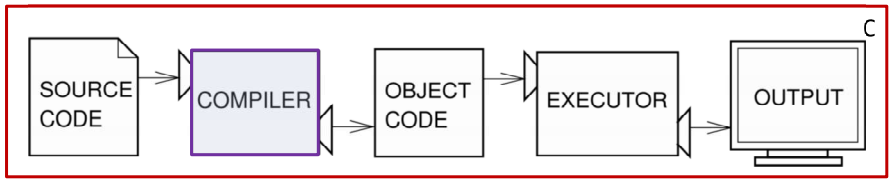
w00-Lec

C for Python Programmers

Assembled for 204112
Semester 2 - 2015
by Kittipitch Kuptavanich

Interpreted vs Compiled





- ภาษา C จัดอยู่ในประเภท **Compiled Language**

Variable Declarations

- ในภาษาตระกูล **Compiled Language** นั้น **Compiler** จำเป็นต้องทราบรายละเอียดหลาย ๆ อย่างของโปรแกรม โดยที่ไม่ต้อง **Run** โปรแกรมนั้น ๆ (จุดประสงค์หลักคือการจองพื้นที่หน่วยความจำ)
- การประกาศตัวแปร (**Variable Declaration**) เป็นการแจ้งให้ **Compiler** ทราบข้อมูลเกี่ยวกับ **variable** ต่าง ๆ ที่ใช้*ก่อน*เกิดการใช้จริง (ก่อน **run time**)
- ในภาษา **C** การประกาศตัวแปร เป็นการกำหนดชนิดของตัวแปรว่าเป็น **integer** (`int`) **floating-point** (`double`) หรือ **character** (`char`) หรือชนิดอื่น ๆ
- ทำได้โดยการใส่ ชื่อชนิดข้อมูลตามด้วย **whitespace** และชื่อตัวแปร แล้วจบ **statement** ด้วย **semicolon** `;` เช่น

```
double x;
```


Whitespaces

- ในภาษา **Python** อักขระว่างต่าง ๆ (**whitespace characters**) เช่น **tab** (`'\t'` ในการย่อหน้า เพื่อแสดง **block**) หรือ **newline** (`'\n'` เพื่อจบ **statement**) มีความหมายต่างกัน
- ในภาษา **C** อักขระว่างมีความหมายแค่เพื่อแยกคำออกจากกัน
  - จบทุก **Statement** ด้วย **semicolon** `;`
  - และแสดง **Block** ด้วยปีกกา (**Braces** `{}`)

http://www.toves.org/books/cpy

Whitespaces

Python Fragment

```
disc = b * b - 4 * a * c

if disc < 0:
    num_sol = 0
else:
    t0 = -b / a
    if disc == 0:
        num_sol = 1
        sol0 = t0 / 2
    else:
        num_sol = 2
        t1 = disc ** 0.5 / a
        sol0 = (t0 + t1) / 2
        sol1 = (t0 - t1) / 2
```

C equivalent #1

```
disc = b * b - 4 * a * c;

if (disc < 0) {
    num_sol = 0;
} else {
    t0 = -b / a;
    if (disc == 0) {
        num_sol = 1;
        sol0 = t0 / 2;
    } else {
        num_sol = 2;
        t1 = sqrt(disc) / a;
        sol0 = (t0 + t1) / 2;
        sol1 = (t0 - t1) / 2;
    }
}
```

C equivalent #2

```
disc=b*b-4*a*c;if(disc<0){
num_sol=0;}else{t0=-b/a;if(
disc==0){num_sol=1;sol0=t0/2
;}else{num_sol=2;t1=sqrt(disc/a;
sol0=(t0+t1)/2;sol1=(t0-t1)/2;}}
```

The same to the Compiler

Comments

```
/* this is a comment */

/* this comment spans
   over multiple lines */

/* this comment spans
 * over multiple lines
 * with extra asterisk*/


// this is an in-line comment
```

- **in-line comment ด้วยเครื่องหมาย `//` เริ่มใช้ได้ในภาษา C version C99 เป็นต้นไป**

The `printf()` function

- **ฟังก์ชัน printf() ใช้เพื่อแสดง output**
- **การเรียกใช้ printf จำเป็นต้องมีการ include library `stdio.h` (standard input/output)**

```
#include <stdio.h>
```

- **Argument แรกของฟังก์ชันจะต้องเป็น string เสมอ โดยมีหน้าที่กำหนด format ในการแสดงผล**
- **Argument ถัด ๆ ไปจะเป็นค่าที่ใช้ในการแสดงผล**

```
printf("# solns: %d\n", num_sol);
```

```
# solns: 2
```


The `printf()` function

```
05      int num_sol = 2;
06      double sol0 = 4.0;
07      double sol1 = 1.0;
08
09      printf("# solns: %d", num_sol);
10      printf("solns: %.2f, %.2g", sol0, sol1);
11      return 0;
```

```
# solns: 2solns: 4.00, 1
```

- **จาก output ที่ได้ จะเห็นว่าฟังก์ชัน `printf()` ทำงานตามที่ระบุโดยไม่มีการเพิ่มอักขระ space หรือขึ้นบรรทัดใหม่ให้**
- **Conversion Control Sequence (e.g. `%d` `%f` `%x`) ทำงานในลักษณะเดียวกันกับในภาษา Python**

Functions

- ในภาษา C ชุดคำสั่งต้องอยู่ภายในฟังก์ชัน
- พิจารณาการกำหนดฟังก์ชัน

```
double square(double b)
{
    return b * b;
}
```

- ฟังก์ชันในภาษา C จะต้องมีการกำหนดชนิดข้อมูลที่ต้องคืนค่า (return type) ในที่นี้คือ *double*
- หากไม่มีการคืนค่า จะต้องระบุโดย keyword *void*
- ฟังชัน main() เป็นฟังก์ชันพิเศษทำหน้าที่เป็นจุดเริ่มต้นของโปรแกรม

Function Prototypes

- ในภาษา C โปรแกรมจะทำงานจากบนลงล่าง
- หากมีการเรียกใช้ฟังก์ชัน (บรรทัดที่ 15) ก่อนที่จะมีการ Define ฟังก์ชัน (บรรทัดที่ 20) จะทำให้ compile ไม่ผ่าน
- จำเป็นต้องมีการสร้างฟังก์ชันโปรโตไทป์ (Function Prototype) ไว้ที่บรรทัดที่ 10 เพื่อให้ Compiler รู้จักฟังก์ชันก่อน

```
08 #include <stdio.h>
09
10 double square(double b);
11
12 int main()
13 {
14     double x = 2.5;
15     double result = square(x);
16     printf("result = %.2f", result);
17     return 0;
18 }
19
20 double square(double b)
21 {
22     return b * b;
23 }
```

1. ชื่อฟังก์ชัน
2. Parameters
3. Return Type

Operators

- Operator ในภาษา Python ถูกออกแบบโดยยึดจาก Operator ในภาษา C เป็นหลัก โดยมีความแตกต่างที่สำคัญคือ

C operator precedence	Python operator precedence
++ -- (postfix) + - ! (unary) * / % + - (binary) < > <= >= == != && || = += -= *= /= %=	** + - (unary) * / % // + - (binary) < > <= >= == != not and or

- ภาษา C ไม่มีเครื่องหมายยกกำลัง
- ภาษา C ใช้ สัญลักษณ์แทน Boolean operator &&, ||, !.

Operators [2]

- ในภาษา C เครื่องหมาย ! (not operator) มี Precedence ที่สูงมาก ควรใช้ด้วยความระมัดระวัง (ใส่วงเล็บหากต้องการให้ทำ operator อื่น ๆ ก่อน)
- ในภาษา C (Assignment) การกำหนดค่า เป็น Operator (ใน Python เป็น statement)
  - = += -= *= /= %=
  - ทำให้ Assignment สามารถเป็นส่วนหนึ่งของ statement อื่นได้

```
while ((a = getchar()) != EOF)
```

  - ในกรณีนี้ จะเป็นการกำหนดค่าให้ ตัวแปร a ก่อนโดยใช้ค่าที่ได้จากฟังก์ชัน getchar() แล้วจึงตรวจสอบว่าค่าที่กำหนดเท่ากับอักขระ EOF (End of File หรือไม่)

Operators [3]

- ในภาษา C เครื่องหมาย `++` และ `--` คือการ increment และ decrement ที่ละ 1 "`i++`" คือ "`i = i + 1`" (หรือ "`i += 1`").
- เครื่องหมายหาร `/` ในภาษา C มีประเภทเดียว โดยหากทั้งตัวตั้งและตัวหาร (Operands ทั้งสองตัว) เป็นจำนวนเต็มจะเป็นการหารแบบปัดเศษทิ้ง (round toward zero) (ไม่ใช่ Floor Division เหมือนเครื่องหมาย `//` ใน Python)
  - `1 / 3` = _____
  - `10 / 3` = _____
  - `10.0 / 3` = _____
  - `10.0 / 3.0` = _____
  - `-10 / 3` = _____
  - `-10.0 / 3` = _____

Basic Types

C does *not* have a Boolean type
0 is False
Everything else is True

- C มี Basic Types (ชนิดข้อมูลพื้นฐาน หรือ Primitive Types) 4 ชนิด

Type	Descriptions	Notes
`char`	a single byte, capable of holding one character in the local character set	
`int`	an integer, typically reflecting the natural size of integers on the host machine	
`float`	single-precision floating point	
`double`	double-precision floating point	

- นอกจากนี้ยังมี Qualifier เพื่อกำหนดคุณสมบัติเพิ่มเติมให้กับข้อมูลที่เป็น Basic Types ได้ ได้แก่ `short long signed unsigned`
  - เมื่อใช้ short และ long คู่กับ int เราสามารถเขียนในรูปย่อได้
    - `short` แทน `short int`
    - `long` แทน `long int`

C Qualifiers

- 9.234 indicates a double literal
- 9.234f indicates a float literal
- 9.234L indicates a long double literal
- 3 (no suffix) indicates int literal

Qualifiers	Data Types	Meaning	Notes
`signed`	`short, int, long, long long`	negative or positive or zero number	
	`float, double, long double`		
	`char`	have values between -128 and 127	
`unsigned`	`short, int, long, long long,`	positive or zero number	
	`float, double, long double`		
	`char`	have values between 0 and 255	

Data Type Ranges

- ในภาษา C ข้อมูลเกี่ยวกับ Range ของ primitive types อยู่ใน library `limits.h` (integral types) และ `float.h` (floating-point types)

```
01 #include <stdio.h>
02 #include <limits.h>
03 #include <float.h>
04
05 int main()
06 {
07     printf("The minimum value of SIGNED CHAR = %d\n", SCHAR_MIN);
08     return 0;
09 }
```

```
The minimum value of SIGNED CHAR = -128
```

Data Type Ranges

Qualifiers	Data Types	Minimum	Maximum
signed	char	SCHAR_MIN	UCHAR_MAX
	short	SHRT_MIN	USHRT_MAX
	int	INT_MIN	UINT_MAX
	long	LONG_MIN	ULONG_MAX
	long long	LLONG_MIN (C99)	ULLONG_MAX(C99)
unsigned	char	UCHAR_MIN	UCHAR_MAX
	short	USHRT_MIN	USHRT_MAX
	int	UINT_MIN	UINT_MAX
	long	ULONG_MIN	ULONG_MAX
	long long	ULLONG_MIN (C99)	ULLONG_MAX(C99)

Integral Data Type Ranges

C data type	Minimum	Maximum
char	−127	127
unsigned char	0	255
short [int]	−32,767	32,767
unsigned short [int]	0	65,535
int	−32,767	32,767
unsigned [int]	0	65,535
long [int]	−2,147,483,647	2,147,483,647
unsigned long [int]	0	4,294,967,295
long long [int]	−9,223,372,036,854,775,807	9,223,372,036,854,775,807
unsigned long long [int]	0	18,446,744,073,709,551,615

Guaranteed ranges for C integral data types.

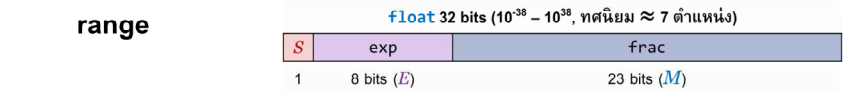
Integral Data Type Ranges [2]

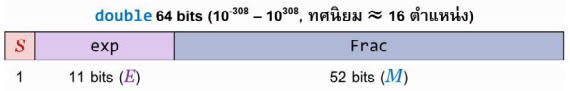
C data type	Minimum	Maximum
char	−128	127
unsigned char	0	255
short [int]	−32,768	32,767
unsigned short [int]	0	65,535
int	−2,147,483,648	2,147,483,647
unsigned [int]	0	4,294,967,295
long [int]	−2,147,483,648	2,147,483,647
unsigned long [int]	0	4,294,967,295
long long [int]	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long long [int]	0	18,446,744,073,709,551,615

Typical ranges for C integral data types o a 32-bit machine

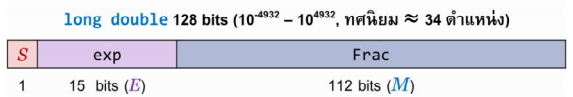
Floating-Point Data Type Ranges

- **C ยึดมาตรฐานการ implement Floating point ตาม IEEE 754 เช่นเดียวกันกับ python**
  - **`float` ในภาษา Python = 64bit = `double` ในภาษา C)**
  - **เนื่องจากการ Encode จำนวนลบ ใช้ sign bit เพียง 1 bit ดังนั้น qualifier `signed` หรือ `unsigned` จึงไม่มีผลต่อความกว้างของ range**





$(-1)^s \times M \times 2^E$



Braces

- วงเล็บปีกกาใช้กำหนด **Block** ในภาษา **C** - หาก **Block** มี **Statement** เดียว อาจเลือกที่จะใส่หรือไม่ใส่ก็ได้
- หากมีมากกว่า **1 Statement** ต้องใส่เสมอ
- การใช้ปีกกาทำให้สามารถแปลง **Nested** `if` เป็น `else if` ได้ **(Chained Condition)**

```
if (first > second)
    max = first;
    min = second;
else
    max = second;
    min = first;
```

```
disc = b*b - 4*a*c;

if (disc < 0) {
    num_sol = 0;
} else {
    if (disc == 0) {
        num_sol = 1;
    } else {
        num_sol = 2;
    }
}
```

```
disc = b*b - 4*a*c;

if (disc < 0) {
    num_sol = 0;
} else
    if (disc == 0) {
        num_sol = 1;
    } else {
        num_sol = 2;
    }
```

```
disc = b*b - 4*a*c;

if (disc < 0) {
    num_sol = 0;
} else if (disc == 0) {
    num_sol = 1;
} else {
    num_sol = 2;
}
```


Statements

A few basic instructions appear in just about every language:

- **Input**
- **Output**
- **Math**
- **Conditional Execution**
- **Repetition**

Statements [2]

- **Input – ฟังก์ชัน `scanf()` ทำงานตรงกันข้ามกับฟังก์ชัน `printf()`**

```
int scanf(char *format, ...)
```

- `scanf()` อ่านอักขระจาก **standard input** แล้วแปลงค่าตาม `format` แล้วเขียนค่าลงในตัวแปรที่ระบุเช่น (สังเกตการใช้ **&**)
- คืนค่าจำนวนค่าที่อ่านและแปลงสำเร็จตาม `format`

```
int num1;
float num2;
double num3;

scanf("%d", &num1); // d for integer
scanf("%f", &num2); // f for floating point
scanf("%d %le", &num1, &num3); // le for double
```


The `scanf()` Function

- **Conversion Specifier for `scanf()`**

Format	Input Value	Use With
`%c or getchar()`	character	`char`
`%d`	decimal	`char, int, short, long, long long`
`%o`	octal	`int, char, short, long, long long`
`%x`	hexadecimal	`int, char, short, long, long long`
`%f`	floating-point	`float, double, long double`

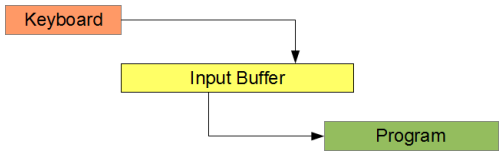
- **Size Specifier**

For integer values:		floating-point values:	
Size Specifier	Use With	Size Specifier	Use With
none	`int`	none	`float`
`hh`	`char`	`l`	`double`
`h`	`short`	`L`	`long double`
`l`	`long`		
`ll`	`long long`		

For example: `scanf("%lf", &double_value);`

Input Buffer

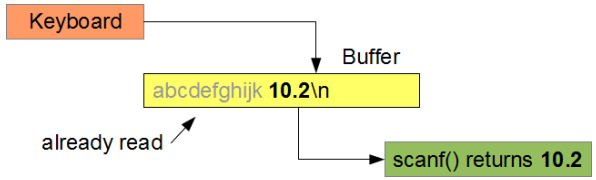
- **Buffer** คือหน่วยความจำส่วนหนึ่งที่มีไว้เพื่อเก็บข้อมูลชั่วคราว
  - เมื่อมีการพิมพ์อักขระต่าง ๆ ลงไปผ่าน **keyboard** แต่ละอักขระก็จะถูกบันทึกไว้ใน **Input Buffer**
  - **User** สามารถ **Edit** สิ่งที่พิมพ์ได้
  - จนกระทั่ง **User** พิมพ์อักขระ `'\n'`
- โปรแกรมจะอ่านและดึงข้อมูลจาก **Buffer** เท่าที่จำเป็น



https://scs.senecac.on.ca/~ipc144/pages/content/formi.html

Input Buffer [2]

- `scanf()` จะอ่านอักขระและแปลงค่าตาม **format** ที่ระบุ



- `scanf()` จะอ่านและดึงอักขระดึงอักขระออกจาก **Input Buffer** จนกว่ากรณีใดกรณีหนึ่งต่อไปนี้เกิดขึ้น
  - อ่านและแปลงค่าตาม **format** ที่ระบุเรียบร้อย
  - พบอักขระที่ไม่ตรงตาม **format**
  - อ่านจน **Input Buffer** ว่าง

https://scs.senecac.on.ca/~ipc144/pages/content/formi.html

Input Buffer [3]

- เนื่องจากเมื่ออ่านอักขระด้วย `scanf()` ในหลาย ๆ กรณี ฟังก์ชันจะเหลือ **white space** หรือ `'\n'` ไว้บน **Input Buffer**
  - เช่นการอ่านด้วย `%d` ก็จะอ่านจนถึง **whitespace** และทิ้งอักขระ **white space** ไว้บน **Input Buffer**
  - การการอ่านอักขระที่ละตัวด้วย `scanf("%c")` หรือ `getchar()` หลังจากนั้น จะพบอักขระว่างที่ยังไม่ได้ถูกดึงค่าเป็นอักขระตัวแรกแทน
- อาจจำเป็นต้องอ่านอักขระว่างกล่าวทิ้ง จนหมดบรรทัด หรือจนกว่าจะพบอักขระที่ต้องการอ่านเช่น

```
while (getchar() != '\n');
```

https://scs.senecac.on.ca/~ipc144/pages/content/formi.html

Statements [3]

- **Output**
  - **We covered** `printf()` **in previous section**
  - ฟังก์ชันอื่น ๆ เช่น `puts()` หรือ `putchar()` จะถูกกล่าวถึงในบทถัด ๆ ไป
- **Math**
  - **We also cover operators**
- **Conditionals**
  - `if` **statement ทำงานเหมือนในภาษา python**
    - **Conditional expression ต้องอยู่ในวงเล็บเสมอ**
    - **ไม่ต้องใช้ : หลัง condition expression**
    - **C ไม่มี** `elif` **แต่ใช้** `else if` **แทน**

http://www.toves.org/books/cpy

Statements [4]

- **Conditionals (cont'd)**
  - `switch` statement
    - ใช้แทน `if…elif…elif…elif…else` chain
    - Test ได้แต่ equality (exact match)
- **Repetitions**
  - `while` statement
    - ทำงานเหมือนใน python
    - Syntax เหมือน if ในภาษา C
  - `break` และ `continue` ทำงานเหมือนใน Python

```
switch (letter_grade) {
case 'A':
    gpa += 4;
    credits += 1;
    break;
case 'B':
    gpa += 3;
    credits += 1;
    break;
case 'C':
    gpa += 2;
    credits += 1;
    break;
case 'D':
    gpa += 1;
    credits += 1;
    break;
case 'W':
    break;
default:
    credits += 1;
}
```


Statements [5]

- **Repetitions (cont'd)**
  - `for` statement:
    - ในวงเล็บประกอบด้วย 3 statement

0 ทำครั้งเดียว ก่อน test เงื่อนไขครั้งแรกสุด

3 ทำทุกครั้งที่วน loop
- หลัง loop_statement_N
- ก่อน test เงื่อนไขทำซ้ำ ครั้งถัดไป

```
for (กำหนดค่าเริ่มต้น; เงื่อนไขการทำซ้ำ; เปลี่ยนค่าตัวแปรควบคุม) {
    loop_statement_1
    loop_statement_2
    loop_statement_3

    loop_statement_N
}
```

2

1 test เงื่อนไขทุกครั้งที่วน loop ก่อน loop_statement_1

Statements [6]

- `for` statement (cont'd):
  - ไม่จำเป็นต้องระบุทั้ง 3 ค่าภายใน บรรทัดที่มีคำสั่ง for (เช่น สามารถกำหนดค่าเริ่มต้นที่บรรทัดอื่น ก่อนคำสั่ง for ได้)
  - แต่จำเป็นต้องใส่ semi-colon ให้ครบทั้งสองอันภายในวงเล็บ

```
03 int main()
04 {
05     int count = 2;
06     int max = 20;
07
08     for (; count <= max; count += 2) {
09         printf("%d ", count);
10     }
11
12     return 0;
13 }
```

```
2 4 6 8 10 12 14 16 18 20
```


Arrays

- **Array มีลักษณะคล้าย list ใน python**
- **แต่ไม่สามารถเพิ่มหรือลดขนาดได้ เช่น**

```
double pops[50];
pops[0] = 897934;
pops[1] = pops[0] + 11804445;
```

  - **เป็นการสร้าง array ขนาด 50 ช่อง เพื่อเก็บตัวแปรชนิด double (index 0 – 49)**
- **เมื่อออกนอก array boundary ในภาษา C โปรแกรมจะไม่เกิด error แต่จะอ่าน หรือเขียนที่หน่วยความจำจากบริเวณที่ระบุแทน แม้จะอยู่นอกช่วงที่เป็น array boundary**

References

- http://www.toves.org/books/cpy
- https://learnxinyminutes.com/docs/c/
- **https://scs.senecac.on.ca/~ipc144/pages/content/formi.html**