ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

Computer Programming II
การเขียนโปรแกรมคอมพิวเตอร์2
LECTURE#2 Introduction to C Language

อ.สถิตย์ ประสมพันธ์

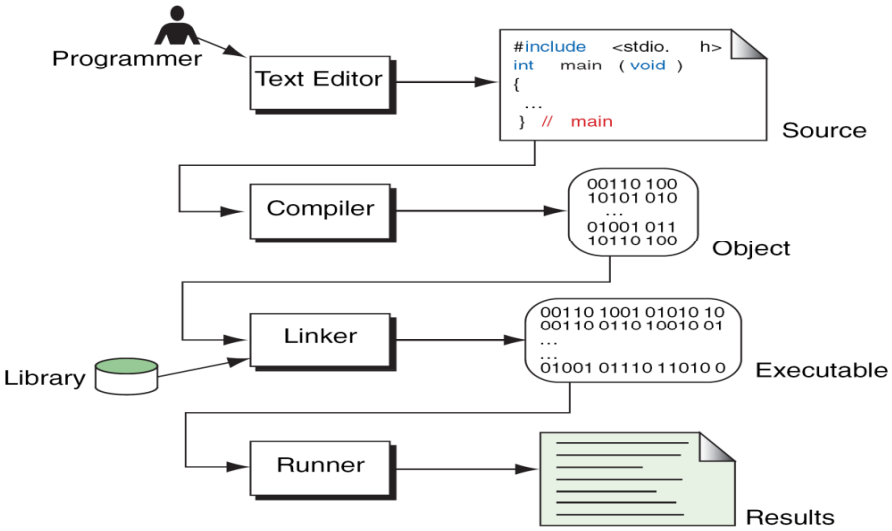
ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

KMUTNB

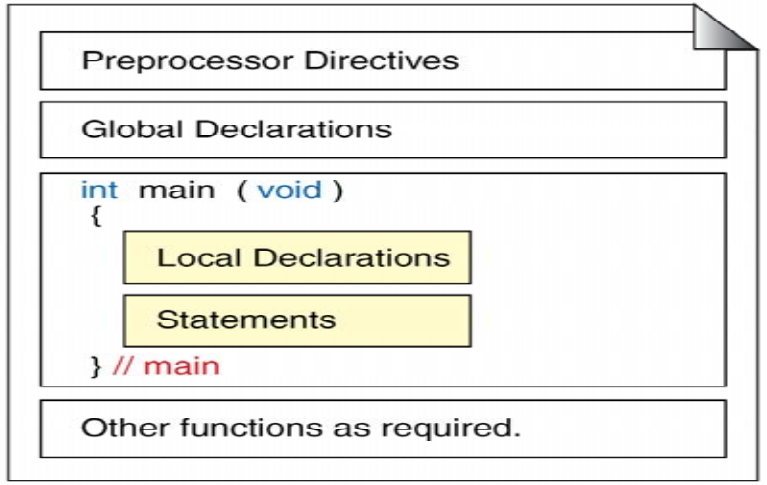
ประวัติความเป็นมาของภาษาซี

- ค.ศ. 1970 มีการพัฒนาภาษา B โดย Ken Thompson ซึ่งทำงานบนเครื่อง DEC PDP-7 ซึ่งทำงานบนเครื่องไมโครคอมพิวเตอร์ไม่ได้ และยังมีข้อจำกัดในการใช้งานอยู่ (ภาษา B สืบทอดมาจากภาษา BCPL ซึ่งเขียนโดย Marth Richards)
- ค.ศ. 1972 Dennis M. Ritchie และ Ken Thompson ได้สร้างภาษา C เพื่อเพิ่มประสิทธิภาพภาษา B ให้ดียิ่งขึ้น ในระยะแรกภาษา C ไม่เป็นที่นิยมแก่นักโปรแกรมเมอร์โดยทั่วไปนัก
- ค.ศ. 1978 Brian W. Kernighan และ Dennis M. Ritchie ได้เขียนหนังสือเล่มหนึ่งชื่อว่า The C Programming Language และหนังสือเล่มนี้ทำ ให้บุคคลทั่วไปรู้จักและนิยมใช้ภาษา C ในการเขียนโปรแกรมมากขึ้น แต่เดิมภาษา C ใช้ Run บนเครื่องคอมพิวเตอร์ 8 bit ภายใต้ระบบปฏิบัติการ CP/M ของ IBM PC ซึ่งในช่วงปี ค. ศ. 1981 เป็นช่วงของการพัฒนาเครื่องไมโครคอมพิวเตอร์ ภาษา C จึงมี บทบาทสำคัญในการนำ มาใช้บนเครื่อง PC ตั้งแต่นั้นเป็นต้นมา และมีการพัฒนาต่อมาอีกหลาย ๆ ค่าย ดังนั้นเพื่อกำหนดทิศทางการใช้ภาษา C ให้เป็นไปแนวทางเดียวกัน ANSI (American National Standard Institute) ได้กำหนดข้อตกลงที่เรียกว่า 3J11 เพื่อสร้างภาษา C มาตรฐานขึ้นมา เรียนว่า ANSI C
- ค.ศ. 1983 Bjarne Stroustrup แห่งห้องปฏิบัติการเบล (Bell Laboratories) ได้พัฒนาภาษา C++ ขึ้นรายละเอียดและความสามารถของ C++ มีส่วนขยายเพิ่มจาก C ที่สำคัญ ๆ ได้แก่ แนวความคิดของการเขียนโปรแกรมแบบ OOP (Object Oriented Programming) ซึ่งเป็นแนวคิดการเขียนโปรแกรมที่เหมาะกับการพัฒนาโปรแกรมขนาดใหญ่ที่มีความสลับซับซ้อนมาก มีข้อมูลที่ใช้ในโปรแกรมจำนวนมาก จึงนิยมใช้เทคนิคของการเขียนโปรแกรมแบบ OOP ในการพัฒนาโปรแกรมขนาดใหญ่ในปัจจุบันนี้

การคอมไพล์และลิงค์โปรแกรมในภาษาซี



โครงสร้างโปรแกรมภาษาซี



โครงสร้างและลำดับการเขียนภาษาซี

- คำสั่งตัวประมวลผลก่อน(Preprocessor statement)
- รหัสต้นฉบับ (Source code) มีลำดับการเขียนดังนี้
  - คำสั่งประกาศครอบคลุม (Global declaration statements)
  - ต้นแบบฟังก์ชัน(function prototypes)
  - ฟังก์ชันหลัก(main function) มีฟังก์ชันเดียว
  - ฟังก์ชัน(functions) มีได้หลายฟังก์ชัน
  - คำสั่งประกาศตัวแปรเฉพาะที่(Local declaration statements)
- หมายเหตุ(Comment) สามารถแทรกไว้ที่ใดก็ได้ภายในโปรแกรม

รหัสต้นฉบับ (Source code)

```
/* This program demonstrates function calls by
calling a small function to multiply two numbers.
      Written by:
      Date:
*/
#include <stdio.h>
int multiply (int num1, int num2);
int main (void)
{
   int multiplier;
   int multiplicand;
   int product;
   printf("Enter two integers: ");
   scanf ("%d%d", &multiplier, &multiplicand);
   product = multiply (multiplier, multiplicand);
   printf("Product of %d & %d is %d\n",
            multiplier, multiplicand, product);
   return 0;
}  // main
/* ================== multiply ================
   Multiples two numbers and returns product.
      Pre   num1 & num2 are values to be multiplied
      Post  product returned
*/
int multiply (int num1, int num2)
{
// Statements
   return (num1 * num2);
}  // multiply
```

// Function Declarations

// main function

// Local Declarations

// function

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Enter two integers: 17 21
Product of 17 & 21 is 357
```


โครงสร้างและลำดับการเขียนภาษาซี

```
/* The greeting program. This program demonstrates
   some of the components of a simple C program.
      Written by:  your name here
      Date:        date program written
*/
#include <stdio.h>

int main (void)
{
// Local Declarations

// Statements

   printf("Hello World!\n");

   return 0;
} // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Hello World!
```


โครงสร้างและลำดับการเขียนภาษาซี

```
/* This program reads two integers from the keyboard
   and prints their product.
      Written by:
      Date:
*/
#include <stdio.h>

int main (void)
{
// Local Definitions
   int number1;
   int number2;
   int result;

// Statements
   scanf  ("%d", &number1);
   scanf  ("%d", &number2);
   result = number1 * number2;
   printf ("%d", result);
   return 0;
}  // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
5 25
125
```

คำสั่งตัวประมวลผลก่อน (Preprocessor statement)

- **ตัวประมวลผลก่อน (Preprocessor)** คือ ส่วนที่คอมไพเลอร์จะต้องทำก่อนทำการแปลโปรแกรม คำสั่งของตัวประมวลผลก่อนจะนำหน้าด้วยเครื่องหมาย # มีคำสั่งต่างๆ ต่อไปนี้

#include	#define	#if	#program
#endif	#error	#ifndef	#undef
#elif	#else	#ifdef	

คำสั่งตัวประมวลผลก่อน (Preprocessor statement)

- **#include** ทำหน้าที่แจ้งให้คอมไพเลอร์อ่านไฟล์อื่นเข้ามาแปลร่วมด้วย มีรูปแบบดังนี้

#include <filename> หรือ #include "filename"

เช่น

#include <dos.h>	อ่านไฟล์ dos.h จากไดเร็คทรอรีที่กำหนด
#include "sample.h"	อ่านไฟล์ sample.h จากไดเร็คทรอรีปัจจุบันหรือที่กำหนด
#include "stdio.h"	อ่านไฟล์ stdio.h จากไดเร็คทรอรีปัจจุบันหรือที่กำหนด

คำสั่งตัวประมวลผลก่อน (Preprocessor statement)

- **#define** ทำหน้าที่ใช้กำหนดค่าคงที่ ที่เป็นชื่อแทน คำ นิพจน์ คำสั่ง หรือคำสั่งหลายคำสั่ง มีรูปแบบการใช้งานดังนี้

#define ชื่อตัวแปร (ชื่อที่ใช้แทน) ค่าที่ต้องการกำหนด

เช่น

#define	TWO	2	กำหนดตัวแปร TWO แทนค่า 2
#define	PI	3.141592654	กำหนดตัวแปร PI แทนค่า 3.141592654

หมายเหตุ (Comment)

- สามารถแทรกไว้ที่ใดก็ได้ภายในโปรแกรม ภาษาซีนิยมการเขียนข้อความอธิบายการทำงานในส่วนต่างๆ ของโปรแกรมเพื่อให้เข้าใจและอ่านโปรแกรมง่ายขึ้น การเขียนอธิบายจะใช้เครื่องหมาย /* และ */ คร่อมข้อความที่ต้องการอธิบาย

/* .....ข้อความที่ต้องการอธิบาย.....*/

- แต่ถ้าต้องการเขียนอธิบายหลายๆบรรทัดจะเขียนได้ดังนี้

/* ....
..... ข้อความที่ต้องการอธิบาย.....
.....*/

```
/* This program demonstrates three ways to use
constants.
      Written by:
      Date:
*/
#include <stdio.h>
#define PI 3.1415926536

int main (void)
{
// Local Declarations
   const double cPi = PI;

// Statements
   printf("Defined constant PI: %f\n", PI);
   printf("Memory constant cPi: %f\n", PI);
   printf("Literal constant:    %f\n", 3.1415926536);
   return 0;
}  // main
```

รูปแบบคำสั่งในภาษาซี

- รูปแบบคำสั่งในภาษาซี มีกฎเกณฑ์ในการเขียนคำสั่ง ดังนี้
  - คำสั่งทุกคำสั่งต้องเขียนด้วยอักษรตัวเล็กเสมอ เช่นคำสั่ง printf , scanf , for
  - ทุกคำสั่งจะใช้เครื่องหมาย ; แสดงการจบของคำสั่ง เช่น printf(“Hello”) ;
  - การเขียนคำสั่ง จะเขียนได้แบบอิสระ (Free Format) คือ สามารถเขียนหลายๆคำสั่งต่อกันได้ เช่น

    printf(“Hello”); printf(“C Programming”); f = 3.414;

    แต่เพื่อความเป็นระเบียบและอ่านง่าย ควรจะเขียน 1 คำสั่งต่อ 1 บรรทัด

ตัวแปร (Variable)

- ชื่อเรียกแทนพื้นที่เก็บข้อมูลในหน่วยความจำ มีชนิดของข้อมูล หรือแบบของตัวแปรคือ char, int, long, float, double, unsigned int, unsigned long int
- การกำหนดตัวแปร ทำได้ 2 แบบ คือ
  - 1. กำหนดไว้นอกกลุ่มคำสั่ง หรือฟังก์ชัน เรียกตัวแปรนี้ว่า **Global Variable** กำหนดไว้นอกฟังก์ชัน ใช้งานได้ทั้งโปรแกรม มีค่าเริ่มต้นเป็น 0 (กรณีไม่ได้กำหนดค่าเริ่มต้น)
  - 2. กำหนดไว้ในกลุ่มคำสั่ง หรือฟังก์ชัน เรียกตัวแปรนี้ว่า **Local Variable** กำหนดไว้ภายในฟังก์ชัน ใช้งานได้ภายในฟังก์ชันนั้น และไม่ถูกกำหนดค่าเริ่มต้นโดยอัตโนมัติ

ชนิดตัวแปร ชื่อตัวแปร , ชื่อตัวแปร, ชื่อตัวแปร,.....;

กฎการตั้งชื่อตัวแปร การตั้งชื่อตัวแปร

- มีข้อกำหนดดังนี้
  - ประกอบด้วย a ถึง z, 0 ถึง 9 และ _ เท่านั้น
  - อักขระตัวแรกต้องเป็น a ถึง z และ _
  - ห้ามใช้ชื่อเฉพาะ
  - ตัวพิมพ์ใหญ่ ตัวพิมพ์เล็ก มีความหมายที่แตกต่างกัน (Case sensitive)
  - ยาวสูงสุดไม่เกิน 31 ตัวอักษร

ตัวอย่างของการตั้งชื่อตัวแปรที่ถูกต้องและไม่ถูกต้อง

Valid Names		Invalid Name	
a	// Valid but poor style	$sum	// $ is illegal
student_name		2names	// First char digit
_aSystemName		sum-salary	// Contains hyphen
_Bool	// Boolean System id	stdnt Nmbr	// Contains spaces
INT_MIN	// System Defined Value	int	// Keyword

กลุ่มคำในภาษาซี

- คำสงวน (Keywords)
  - คำที่ภาษาซีกำหนดไว้ก่อนแล้ว เพื่อใช้งาน ได้แก่

auto	**default**	**float**	**register**	**struct**	**volatile**	**break**	**void**
do	for	extern	const	long	return	static	enum
goto	short	char	int	sizeof	case	while	continue
union	unsigned	typedef	if	else			

กลุ่มคำในภาษาซี

- **คำที่ผู้ใช้ตั้งขึ้นใหม่** (User Defines words)
- กลุ่มอักษรที่นิยามขึ้นใช้ในโปรแกรม โดยผู้เขียนโปรแกรมกำหนดขึ้นเอง มีข้อกำหนดดังนี้
  - ตัวอักษรภาษาอังกฤษตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ภาษาซีถือว่าเป็นคนละตัวกัน เช่น Area และ area เป็นตัวแปรคนละตัวกัน
  - ตัวอักษรตัวแรกต้องเป็นตัวอักษรหรือ _ จะเป็นตัวเลขไม่ได้
  - ตัวอักษรที่ไม่ใช่ตัวแรกจะเป็นตัวอักษรหรือ _ หรือตัวเลขก็ได้
  - ก่อนการใช้ชื่อใดๆ ต้องนิยามก่อนเสมอ
  - ห้ามตั้งชื่อซ้ำกับคำสงวน
  - ภายในกลุ่มคำสั่ง สามารถกำหนดชื่อขึ้นใหม่ได้ ชื่อนั้นจะถูกใช้งานภายในกลุ่มคำสั่ง และกลุ่มคำสั่งที่ย่อยลงไปเท่านั้น หากชื่อในกลุ่มคำสั่งไปซ้ำกับที่นิยามไว้ภายนอก จะถือเอาชื่อที่นิยามใหม่เป็นหลัก
  - ความยาวชื่อจะขึ้นอยู่กับตัวแปรในภาษาซี สำหรับโปรแกรม Borland C ได้ 32 ตัวอักษร

ชนิดของข้อมูลในภาษาซี

ชนิด	ขนาดความกว้าง	ช่วงของค่า
char	8 บิต	ASCII character (-128 ถึง 127)
unsigned char	8 บิต	0-255
int	16 บิต	-32768 ถึง 32767
long int	32 บิต	-2147483648 ถึง 2147483649
float	32 บิต	3.4E-38 ถึง 3.4E+38 หรือ ทศนิยม 6 ตำแหน่ง
double	64 บิต	1.7E-308 ถึง 1.7E+308 หรือ ทศนิยม 12 ตำแหน่ง
unsigned int	16 บิต	0 ถึง 65535
unsigned long int	32 บิต	0 ถึง 4294967296

การหาขนาดของชนิดตัวแปรต่าง ๆ จะใช้คำสั่ง sizeof(ประเภทข้อมูล) โดยลำดับขนาดของประเภทข้อมูลเรียงลำดับจากน้อยไปหามากมีดังนี้

sizeof (short) ≤ sizeof (int) ≤ sizeof (long) ≤ sizeof (long long)

ค่าคงที่ constant

ตัวอย่างค่าคงที่	ชนิด	ความหมาย
255	decimal int	จำนวนเต็มฐานสิบ
0xFF	hexadecimal int	จำนวนเต็มฐานสิบหก
0377	octal int	จำนวนเต็มฐานแปด
255L หรือ 255l	long int	จำนวนเต็มฐานสิบแบบยาว
255U หรือ 255u	unsigned int	จำนวนเต็มฐานสิบไม่คิดเครื่องหมาย
0xFFUL	unsigned long int	เลขฐานสิบหกแบบยาวไม่คิดเครื่องหมาย
15.75E2	floating point	เลขทศนิยมแบบยกกำลัง
-1.23	floating point	เลขทศนิยมแบบค่าติดลบ
.123	floating point	เลขทศนิยม
123F	floating point	เลขทศนิยม
'a'	character	ตัวอักษร
""	string	ประโยค, ข้อความ

คือ ค่าของข้อมูลที่มีจำนวนแน่นอน เขียนได้ 3 ลักษณะ คือ

1 ค่าคงที่จำนวนเต็ม เขียนอยู่ในรูปตัวอักษร อาจมีเครื่องหมายลบนำหน้า

2 ค่าคงที่จำนวนจริง เขียนในรูปตัวเลขมีทศนิยม

3 ค่าคงทีที่หมายถึงรหัสตัวอักษร (ตัวอักษรถูกจำในรูปแบบตัวเลข ตามรหัส ASCII)

การแสดงผลข้อมูล

- ฟังก์ชัน printf() เป็นฟังก์ชันจากคลังที่มาพร้อมกับตัวแปลโปรแกรมภาษาซี ใช้สำหรับการแสดงผล มีรูปแบบดังนี้

```
printf("สายอักขระควบคุม", ตัวแปร);
```

- 1. สายอักขระควบคุมประกอบด้วย 3 ส่วนคือ
  - -ตัวอักขระที่จะแสดง
  - -รูปแบบการแสดงผล ขึ้นต้นด้วยเครื่องหมายเปอร์เซ็นต์(%)
  - -ลำดับหลีก (escape sequence)
- 2. ตัวแปร คือชื่อของตัวแปรที่จะแสดงผล (format specifies)

การกำหนดรูปแบบการแสดงผล

ขึ้นต้นด้วยเครื่องหมายเปอร์เซ็นต์(%) ตามด้วยอักขระ 1 ตัว หรือหลายตัวโดยที่อักขระมีความหมายดังนี้

อักขระ	ชนิดข้อมูล	รูปแบบการแสดงผล
c	char	อักขระเดียว
d	int	จำนวนเต็มฐานสิบ
o	int	จำนวนเต็มฐานแปด
x	int	จำนวนเต็มฐานสิบหก
f	float	จำนวนที่มีทศนิยมในรูปฐานสิบ

ลำดับหลีก(Escape sequence)

ลำดับหลีก	ผลการกระทำ
\n	ขึ้นบรรทัดใหม่ (new line)
\t	เลื่อนไปยังจุดตั้งระยะ tab ต่อไป
\a	เสียงกระดิ่ง (bell)
\b	ถอยไปหนึ่งที่ว่าง (backspace)
\f	ขึ้นหน้าใหม่ (form feed)
\\	แสดงเครื่องหมายทับกลับหลัง (backslash)
\'	แสดงเครื่องหมายฝนทอง (single quote)
\"	แสดงเครื่องหมายฟันหนู (double quote)

การแสดงค่าของตัวแปร

ชนิดข้อมูล	ชนิดตัวแปร	รูปแบบสำหรับ printf
จำนวนเต็ม	short integer long unsigned short unsigned int unsigned long	%hd หรือ %hi **%d** หรือ %i %ld หรือ %li %hu %u %lu
จำนวนจริง	float double	**%f** %lf
อักขระ สายอักขระ	char char s[]	**%c** **%s**
เลขฐาน	จำนวนเต็มฐาน 10 จำนวนเต็มฐาน 8 จำนวนเต็มฐาน 16	**%d** %o %x
	แสดงเครื่องหมาย %	**%%**

การจัดรูปแบบผลลัพธ์ printf()

“%	Flag	Maximum width	.Pecision	Size	Conversion code	”

Flag คือ - หมายถึงการจัดให้ชิดซ้าย
+ ให้แสดงเครื่องหมาย + หน้าตัวเลข

Maximum width คือ จำนวนสดมภ์ที่จองสำหรับแสดงผลข้อมูล

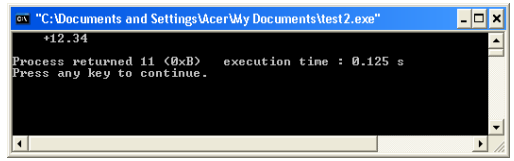
Precision คือ จำนวนตัวเลขหลังจุดทศนิยม

Size คือ ชนิดตัวแปร เช่น long จะเป็น l

การจัดรูปแบบผลลัพธ์ printf()

“%	+	10	.2		f	”

```
#include <stdio.h>
main ( )
{
    printf ("%+10.2f\n", 12.34);
}
```



				+	1	2	.	3	4
1	2	3	4	5	6	7	8	9	10

การจัดรูปแบบผลลัพธ์ printf()

Conversion code คือ การกำหนดชนิดข้อมูลในการแสดงผล

%d : พิมพ์ int ด้วยตัวเลขฐานสิบ

%o : พิมพ์ int ด้วยตัวเลขฐานแปด

%x : พิมพ์ int ด้วยตัวเลขฐานสิบหก

%f : พิมพ์ float, double แบบจุดทศนิยม (หกตำแหน่ง)

%e : พิมพ์ float, double แบบวิทยาศาสตร์ เช่น 1.23e+23

%c : พิมพ์ char ตัวอักษร 1 ตัว

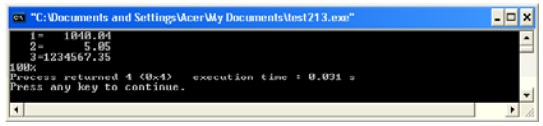
%s : พิมพ์ข้อความ เช่น “Hello”

การจัดรูปแบบผลลัพธ์ printf()

```
#include <stdio.h>
main ( )
{
    int a;
    double b;

    a=1;
    b=1040.041;
    printf ("%4i=%10.2f\n", a, b);
    a=2;
    b=5.05;
    printf ("%4i=%10.2f\n", a, b);
    a=3;
    b=1234567.351;
    printf ("%4i=%10.2f\n", a, b);
    printf("100%%");
}
```

			1	=				1	0	4	0	.	0	4
			2	=							5	.	0	5
			3	=	1	2	3	4	5	6	7	.	3	5
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



การรับค่าข้อมูล

- **ฟังก์ชัน scanf()**
- เป็นฟังก์ชันจากคลังใช้ในการรับข้อมูลจากแป้นพิมพ์ โดยจะบอกเลขที่อยู่ของตัวแปรในหน่วยความจำ แล้วจึงนำค่าที่รับมาไปเก็บไว้ตามที่อยู่นั้น ฟังก์ชัน scanf() มีรูปแบบดังนี้

  scanf(“%รูปแบบ”, &ตัวแปร);
- โดยที่ & ตัวแปร หมายถึงเลขที่อยู่ (address) ของตัวแปรที่จะรับค่ามาเก็บในหน่วยความจำ

การรับค่าข้อมูล

- **การรับค่า** scanf()

  ฟังก์ชัน scanf เป็นการรับข้อมูลจากแป้นพิมพ์ ต้อง #include <stdio.h> จึงจะสามารถใช้คำสั่ง scanf ได้
- **รูปแบบ**

  scanf("%รูปแบบ", &ตัวแปร);
- เช่น การรับข้อมูลชนิด int แล้วไปเก็บไว้ในตัวแปร age
- **รูปแบบ**

  scanf("%d", &age);
- สามารถรับข้อมูลที่ละหลายตัวแปรได้
- **รูปแบบ**

  scanf("%s %f", name, &GPAX);

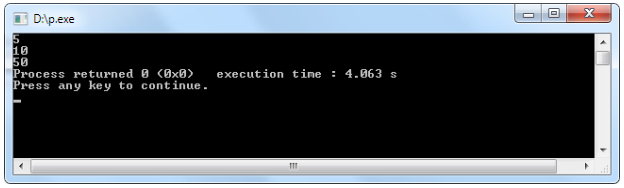
การรับข้อมูล int float char ต้องมีเครื่องหมาย & แต่ string ไม่ต้องมี &

การรับค่าข้อมูล

```
/* This program reads two integers from the keyboard
   and prints their product.
      Written by:
      Date:
*/
#include <stdio.h>

int main (void)
{
// Local Definitions
   int number1;
   int number2;
   int result;

// Statements
   scanf  ("%d", &number1);
   scanf  ("%d", &number2);
   result = number1 * number2;
   printf ("%d", result);
   return 0;
}  // main
```



คำอธิบาย

#include <stdio.h> คือการบอกคอมไพล์เลอร์ให้นำไฟล์ stdio.h มารวมด้วย

main คือชื่อของฟังก์ชัน โปรแกรมจะเริ่มทำงานที่นี่ และเมื่อจบฟังก์ชัน main หมายถึงจบโปรแกรมด้วย

scanf ("%d", &number1); คือการใช้ฟังก์ชัน scanf รับค่าจากผู้ใช้เก็บใส่ในตัวแปร number1

result = number1 * number2; คือ ผลคูณของตัวเลขที่อยู่ในตัวแปร number1 และ number2 และเก็บลงในตัวแปร result

printf ("%d", result); คือการใช้ฟังก์ชัน printf พิมพ์ข้อความที่อยู่ในเครื่องหมาย " " ออกทางอุปกรณ์เอาท์พุทมาตรฐาน

return 0 คือ การคืนค่าไปยังโปรแกรมที่เรียกใช้ฟังก์ชัน main() ซึ่งจากตัวอย่างเป็นฟังก์ชัน main() ดังนั้น return 0 จึงหมายถึงการจบการทำงานของโปรแกรม

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- **ฟังก์ชัน getchar ( )** เป็นฟังก์ชันที่ทำงานกับตัวอักษรโดยเฉพาะ
- getchar ( ) เป็นฟังก์ชันที่ใช้รับข้อมูลเข้ามาทางแป้นพิมพ์ทีละ 1 ตัวอักษร โดยต้องกด enter ทุกครั้งเมื่อสิ้นสุดข้อมูล และข้อมูลที่ป้อนจะปรากฏให้เห็นบนหน้าจอภาพด้วย
- ***รูปแบบ***

  ชื่อตัวแปร =getchar ( ) ;
- ***ตัวอย่าง*** เครื่องจะรอรับข้อมูลจากแป้นพิมพ์ที่ผู้ใช้ป้อน จำนวน 1 ตัวอักษรเก็บไว้ในตัวแปร ch หลังจากที่ผู้ใช้ต้องกดปุ่ม enter เพื่อให้ฟังก์ชันรับค่าข้อมูล

```
#include<stdio.h>
main ( )
{
  char  ch;
  ch=getchar ( );
}
```

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- **ฟังก์ชัน getch ( )**

  getch ( ) เป็นฟังก์ชันที่ใช้รับข้อมูลที่เป็นตัวอักษร 1 ตัว เข้ามาทางแป้นพิมพ์ โดยเมื่อป้อน ข้อมูลเสร็จไม่ต้องกดปุ่ม enter และอักษรที่ป้อนเข้ามาจะไม่ปรากฏบนจอภาพ ต้อง #include <conio.h> จึงจะสามารถใช้ฟังก์ชัน getch ( ) ได้

- ***รูปแบบ***

  ชื่อตัวแปร =getch ( ) ;

- ***ตัวอย่าง*** เครื่องจะรอรับข้อมูลจากแป้นพิมพ์เข้ามา 1 ตัวเพื่อนำมาเก็บไว้ในตัวแปร x โดยผู้ใช้ไม่ต้องกด enter หลังจากที่ป้อนข้อมูลเสร็จแล้ว

```
#include <stdio.h>
main ( )
{
      char x;
      x = getch( ) ;
}
```


ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

ฟังก์ชัน gets ( )

เป็นฟังก์ชันที่ใช้รับข้อมูลที่เป็นข้อความ (ตัวอักษรจำนวนหนึ่ง) จากแป้นพิมพ์เข้ามาเก็บไว้ในตัวแปร gets มาจากคำว่า get string

รูปแบบ

gets (ชื่อตัวแปร) ;

```
#include <stdio.h>
main( )
{
  char name[10];
  gets(name);
}
```

โดยรับค่าข้อความจากแป้นพิมพ์ ฟังก์ชันจะทำการใส่ ' \0 ' เอาไว้ที่ตัวสุดท้ายของข้อความ เพื่อแสดงการสิ้นสุดของข้อความที่รับเข้ามาเมื่อผู้ใช้กดปุ่ม enter

เครื่องจะจองที่ตัวแปรชุดที่ชื่อ name ซึ่งเป็นอักขระ ไว้ 10 ตัว และรอรับค่าที่เป็นข้อความเข้ามาเก็บไว้ในตัวแปรชุดที่ชื่อ nameได้ยาวไม่เกิน 9 ตัวอักษรเพื่อให้ name ตัวที่ 10 (ตัวสุดท้าย) เก็บ \0 เอาไว้

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- **ฟังก์ชัน putchar( )**
- เป็นฟังก์ชันที่ใช้ให้คอมพิวเตอร์แสดงผลบนจอภาพทีละ 1 ตัวอักษร

- ***รูปแบบ***

  putchar (ชื่อตัวแปร) ;

```
#include<stdio.h>
main ( )
{
  char x;
   x=getch ( ) ;
   printf ("The result is \n") ;
   putchar ( x );
}
```

ลักษณะข้อมูลที่ป้อน
A
ผลลัพธ์
The result is
A

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- **ฟังก์ชัน puts ( )**
- เป็นฟังก์ชันที่ใช้แสดงผลข้อมูลที่เป็นข้อความที่เก็บไว้ในตัวแปรชุดออกมาบนจอภาพ puts() มาจากคำว่า put string ใช้สำหรับพิมพ์สตริงออกทางจอภาพ
- ***รูปแบบ***

  puts (ชื่อตัวแปร) ;

- **ตัวอย่าง** เครื่องจะนำค่าที่เก็บในตัวแปรชุด name มาแสดงผลบนจอภาพ

```
#include <stdio.h>
main ( )
{
  char name [10];
  gets (name) ;
  puts (name);
}
```

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- **ฟังก์ชัน getche ( )**
- getche ( ) เป็นฟังก์ชันในการรับข้อมูล 1 ตัวอักษรโดยจะปรากฏตัวอักษรให้เห็นในการป้อนข้อมูล โดยไม่ต้องกด Enter การใช้ getche() จะต้องทำการ #include <conio.h> จึงจะสามารถใช้ getche() ได้
- ***รูปแบบ***

  ชื่อตัวแปร = getche() ;

รหัส Ascii

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com