Variable

Lecturer: Watcharin Sarachai, Ph.D.

What is a program variable?

- เป็นชื่อที่ใช้แทนตำแหน่งในการจัดเก็บข้อมูล
- สามารถจัดเก็บข้อมูลได้เพียงชนิดเดียว เช่น ชนิดข้อมูลแบบ integers หรือ characters เท่านั้น
- ทุก ๆ ตัวแปรภายในโปรแกรมจะต้องถูกประกาศก่อนการใช้งานเสมอ
- การประกาศตัวแปรจะเกี่ยวข้องกับชื่อที่ใช้แทนตำแหน่งในการจัดเก็บข้อมูลภายในหน่วยความจำและการระบุชนิดของข้อมูลที่ต้องการจัดเก็บ
- รูปแบบการประกาศตัวแปร

  <type> <name> ;

  - **float gpa;**
  - **int number;**

Variables in C program

- ตัวแปรใช้สำหรับการจัดเก็บค่าข้อมูล
- แต่ละตัวแปรประกอบไปด้วยชื่อ
- กฎเกณฑ์ในการตั้งชื่อตัวแปร
  - ประกอบไปด้วยตัวอักษร ตัวเลข และเครื่องหมาย _ (underscore)
  - ไม่สามารถเริ่มต้นด้วยตัวเลข
  - ต้องไม่มีช่องว่างหรืออักขระพิเศษ ($, #)
  - มีลักษณะเป็น case sensitive
- ควรพิจารณาใช้ชื่อที่สื่อความหมายได้ชัดเจน
- ไม่ซ้ำกับคำสงวนที่ถูกกำหนดไว้แล้ว

กฎเกณฑ์การเขียนโปรแกรมภาษาซี

1. จะต้องกำหนดพรีโปรเซสเซอร์ที่ต้นโปรแกรมก่อน เช่น #include<stdio.h>
2. คำสั่งต่าง ๆ จะใช้อักษรตัวพิมพ์เล็ก
3. ตัวแปรที่ใช้งานในโปรแกรม จะต้องถูกประกาศไว้เสมอ
4. ภายในโปรแกรมจะต้องมีอย่างน้อยหนึ่งฟังก์ชัน ซึ่งก็คือ ฟังก์ชัน main()
5. ใช้เครื่องหมาย { เพื่อบอกจุดเริ่มต้นของชุดคำสั่ง และเครื่องหมาย } เพื่อบอกจุดสิ้นสุดของชุดคำสั่ง โดยสามารถมีเครื่องหมาย { } ซ้อนอยู่ภายในได้
6. สิ้นสุดของแต่ละประโยคคำสั่ง จะต้องจบด้วยเครื่องหมาย ;
7. สามารถใช้เครื่องหมาย /* comment */ หรือ // comment เพื่อระบุหมายเหตุภายในโปรแกรม โดยข้อความอธิบายภายในเครื่องหมายดังกล่าว จะไม่ถูกนำมาประมวลผล

กฎการตั้งชื่อตัวแปร (Variable Names)

การตั้งชื่อตัวแปรและค่าคงที่ จำเป็นต้องคำนึกถึงกฎเกณฑ์ต่าง ๆ ดังนี้

1. สามารถนำตัวอักขระ ไม่ว่าจะเป็นอักษรพิมพ์ใหญ่ **A-Z** อักษรตัวพิมพ์เล็ก **a-z** ตัวเลข **0-9** รวมถึงเครื่องหมาย **_ (Underscore)** มาใช้เพื่อการตั้งชื่อตัวแปรได้ แต่มีข้อแม้ว่า ตัวอักขระตัวแรกจะต้องเป็นตัวอักษรเท่านั้น ซึ่งเครื่องหมาย _ ถือว่าเป็นตัวอักษรตัวหนึ่ง ดังนั้นจึงสามารถนำมาตั้งชื่อตัวแปรตัวแรกได้
2. ตัวอักษรพิมพ์ใหญ่ และตัวอักษรตัวพิมพ์เล็ก มีความแตกต่างกัน เช่น **Std_Code และ std_code** ถือว่าเป็นตัวแปรคนละตัว
3. ชื่อตัวแปรจะต้องไม่ต้องกับคำสงวน
4. สามารถกำหนดความยาวของตัวแปรได้ไม่จำกัด แต่จะพิจารณาความแตกต่างของตัวอักษร 31 ตัวแรก (ตามมาตรฐาน ANSI C)
5. ชื่อตัวแปรไม่อนุญาตให้มีช่องว่าง ดังนั้นหากต้องการแยกคำ ให้ใช้เครื่องหมาย _ ช่วยได้
6. ควรตั้งชื่อตัวแปรที่สามารถสื่อความหมาย เพื่อบอกให้รู้ว่าตัวแปรนั้น ๆ นำไปใช้เพื่อการใด

Reserved Word In C

- ภาษาซีมีคำสงวนที่ได้กำหนดความหมายไว้ก่อน คำสงวนเหล่านี้จะใช้ได้เฉพาะเพื่อวัตถุประสงค์ตามที่กำหนดไว้เท่านั้น ผู้ใช้ไม่สามารถที่จะกำหนดความหมายของคำสงวนเหล่านี้ขึ้นมาใหม่ได้

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

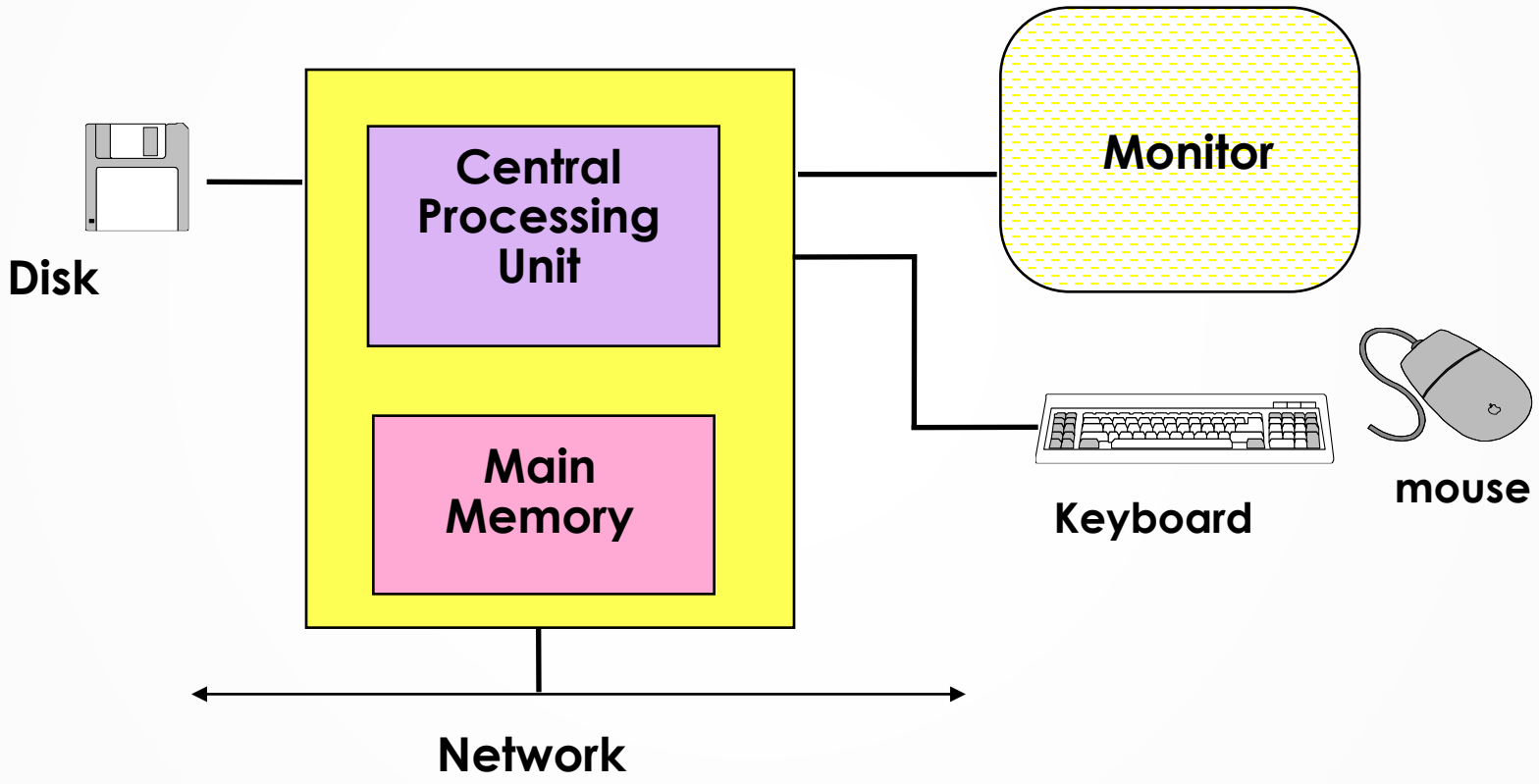
ตัวอย่าง การตั้งชื่อตัวแปรที่ถูกต้อง

a	month	MONTH	_var1
salary	stdCode	num1	day_of_week
EmpID	NAME	FLOAT	topofWindow

ตัวอย่าง การตั้งชื่อตัวแปรที่ผิด

586_cpu	emp no	do	total$
*point	sub-total	num1,2	@email

Review: Computer Organization



Data types in C program

- ชนิดข้อมูลหลักออกเป็น 3 ชนิดตามขนาดของการจัดเก็บ (Storage Area) ของตัวแปรแต่ละชนิดภายในหน่วยความจำ

ชนิดตัวแปร	ขนาด	เลขยกกำลัง	ค่าสูงสุด
character	1 ไบต์ (8 บิต)	$2^8 - 1$	255
integer	4 ไบต์ (32 บิต)	$2^{32} - 1$	2,147,483,647
floating point	4 ไบต์ (32 บิต)	$2^{32} - 1$	6 decimal places
double	8 ไบต์ (64 บิต)	$2^{64} - 1$	15 decimal places

https://www.tutorialspoint.com/cprogramming/c_data_types.htm

Data types in C program

- นอกจากนั้นภาษาซียังอนุญาตให้ผู้ใช้นำคำสงวน มาใช้ประกอบกับชนิดของข้อมูล เพื่อให้สามารถเก็บค่าได้อย่างมีประสิทธิภาพมากขึ้น ได้แก่
  - short_ ใช้ร่วมกับตัวแปรแบบเลขจำนวนเต็ม การกำหนดขนาดแบบนี้จะมีผลทำให้เกิดการประหยัด เนื้อที่ของตัวแปรลงครึ่งหนึ่งทั้งนี้จะขึ้นอยู่กับการนำไปใช้
  - long เป็นการกำหนดขนาดอีกแบบหนึ่งซึ่งใช้กับตัวแปรแบบเลขจำนวนเต็ม ซึ่งจะตรงกันข้ามกับแบบแรก นั่นคืออาจมีผลทำให้ขนาดของตัวแปรเพิ่มขึ้นอีกหนึ่งเท่า
  - double จะทำงานคล้ายกับการกำหนดขนาดแบบ long แต่จะแตกต่างกันที่ double จะใช้กับตัวแปรแบบทศนิยมคงที่เท่านั้น

Data types in C program

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	8 bytes or (4bytes for 32 bit OS)	-9223372036854775808 to 9223372036854775807
unsigned long	8 bytes	0 to 18446744073709551615

9223372036854775807

Data types in C program

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Variable's Declaration

- การประกาศตัวแปรจะขึ้นต้นด้วยการระบุชนิดของตัวแปร ตามด้วยชื่อของตัวแปรตามลำดับ
- ***รูปแบบ: ชนิดของข้อมูล ชื่อตัวแปร;***
- ตัวอย่างการประกาศตัวแปรของชนิดข้อมูลหลัก ๆ มีดังต่อไปนี้

ตัวแปรอักขระ	ตัวแปรเลขจำนวนเต็ม	ตัวเลขทศนิยมคงที่

main() {
char ch;
..... ;
}
	main() {
	int x;
	..... ;
	}
		main() {
		float f;
		..... ;
		}

การกำหนดค่าให้กับตัวแปร

การกำหนดค่าให้กับตัวแปร จะต้องกำหนดให้เป็นไปตามกฎ และต้องสัมพันธ์กับชนิดข้อมูลที่ประกาศไว้เป็นสำคัญ

1. กฎเกณฑ์การกำหนดค่าชนิดเลขจำนวนเต็ม (Integer)

1. ค่าตัวเลขจะต้องไม่มีทศนิยม
2. สามารถเป็นได้ทั้งบวก หรือค่าลบ
3. สำหรับค่าบวก ไม่จำเป็นต้องใส่เครื่องหมาย + นำหน้าค่า
4. ห้ามใช้เครื่องหมาย , หรือช่องว่าง กำกับระว่างตัวเลข เช่น 14,25 ถือว่าผิด
5. ช่วงค่าตัวเลขจำนวนเต็ม จะอยู่ระหว่าง -2,147,483,648 ถึง 2,147,483,648
6. สามารถใช้ Suffix ต่อท้ายค่าได้ เช่น เลขจำนวนเต็มแบบยาว ก็จะใช้อักษร L (ตัวพิมพ์เล็กหรือใหญ่) ต่อท้ายค่า ส่วนค่าที่เป็น unsign จะใช้อักษร U ต่อท้าย และใช้ UL ต่อท้ายค่าที่เป็น unsigned long

การกำหนดค่าให้กับตัวแปร

2. กฎเกณฑ์การกำหนดชนิดเลขจำนวนจริง

1. ค่าตัวเลขสามารถมีจุดทศนิยมได้

2. สามารถเป็นได้ทั้งค่าบวก หรือค่าลบ

3. สำหรับค่าบวก ไม่จำเป็นต้องใส่เครื่องหมาย + นำหน้าค่า

4. สามารถกำหนดค่าแบบเอ็กโปเนนต์ได้ ด้วยการใช้อักษร E ต่อท้ายค่า

5. ค่าที่ถูกกำหนดเป็นเอ็กซ์โปเนนต์ สามารถเป็นได้ทั้งค่าบวก หรือค่าลบ

6. ช่วงของค่าตัวเลขชนิดจำนวนจริงเป็นไปตามชนิดข้อมูล ซึ่งอาจถูกกำหนดเป็น float , double หรือ long double

7. สำหรับค่าเลขจำนวนจริงชนิด double จะใช้ F ต่อท้ายค่า และใช้ L ต่อท้ายค่าที่กำหนดชนิดข้อมูลเป็น long double

```
n1 = 41.9 ;     // double
n2 = 12.34F;   // float
n3 = 12.34L;   // long double
```

การกำหนดค่าให้กับตัวแปร

3. กฎเกณฑ์การกำหนดค่าชนิด

1. ค่าที่กำหนด จะต้องเป็นอักขระเพียงค่าเดียว และจะต้องอยู่ภายในเครื่องหมาย ‘ ’ ไม่ใช่เครื่องหมาย “ ” ดังนั้นต้องระวังเป็นพิเศษ

```
char   c1;
c1 = ‘A’;
```

2. ขนาดจำนวนตัวอักษรสูงสุดคือ 1 ตัวอักษรเท่านั้น

ในกรณีที่ต้องการตัวแปรเก็บค่าข้อความ ซึ่งมีหลายตัวอักขระรวมกัน หรือที่เรียกว่าข้อความแบบสตริง (String) ในภาษาซี จะใช้ตัวแปรอาร์เรย์ในการจัดการกับข้อความเหล่านั้น โดยข้อความดังกล่าว จะถูกกำหนดอยู่ในภายในเครื่องหมาย “ ” (double Quotes) เช่น

```
char prompt[28];
char msg[ ] = “press any key to continue...” ;
```

ค่าคงที่ (Constants)

ในภาษาซีสามารถกำหนดค่าคงที่ในรูปแบบต่าง ๆ ได้ดังนี้

1. **ค่าคงที่ที่ระบุค่าโดยตรงลงไป (Literal Constants)**

   เป็นค่าคงที่ที่ระบุแบบเจาะจงลงไปในโปรแกรมเลย ด้วยการคำสั่งพิมพ์ข้อความดังกล่าวโดยตรง ไม่มีการนำตัวแปรมาเก็บค่าดังกล่าว ดังนั้นค่าคงที่ประเภทนี้จะไม่สามารถถูกเรียกใช้งานผ่านตัวแปรได้

   ตัวอย่าง การกำหนดค่าคงที่แบบ Literal ชนิดข้อความ ผ่านฟังก์ชัน printf()

```
printf ("The C Programming Language");
printf("%s”, “You can learning by yourself! ");
printf("Now. The speed limit here is %d.\n",55);
```

ตัวอย่าง การใช้คำสั่ง const สร้างค่าคงที่

```
#include<stdio.h>
main()
{
  const double pi=3.14;
  const float K=4;
  const char ch= 'A';
  const char company[10]="INTER";
  printf("pi = %d\n",pi);
  printf("K = %f\n",K);
  printf("ch = %d\n",ch);
  printf("company name = %s",company);
}
```

2. ค่าคงที่ที่นิยมด้วย #define (Defined Constants)

ในภาษาซีได้เตรียมพรีโปรเซสเซอร์ไดเรกทีฟชื่อ #define เพื่อนำมาใช้สำหรับกำหนดค่าคงที่ให้โปรแกรม ซึ่งจะถูกกำหนดไว้ที่ต้นโปรแกรม ดังนี้

```
#define SHORTCUT  value
#define MAX 100
#define TXT “Hello”
#define BELL  ‘\007’
```

#define	หมายถึง	พรีโปรเซสเซอร์ไดเรกทีฟ
SHORTCUT	หมายถึง	ชื่อของค่าคงที่ ซึ่งมักกำหนดเป็นอักษรตัวพิมพ์ใหญ่
value	หมายถึง	ค่าของค่าคงที่

ในการกำหนดค่าคงที่ด้วย #define จะไม่ใช้เครื่องหมาย = และจะไม่จำเป็นต้องมีเครื่องหมาย ; ปิดท้าย โดยปกติค่าคงที่นิยามไว้ใน #define ส่วนใหญ่มักใช้อักษรพิมพ์ใหญ่

การใช้คำสั่ง #define สร้างค่าคงที่

```
#include<stdio.h>
#define PI 3.14
#define NAME "SASALAK"
#define CH 'a'
void main()
{
  printf(“PI = %f\n”,PI);
  printf(“NAME = %s\n”,NAME);
  printf(“PI = %c\n”,CH);
}
```

3. ค่าคงที่ที่เก็บไว้ในตัวแปร (Memory Constants)

สำหรับการกำหนดค่าคงที่ คือ การกำหนดค่าคงที่ให้กับตัวแปร ซึ่งมีรูปแบบดังนี้

```
const  data_type variable_name = value;
```

const	หมายถึง การระบุว่าตัวแปรที่กำหนดต่อไปนี้เป็นค่าคงที่
data_type	หมายถึง ชนิดของข้อมูล
variable_name	หมายถึง ชื่อตัวแปรที่เก็บค่าคงที่
Value	หมายถึง ค่าคงที่ที่จัดเก็บอยู่ในตัวแปร

การประกาศตัวแปร (Declarations)

ตัวแปรทุกตัวจะต้องถูกประกาศก่อนการใช้งานเสมอ โดยมีรูปแบบการประกาศตัวแปรได้หลายรูปแบบ ดังนี้

- **1. การประกาศตัวแปรทีละตัว ทีละบรรทัด**

  เป็นการประกาศตัวแปรที่ใช้งานแต่ละตัว ทีละบันทัด เช่น

  ```
  int  lower;
  int  upper;
  char c;
  char line [1000];
  ```

- **2. การประกาศกลุ่มตัวแปรพร้อมกัน**

  เป็นการประกาศตัวแปรที่มีชนิดข้อมูลเหมือนกันภายในบรรทัดเดียวกัน (Multiple Declarations) โดยใช้เครื่องหมาย , คั่นระหว่างชื่อตัวแปรแต่ละตัว เช่น

  ```
  int lower , upper , step;
  ```

การประกาศตัวแปร (Declarations)

- **3. การประกาศตัวแปร พร้อมกำหนดค่าเริ่มต้น**

  เป็นการประกาศตัวแปร พร้อมกำหนดค่าเริ่มต้นให้กับตัวแปรภายในบรรทัดนั้น

```
char esc = '\\' ;
int   i  = 0 ;
int   limit  = MAXLINE + 1 ;
float   eps  = 1.0e-5 ;
```

- **4. ประกาศตัวแปรชนิดค่าคงที่**

  เป็นการประกาศตัวแปรพร้อมกำหนดค่าให้กับตัวแปรชนิดค่าคงที่ ซึ่งตัวแปรชนิดค่าคงที่นั้น จะไม่สามารถเปลี่ยนแปลงค่าในภายหลังได้อีก

```
const  double   e   = 2.71828182459028;
const  char  msg[ ]  = "Warning..." ;
```

Assigning Values to Variables

- การกำหนดค่าให้กับตัวแปรสามารถทำได้โดยใช้เครื่องหมาย “=” เรียกว่า assignment statement
  - ตัวแปรจะต้องถูกประกาศก่อนใช้งานเสมอ
- รูปแบบของการกำหนดค่า:

  <name> = *<value>* ;
  - int number = 3;
  - double gpa = 3.25;

number = 2.5; // ERROR: number can only store int

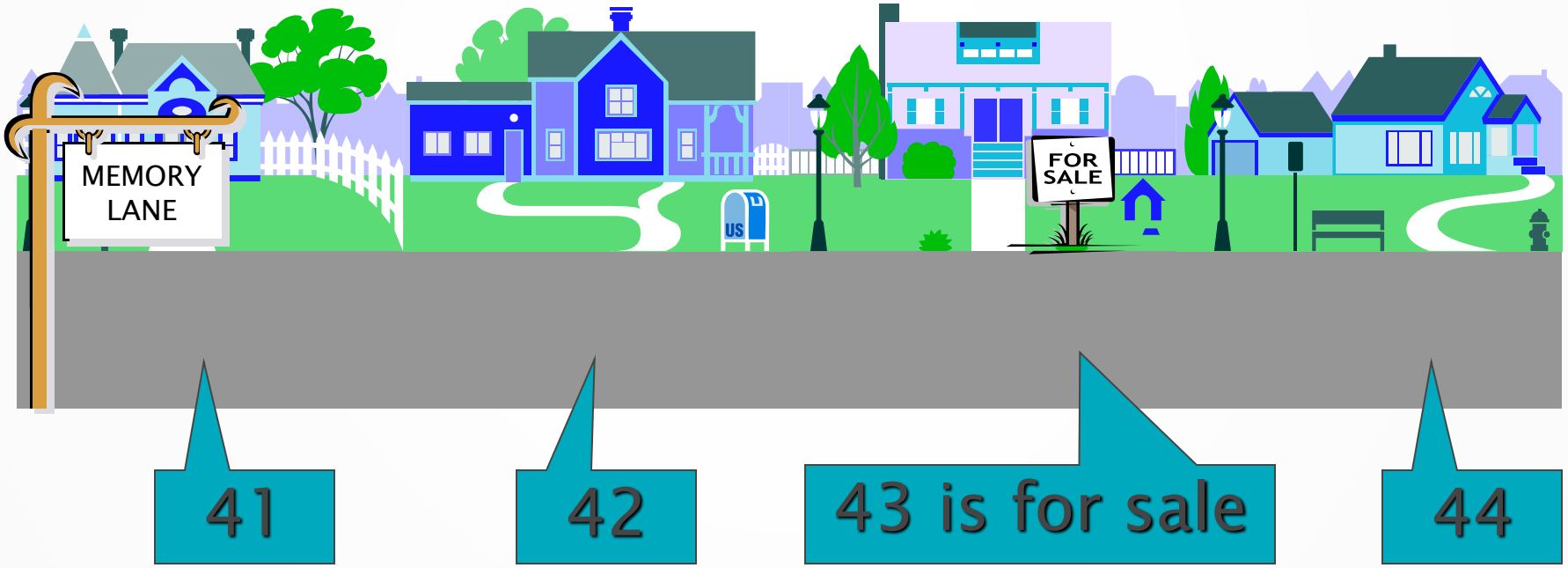
“Memory” & Variable

- เมมโมรี่เป็นเสมือนตารางขนาดใหญ่ที่ประกอบด้วยช่องที่มีลักษณะเป็น slots สำหรับเก็บไบต์ข้อมูล
- จำนวนของ slot จะถือเป็น Address ค่าจำนวน 1 ไบต์จะถูกจัดเก็บลงในแต่ละ slot
- ชื่อของตัวแปรจะถูกกำหนดไว้ ณ ตำแหน่งของเมมโมรี่ที่ใช้ในการจัดเก็บค่า
- ตัวอย่างเช่น การประกาศตัวแปรและการกำหนดค่า

```
char x;
char y='e';
```

Symbol	Addr	Value
	0	
	1	
	2	
	3	
x	4	?
y	5	‘e’ (101)
	6	
	7	
	8	
	9	
	10	
	11	
	12	

Addressing: an analogy



Multi-byte Variables

- ชนิดข้อมูลที่แตกต่างกันจะใช้พื้นที่ในการจัดเก็บข้อมูลที่แตกต่างกันตามไปด้วย ตัวอย่างเช่น

```
char x;
char y='e';
int z = 65400;
```

char ใช้พื้นที่จำนวน 1 ไบต์

int z ใช้พื้นที่จำนวน 2 ไบต์

Symbol	Addr	Value
	0	
	1	
	2	
	3	
x	4	?
y	5	‘e’ (101)
	6	
	7	
z	8	65400
	9	
	10	2
	11	1
	12	

More about assignment

- การกำหนดค่าให้กับตัวแปร สามารถกระทำผ่านนิพจน์ทางคณิตศาสตร์ โดยพิจารณาจากลำดับความสำคัญของเครื่องหมายทางคณิตศาสตร์

  เช่น `x = 2 + 3 * 5;`

  `x` | `17`

- ตัวแปรสามารถกำหนดค่าได้มากตามจำนวนครั้งที่ต้องการ แต่สามารถจัดเก็บค่าได้เพียงค่าเดียวในช่วงเวลาหนึ่ง

```
int x;
x = 3;
x = 4 + 7;
```

Multiple declarations per line

- การประกาศตัวแปรสามารถทำได้หลาย ๆ ตัวแปรในหนึ่งบรรทัด

  <type> <name>, <name>, ..., <name> ;

  - ตัวอย่างเช่น: int a, b, c;

    double x, y;

- การประกาศตัวแปรและการกำหนดค่าสามารถทำได้พร้อม ๆ กันในครั้งเดียว

  <type> <name> = <value> , ..., <name> = <value> ;

  - ตัวอย่างเช่น: int a = 2, b = 3, c = -4;

    double grade = 3.5, delta = 0.1;

- หมายเหตุ กรณีที่ต้องการประกาศหลาย ๆ ตัวแปรในคำสั่งเดียวจะต้องมีชนิดข้อมูลแบบเดียวกันเท่านั้น

Variable's Scope

ขอบเขตการเข้าถึงของตัวแปรขึ้นกับตำแหน่งการประกาศตัวแปรสามารถทำได้ 3 ประเภท

1. การประกาศตัวแปรภายในฟังก์ชัน (Automatic local) จะเป็นการประกาศตัวแปรไว้ภายในฟังก์ชัน main() ค่าตัวแปรนี้จะมีขอบเขตการใช้งานได้เฉพาะภายในฟังก์ชัน main() เท่านั้น
2. การประกาศตัวแปรภายนอกฟังก์ชัน (Global) จะเป็นการประกาศตัวแปรไว้ภายนอกฟังก์ชัน main() โดยประกาศแบบนี้จะวางไว้เหนือฟังก์ชัน main() เสมอ ค่าตัวแปรเหล่านี้สามารถนำไปใช้ในโปรแกรมโดยไม่มีการจำกัดขอบเขตการใช้งาน
3. การประกาศในรูปของการส่งผ่านค่าไปยังฟังก์ชัน (Formal Parameter) จะเป็นการประกาศตัวแปรที่มีจุดประสงค์ในการผ่านค่าไปยังฟังก์ชันอื่น ๆ การประกาศตัวแปรแบบที่สามจะอ้างถึงต่อไปในบทที่เกี่ยวข้องกับฟังก์ชัน

Ex. Automatic local & Global variable

```
main(){
  /*Automatic local */
  int event = 5;
  char heat = 'C’;
  float time = 28.35;
  printf("The winning time in heat %c ",heat);
  printf("of event %d was %.2f .",event,time);
  getch();
}
```

```
/*Global Variable*/
int event = 5;
char heat = 'C';
float time = 28.35;
main(){
  printf("The winning time in heat %c ",heat);
  printf("of event %d was %.2f .",event,time);
  getch();
}
```

Arithmetic Expressions

- *Expression* – เป็นนิพจน์ที่ประกอบไปด้วย operands ตั้งแต่หนึ่งหรือมากกว่าที่ใช้เป็นตัวกระทำ
- สนับสนุนการคำนวณพื้นฐานที่ใช้เครื่องหมายทางคณิตศาสตร์ดังต่อไปนี้

Operator	Meaning	integer types	real types
+	addition	5 + 2 -> 7	5.0 + 2.0 -> 7.0
-	subtraction	5 - 2 -> 3	5.0 - 2.0 -> 3.0
*	multiplication	5 * 2 -> 10	5.0 * 2.0 -> 10.0
/	division	5 / 2 -> 2	5.0 / 2.0 -> 2.5
%	remainder	5 % 2 -> 1	*not applicable*

Division and Remainder

- ในกรณีที่operands ทั้งคู่เป็น integer ถูกนำมาหารกัน (/) ผลลัพธ์ที่ได้จะต้องเป็น integer (ส่วนที่เป็นทศนิยมจะถูกตัดทิ้งไปโดยอัตโนมัติ)

```
14 / 3      equals?   4
8 / 12      equals?   0
```

- ในกรณีที่ใช้การ modulus ด้วยเครื่องหมาย (%) ผลลัพธ์ที่ได้จะเป็นเศษที่ได้จากการคำนวณ (ในกรณีนี้จะทำได้เฉพาะกับ integer types)

```
14 % 3      equals?   2
8 % 12      equals?   8
```

Time Example : % Usage

% คือคำสั่งสำหรับ mod หรือหาค่า remainder เช่น :

299 % 100 → 99

6 % 4 → 2

5 % 6 → 5

กำหนดให้: total_minutes 359

ต้องการหา: hours 5

minutes 59

hours = total_minutes / 60 ;

minutes = total_minutes % 60 ;

Operator Precedence

- การประเมินผลทางคณิตศาสตร์อาศัยลำดับความสำคัญของเครื่องหมาย ได้แก่
  1. เครื่องหมาย *, / และ %
  2. เครื่องหมาย + และ -
  3. เครื่องหมายทางคณิตศาสตร์ที่มีลำดับความสำคัญเท่ากันจะถูกประเมินจากซ้ายไปขวา

```
a + b + c + d + e
  1   2   3   4

a + b * c - d / e
    3   1   4   2

a / (b + c) - d % e
  2     1     4   3
```

ตัวอย่างการใช้งาน

เครื่องหมาย	ตัวแปร	การใช้งาน	คำอธิบาย
+	A+B	4+2 = 6	4 บวก 2 เท่ากับ 6
-	A-B	4-2 = 2	4 ลบด้วย 2 เท่ากับ 2
*	A*B	4*2 = 8	4 คูณด้วย 2 เท่ากับ 8
/	A/B	4/2 = 2	4 หารด้วย 2 เท่ากับ ค่าที่ได้ คือ 2
%	A%B	5%2 = 1	5 โมดูเลท 2 เท่ากับ ค่าที่ได้ คือ 1

Arithmetic Operators – Order of Evaluation

x = 2 * 3 - (4 + 5) + 8 % 7;
x = 2 * 3 - 9 + 8 % 7;
x = 6 - 9 + 8 % 7;

x = 6 - 9 + 1;

x = -3 + 1;

x = -2;

Another example:

x = 6 / 2 + 1 - 3 + 8 * 4;

x = 33;
x = 6 / (2 + 1) - (3 + 8) * 4;
x = -42;

More Assignment Operators

Operator:	Example:	Meaning:
=	x = 5 ;	x = 5 ;
+=	x += 5 ;	x = x + 5 ;
–=	x –= 5 ;	x = x – 5 ;
/=	x /= 5 ;	x = x / 5 ;
*=	x *= 5 ;	x = x * 5 ;
%=	x %= 5;	x = x % 5 ;

ตัวอย่างของ Assignment Operator กำหนดให้ **int a = 4, b = 2, c = 36 ;**

a += b ;

[Answer: a = a + b, so a = 4 + 2 or a = 6]

c /= a + b ;

[Answer: c = c / (a + b), and a = 6 now,

so c = 36 / (6 + 2), so c = 36 / 8 or c = 4]

Increment/Decrement Operators

- เป็นตัวดำเนินการที่ต้องการตัวกระทำ (operand) เพียงตัวเดียวในการทำงานร่วมกับเครื่องหมายบวกหรือลบเพื่อการเพิ่มหรือลดค่ามีดังต่อไปนี้
  - ++ เครื่องหมายเพิ่มค่าทีละ 1
  - -- เครื่องหมายลบค่าทีละ 1

เครื่องหมาย	คำสั่งในการทำงาน	ค่า x	ค่า y
x++	x = 1; y = x++;	2	1
x--	x = 1; y = x--;	0	1
++x	x = 1; y = ++x;.	2	2
--x	x = 1; y = --x;	0	0

เครื่องหมายการคำนวณทางคณิตศาสตร์
ประเภทการเพิ่มหรือลดค่าทีละหนึ่ง

เครื่องหมาย	การดำเนินการ	ตัวอย่างการใช้งาน	ความหมาย
++	เพิ่มค่าทีละหนึ่ง (Increment)	Y= ++x	บวกค่าในตัวแปร x เพิ่มขึ้น 1 ก่อนที่จะกำหนดค่า x ให้กับตัวแปร y
		Y= x++	กำหนดค่าในตัวแปร x ให้กับตัวแปร y ก่อนที่จะเพิ่มค่า x ขึ้นทีละ 1
- -	ลดค่าทีละหนึ่ง (Decrement)	Y= - -x	ลบค่าในตัวแปร x ลง 1 ก่อนที่จะกำนดค่า x ให้กับตัวแปร y
		Y= x- -	กำหนดค่าในตัวแปร xให้กับตัวแปร y ก่อนที่จะลดค่า x ลง 1

```
n=5
x=++n
x=6
```

เป็นการบวกค่าเพิ่มเข้าไปทีละ1ให้กับค่า n แล้วค่อยโอนค่าให้กับค่า x ดังนั้น x =6

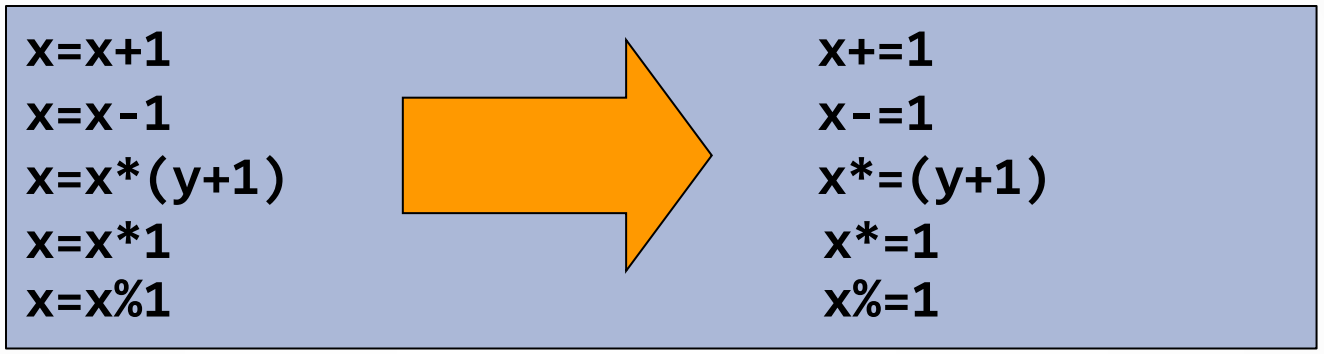
```
n=5
x=n++
x=5
a=x
x=6
```

จะทำการโอนค่า n ให้กับค่า x ก่อนแล้วจึงทำ การบวกค่าเพิ่มเข้าไปทีละ1ให้กับค่า x โดยผลลัพธ์ที่ได้ x= 5 และ n = 6 และผลลัพธ์ X จะเท่ากับ 6 ได้นั้น ก็ต่อเมื่อ x อยู่บรรทัดถัดไป

เครื่องหมายประเภทที่เรียกว่าลดรูป

เครื่องหมาย	ตัวอย่างการใช้งาน	ความหมาย
+=	y +=x	บวกค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
-=	y -=x	ลบค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
*=	y *=x	คูณค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
/=	y /=x	หารค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
%=	y %=x	หารค่าในตัวแปร y ด้วยค่าในตัวแปร x เศษจากการหารเป็นผลลัพธ์กำหนดกลับไปให้ y

ตัวอย่างการรวมตัวแปรเข้าด้วยกัน หรือการลดรูป



เครื่องหมายที่ใช้กับตัวแปร

int a = 5;

int b = 6;

คำสั่ง	มีผลเหมือนกับ	ผลที่ได้
a+=4;	a=a+4;	a=9
b--;	b=b-1;	b=5
a*=2;	a=a*2;	a = 10
a/=2	a=a/2;	a =2

Conversions in Assignments

```
int total, count, value;

float avg;

total = 97 ;

count = 10;

avg = total / count ;   /*avg is 9.0!*/

value = total*2.2;      /*bad news*/
```

implicit conversion to double

implicit conversion to int – drops fraction with no warning

ตัวดำเนินการเปรียบเทียบและตรรกะ (Comparison and Logical Operators)

- คือ เครื่องหมายที่ใช้ในการเปรียบเทียบและตัดสินใจ ซึ่งผลการเปรียบเทียบจะได้เป็น 2 กรณี คือ จริงจะให้ค่าเป็น 1 และเท็จจะให้ค่าเป็น 0

เครื่องหมาย	ความหมาย	ใช้กับตัวแปร	ตัวอย่าง
>	มากกว่า	ทุกชนิด	A > B
>=	มากกว่าท่ากับ	ทุกชนิด	A >= B
<	น้อยกว่า	ทุกชนิด	A < B
<=	น้อยกว่าเท่ากับ	ทุกชนิด	A <= B
= =	เท่ากับ	ทุกชนิด	A = = B
!=	ไม่เท่ากับ	ทุกชนิด	A != B

ต.ย. การใช้เครื่องหมายเปรียบเทียบ

```
#include<stdio.h>
main()
{
  int age = 15;
      printf("Is age less than = 21?%d\n",age<21);
      age=30;
      printf("Is age less than = 21?%d\n",age<21);

}
```

Output: Is age less than 21? 1
Is age less than 21? 0

ต.ย. การใช้เครื่องหมายเครื่องหมายการเปรียบเทียบ

เครื่องหมาย	ตัวอย่างตัวแปร	ตัวอย่างการใช้งาน	ผลที่ได้	คำอธิบาย
= =	A = = B	(4+2) = = (4-2)	False	6 = = 2
!=	A != B	(4+2) ! = (4-2)	True	6 != 2
>	A > B	(4+2) > (4-2)	True	6 > 2
<	A < B	(4+2) < (4-2)	False	6 < 2
>=	A >= B	(4+2) >= (4-2)	True	6 >= 2
<=	A <= B	(4+2) <= (4-2)	False	6 <= 2

เครื่องหมายทางตรรกศาสตร์

- คือ เครื่องหมายที่ใช้ในการเปรียบเทียบและตัดสินใจโดยเอาเงื่อนไขตั้งแต่ 2 เงื่อนไขมาเปรียบเทียบกัน ผลที่ได้จากการเปรียบเทียบเป็น 2 กรณี คือ จริง ซึ่งให้ค่าเป็น 1 และเท็จ ซึ่งให้ค่าเป็นศูนย์

เครื่องหมาย	ความหมาย	ใช้กับตัวแปร	ตัวอย่าง
&&	(And) และ	จำนวนเต็ม	A && B
||	(or) หรือ	จำนวนเต็ม	A || B
!	(not) ไม่	จำนวนเต็ม	!A

การดำเนินการโดยใช้เครื่องหมาย &&

x	y	x&&y
T	T	T
T	F	F
F	T	F
F	F	F

การดำเนินการโดยใช้เครื่องหมาย ||

x	y	x||y
T	T	T
T	F	T
F	T	T
F	F	F

การดำเนินการโดยใช้เครื่องหมาย !

y	!y
T	F
F	T

ตัวอย่างการเปรียบเทียบนิพจน์ทางคณิตศาสตร์

```
Int x=5;
float y = -2.5;
char a= 'A';
char b;
```

(x !=y) && (x <a)

(5 !=-2.5) && (5 < 65)

T && T

Output: T

หมายเหตุ

- ข้อมูลชนิดตัวเลขทศนิยม ถ้า เป็นทศนิยมที่อยู่ในรูปยกกำลัง จะใช้ E หรือ e เช่น 1.25*10-2 เราจะเขียนแทนด้วย 1.25e-02
- ข้อมูลชนิดเลขฐานแปด ในภาษาซี ต้องขึ้นต้นด้วย 0 ก่อน เช่น 0124 แทน ตัว T int otc=0124;
- ข้อมูลชนิดเลขฐานสิบหก ในภาษาซีต้องขึ้นต้นด้วย0X ก่อน เช่น 0X54 แทน ตัว T int hex=0X54;

ตารางแสดงสัญลักษณ์แสดงผล

สัญลักษณ์	ใช้สำหรับ
%d	แสดงค่าที่เป็นเลขจำนวนเต็ม
%s	แสดงค่าที่เป็นสตริง
%f	แสดงค่าที่เป็นเลขทศนิยม
%c	แสดงค่าที่เป็นตัวอักษร 1 ตัว
%o	แสดงค่าของตัวเลขในรูปฐานแปด
%x	แสดงค่าของตัวเลขในรูปฐานสิบหก

ตัวอย่าง การนำตัวแปรไปใช้ในการคำนวณ

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

   a = 20;
   b = 40;
   c = 5;

   ans = (a+b)/c;
   printf("Answer is %d\n", ans);

   }
```

Answer is 12

Explicit Conversions

- ใช้ cast ในกรณีที่ต้องการแปลงผลลัพธ์ของนิพจน์ที่มีชนิดข้อมูลที่แตกต่างกัน

Format example: **(type) expression**

(double) myage

(int) (balance + deposit)

```
int total, count ;       /* explicit conversion to double (wrong way)*/
double avg;                      avg = (double) (total / count) ;
total = 97 ;    count = 10 ;                    /*avg is 9.0*/
/* explicit conversion to double (right way)*/
avg = (double) total / (double) count);     /*avg is 9.7 */
```

Formatted input: scanf

- **scanf()** เป็นฟังก์ชันมาตรฐานทำหน้าที่หยุดรอรับข้อมูลจากการกดแป้นพิมพ์โดยผู้ใช้ และจะสิ้นสุดการรับค่าต่อเมื่อผู้ใช้กด **enter key**
- การรับค่าจะไม่สามารถกระทำได้หากมีช่องว่างที่เกิดขึ้นในระหว่างการกดแป้นพิมพ์ เช่น สัญลักษณ์ที่เป็นช่องว่าง **(Space),** แท็บตั้งระยะ **(Tab)** และการขึ้นบรรทัดใหม่ **(New-line)** เป็นต้น
- **arguments** จะได้แก่ สัญลักษณ์ที่ใช้แสดงค่าตำแหน่งแอดเดรส ได้แก่ เครื่องหมาย *'&'* ซึ่งใช้ในการระบุตำแหน่งภายในหน่วยความจำ ตามด้วยชื่อของตัวแปรที่กำหนด
- *scanf("%d ", &n);*

Input data with scanf()

- **scanf()** ทำงานโดยการแปลงค่าตัวอักขระจากการกดแป้นพิมพ์โดยผู้ใช้ โดยการกำหนดรูปแบบแทนชนิดข้อมูลต่างๆ เช่น %c, %d, %f เป็นต้น

```
main() {
    float years,days;
        printf ("Please type your age in year :   ");
        scanf ("%f", &years);
        days = years * 365;
        printf("You are %.1f days old\n",days);
}
```

Output: **Please type your age in year : 2**
You are 730 days old

Input character with getch()

- **getch()** ทำหน้าที่ในการรับค่าจากการกดแป้นพิมพ์เพียงค่าเดียว โดยจะทำงานและสิ้นสุดการรับค่าทันทีที่มีการกดค่าจากแป้นพิมพ์คีย์ใดคีย์หนึ่ง การใช้ฟังก์ชันนี้จะไม่แสดงค่าผลการรับค่าออกทางหน้าจอ

  รูปแบบ: ***Variable = getch();***

- การประกาศตัวแปรแบบอักขระหากทำการกำหนดค่าเริ่มต้น ค่าของตัวแปรจะต้องอยู่ภายในเครื่องหมาย Quote (' ') เสมอ

- หากผู้ใช้ต้องการให้มีผลปรากฏบนหน้าจอควรจะเลือกใช้ฟังก์ชัน getche() ซึ่งจะมีรูปแบบและวิธีการใช้เหมือนกันดังนี้

Input character with getch()

- เนื่องจากฟังก์ชัน **getch()** หยุดรอรับค่าจากการกดคีย์แป้นพิมพ์เพียงค่าเดียว ดังนั้นในบางครั้งฟังก์ชันอาจถูกนำมาใช้ในกรณีที่ต้องการหยุดดูผลลัพธ์การทำงานของโปรแกรมทางหน้าจอได้ โดยวางฟังก์ชันนี้ไว้ที่บริเวณส่วนท้ายสุดของโปรแกรม

```
main() {
   char ch;
   printf("Type any character :   ");
      ch = getche( );
      printf("\nThe letter you typed was %c",ch);
      getch();
}
```

Output : **Type any character : x**
The letter you typed was x

Q&A