CP 111 Computer Programming (2/2554 Section B01-B02)

Handout 4*

<u>หัวข้อ</u>

- ทบทวน
- Functions (Chapter 14.1, 14.2, 14.4)

<u>ทบทวน</u>

- Variables, literals, operators, expression, statement
- Conditional constructs: `if`, `if-else`, `switch` statements
- Iterative constructs: `while`, `do-while`, `for`, `break`, `continue` statements

<u>Functions</u>

1. จาก Lecture แรก

ภาษา C เป็นภาษาโปรแกรมคอมพิวเตอร์แบบ Procedural (Procedural programming language) โปรแกรมภาษา C หนึ่งๆ จะประกอบไปด้วยหลายๆ โพรซีเยอร์ (หรือ Subroutine) และภาษา C จะเรียก โพรซีเยอร์ (หรือ Subroutine) ว่า Function

2. Function คือ subprogram (โปรแกรมย่อย),
   - Smaller, simpler, subcomponent of program
   - Subprogram เป็นหัวใจสำคัญของภาษาโปรแกรมสมัยใหม่ (the soul of modern programming languages)

3. Why? ทำไม Function จึงสำคัญมาก =>ทำให้เราสามารถทำ *<u>Abstraction</u>* (the main theme in CP121)
   - ซ่อนรายละเอียด (hide low level details)
   - ทำให้เห็นโครงสร้างภาพรวมของโปรแกรมและทำให้โปรแกรมดูเข้าใจง่าย
   - ทำให้การเขียนโปรแกรมสามารถแยกเป็นส่วนๆ และแบ่งไปให้โปรแกรมเมอร์หลายๆคนไปช่วยกันพัฒนาได้
   - ตัวอย่างข้างล่าง โปรแกรมในฟังก์ชัน main จะประกอบไปด้วย 5 functions โปรแกรมนี้เป็นโครงของโปรแกรมเกมหมากกระดาน (Board game)

```
int main()
{
  SetupBoard();  /* place pieces on board */
  DetermineSides();  /* choose black/white */
  /* Play game */
  do {
    WhitesTurn();
    BlacksTurn();
  }
  while (NoOutcomeYet());
  return 0;
}
```

4. Function ในภาษา C เทียบเท่ากับ subroutine ใน LC-3 assembly language

* Handout นี้ เรียงเรียงจาก Slide ประกอบหนังสือ Introduction to computing systems by Y.Patt and S.J.Patel และ slide วิชา ECE190 UIUC by V.Kindratenko (https://wiki.engr.illinois.edu/display/ece190/Lectures)

5. การเขียน/เรียกใช้ Function ในโปรแกรมภาษา C ต้องการสิ่งต่อไปนี้

```
01 #include <stdio.h>
02
03 int Factorial(int);   /* Function declaration */
04 /* or int Factorial(int n); */
05 int main()
06 {
07    int number,answer;
08
09         printf("Input a number: ");   /* Function call to printf()      */
10         scanf("%d",&num);             /* Function call to scanf()       */
11         answer=Factorial(num);        /* Function call to Factorial()   */
12                                       /* returned value from Factorial()is saved into answer.*/
13         printf("Factorial(%d)=%d\n",num,answer) ;
14         return 0;
15 }
16
17 int Factorial(int n)              /* Function definition */
18 {
19    int i,result;                  /* Codes for function */
20      result=1;
21      for(i=1;i<=n;i++)
22        result=result*i;
23
24      return(result);         /* return to caller with the value of result */
25 }
```

(5.1) Function declaration (Function prototype): บรรทัดที่ 03 ในโปรแกรมข้างบน

Syntax:

```
return_type function_name (type_of_argument_1,…, type_of_argument_N);
```

เช่น `int Factorial(int n);`

ฟังก์ชัน Factorial มี argument (อินพุต) 1 ตัวชนิด int และคืนค่าชนิด int (อนุญาตให้มีชื่อหลัง type เช่น n)

- ฟังก์ชันในภาษา C จะคืนค่ากลับไปให้ caller ได้ 1 ค่าผ่าน statement `return`
- ในกรณีที่ฟังก์ชันไม่มีการ return ค่า และ/หรือ ไม่มี argument ให้ประกาศชนิดข้อมูลเป็น `void` เช่น
  `void clear_screen(void);`
- อย่าลืม semicolon (;) ก่อนจบบรรทัด
- วางไว้ที่ต้นไฟล์ก่อน ฟังก์ชันที่จะเรียก (เช่น main)
- เหตุที่ต้องมี function declaration เพื่อทำให้ compiler สามารถตรวจสอบคุณสมบัติของฟังก์ชันได้
- #include <stdio.h> เป็นการนำ function declaration ที่อยู่ใน stdio.h มาแปะไว้ที่ต้นไฟล์โปรแกรมที่เราเขียน ซึ่งในไฟล์ stdio.h จะมี function declaration ของ scanf() และ printf() อยู่

(5.2) Function definition: บรรทัดที่ 17 ถึง 25 ในโปรแกรมข้างบน

- โค้ดที่ระบุการทำงานของฟังก์ชัน ประกอบด้วย statement ต่างๆ
- รูปแบบ

Syntax:

```
return_type function_name (type arg1,…, type arg_N)
{
  declarations and statements
}
```

ไม่มี semi-colon

- บรรทัดแรกของ function declaration จะคล้ายกับ function declaration แต่ไม่มี semicolon (;)
- `return` statement ใช้สำหรับการออกจากฟังก์ชันแล้วกลับไปที่ caller (function) และทำการคืนค่าที่อยู่ภายในวงเล็บ เช่น `return(result);` จะคืนค่าปัจจุบันของตัวแปร result

(5.3) Function call

- เขียนเป็น statement ระบุชื่อฟังก์ชันและตามด้วยค่าที่จะส่งเป็น argument
- ถ้าฟังก์ชันมีการคืนค่า ต้องมีตัวแปรมารับค่าในรูปแบบ assignment statement (=)
- ตัวอย่าง

(i) บรรทัดที่ 11 ในโปรแกรมข้างบน สำหรับการเรียก (กระโดดเข้าไปทำงานใน) ฟังก์ชัน `Factorial` โดยส่งค่าของตัวแปร `num` เป็น argument `n` ให้กับฟังก์ชัน และค่าที่คืนมาจะถูกเก็บไว้ที่ตัวแปร `answer`

(ii) บรรทัดที่ 9 ในโปรแกรมข้างบน สำหรับการเรียก (กระโดดเข้าไปทำงานใน) ฟังก์ชัน `printf` และส่งอาร์กิวเมนต์ดังในโปรแกรม แต่ไม่มีการคืนค่า

(iii) บรรทัดที่ 10 และ13 ในโปรแกรมข้างบน สำหรับการเรียกฟังก์ชัน `scanf` และ `printf`

Note: ฟังก์ชัน `printf` และ `scanf` เป็น standard library functions เราเรียกใช้อย่างเดียว ไม่ต้องเขียน function definition ส่วน function declaration เก็บในไฟล์ stdio.h เราใช้ #include เพื่อนำมาแปะไว้ที่ต้นไฟล์โปรแกรมของเรา

- **Note**: ขอบเขตของตัวแปรและ argument จะอยู่ภายในฟังก์ชัน เช่น ตัวแปร `number` และ `answer` จะมีผล (สามารถอ้างถึงค่าตัวแปรผ่านชื่อของมันได้) ภายในฟังก์ชัน `main()` ส่วนตัวแปร `i, result` และ argument n จะมีผลภายในฟังก์ชัน `factorial()`

6. โจทย์ตัวอย่าง Function: จงระบุผลลัพธ์การทำงานของโปรแกรม

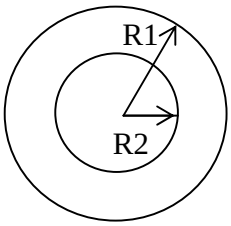
```
#include <stdio.h>
int func_AA(int a);
int func_BB(int any_identifier);
int main()
{
int a,b,x,y,z;  /* ค่าของ a, b, x, y, z จะมีผลภายใน main นี้ และเป็นคนละตัวแปรกับ a, b, x, y, z ใน func_BB และ func_AA  */

    for(x=2;x<=3;x++)
    {
       y=func_AA(x);                /*M1*/
       z=func_BB(y);                /*M2*/
       printf("%d %d %d\n",x,y,z); /*M3*/
    }
    return 0;
}
int func_AA(int a)      /* ค่าของ a, b, x, y, z จะมีผลภายใน { } นี้ และเป็นคนละตัวแปรกับ a, b, x, y, z ใน func_BB และ main  */
{
int b,x,y,z;
       b=a*a;         /*AA_1*/
       return(b);     /*AA_2*/
}
int func_BB(int a)     /* ค่าของ a, b, x, y, z จะมีผลภายใน { } นี้ และเป็นคนละตัวแปรกับ a, b, x, y, z ใน func_AA และ main  */
{
int b,x,y,z;
       b=func_AA(a);  /*  BB_1  */
       b=a*b;         /*  BB_2  */
       return(b);     /*  BB_3  */
}
```

x=2	หลัง M1 y=	หลัง M2 z=

x=3	หลัง M1 y=	หลัง M2 z=

7. โจทย์ตัวอย่าง Function: จงทำโปรแกรมข้างล่างให้สมบูรณ์เพื่อทำให้โปรแกรมนี้ทำการคำนวณหาพื้นที่ของวงแหวน โดยเขียนและเรียกใช้ฟังก์ชันที่คำนวณหาพื้นที่วงกลม




```
#include <stdio.h>

……................................................................................
int main()
{
double R1,R2,a1,a2,area;
        printf("Enter radius:");
        scanf("%lf %lf",&R1,R2);

         ……................................................................

         ……................................................................
        area=a1-a2;
        printf("Area of the ring=%.3f\n",area);
        return 0;
}

……................................................................................
{




}
```

8. โจทย์ตัวอย่างการใช้ `break` statement: จงทำโปรแกรมข้างล่างให้สมบูรณ์เพื่อทำการรับค่าตัวเลขจำนวนเต็มบวกที่ไม่เท่ากับศูนย์ 2 จำนวนจากผู้ใช้ หลังจากนั้นให้ทำการหาค่าตัวคูณร่วมน้อย หรือ ค.ร.น. (Least Common Multiple or LCM) ของ 2 จำนวนนี้แล้วนำมาแสดงผลออกสู่หน้าจอ เช่น ถ้าป้อน 8 12 โปรแกรมจะแสดง 24

```
#include <stdio.h>
int main()
{
int x,y,lcm,i,start,stop;
      scanf("%d %d",&x,&y);
      if(x>y)
         start=_____;
      else
         start=_____;
      stop=_________;
      for(i=_____;__________;_______)
      {
            if(_________________________)
            {
                  lcm=____________________;
                  _________________________;
            }
      }
      printf("%d\n",lcm);
      return 0;
}
```

9. โจทย์ตัวอย่างการใช้ `continue` statement: จงทำโปรแกรมข้างล่างให้สมบูรณ์เพื่อทำให้โปรแกรมหาค่าเฉลี่ยของข้อมูลที่เป็นค่าบวก (ไม่สนใจ 0 หรือค่าลบ) โดยข้อมูลนำเข้าบรรทัดแรกจะเป็นจำนวนข้อมูล (N) และ N บรรทัดต่อมาจะเป็นข้อมูลซึ่งเป็นไปได้ทั้งค่าบวกและลบ

```
#include <stdio.h>
int main()
{
 int N,i,x,sum,count;
double avg;
            scanf("%d",&N);
            sum=_______________________;
            count=_____________________;
            for(i=0;i<N;i++)
            {
               scanf("%d",&x);
               if(_______________________)
                  _______________________;

                sum=____________________;
                count=__________________;
            }
            avg=_______________________;
            printf("%.2f\n",avg);
           return(0);
}
```

10. โจทย์ตัวอย่าง nested loop: จงเขียนโปรแกรมเพื่อแสดงเครื่องหมายกากบาทโดยใช้ตัวอักษร ‘O’ บนพื้นหลังที่แสดงโดยตัวอักษร ‘.’ โดยโปรแกรมจะรับค่าจำนวนเต็มบวก 1 ค่าสำหรับระบุค่าความกว้าง/ยาวของรูปนี้ (ค่าที่ป้อนเข้ามาจะเป็นเลขคี่เสมอ) ตัวอย่างข้างล่างเป็นรูปของเครื่องหมายกากบาทเมื่อป้อนค่า 11 (ขนาด 11x11)

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int x,y,N,Abx,Aby;

      scanf("%d",&N);
      for(________;________;________)
      {
            for(________;________;________)
            {
                  Abx=
                  Aby=
                  if(_______________________)

                        printf(________);
                  else

                        printf(________);
            }
            printf("________");
      }
      return 0;
}
```

```
O.........O
.O.......O.
..O.....O..
...O...O...
....O.O....
.....O.....
....O.O....
...O...O...
..O.....O..
.O.......O.
O.........O
```

11. โจทย์ตัวอย่าง Function: จงทำโปรแกรมข้างล่างให้สมบูรณ์เพื่อทำให้โปรแกรมนี้ทำการคำนวณหาค่า $\sum_{k=-N}^{N} e^k$ โดยที่ค่า N จะถูกป้อนเข้ามาทางคีย์บอร์ด โดยกำหนดให้ค่า e^x คำนวณโดยใช้สูตรการประมาณดังต่อไปนี้

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + ... + \frac{x^m}{m!} \quad where \quad m = 50$$

หลังจากนั้นให้แสดงค่า $\sum_{k=-N}^{N} e^k$ ที่คำนวณได้ในบรรทัดถัดไป โดยแสดงเป็นตัวเลขทศนิยม 2 ตำแหน่ง ตัวอย่างเช่นถ้าป้อน N=5 โปรแกรมจะคำนวณหา $e^{-5} + e^{-4} + e^{-3} + e^{-2} + e^{-1} + e^{-0} + e^{5} + e^{4} + e^{3} + e^{2} + e^{1} = 234.78$

```
#include <stdio.h>
_____________________________________________/* Function declaration */
int main()
{
double val,sum;
int k,N;

      scanf("%d",&N);
      sum=
      for(        ;        ;         )
      {
            val=
            sum=
      }
      printf("%.2f\n",sum);
      return 0;
}

_____________________________________________________/* Function definition */
{ /* คำนวณและคืนค่า ex */

}
```