โครงสร้างภาษาซี

standard input output

```
# include <stdio.h>   ← header

int main(){            ← body

    printf("hello world 6410401043");

}
```

การ comment

```
// in line comment

/* box comment */
```

printf() กับ puts()

-ex1:

```
printf("hello");

printf("world");
```

output : helloworld

-ex2:

```
puts("hello");

puts("world");
```

output : hello
world

ขึ้นบรรทัดใหม่อัตโนมัติ printf() ใช้ \n

```
printf("ข้อความ or format",ตัวแปร);
```

format → %d int, %f float, %c char, %s char[n], etc.
String

การประกาศตัวแปร

ภาษาซีต้องมีการประกาศตัวแปรก่อนนำมาใช้เสมอ !!

```
datatype variablename ;

datatype variablename = value ;
```

input

```
charvalue = getchar()   ← เก็บ char 1 ตัว

gets(strvariable);

fgets(strvariable ,n ,stdin);   ต้อง # include <stdlib.h> ด้วย!

scanf("format",&variable);
```

Arrays เก็บได้ชนิดเดียว

✗ {'a', 1, 2.5}

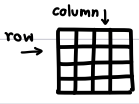
```
int variable[n];      n = จำนวนเต็ม

char variable2[n];    char Arrays = string
```

✿ เวลาส่ง Arrays ให้ f() ต้องส่งไซส์ไปด้วย !

Arrays 2 มิติ

```
int A[rowsize][columnsize];
```



while loop

```
while (เงื่อนไข){
    ถ้าเป็นจริงจะทำตรงนี้ ;
}
```

ใน c ไม่มี true , false ดังนั้นลูป while true เขียนเป็น

```
while(1){
    if(condition){
        break;
    }
}
```

for loop รู้จำนวนรอบชัดเจน

```
for (start;stop;step){
    ถ้าเป็นจริงจะทำตรงนี้ ;
}
```

สามารถละได้ แต่ต้องมี ; ครบ 2 อัน!

if-else

```
if (เงื่อนไข){
    statement;
}
else if (เงื่อนไข){
    statement;
}
else{
    statement;
}
```

ตัวดำเนินการทางตรรกศาสตร์

and	&&
or	||
not	!

atoi(),atof(),atol(),atoll() เปลี่ยน str เป็น int,float,long,long long

```
int variable = atoi(strvariable);
float variable = atof(strvariable);
long variable = atol(strvariable);
long long  variable = atoll(strvariable);
```

function

ประกาศฟังก์ชันก่อน main()

parameter

ไพบีที่ฟังก์ชันจะ return กลับ (เค้าไม่มีใช้ void)

```
int f1 (int variable){
    return variable * 2 ;
}
```

pass by value , call by value : การส่งค่าข้อมูลผ่าน parameter

pass by reference : การส่งค่าข้อมูลผ่าน address

ประกาศฟังก์ชันแต่ยังไม่เขียนฟังก์ชัน

function prototypes → returnvaluetype functionname(parametertype);

recursion ฟังก์ชันที่สามารถเรียกใช้ตนเองได้

```
ex: long long fac(long long n){
        if (n==0)
            return 1;
        else
            return n*fac(n-1);
    }
```

pointers เก็บ address

ตำแหน่ง

```
int *pointername = &variable;
```

ประเภทของตัวแปร

ex: int *ptr = &x;

การปริ้นท์ pointer

printf("%p", ptr); ← address ที่ ptr เก็บอยู่

printf("%d", *ptr); ← ค่าใน address ที่ ptr เก็บอยู่

pointer สามารถนำไปคำนวณได้

ex: ptr += 1 ; (ptrจะบวกไป 1 int = 4 bytes)

ptr = 1 don't

*ptr = 1 do

เปลี่ยนค่าใน address ที่อยู่ใน pointer

ex: *ptr += 1 ; (ค่าใน address ptr บวกไป 1)

ชื่อ array เป็น address

ex: int data[3];

int *dataptr = data; ความหมายเดียวกับ int *dataptr = &data[0];

★ รู้ชิ้นแรก หาตำแหน่งชิ้นต่อๆไปก็ ++ ๆ