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

Handout 5

<u>หัวข้อ</u>

- ทบทวน
- variable storage class: local variables และ global variables (Chapter 12.2.3 หน้า 311)
- Testing and Debugging (Chapter 15)

<u>ทบทวน</u>

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

<u>Local variables VS global variables</u>

1. Local variables เป็นตัวแปรที่ประกาศภายในฟังก์ชัน (รวมทั้ง argument ของฟังก์ชัน โดยที่ ขอบเขต หรือ scope ของตัวแปรจะมีอยู่ภายในฟังก์ชันเท่านั้น ตัวแปรที่เราได้เรียนกันก่อนหน้านี้ถือเป็นตัวแปรแบบ local ทั้งหมด

- ตัวอย่าง

```
01 #include <stdio.h>
02 int Func1(int);
03 int count;
04 double gVal;
05 int main()
06 {
07    int n,i,sum;
08       scanf("%d",&n);
09       i=100;
10       count=0;
11       sum=Func1(n);
12       printf("m:%d %d %d\n",n,i,sum);
13       printf("m:%d %.2lf\n",count,gVal);
14       return 0;
15 }
16 int Func1(int n)
17 {
18    int i,sum;
19      sum=0;
20      gVal=1.23;
21      i=1;
22      while(n>0)
23      {
24         sum=sum+n;
25         n=n-1;
26      }
27      printf("F1:%d %d %d",n,i,sum);
28      printf("m:%d %.2lf\n",count,gVal);
29      count=sum;
30      return(sum);
31  }
```

- ในโปรแกรมนี้จะมี ตัวแปร n, i, sum อยู่ 2 ชุด ซึ่งมีชื่อเหมือนกัน แต่มองได้เป็นคนละตัวแปร

- ตัวแปร n, i, sum ที่ประกาศไว้ที่บรรทัด 7 จะมี scope อยู่ในฟังก์ชัน main() ตั้งแต่ { บรรทัด 6 ถึง } บรรทัด 13

- argument n ที่บรรทัด 14 และตัวแปร i, sum ที่ประกาศไว้ที่บรรทัด 16 จะมี scope อยู่ในฟังก์ชัน Func1() ตั้งแต่ { บรรทัด 16 ถึง } บรรทัด 26

- ตัวแปร count และ gVal เป็น global variable ซึ่งสามารถอ้างถึงได้ทั้งภายใน main() และ Func1()

- ผลลัพธ์ที่หน้าจอคืออะไร ถ้าผู้ใช้ป้อน 5

2. Global variables เป็นตัวแปรที่ประกาศไว้ข้างนอกฟังก์ชัน ดังเช่น ตัวแปร count และ gVal ที่ประกาศในบรรทัดที่ 4 และ 5 ซึ่งจะมีขอบเขตในทุกฟังก์ชันในไฟล์โปรแกรม (สามารถตั้งชื่อตัวแปรให้เหมือนชื่อ local variables ได้ ถ้าไม่จำเป็น ไม่แนะนำให้ใช้ global variable แต่ถ้าจำเป็นให้ตั้งชื่อโดยใช้ตัว g นำหน้า เช่น gVal, gCount)

3. โจทย์ตัวอย่าง local vs global variables จงระบุผลลัพธ์ของโปรแกรมต่อไปนี้

```
#include <stdio.h>
int Multiply(int,int);
int d;
int main()
{
  int a,b,c;
  int e;
   a=1;
   b=2;
   d=3;
   e=4;
   c=Multiply(a,b);
   printf("%d %d %d %d %d\n",a,b,c,d,e);
}
int Multiply(int c,int e)
{
  int a;
      a=2;
      e=3;
      d=d+10;
      return(a*e);
}
```

4. โจทย์ตัวอย่าง: จงเขียนโปรแกรมเพื่อหาค่า $\int_a^b x^2 + 2x + 3\,dx$ โดยที่ a และ b จะถูกป้อนมาทางคีย์บอร์ด

Note: Algorithm ในการคำนวณ numerical integration $\int_a^b f(x)\,dx$ แสดงได้ด้วยสูตรข้างล่าง (Riemann Integral Sum)

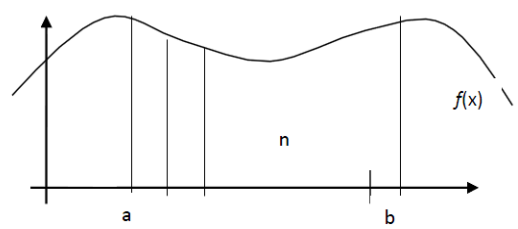
```
#include <stdio.h>

_____________________________________/**/
int main()
{
____________a,b,sum,c,x,y;
int i,n;
     scanf(_________________________);
     n=10000000;
     sum=_________;
     c =_____________________;
     for(_________ ;_________;_________)
     {
        x=_____________________________;
        y=_____________________________;
        sum=___________________________;
     }
     printf("%.6f\n",sum);
     return(0);
}

_____________________________________/**/
{




}
```

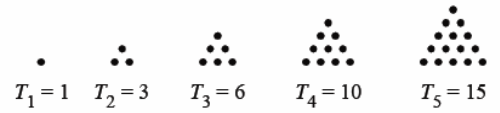


$$\int_a^b f(x)dx = \lim_{n\to\infty}\sum_{i=0}^{n-1} f\left(a + \frac{b-a}{n}i\right)\frac{b-a}{n}$$

5. โจทย์ตัวอย่าง Nested loop: จงเขียนโปรแกรมเพื่อแสดงรูปแบบ Diamond โดยใช้ตัวอักษร ‘x’, ‘o’ และ ‘.’ ดังตัวอย่างรูปข้างล่าง โดยกำหนดให้ผู้ใช้ป้อนค่าตัวเลขจำนวนเต็มบวกที่เป็นเลขคี่ สมมติว่าชื่อ N ที่สามารถแทนอยู่ในรูป 4k+1 โดยที่ k=4, 5, 6,…. (นั้นคือ N มีค่า 17, 21, 25, 29, 33, 37, 41, …..) โดยที่โปรแกรมจะนำค่า N นี้มาใช้เป็นจำนวนแถวและคอลัมน์ของรูปที่จะแสดงออกบนหน้าจอ ตัวอย่างด้านข้างกำหนด N=17

```
17 ↵
oooooooo.oooooooo
ooooooo.x.ooooooo
oooooo.xxx.oooooo
ooooo.xxxxx.ooooo
oooo.xxxxxxx.oooo
ooo.x.xxxxx.x.ooo
oo.xxx.xxx.xxx.oo
o.xxxxx.x.xxxxx.o
.xxxxxxx.xxxxxxx.
o.xxxxx.x.xxxxx.o
oo.xxx.xxx.xxx.oo
ooo.x.xxxxx.x.ooo
oooo.xxxxxxx.oooo
ooooo.xxxxx.ooooo
oooooo.xxx.oooooo
ooooooo.x.ooooooo
oooooooo.oooooooo
```

6. โจทย์ตัวอย่าง Function: <u>นิยาม</u> จำนวนเต็มหนึ่งๆ สมมติว่าคือ x จะเป็น triangular number ก็ต่อเมื่อ ถ้าเรามีก้อนหินจำนวน x ลูกแล้วเราสามารถนำมาเรียงเป็นสามเหลี่ยม ดังรูปข้างล่าง 10 จำนวนเต็มแรกที่เป็นtriangle number คือ 0, 1, 3, 6, 10, 15, 21, 28, 36, and, 45 จงเขียนโปรแกรมที่รับตัวเลขจำนวนเต็มบวก 1 จำนวน แล้วตรวจสอบว่าจำนวนนั้นเป็น triangle number หรือไม่ โดยกำหนดเขียนและเรียกใช้ฟังก์ชัน int isTriangular(int n) ซึ่งจะคืนค่า 1 ถ้า n เป็น triangular number มิฉะนั้นคืนค่า 0



Testing and Debugging

1. Testing หา bug ของโปรแกรม: โปรแกรมอาจจะมีหรือไม่มี bug ก็ได้

2. Debugging: แก้ bug ของโปรแกรม: โปรแกรมมี bug บางอย่างที่เรารู้ เช่น ป้อนอินพุตค่าหนึ่งๆ แต่ไม่ได้ output ที่ต้องการ โปรแกรมเมอร์ต้องแก้โค้ดของโปรแกรมเพื่อให้ bug นี้หายไป

3. Types of errors:

- Syntactic error คือความผิดพลาดที่เขียนไม่เป็นไปตามรูปแบบไวยากรณ์ (syntax) ของภาษา C และสามารถตรวจเจอได้ในขั้นตอนการ compile

- Semantic error คือความผิดพลาดในการเขียนโปรแกรมที่ทำให้โปรแกรมทำงานผิดไปจากวัตถุประสงค์ที่เราต้องการ และข้อผิดพลาดไม่ผิดทาง syntax ซึ่งจะทำให้ compiler ไม่สามารถตรวจเจอได้

Algorithmic errors คือความผิดพลาดที่เริ่มต้นตั้งแต่การออกแบบโปรแกรม แต่การเขียนโปรแกรมถูกต้องตรงตามการออกแบบและไม่มี error ใดๆ เช่น Y2K bug

ตัวอย่าง Syntactic error	ตัวอย่าง Semantic error โปรแกรมคำนวณค่า 1+2+..+10	ตัวอย่าง Semantic error โปรแกรมพิมพ์สูตรคูณ x7

`int main()`
`{`
`int i`
`int j;`
`  i=10`
`  j=i*(i+1;`
 	
`return 0`
`}`
	`int main()`
	`{`
	`int i,sum;`
	
	`  for(i=0;i<10;i++);`
	`  sum=sum+i;`
	`  printf("%d\n",sum);`
	`  return 0;`
	`}`
		`int main()`
		`{`
		`int i,x;`
		
		`  for(i=1;i<=10;i++)`
		`  x=i*7;`
		`  printf("%dx7=%d\n",i,x);`
		`  return 0;`
		`}`

4. Testing แบ่งออกเป็น 2 ประเภท (ตาม textbook) ได้แก่

- Black-box testing: การตรวจสอบ input และ output ของโปรแกรมว่าเป็นไปตามข้อกำหนดหรือไม่ ชุดทดสอบ (test case) ที่ให้ในโจทย์ programming ในวิชานี้จัดอยู่ในกลุ่มนี้

- White-box testing: ตรวจสอบแต่ละ component ของโปรแกรม เช่นตรวจที่ละ ฟังก์ชันหลักๆที่ได้ออกแบบไว้

5. เทคนิคการ debugging ตาม textbook พูดไว้ 2 แนวทาง ได้แก่

- Ad-hoc technique: แบบใช้เฉพาะสถานการณ์ ไม่มีหลักตายตัว เช่น การแทรก printf() เข้าไปใน program เพื่อแสดงข้อมูลตัวแปรภายในของโปรแกรม

- Source-level debuggers: ใช้ debugger (เช่น เครื่องมือที่มากับ Visual Studio) เพื่อตรวจดูการทำงานภายในโปรแกรม

(i) Breakpoints, Conditional breakpoints (โปรแกรมจะหยุดที่ breakpoint เมื่อ condition เป็นไปตามข้อกำหนด), watchpoints (โปรแกรมจะหยุด ณ ที่ใดๆ ในโปรแกรมเมื่อ condition เป็นไปตามข้อกำหนด)

(ii) Single-stepping

(iii) Display values

6. Programming for correctness: ใน textbook หน้า 417 แนะนำ tip ในการเขียนโปรแกรมไม่ให้มี bug โดยเทคนิคที่แนะนำคือ Defensive programming (ป้องกันไม่ให้มี bug ตั้งแต่ขั้นตอนการเขียนโปรแกรม) โดยแนะนำ tip ต่อไปนี้

Comment your code **Adopt a consistent coding style** **Avoid assumption**

Avoid global variables **Rely on the compiler**

7. โจทย์ตัวอย่าง จาก Textbook Exercuses 15.5 หน้า 424: จงใช้ source level debugger ทำการตรวจสอบการทำงานภายในโปรแกรม ดังต่อไปนี้

(a) ให้เซ็ต breakpoint ที่บรรทัดแรกของฟังก์ชัน `IsDivisibleBy` และตรวจสอบว่าค่าของ argument ที่ส่งเข้ามาของ function call 10 ครั้งแรกคืออะไร

(b) ค่าของ f หลังจาก for loop ข้างใน main ณ iteration ที่ i เท่ากับ 660

```
#include <stdio.h>
int IsDivisibleBy(int,int);
int main()
{
int i,j,f;
    for(i=2;i<1000;i++)
    {
      f=0;
      for(j=2;j<i;j++)
      {
        if(IsDivisibleBy(i,j))
             f++;
      }
      printf("The number %d has %d factors\n”,i,f);
   }
   return 0;
}
int IsDivisibleBy(int dividend,int divisor)
{
   if(dividend%divisor==0)
      return 1;
   else
      return 0;
}
```

8. โจทย์ตัวอย่าง จาก Textbook Exercises 15.6 หน้า 425: จงใช้ source level debugger ทำการตรวจสอบการทำงานภายในโปรแกรม เพื่อดูว่าฟังก์ชัน Mystery จะคืนค่า 0 เมื่อ argument ของฟังก์ชัน (a,b,c) มีค่าเป็นอะไรบ้าง

```
#include <stdio.h>
int Mystery(int,int,int);
int main()
{
int i,j,k,sum;
    sum =0;
    for(i=100;i>0;i--)
       for(j=1;j<i;j++)
          for(k=j;k<100;k++)
             sum=sum+Mystery(i,j,k);

   return 0;
}
int IsDivisibleBy(int dividend,int divisor)
{
int out;
    out= 3*a*a + 7*a  - 5*b*b + 4*b + 5*c;
    return(out);
}
```

9. โจทย์ตัวอย่าง: เมื่อ compile โปรแกรมข้างล่างจะมีข้อความเตือน (warning) จาก compiler ว่าอะไรและผลลัพธ์การทำงานของโปรแกรมข้างล่างคืออะไร

```
#include<stdio.h>
main()
{
   int i;
   int a;
   for (i = 0; i < 4; i++)
       a = a * i;
   printf("%d\n", a);
}
```

```
#include<stdio.h>
main()
{
   int i;
   int a;
   for (i = 0; i < 4; i++)
       a = a * i;

   printf("%d\n", a);
}
```

10. โจทย์ตัวอย่างจาก Textbook Exercises 15.3 หน้า 422: โปรแกรมข้างล่างจะทำการหาค่าน้อยที่สุด (minimum) จากข้อมูลจำนวนเต็มบวกที่ป้อนเข้ามาจากคีย์บอร์ด ผู้ใช้จะระบุจุดสิ้นสุดของข้อมูลเมื่อเจอค่า -1 หลังจากข้อมูลทุกตัวป้อนครบ โปรแกรมจะแสดงผลลัพธ์ข้อมูลน้อยสุด เช่น ถ้าป้อน 4 5 3 7 2 10 -1 โปรแกรมควรจะแสดงค่า 2 แต่โปรแกรมนี้มี bug ทำให้ไม่ทำงานตามที่กำหนด จงใช้ source level debugger ค้นหาและแก้ไข bug โดยให้ระบุจุดที่ทำให้เกิด bug และวิธีแก้ไข

```
#include <stdio.h>
int main()
{
 int smallestNumber=0;
int nextInput;
      scanf("%d",&nextInput);
      while(nextInput!=-1)
      {
         if(nextInput<smallestNumber)
            smallestNumber=nextInput;
         scanf("%d",&nextInput);
      }
      printf("The smallest number is %d\n",smallestNumber);
      return 0;
}
```

11. โจทย์ตัวอย่าง: โปรแกรมข้างล่างจะทำการหาผลบวก (sum) และผลคูณ (product) จากข้อมูลจำนวนเต็ม 10 จำนวนที่ป้อนเข้ามาจากคีย์บอร์ด แต่โปรแกรมนี้มี bug ทำให้ไม่ทำงานตามที่กำหนด จงใช้ source level debugger ค้นหาและแก้ไข bug โดยให้ระบุจุดที่ทำให้เกิด bug และวิธีแก้ไข

```
#include <stdio.h>
int main()
{
   int i,sum,product,input;
   i = 0;
   sum = 0;
   product = 0;
   for (i = 0; i < 10; i++) {
       scanf("%d", &input);
       sum += input;
       product *= input;
   }
   printf("Sum= %d, product= %d\n", sum, product);
   return 0;
}
```