ทบทวนการเขียนโปรแกรม
Programming Review

- การแก้ปัญหา
- การออกแบบอัลกอริทึม
- โครงสร้างควบคุมแบบเงื่อนไขและทำซ้ำ
- ฟังก์ชันและการส่งผ่านค่าพารามิเตอร์

ศิลปะของการแก้ปัญหา(ทั่วไป)

1. ทำความเข้าใจกับปัญหา
2. ออกแบบแผนการแก้ปัญหา
3. ดำเนินการตามแผน
4. ติดตามและประเมินผลว่าขั้นตอนการแก้ปัญหาที่ออกแบบนั้นถูกต้องแม่นยำดีหรือไม่ เพื่อประยุกต์ขั้นตอนนั้นกับปัญหาอื่นๆ ได้

Problem Solving

- Analysis (การวิเคราะห์ ปัญหา)
  - ทำความเข้าใจปัญหาอย่างละเอียด
  - ปัญหาที่ซับซ้อน ต้องถูกแบ่งหรือแตกออก(Decomposing)เป็นปัญหาย่อยๆ
  - ต้องบันทึกความสัมพันธ์ระหว่างปัญหาย่อยเหล่านั้น
- Synthesis (การสังเคราะห์ คำตอบ)
  - แก้ปัญหาย่อย
  - นำคำตอบหรือการแก้ปัญหาย่อยประกอบเข้าด้วยกันตามความสัมพันธ์ของปัญหา

วิธีแก้ปัญหาด้วยคอมพิวเตอร์

การแก้ปัญหาด้วยคอมพิวเตอร์ ต้องการความเข้าใจในประเด็นต่างๆดังนี้

- เข้าใจในงานหรือปัญหาให้ชัดเจน
- ผลลัพธ์ที่ได้จากการทำงานคืออะไร
- ข้อมูลจะเข้าสู่งานหรือระบบได้อย่างไร
- มีวิธีจัดการกับข้อมูลอย่างไร
- แนวทางการแก้ปัญหานั้น มีขั้นตอนอย่างไร
- ผู้ปฏิบัติงานทำงานได้ง่ายและสะดวกขึ้นหรือไม่

วัฏจักรการพัฒนาโปรแกรม

- การวิเคราะห์และกำหนดคุณสมบัติของปัญหา (Analyzing and Defining the problem)
- การออกแบบแนวทางการแก้ปัญหา (Designing the solution)
- การเขียนเป็นภาษาคอมพิวเตอร์ (Coding the program)
- การทดสอบโปรแกรม (Testing the program)
- การจัดทำเอกสารประกอบและการบำรุงรักษาโปรแกรม (Documenting the program and maintenance)

การวิเคราะห์และการกำหนดคุณสมบัติของปัญหา

- ความต้องการเกี่ยวกับผลลัพธ์ (Output Requirement)
  - วัตถุประสงค์ของการแก้ปัญหา
  - กำหนดรูปแบบการนำเสนอผลลัพธ์
- การรับข้อมูล (Input Requirement)
  - ปัญหาบอกข้อมูลอะไรบ้าง
  - กำหนดรูปแบบของข้อมูลที่รับเข้า
- วิธีการคำนวณ(Method of Calculation)
  - กำหนดสูตรหรือวิธีการ
  - กำหนดเงื่อนไขของการคำนวณ (ถ้ามี) เพื่อป้องกันไม่ให้โปรแกรมทำงานผิดพลาด เช่น
    - กรณีการหารด้วยศูนย์
    - การวนซ้ำไม่รู้จบ

ข้อกำหนดและคุณสมบัติต่างๆของโปรแกรม จะเป็นเอกสารอ้างอิงและใช้ในการทดสอบระบบ

ตัวอย่าง

- จงเขียนโปรแกรมภาษาซี เพื่อรับข้อมูลซึ่งเป็นเลขจำนวนเต็ม มีค่าอยู่ในช่วง -1000 ถึง 1000 (นั่นคือ ไม่รวม -1000 และ 1000) เพื่อหาค่าสูงสุดและค่าต่ำสุดของข้อมูลชุดนี้ซึ่ง*ไม่ทราบจำนวนข้อมูลทั้งหมด* โดยให้มีฟังก์ชัน Is_in_Range() ทำหน้าที่ตรวจสอบว่าข้อมูลอยู่ในช่วงที่กำหนดหรือไม่ โดยฟังก์ชันจะมีการส่งค่ากลับเป็น
  - จริง เมื่อข้อมูลมีค่าอยู่ในช่วงที่กำหนด
  - เท็จ เมื่อข้อมูลมีค่าไม่เป็นตามกำหนด

จงเขียนผลการวิเคราะห์และการออกแบบฟังก์ชัน พร้อมทั้งชุดคำสั่งภาษาซี

การวิเคราะห์และการกำหนดคุณสมบัติของปัญหา

- ความต้องการเกี่ยวกับผลลัพธ์ (Output Requirements)
  - วัตถุประสงค์ของการแก้ปัญหา
    1. หาค่ามากที่สุด(*max*)
    2. หาค่าน้อยที่สุด(*min*)
  - กำหนดรูปแบบการนำเสนอผลลัพธ์
    1. Maximum of this data group is *max*.
    2. The minimum is *min*.

การวิเคราะห์และการกำหนดคุณสมบัติของปัญหา

- การรับข้อมูล (Input Requirements)
  - ปัญหาบอกข้อมูลอะไรบ้าง

  เลขจำนวนเต็ม (*in*) และ*ไม่ทราบจำนวนข้อมูลทั้งหมด*
  - กำหนดรูปแบบของข้อมูลที่รับเข้า

  Please enter integer number in range of -1000 and 1000, input number which is out of range to stop.

  Input number : *in*

Cursor รอรับการป้อนข้อมูล
ข้อมูลที่ป้อนจะเข้าเป็นค่าของตัวแปร in

การวิเคราะห์และการกำหนดคุณสมบัติของปัญหา

- วิธีการคำนวณ(Method of Calculation) คือตอบคำถาม WHAT TO DO และกำหนดเงื่อนไขของการคำนวณ
  1. กำหนดค่าเริ่มต้น ให้กับ max และ min ด้วยค่าที่มั่นใจว่าจะต้องถูกเปลี่ยนด้วยข้อมูลที่รับเข้า(in)
  2. รับค่า in
  3. นำ in ไปตรวจสอบว่าเป็น ค่า max หรือ min ไหม

  ทำซ้ำ ในขณะที่ข้อมูล in อยู่ในช่วง
  4. แสดงผลลัพธ์ max และ min ตามรูปแบบที่ได้ออกแบบไว้

แบบร่วมฝึก

- จัดนักศึกษาเป็น 2 กลุ่ม ตามที่นั่ง
- ให้แต่ละกลุ่ม ช่วยกันวิเคราะห์ปัญหา การแก้สมการควอดดราติก (Quadratic Equation)

$$aX^2+bX+c=0$$

- ให้เวลา 10 นาที

ทบทวนการเขียนโปรแกรม
Programming Review

- การแก้ปัญหา
- การออกแบบอัลกอริทึม
- โครงสร้างควบคุมแบบเงื่อนไขและทำซ้ำ
- ฟังก์ชันและการส่งผ่านค่าพารามิเตอร์

วัฏจักรการพัฒนาโปรแกรม

- การวิเคราะห์และกำหนดคุณสมบัติของปัญหา (Analyzing and Defining the problem)
- การออกแบบแนวทางการแก้ปัญหา (Designing the solution)
- การเขียนเป็นภาษาคอมพิวเตอร์ (Coding the program)
- การทดสอบโปรแกรม (Testing the program)
- การจัดทำเอกสารประกอบและการบำรุงรักษาโปรแกรม (Documenting the program and maintenance)

การออกแบบวิธีแก้ปัญหา

- เทคนิคและวิธีมาตรฐานที่ได้รับการยอมรับ
  - Divide-and-Conquer
  - Modularization
  - Top-down Design
  - Bottom-up Design
- หลักการสำคัญ
  - การเลือกโครงสร้างข้อมูล
  - การออกแบบอัลกอริทึม (Algorithm)

WORTH

โครงสร้างข้อมูล + อัลกอริทึม = โปรแกรม

Abstraction: Definitions

- Abstraction = the distinction between the external properties of an entity and the details of the entity's internal composition.
- Abstract tool = a component of a larger system whose internal composition we ignore.

Uses of abstraction

- Abstraction allows us to use things we don't fully understand.
- We all can use electrical devices, food, etc. that we either do not understand or cannot produce.
- Computer scientists can use algorithms implemented by others without understanding their details.

Algorithms: levels of abstraction

- Problem = motivation for algorithm
- Algorithm = procedure to solve the problem
  - Often one of many possibilities
- Representation = description of algorithm sufficient to communicate it to the desired audience
  - Always one of many possibilities

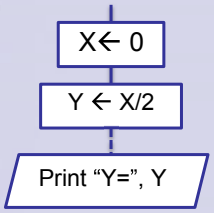
Representation of Algorithms

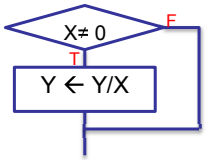
- ภาษาธรรมชาติ
  - เขียนด้วยภาษาทั่วไป โดยมีหมายเลขกำกับขั้นตอนอย่างชัดเจน
  - ภาษารูปภาพ
- *คำสั่งลำลอง* (Pseudo code) : อธิบายโครงสร้างการทำงานของโปรแกรมโดยใช้ภาษาทั่วไป
- *ผังงาน* (Flowchart) : กำหนดสัญลักษณ์ที่มีความหมายเฉพาะ อธิบายการทำงานผ่านสัญลักษณ์ ลำดับการทำงานสอดคล้องกับลำดับของสัญลักษณ์ที่ปรากฏ

การโปรแกรมแบบโครงสร้าง

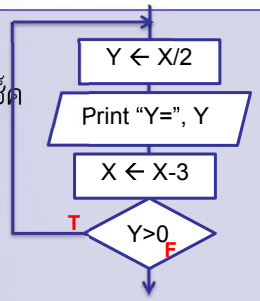
- มีโครงสร้างการดำเนินการอย่างชัดเจน
  - แบบลำดับ
  - แบบเลือกดำเนินการอย่างมีเงื่อนไข
    - ทางเลือกไม่เกิน 2 ทาง (if-else)
    - ทางเลือกมากกว่า 2 ทาง (switch/case)
  - แบบทำซ้ำ
    - ตรวจสอบเงื่อนไขก่อนดำเนินการ(CHECK first DO later)
    - ดำเนินการแล้วตรวจสอบเงื่อนไข (DO first CHECK last)
- แต่ละโครงสร้างมีรูปแบบการเขียนเฉพาะตัว

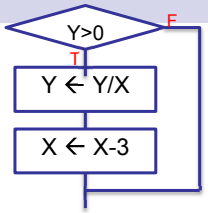
รูปแบบการเขียนของแต่ละโครงสร้าง

โครงสร้าง	ผังงาน	Pseudo code	C statement
ลำดับ		X ← 0 Y ← X / 2 : Display "Y=", Y	int X, Y; : X = 0; Y = X/2; : printf("Y=%d\n", Y);

เลือกไม่เกิน 2 ทาง		If X≠ 0 then
		Y ← Y/X
		Endif
			if (X != 0)
			Y = Y/X;

รูปแบบการเขียนของแต่ละโครงสร้าง

โครงสร้าง	ผังงาน	Pseudo code	C statement
ทำซ้ำ ทำก่อนเช็ค		Repeat Y ← X / 2 Display "Y=", Y X ← X - 3 While Y>0	do { Y = X/2; printf("Y=%d\n", Y); X -= 3; } while (Y>0);

เช็คก่อนทำ		while Y > 0 do
		(Y ← Y/X
		X ← X - 3)
			while (Y > 0) {
			Y = Y/X;
			X -= 3;
			}

ตัวอย่าง

```
#include <stdio.h>
void main() {
   int  a = 1, b = 2,  c = 3;
   x = a + b;
   print("x = %d\n", x);
}
```

เมื่อทำการ compile โปรแกรมนี้จะมีข้อผิดพลาดปรากฏดังนี้

```
Error EX1.C  7: Undefined symbol 'x' in function main
Warning EX1.C 9: 'c' is assigned a value that is never used
```


โปรแกรมภาษาซี

```
/* ข้อความนี้เป็นเพียงคำอธิบาย ไม่มีผลต่อขั้นตอนการทำงานของโปรแกรม */
/* A simple program to display a line of text */
#include  < stdio.h >
void main ( )
{
   printf ("Hello, welcome to C programming\n") ;
}
```

ผลลัพธ์ที่จะได้เมื่อ run โปรแกรมนี้ คือ

```
Hello, welcome to C programming
[]
```

Cursor

โปรแกรมภาษาซี

```
/* โปรแกรมแบบง่ายๆ แสดงข้อความ 1 บรรทัด
   ตัวโปรแกรมดูแตกต่าง แต่ผลลัพธ์ที่ได้ จะเหมือนกัน */
#include  < stdio.h >
void main ( )
{
   printf ("Hello, ");
   printf ("welcome to C programming");
   printf ("\n") ;
}
```

ผลลัพธ์ที่จะได้เมื่อ run โปรแกรมนี้ คือ

```
Hello, welcome to C programming
```



```
#include  < stdio.h >
void main ( ) {
   /* ประกาศตัวแปรหนึ่งตัวชื่อ number มีชนิดข้อมูลเป็นเลขจำนวน
   เต็ม */
   int   number;

   /* กำหนดค่า 8 ให้กับตัวแปร number  */
   number = 8;
   printf ("The value of number is %d . ", number);
}
```

ผลลัพธ์ที่จะได้เมื่อ run โปรแกรมนี้ คือ

```
The value of number is 8 .
```



```
#include  < stdio.h >
#define       SIXTY  60   /* เรียก SIXTY ว่าเป็นค่าคงที่สัญลักษณ์ */
void main ( ) {
    float hour;  /* ตัวแปรชื่อ hour ถูกประกาศให้มีชนิดข้อมูลเป็นเลขจำนวนจริง */
    int   minute, second; /* ตัวแปร 2 ตัวถูกประกาศ โดยมีชนิดข้อมูลเป็นเลขจำนวนเต็ม */

    hour = 1.5; /* กำหนดค่า 1.5 ให้กับตัวแปร hour  */
    /* คูณค่าของตัวแปร hour ด้วย 60 แล้วกำหนดให้เป็นค่าของตัวแปร minute  */
    minute = hour * SIXTY;
    /* คูณค่าของตัวแปร minute ด้วย 60 แล้วกำหนดให้เป็นค่าของตัวแปร second  */
    second = minute * SIXTY;
    printf ("In one period : \n %.2f hours\n", hour);
    printf ("%d minutes\n%d seconds", minute, second);
}
```

```
In one period :
1.50 hours
90 minutes
540 seconds
```


รูปแบบโปรแกรมภาษาซีอย่างง่าย

Preprocessing Directives

ส่วนที่ตัวแปลภาษาต้องดำเนินการก่อนทำการแปล

```
void main ( )
{
```

Declarations

ส่วนของการประกาศตัวแปรที่จะใช้ภายในบล็อคนี้ซึ่งถูกคร่อมด้วย { และ}

Statements

คำสั่งต่างๆ ซึ่งจะมีผลต่อขั้นตอนการทำงาน

```
}
```

Formatted output function printf( )

- printf() เป็นฟังก์ชันสำหรับแสดงค่าทางอุปกรณ์แสดงผลมาตรฐาน (จอภาพ)
- รูปแบบการเรียกใช้ฟังก์ชัน

  printf (*control*, arg_1, arg_2, ..., arg_n);
- arg_1, arg_2, ..., arg_n : นิพจน์ที่ต้องการแสดงค่า
- *control* : "ข้อความที่ต้องการแสดง และ *รูปแบบของการแสดงผลข้อมูล*"
- *รูปแบบของการแสดงผลข้อมูล* : %สัญลักษณ์

%สัญลักษณ์	สำหรับนิพจน์ที่มีชนิดข้อมูลเป็น
%c	ตัวอักขระ
%d	เลขจำนวนเต็ม
%f	เลขจำนวนจริงหรือเลขมีจุดทศนิยม
%s	สายอักขระ

โปรแกรมภาษาซี

```
#include  < stdio.h >

void main ( ) {
   int   years;

   printf ("How long have you been here? ");
   scanf ("%d", &years);
   printf ("You've been here for %d years.", years);
   printf ("\tReally?");
}
```

How long have you been here? 20
You've been here for 20 years. Really?

ค่าที่ป้อน

ผลของ \t ซึ่งหมายถึง ช่องว่าง 1 tab

โปรแกรมภาษาซี

```
#include  < stdio.h >
#define     PI   3.14159   /* กำหนดให้ค่าคงที่สัญลักษณ์ PI มีค่าเป็น 3.14159*/
void main ( ) {
   float    r,  area;  /* ตัวแปร 2 ตัวถูกประกาศ โดยมีชนิดข้อมูลเป็นเลขจำนวนจริง */

   printf ("This program computes the area of a circle\n");
   r = 12.5; /* กำหนดให้ รัศมีมีค่าเป็น 12.5 */
   area = PI * r * r; /* พื้นที่ของวงกลม = ¶ r^2  */
   printf ("Area = Pi x radius x radius\n");
   printf ("       = %f x %.2f x %.2f\n", PI, r, r);
   printf ("       = %f\n", area);
}
```

This program computes the area of a circle
Area = Pi x radius x radius
= 3.141590 x 12.50 x 12.50
= 490.873444

Formatted input function scanf( )

- scanf() เป็นฟังก์ชันสำหรับรับค่าจากอุปกรณ์นำเข้ามาตรฐาน (แป้นพิมพ์)
- รูปแบบการเรียกใช้ฟังก์ชัน

  scanf (*control*, arg_1, arg_2, ..., arg_n);
- arg_1, arg_2, ..., arg_n : address ของตัวแปรที่ใช้รับค่า
- *control* : "*รูปแบบของข้อมูลที่ต้องการรับ*"
- *รูปแบบของข้อมูลที่ต้องการรับ* : %สัญลักษณ์

%สัญลักษณ์	ข้อมูลที่อ่านได้หรือที่ป้อนเข้าจะถูกเปลี่ยนเป็น
%c	ตัวอักขระ
%d	เลขจำนวนเต็ม
%f	เลขจำนวนจริงหรือเลขมีจุดทศนิยม
%s	สายอักขระ

```
#include  < stdio.h >
 void main ( ) {
    int   years;
    printf ("How long have you been here? ");
    scanf ("%d", &years);
    printf ("You've been here for %d years.", years);
    printf ("\tReally?");
}
```

กำหนดรูปแบบข้อมูลที่จะรับเข้าเป็นเลขจำนวนเต็ม

& เป็นตัวดำเนินการ ซึ่งกระทำกับตัวแปร years ผลจากการดำเนินการจะได้เป็น address หรือตำแหน่งในหน่วยความจำที่ใช้บันทึกค่าของตัวแปร years

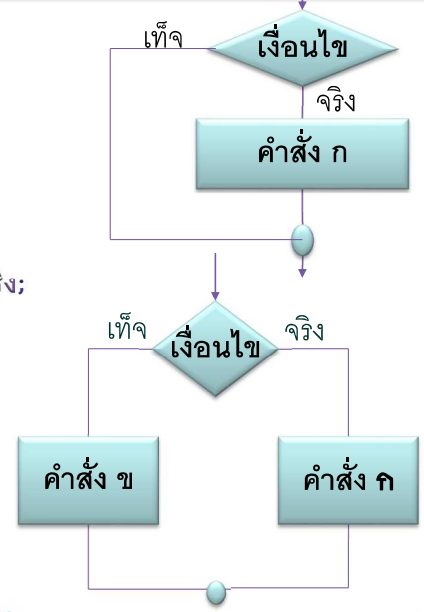

```
#include  < stdio.h >

void main ( ) {
    float          x;
    printf ("Please enter any number, ");
    scanf ("%f", &x);
    if (x < 0.0)
          printf("Your input %.2f is negative...", x);
    else
          printf("Your input %.2f is positive...", x);
}
```

Please enter any number, 0.85
Your input 0.85 is positive...

Please enter any number,-15.5
Your input -15.50 is negative...

Selection statement

- if (เงื่อนไข)
  คำสั่งที่ทำเมื่อเงื่อนไขเป็นจริง;
- if (เงื่อนไข) {
  คำสั่ง 1 ที่ทำเมื่อเงื่อนไขเป็นจริง;
  คำสั่ง 2 ที่ทำเมื่อเงื่อนไขเป็นจริง;
  ... คำสั่ง n ที่ทำเมื่อเงื่อนไขเป็นจริง;
  }
- if (เงื่อนไข)
  คำสั่งที่ทำเมื่อเงื่อนไขเป็นจริง;
  else
  คำสั่งที่ทำเมื่อเงื่อนไขไม่เป็นจริง;




```
#include  < stdio.h >
void main ( ) {
    int  i = 1, sum = 0;
    while (i<=10) {
          sum = sum + i;
          i = i + 1;
    }
    printf ("Summation of integer 1-10 is %d ", sum);
}
```

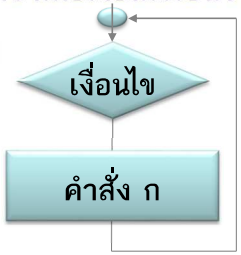
เงื่อนไขที่ต้องตรวจสอบ ในที่นี้หมายถึง ตรวจสอบว่าค่า ของ i น้อยกว่าหรือเท่ากับ 10 หรือไม่

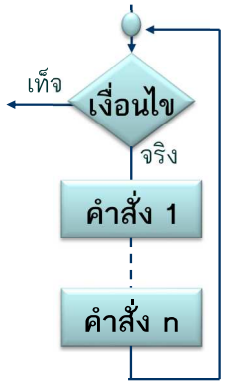
สองคำสั่งนี้จะถูกดำเนินการเมื่อ ค่าของ i น้อยกว่าหรือเท่ากับ10 เท่านั้น

Summation of integer 1-10 is 55

while statement

- while (เงื่อนไข)
  คำสั่งที่ทำซ้ำเมื่อเงื่อนไขเป็นจริง;





- while (เงื่อนไข) {
  คำสั่ง 1 ที่ทำเมื่อเงื่อนไขเป็นจริง;
  คำสั่ง 2 ที่ทำเมื่อเงื่อนไขเป็นจริง;
  ... คำสั่ง n ที่ทำเมื่อเงื่อนไขเป็นจริง;
  }


```
#include  < stdio.h >
int square (int ); /* Prototype of function */
void main ( ) {
   int   x, xx;

   printf("Enter any integer x = ");
   scanf("%d", &x);
   xx = square (x); /* เรียกใช้ฟังก์ชัน square() พร้อมส่งค่า x */
 /*เมื่อฟังก์ชันทำงานเสร็จจะให้ผลลัพธ์เป็น x2  แล้วกำหนดให้เป็นค่าของ xx*/
   printf ("Square of %d is %d\n ", x, xx);
}

int square (int  a) {
    return (a * a);
}
```

```
Enter any integer x = 5
Square of 5 is 25
```


General form of C source program

Preprocessing directives

Global Declarations

void main() {

Local declarations

Statements

}

มีได้เพียง 1 ฟังก์ชัน main() เท่านั้น

อาจมีได้หลายฟังก์ชัน แต่ชื่อต้องไม่ซ้ำกัน

User-defined function () {

Local declarations

Statements

}

พักยก ฝึกคิด

จงเขียนฟังก์ชันชื่อ reverse เพื่อทำการกลับหลักของเลขจำนวนหนึ่ง

เช่น เรียกฟังก์ชันเป็น

result = reverse(325);

เมื่อทำงานสำเร็จ result จะมีค่าเป็น 523

สิ่งที่ต้องนำเสนอ

1. ผลการวิเคราะห์ปัญหา
2. การออกแบบฟังก์ชัน
3. ผังงานของฟังก์ชัน reverse()
4. ชุดคำสั่งภาษาซีของฟังก์ชัน reverse()

325	0
32	5 = 0 + 5
3	52= 50 + 2
0	523= 520 + 3

Function

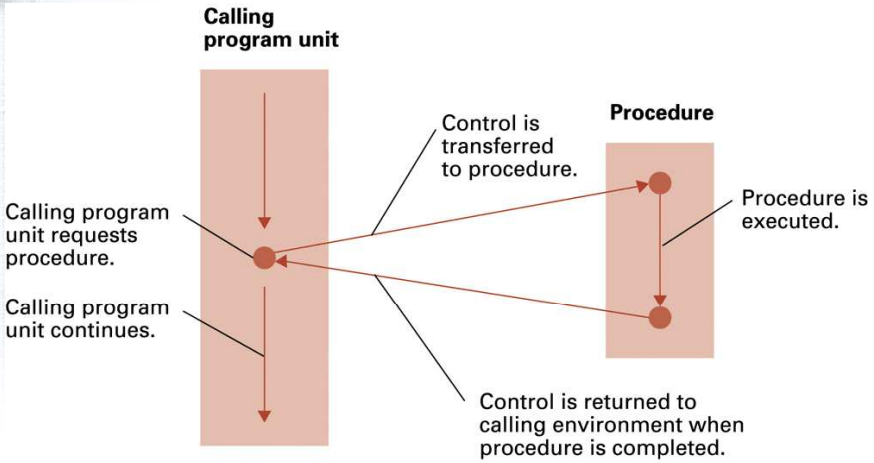
ฟังก์ชัน

ความหมาย

การประกาศ

และการใช้

The flow of control involving a procedure



The procedure ProjectPopulation written in the programming language C

Starting the head with the term "void" is the way that a C programmer specifies that the program unit is a procedure rather than a function. We will learn about functions shortly.

The formal parameter list. Note that C, as with many programming languages, requires that the data type of each parameter be specified.

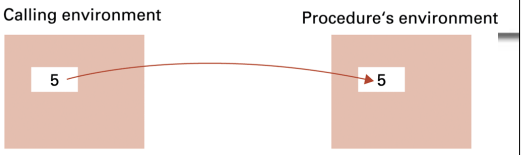
```
void ProjectPopulation (float GrowthRate)
{ int Year;
  Population[0] = 100.0;
  for (Year = 0; Year =< 10; Year++)
  Population[Year+1] = Population[Year] + (Population[Year] * GrowthRate);
}
```

This declares a local variable named Year.

These statements describe how the populations are to be computed and stored in the global array named Population.

Executing the procedure Demo and passing parameters by value

a. When the procedure is called, a copy of the data is given to the procedure

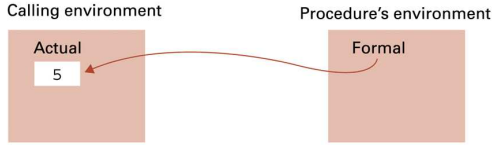


b. and the procedure manipulates its copy.

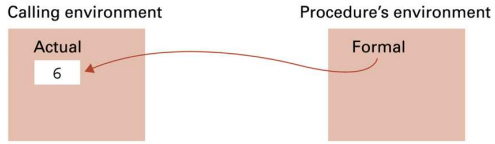
c. Thus, when the procedure has terminated, the calling environment has not been changed.

Executing the procedure Demo and passing parameters by reference

a. When the procedure is called, the formal parameter becomes a reference to the actual parameter.



b. Thus, changes directed by the procedure are made to the actual parameter



c. and are, therefore, preserved after the procedure has terminated.

The function CylinderVolume written in the programming language C

The function header begins with the type of the data that will be returned.

```
float CylinderVolume (float Radius, float Height)
{ float Volume;
  Volume = 3.14 * Radius * Radius * Height;
  return Volume;
}
```

Declare a local variable named Volume.

Compute the volume of the cylinder.

Terminate the function and return the value of the variable Volume.

ฟังก์ชัน

- การเขียนโปรแกรมแบบโครงสร้าง
- ชนิดของฟังก์ชัน
  - ฟังก์ชันมาตรฐาน (Library functions)
  - ฟังก์ชันที่พัฒนาเองโดยโปรแกรมเมอร์ (User defined functions)
- การใช้ฟังก์ชัน
  - การกำหนดฟังก์ชัน (Function definition)
  - การเรียกใช้ฟังก์ชัน (Function invocation)
  - โปรโตไทป์ของฟังก์ชัน(Function prototype)
- ฟังก์ชันมาตรฐานสำหรับการประมวลผลอักขระ

ฟังก์ชันมาตรฐาน (Library Functions)

- โปรโตไทป์ของฟังก์ชัน
  - ประกาศไว้ในแฟ้มข้อมูลที่มีนามสกุล เป็น .h

```
int getch ( void ) ;
```

  - เรียกการประกาศนี้โดยใช้

```
#include <ชื่อแฟ้ม.h>
#include <conio.h>
```

- การเรียกใช้ฟังก์ชัน : ขึ้นกับรูปแบบโปรโตไทป์ของฟังก์ชันนั้นๆ เช่น

```
ch = getch ( ) ;
```

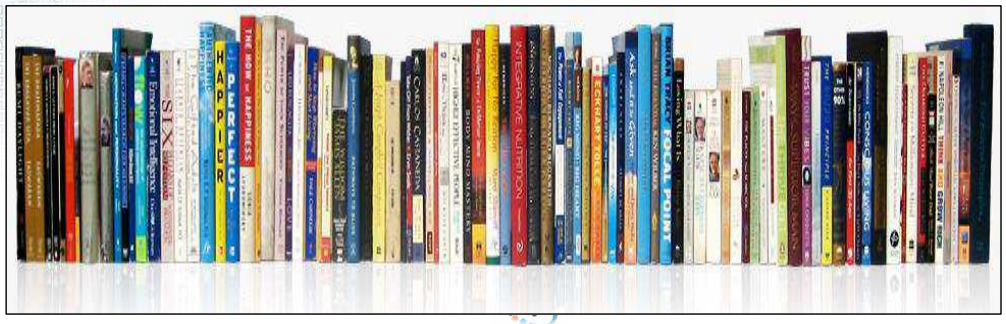
- การกำหนดฟังก์ชัน : มีพร้อมแล้วในไลบรารี

ฟังก์ชันมาตรฐานสำหรับการประมวลผลอักขระ

- **ฟังก์ชันสำหรับรับอักขระ**
  - **int getch(void);** /*read a character from the console, no echo*/
  - **int getche(void);** /*read a character from the console, echo */
  - **int getchar(void);** /*read a character from stdin*/
- **ฟังก์ชันสำหรับแสดงอักขระ**
  - **int putch(int);** /* write a character to the console */
  - **int putchar(int);** /* write a character to stdout */
- **ฟังก์ชัน/แมคโคร ใน ctype.h : ฟังก์ชันหรือแมคโครที่ช่วยอำนวยความสะดวกในการเขียนโปรแกรม เช่น ฟังก์ชันตรวจสอบว่าตัวอักขระนั้นเป็นตัวพิมพ์ใหญ่หรือไม่**

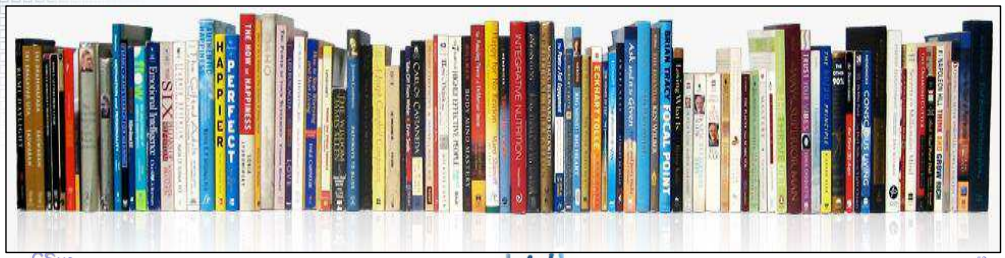
การใช้ฟังก์ชันมาตรฐานรับอักขระ getch()

- **เพื่อทำการอ่าน ตัวอักขระ จากแป้นพิมพ์ โดยไม่มีการสะท้อนให้เห็นตัวอักขระที่ป้อน**
- **Include file : <conio.h>**
- **Prototype : int getch (void);**
- **Argument : ไม่มี หรือไม่ต้องระบุ**
- **Return : ค่ารหัส ASCII ของอักขระที่อ่านได้**



การใช้ฟังก์ชันมาตรฐานรับอักขระ getche()

- **เพื่อทำการอ่าน ตัวอักขระ จากแป้นพิมพ์ โดยมีการสะท้อนให้เห็นตัวอักขระที่ป้อน**
- **Include file : <conio.h>**
- **Prototype : int getche (void);**
- **Argument : ไม่มี หรือไม่ต้องระบุ**
- **Return : ค่ารหัส ASCII ของอักขระที่อ่านได้**



การใช้ฟังก์ชันมาตรฐานแสดงอักขระ putch()

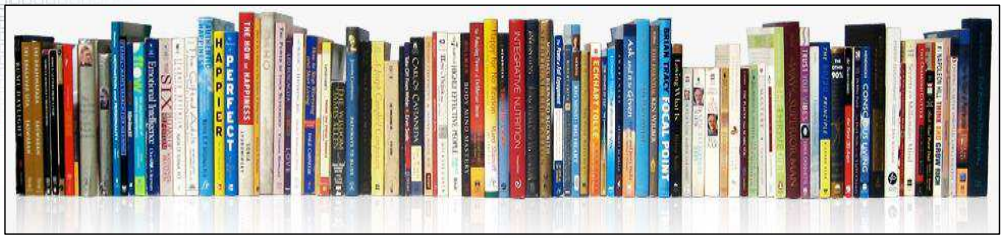
- เพื่อทำการแสดงตัวอักขระออกทางจอภาพ
- Include file : <stdio.h>
- Prototype : int putch (int);
- Argument : อักขระที่ต้องการแสดง
- Return : ในกรณีที่แสดงได้ จะให้ค่ารหัส ASCII ของอักขระที่แสดง
  ในกรณีที่แสดงไม่ได้ จะให้เป็นค่า EOF

```
putch(48); /* แสดง 0 */
putch('\a'); /* แสดงเสียงบี๊บ */
putch('A'+5); /* แสดงตัวอักษร F*/
```

```
int   c, i;
for (i=1; i<5; i++) {
    c = getch();
    putch('*');
}
```


การใช้ฟังก์ชันมาตรฐานรับอักขระ getchar()

- เพื่อทำการอ่านทีละตัวอักขระ จากบัฟเฟอร์ของแป้นพิมพ์ (Keyboard buffer) โดยมีการสะท้อนให้เห็นตัวอักขระที่ป้อน
- Include file : <stdio.h>
- Prototype : int getchar (void);
- Argument : ไม่มี หรือไม่ต้องระบุ
- Return : อักขระที่อ่านได้ ถ้าอ่านไม่ได้ จะให้ค่าเป็นEOF



การใช้ฟังก์ชันมาตรฐานแสดงอักขระ putchar()

- เพื่อทำการแสดงตัวอักขระทางอุปกรณ์แสดงผลมาตรฐาน
- Include file : <conio.h>
- Prototype : int putchar (int);
- Argument : รหัส ASCII ของอักขระที่ต้องการแสดง
- Return : อักขระที่แสดงหรือปรากฏ
  ในกรณีที่แสดงไม่ได้ จะให้เป็นค่า EOF

```
putchar(48); /* แสดง 0 */
putchar('\a'); /* แสดงเสียงบี๊บ */
putchar('A'+5); /* แสดงตัวอักษร F*/
putchar('\n'); /*ขึ้นบรรทัดใหม่ cursor อยู่ต้นบรรทัด*/
```


ตัวอย่างการใช้ฟังก์ชันสำหรับการประมวลผลอักขระ

จงเขียนโปรแกรมเพื่อรับอักขระ แล้วแสดงชุดของอักขระที่รับ โดยแสดงเป็นตัวพิมพ์ใหญ่ทั้งหมด พร้อมทั้งแทรกบรรทัดว่างหากเจอการขึ้นบรรทัดใหม่ หยุดการป้อนข้อมูลด้วยการกด Control-D

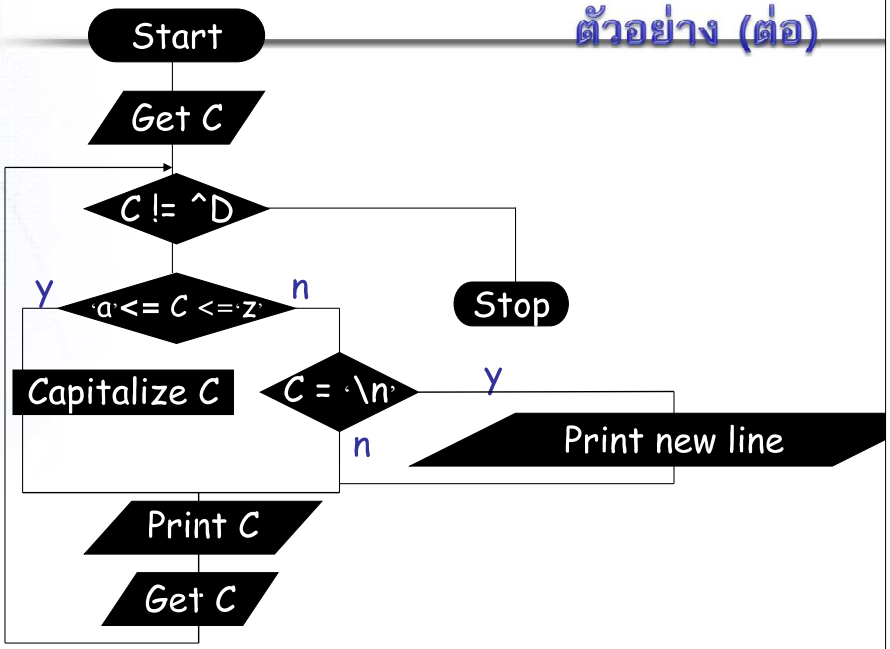
เช่น รับข้อมูลเป็น I'll be a champion! <**กด** enter>

ผลลัพธ์ได้เป็น I'LL BE A CHAMPION!

<**เว้น 1 บรรทัด**>

<cursor **รอรับการป้อนข้อมูล**>

ตัวอย่าง (ต่อ)



ตัวอย่างการใช้ฟังก์ชันสำหรับการประมวลผลอักขระ

```
#include <stdio.h>
#define  ControlD        4
#define  NEWLINE     '\n'
void main ( ) {
  int c;
  /* get characters, stop at end of file */
  while ((c = getchar()) != ControlD) {
     if ('a' <= c && c <= 'z')
          c = c + 'A' - 'a'; /*Capitalize the character c */
     else
          if (c == NEWLINE)
            putchar(NEWLINE); /*write the newline character*/
     putchar(c); /* output the character*/
   } /* End of while loop */
}
```


The macro in ctype.h

Macro	Nonzero is returned if :
isalpha(c)	*c is a letter*
isupper(c)	*c is an uppercase letter*
islower(c)	*c is a lowercase letter*
isdigit(c)	*c is a digit*
isalnum(c)	*c is a letter or digit*
isxdigit(c)	*c is a hexadecimal digit*
isspace(c)	*c is a white space character*
ispunct(c)	*c is a punctuation character*
iscntrl(c)	*c is a control character*
isascii(c)	*c is an ASCII code*
Function / Macro effect	
toupper(c)	*change c to uppercase*
tolower(c)	*change c to lowercase*
toascii(c)	*change c to ASCII code*

ตัวอย่างการใช้ฟังก์ชันสำหรับการประมวลผลอักขระ

```
#include <stdio.h>
#include  <ctype.h>
#define  ControlD        4
#define  NEWLINE         '\n'
void main ( ) {
  int c;
  while ((c = getchar()) != ControlD) {
     if (islower(c))
       c = toupper(c); /*Capitalize the character c */
     else
        if (c == NEWLINE)
          putchar(NEWLINE); /*write the newline character*/
     putchar(c); /* output the character*/
   }
}
```

ขอบเขตของตัวแปร (Scope of Variables)

- Global Variable
  - โดยตำแหน่งของการประกาศ
    - ประกาศไว้นอกกรอบของฟังก์ชันใด
    - ถูกอ้างถึงหรือเรียกใช้ได้จากคำสั่งที่อยู่ถัดจากบรรทัดคำสั่งของการประกาศทันที
  - โดย คำเฉพาะ extern *<type> <global variable>*
- Local Variables (ตัวแปรเฉพาะถิ่น)
  - จะถูกเรียกใช้ได้เฉพาะภายในกรอบที่ประกาศตัวแปรนั้นๆ เท่านั้น
  - สามารถใช้ {} กำหนดกรอบหรือขอบเขตของตัวแปรนั้นได้

ตัวอย่าง

```
int  a=10, b=8;
int mo_a () {
  int x = 5;
  x += a;
  printf("x = %d\n", x);
  a++;
  return (a);
}
```

```
void main () {
   int b = 500;
   printf ("global a is %d\t", a);
   printf("b+a=%d\n", b+a);
   printf("a in mo_a is %d\n", mo_a());
   { int a = -1;
     printf("a in block is %d\n", a); }
   printf("a out of block is %d\n", a);
   printf("a in mo_a is %d\n", mo_a());
}
```


ฟังก์ชันที่พัฒนาโดยโปรแกรมเมอร์

- **การกำหนดฟังก์ชัน** (Function definition)
- **การเรียกใช้ฟังก์ชัน** (Function invocation)
- **การประกาศโครงร่างหรือรูปแบบการเรียกใช้ฟังก์ชัน**(Function prototype)
- **ลำดับการวาง มี 2 ลักษณะ**

1. Function definition	1. Function prototype
2. Function invocation	2. Function invocation
	3. Function definition

ตัวอย่าง

```
#include <stdio.h>
void prn_msg(void) {
  printf("Message for you only :\n");
  printf("Have a nice day!!! \n");
}
void main() {
   prn_msg(); /* function is invoked here*/
}
```

การกำหนดฟังก์ชัน

Function header

ชนิดข้อมูลที่ส่งกลับ *ชื่อฟังก์ชัน* (การประกาศ พารามิเตอร์)

{

การประกาศตัวแปรภายในฟังก์ชัน;

คำสั่ง ;

return *ค่าที่ต้องการส่งกลับ* ;

}

Function body

```
int  min (int  x,  int  y) {
   return (x<y? x : y);
}
```


return statement

- คำสั่งหยุดการทำงานของฟังก์ชันที่ถูกเรียกและย้ายการทำงานกลับไปยังฟังก์ชันที่เรียก
- โดยในการย้ายการทำงานกลับนั้นอาจมีการส่งค่ากลับหรือกำหนดค่าให้กับฟังก์ชันด้วย
  - กรณีไม่มีค่าส่งกลับ return;
  - กรณีมีค่าส่งกลับ

return *นิพจน์*;

โดยที่ชนิดข้อมูลของนิพจน์และของฟังก์ชัน ต้องเป็นชนิดเดียวกัน

ตัวอย่าง

```
void ex_sub(int x) {
  if (x < 0) return;
  printf("Have a nice day!!! \n");
}
int max(int x, int y) {
   return (x > y ? x : y);
}
```


การเรียกใช้ฟังก์ชัน

- รูปแบบ จะต้องสอดคล้องกับที่ประกาศ
  - ในส่วนหัวของฟังก์ชัน หรือ
  - ตามรูปแบบโปรโตไทป์ของฟังก์ชัน
- หากฟังก์ชัน มีชนิดข้อมูลเป็น void จะเรียกใช้ฟังก์ชัน ดังนี้

Function name **()**; หรือ

Function name **(***argument list***)**;

- หากฟังก์ชัน มีชนิดข้อมูลเป็นอย่างอื่น(ที่ไม่ใช่ void) จะเรียกใช้ฟังก์ชัน โดยมีตัวแปรรอรับค่า หรือไม่ก็ได้
- โดยการส่งค่าของอาร์กิวเมนต์ให้กับพารามิเตอร์เป็นลักษณะของการคัดลอกค่า (Call by value)

ตัวอย่างการเรียกใช้ (๑)

```
void ex_sub(int x){
  if (x < 0) return;
  printf("Have a nice day!!! \n");
}
```

x เป็นพารามิเตอร์

อาร์กิวเมนต์

ex_sub(8); /*Have a nice day!!! will be displayed*/

ex_sub(-1); /* Nothing's displayed */

ตัวอย่างการเรียกใช้ (๒)

```
int max (int x, int y) {
   return (x > y ? x : y);
}
```

```
int    maximum;
printf("Maximum of 8 and 80 is %d\n", max(8, 80));
/*80 will be displayed*/
maximum = max(8, 80);
printf("Maximum of 8 and 80 is %d\n", maximum);
```


แบบฝึกหัด

จงหาแสดงผลลัพธ์ เมื่อทำการรันโปรแกรมต่อไปนี้

```
#include  <stdio.h>
void A (int x) {
   printf("%d : ", x);
}
void B (int x) {
   A(++x);
   printf("%d : ", x);
}
void main () {
   B(8);
   A(2);
   B(10);
}
```


ฟังก์ชันที่พัฒนาโดยโปรแกรมเมอร์ (2)

- โปรโตไทป์ของฟังก์ชัน
  - จำเป็นต้องมีในกรณีที่การกำหนดฟังก์ชันถูกประกาศไว้หลังการเรียกใช้
  - ประกาศไว้ก่อนการเรียกใช้ และก่อนการกำหนดฟังก์ชัน
  - มีรูปแบบเหมือนส่วนหัวของฟังก์ชันนั่นเอง แต่ไม่มีชื่อพารามิเตอร์
  - รูปแบบ

ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน (*ชนิดข้อมูลของพารามิเตอร์*);

int max (int, int) ;

ตัวอย่าง

```
#include <stdio.h>
int max (int, int);   /* Function prototype */
void main ( ) {
   int    maximum;
   printf("Maximum of 8 and 80 is %d\n", max(8, 80));
   /*80 will be displayed*/
   maximum = max(8, 80);
   printf("Maximum of 8 and 80 is %d\n", maximum);
}
int max(int x, int y) {
   return (x > y ? x : y);
}
```


ฟังก์ชันที่พัฒนาโดยโปรแกรมเมอร์(3)

- ประเภทของฟังก์ชัน ~ จำแนกตามลักษณะการส่งผ่านข้อมูล
  - ฟังก์ชันที่ไม่มีการส่งผ่านข้อมูล
  - ฟังก์ชันที่ส่งค่าทางเดียว
  - ฟังก์ชันที่มีการส่งค่ากลับไปที่ชื่อฟังก์ชัน
  - ฟังก์ชันที่มีการผ่านค่าไปกลับ

  ฟังก์ชันมาตรฐาน อาจถูกจำแนกได้ในลักษณะเดียวกันนี้

- การส่งผ่านค่า มี 2 แบบ
  - Call by value
  - Call by reference

ฟังก์ชันที่ไม่มีการส่งผ่านข้อมูล

- รูปแบบการกำหนดฟังก์ชัน

```
void ชื่อฟังก์ชัน ( ) {
      การประกาศตัวแปรภายในฟังก์ชัน
      คำสั่งหรือชุดคำสั่ง
}
```

อาจเป็นคำสั่งเพื่อเรียกใช้ฟังก์ชันอื่นได้ ยกเว้นฟังก์ชัน main()

- รูปแบบของโปรโตไทป์

```
void ชื่อฟังก์ชัน ( ) ;
```


ตัวอย่าง

```
#include <stdio.h>

void makeline
   int i=0;
   for (; i<= 80; i++)
      putchar('=');
   putchar('\n');
}

void main ( ) {
   makeline( );
   printf("Heading");
   makeline( );
}
```

```
#include <stdio.h>
void makeline ( );

void main ( ) {
     makeline( );
     printf(" Heading" );
     makeline( );
}

void makeline (  ) {
     int i=0;
     for (; i<= 80; i++)
        putchar('=');
     putchar('\n');
}
```

ฟังก์ชันที่ส่งค่าทางเดียว

- การกำหนดฟังก์ชัน

```
void ชื่อฟังก์ชัน (การประกาศพารามิเตอร์) {
        การประกาศตัวแปรภายในฟังก์ชัน
        คำสั่งหรือชุดคำสั่ง
}
```

อาจเป็นคำสั่งเพื่อเรียกใช้ฟังก์ชันอื่นได้ ยกเว้นฟังก์ชัน main()

- การประกาศโปรโตไทป์

```
void ชื่อฟังก์ชัน ( ชนิดข้อมูลของพารามิเตอร์);
```

ชนิดข้อมูลและจำนวน ของอาร์กิวเมนต์ เป็นอย่างเดียวกับพารามิเตอร์

- การเรียกใช้

```
ชื่อฟังก์ชัน (อาร์กิวเมนต์);
```


ตัวอย่าง - ฟังก์ชันที่ส่งค่าทางเดียว

```
/*Demonstrate passing by value*/
#include <stdio.h>
#define Heading  "Report Heading"
void makeline (int);
void main ( ) {
  makeline(10);
  printf(Heading); putchar('\n');
  makeline(strlen(Heading));
}
void makeline (int a) {
    for ( ; a > 0; a--)
       putchar('=');
    putchar('\n');
}
```

```
==========
Report Heading
==============
_
```


ฟังก์ชันที่มีการส่งค่ากลับไปที่ชื่อฟังก์ชัน

- การกำหนดฟังก์ชัน

```
ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน() {
        การประกาศตัวแปรภายในฟังก์ชัน
            คำสั่งหรือชุดคำสั่ง
        return ค่าที่มีชนิดข้อมูลเดียวกับฟังก์ชัน ;
}
```

- โปรโตไทป์ของฟังก์ชัน

```
ชนิดข้อมูล  ชื่อฟังก์ชัน ();
```

- การเรียกใช้ฟังก์ชัน

```
ตัวแปรที่มารับค่า = ชื่อฟังก์ชัน ( );
```


ตัวอย่าง

```
# include <stdio.h>
int getdigit( ) {
    char  c;
    c = getche( );
    return ((' 0 ' <= c && c <= ' 9 ') ? c - ' 0 ' : -1);
}
int getnum ( ) {
     int          i, num=0;
     while ((i=getdigit()) != -1)
          num = num*10 + i;
     return num;
}

void main ( ) {
  printf("Input number is %d\007\n", getnum());
}
```

getnum.h

ตัวอย่าง(ต่อ)

```
# include <stdio.h>

#include  "getnum.h"

void main ( ) {
  printf("Input number is %d\007\n", getnum());
}
```


ฟังก์ชันที่มีการผ่านค่าไปกลับ

- การกำหนดฟังก์ชัน

ชนิดข้อมูลที่ส่งกลับ *ชื่อฟังก์ชัน* (*การประกาศพารามิเตอร์*) {

การประกาศตัวแปรภายในฟังก์ชัน

คำสั่งหรือชุดคำสั่ง

return *ค่าที่มีชนิดข้อมูลเดียวกับฟังก์ชัน*;

}

- *ชนิดข้อมูล* ชื่อฟังก์ชัน (*ชนิดข้อมูลของพารามิเตอร์*);

- ตัวแปรที่มารับค่า = ชื่อฟังก์ชัน (*อาร์กิวเมนต์*) ;

ตัวอย่าง - ฟังก์ชันคำนวณค่าแฟคตอเรียล

```
#include <stdio.h>
#include <getnum.h>
long factor (int );
void main ( ) {
  int n;
  printf("Enter a positive number, finish by pressing Enter");
  n = getnum( );
  printf("Factorial of %d = %ld\n", n, factor(n));
}

long factor ( int n) {
   long fl=1;
   for (; n > 0; n--) fl *= n;
   return fl;
}
```


แบบฝึกหัด

- **จงเขียนโปรแกรมเพื่อพิมพ์ปฏิทินของแต่ละเดือน ตามต้องการ โดยไม่ใช้ตัวแปรชุด กำหนดให้ วันที่ 1 มกราคม พ.ศ. 2545 ตรงกับวันอังคาร**
- **จงเขียนโปรแกรมทำการ Zapping ข้อมูลที่ป้อน 1 บรรทัด โดยให้ตัด ช่องว่างออก และเปลี่ยนตัวอักษรแรกของคำที่ป้อนให้เป็นพิมพ์ใหญ่**

เช่น ป้อนข้อมูลเป็น The final day is Sunday.

ผลลัพธ์ที่ได้เป็น TheFinalDayIsSunday.