

**DATA
TYPEF**



បណ្តុះបន្ទាន់ខ្លួន

ຈະមើលឲ្យ 3 បណ្តកំឲត់ការងារ

1. **int** - ការបានឱ្យកិច្ចបន្ទាន់ខ្លួនត្រូវលេខកំណត់ដែលត្រូវបានគេបង្ហាញឡើង ដូច 123, -123
2. **float** - ការបានឱ្យកិច្ចបន្ទាន់ខ្លួនត្រូវលេខកំណត់ដែលត្រូវបានគេបង្ហាញឡើង ដូច 12.5, -123.353
3. **char** - ការបានឱ្យកិច្ចបន្ទាន់ខ្លួនត្រូវលេខកំណត់ដែលត្រូវបានគេបង្ហាញឡើង ដូច 'a', 'A'

** បានឱ្យកិច្ចបន្ទាន់ខ្លួនត្រូវបានគេបង្ហាញឡើង ត្រូវបានគេបង្ហាញឡើង ត្រូវបានគេបង្ហាញឡើង

ตัวอักษรในภาษา C จะมีรหัส ASCII ซึ่งทำให้ตัวอักษรเป็น char และ int

Decimal - Binary - Octal - Hex – ASCII Conversion Chart																			
Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	\$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int num1 = 'A';
6
7     char chr1 = 65;
8
9     printf("num1 = %d\n", num1);
10
11    printf("chr1 = %c\n", chr1);
12
13    return 0;
14 }
```

OUTPUT

```
[kitten@Kitten]~/mnt/d/ce-boostup-xi/boostupxi-api]
$ ./main.exe
num1 = 65
chr1 = A

[kitten@Kitten]~/mnt/d/ce-boostup-xi/boostupxi-api]
$
```

นอกจาก 3 ชนิดข้างบนแล้วก็ยังมีตัวประบิดอื่นอยู่อีก ตามตัวอย่างในตาราง

Type	Definition	Control Character	Limits
int	Integer		-2147483648 to 2147483647
short	Short Integer		-32768 to 32767
long	Long Integer	I or L	-2147483648 to 2147483647
float	Floating Decimal Number	f or F	1.17549e-038 to 3.40282e+038
double	Double Decimal Number		2.22507e-308 to 1.79769e+308
long double	Long Decimal Number		2.22507e-308 to 1.79769e+308
char	Character		-128 to 127
unsigned int	Unsigned Integer		0 to 4294967295
unsigned short	Unsigned Short Integer		0 to 65535
unsigned long	Unsigned Long Integer		0 to 4294967295
unsigned char	Unsigned Character		0 to 255
bool	True or False		True = 1 and False = 0

แต่เราจะพบว่า ในภาษา C ไม่มีชนิดของตัวแปรที่เป็น String

แล้วถ้าเราอยากจะสร้างตัวแปรที่เป็น String เราจะทำยังไง

เราต้องใช้การกำหนดตัวแปรเป็น Char ร่วมกับการกำหนดขนาดแทน

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char fruit[20] = "Apple";
6
7     printf("%s", fruit);
8
9     return 0;
10 }
```

char variableName[size] = value;

OUTPUT

```
└─(kitton㉿Kitton)-[/mnt/d/boostup-xi/example]
└─$ ./main.exe
Apple
└─(kitton㉿Kitton)-[/mnt/d/boostup-xi/example]
└─$
```

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int myNum = 15;
6     float myFloat = 3.14;
7     char myLetter = 'a';
8     char myString[] = "Hello World";
9
10    printf("myNum = %d\n", myNum);
11    printf("myFloat = %f\n", myFloat);
12    printf("my Letter = %c\n", myLetter);
13    printf("myString = %s", myString);
14
15    return 0;
16 }
```

OUTPUT

```
[kitten@Kitton]-[/mnt/d/boostup-xi/example]
$ ./main.exe
myNum = 15
myFloat = 3.140000
my Letter = a
myString = Hello World
[kitten@Kitton]-[/mnt/d/boostup-xi/example]
$
```

VARIABLES



ตัวแบบ คืออะไร ?

ตัวแปร คือ การจ่องพื้นที่ในหน่วยความจำของคอมพิวเตอร์สำหรับเก็บข้อมูลที่ต้องใช้ในการทำงานของโปรแกรม

การจ่องพื้นที่ คือ ในการกำหนดตัวแปร ชนิดข้อมูลต่างๆ จะมีการกำหนดพื้นที่หน่วยความจำ ต่างกัน โดย

- int ຈະຈອງພື້ນຖານທີ່ 2 byte
 - float ຈະຈອງພື້ນຖານທີ່ 4 byte
 - char ຈະຈອງພື້ນຖານທີ່ 1 byte

สรุปสันๆ ตัวแปรมันเอาไว้เก็บข้อมูลต่างๆในโปรแกรม เช่น ตัวเลข ตัวอักษร ข้อความ

ຕົວຢ່າງຂາດກາຈອນພື້ນຖະຂອງແຕ່ລະບົດຂ້ອມູລ

Type	Description	Size	Domain
char	Signed character/byte. Characters are enclosed in single quotes .	1	-128..127
double	Double precision number	8	ca. $10^{-308}..10^{308}$
int	Signed integer	4	$-2^{31}..2^{31}-1$
float	Floating point number	4	ca. $10^{-38}..10^{38}$
long (int)	Signed long integer	4	$-2^{31}..2^{31}-1$
long long (int)	Signed very long integer	8	$-2^{63}..2^{63}-1$
short (int)	Short integer	2	$-2^{15}..2^{15}-1$
unsigned char	Unsigned character/byte	1	0..255
unsigned (int)	Unsigned integer	4	0.. $2^{32}-1$
unsigned long (int)	Unsigned long integer	4	0.. $2^{32}-1$
unsigned long long (int)	Unsigned very long integer	8	0.. $2^{64}-1$
unsigned short (int)	Unsigned short integer	2	0.. $2^{16}-1$

การสร้างตัวแปร

คือ การสร้างตัวแปร เพื่อใช้งานในโปรแกรมของเรา

type variableName = value;

type - ชนิดของข้อมูล เช่น int, float, char

variableName - ชื่อของตัวแปร

value - ค่าของตัวแปร

การตั้งชื่อตัวแปร

- ต้องขึ้นต้นด้วยตัวอักษร A-Z หรือ a-z หรือเครื่องหมาย _(Underscore) เท่านั้น
- ภายในชื่อตัวแปรสามารถใช้ตัวอักษร A-Z หรือ a-z หรือตัวเลข 0-9 หรือเครื่องหมาย _
- ภายในชื่อห้ามเว้นช่องว่าง หรือใช้สัญลักษณ์ใดๆ ก็ได้หากเป็นสองคำ
- ตัวอักษรเหล่านี้มีความหมายแตกต่างกัน
- ห้ามตั้งชื่อซ้ำกับคำสংวน (Reserved Word) ดังนี้

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

ป. ชื่อตัวแปรควรตั้งให้สื่อความหมายมากที่สุด

การตั้งชื่อตัวแปร

Camel Case : แต่ละคำในชื่อตัวแปร ยกเว้นคำแรก จะขึ้นต้นด้วยตัวอักษรพิมพ์ใหญ่

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char myName[] = "Groot";
6     char myPhone[] = "iPhone";
7     char myPhoneNumber[] = "0999999999";
8     return 0;
9 }
```

Pascal Case : ทุกคำในชื่อตัวแปรจะขึ้นต้นด้วยตัวอักษรพิมพ์ใหญ่

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char MyName[] = "Groot";
6     char MyPhone[] = "iPhone";
7     char MyPhoneNumber[] = "0999999999";
8     return 0;
9 }
```

การตั้งชื่อตัวแปร

Snake Case : คั่นแต่ละคำด้วยเครื่องหมาย Underscore (_)

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char my_name[] = "Groot";
6     char my_phone[] = "iPhone";
7     char my_phone_number[] = "0999999999";
8     return 0;
9 }
```

ตัวอย่างการตั้งชื่อตัวแปรกึ่งแบบ ถูกต้อง และไม่ถูกต้อง

bath_room	ถูกต้อง
n-sync	ผิดหลักการ เนื่องจากมีเครื่องหมาย - ปรากฏในชื่อ
108dots	ผิดหลักการ เนื่องจากขั้นต้นด้วยตัวเลข
Year#	ผิดหลักการ เนื่องจากมีเครื่องหมาย # อยู่ในชื่อ
-good	ถูกต้อง
_1234	ถูกต้อง
goto	ผิดหลักการ เนื่องจากเป็นคำสั่งวน
work	ถูกต้อง
break	ผิดหลักการ เนื่องจากเป็นคำสั่งวน

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int price = 200;
6
7     float rate = 0.5;
8
9     int discount = price * rate;
10
11    printf("Discount is %d THB", discount);
12
13    return 0;
14 }
```

OUTPUT

```
└─(kitton㉿Kitton)-[~/mnt/d/boostup-xi/example]
└─$ ./main.exe
Discount is 100 THB
└─(kitton㉿Kitton)-[~/mnt/d/boostup-xi/example]
└─$
```

จงเขียนคำสั่งประการค่าตัวแปรเพื่อเก็บข้อมูลชนิดต่างๆ ดังต่อไปนี้

1. สร้างตัวแปรชื่อ `a` สำหรับเก็บเลขจำนวนเต็ม 45
2. สร้างตัวแปรชื่อ `b` สำหรับเก็บตัวอักษร 'x'
3. สร้างตัวแปรชื่อ `c` สำหรับเก็บเลขทศนิยม 150.75
4. สร้างตัวแปรชื่อ `d` สำหรับเก็บตัวอักษร "Welcome"
5. สร้างตัวแปรชื่อ `e` สำหรับเก็บค่าตัวเลข 1.5e08

OPERATOR

Arithmetic Operator

Assignment Operator

Relational Operator

Unary Operator

Logical Operator

Arithmetic Operator

Arithmetic Operators

Operator	Context
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Division

Arithmetic Operations on Pointers in C



```
1 #include <stdio.h>
2
3 int main()
4 {
5     int num1 = 10, num2 = 3, result;
6
7     result = num1 + num2;
8     printf("Addition result: %d\n", result);
9
10    result = num1 - num2;
11    printf("Subtraction result: %d\n", result);
12
13    result = num1 * num2;
14    printf("Multiplication result: %d\n", result);
15
16    result = num1 / num2;
17    printf("Division result: %d\n", result);
18
19    result = num1 % num2;
20    printf("Modulus result: %d\n", result);
21
22    return 0;
23 }
```

OUTPUT

```
[kitten@Kitton]~/mnt/d/boostup-xi/example]
$ ./main.exe
Addition result: 13
Subtraction result: 7
Multiplication result: 30
Division result: 3
Modulus result: 1

[kitten@Kitton]~/mnt/d/boostup-xi/example]
$
```

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char num1 = 'A';
6     int num2 = 6;
7
8     printf("Sum of num1 and num2 is %d\n", num1 + num2);
9
10    printf("Difference of num1 and num2 is %d\n", num1 - num2);
11
12    printf("Product of num1 and num2 is %d\n", num1 * num2);
13
14    printf("Division of num1 and num2 is %d\n", num1 / num2);
15
16    printf("Modulus of num1 and num2 is %d\n", num1 % num2);
17
18    return 0;
19 }
```

OUTPUT

```
└─(kitton㉿Kitton)-[/mnt/d/ce-boostup-xi/boostupxi-api]
└─$ ./main.exe
Sum of num1 and num2 is 71
Difference of num1 and num2 is 59
Product of num1 and num2 is 390
Division of num1 and num2 is 10
Modulus of num1 and num2 is 5

└─(kitton㉿Kitton)-[/mnt/d/ce-boostup-xi/boostupxi-api]
└─$
```

Assignment Operator

Assignment Operators

Assignment Operator	Shorthand operation
<code>a = a + b</code>	<code>a += b</code>
<code>a = a - b</code>	<code>a -= b</code>
<code>a = a * b</code>	<code>a *= b</code>
<code>a = a / b</code>	<code>a /= b</code>
<code>a = a % b</code>	<code>a %= b</code>

Assignment Operations on Pointers in C



```
1 #include <stdio.h>
2
3 int main() {
4     int x = 5;
5     int y = 2;
6
7     printf("x = %d\n", x);
8     printf("y = %d\n", y);
9
10    x = y;
11
12    printf("After assignment:\n");
13    printf("x = %d\n", x);
14    printf("y = %d\n", y);
15
16    return 0;
17 }
18
```

OUTPUT

```
└─(kitton㉿Kitton)-[/mnt/d/boostup-xi/example]
$ ./main.exe
x = 5
y = 2
After assignment:
x = 2
y = 2

└─(kitton㉿Kitton)-[/mnt/d/boostup-xi/example]
$
```

```
1 #include <stdio.h>
2
3 int main() {
4     int x = 5, y = 2, z = 10;
5
6     x += 2;
7     printf("x = %d\n", x);
8
9     y -= 1;
10    printf("y = %d\n", y);
11
12    z *= 3;
13    printf("z = %d\n", z);
14
15    x /= 2;
16    printf("x = %d\n", x);
17
18    z %= 4;
19    printf("z = %d\n", z);
20
21    return 0;
22 }
23
```

OUTPUT

```
└─(kitton㉿Kitton)-[/mnt/d/boostup-xi/example]
$ ./main.exe
x = 7
y = 1
z = 30
x = 3
z = 2

└─(kitton㉿Kitton)-[/mnt/d/boostup-xi/example]
$ |
```

Unary Operator

Unary Operators

Operator	Context
<code>++</code>	Increments by 1
<code>--</code>	Decrements by 1

Unary Operations on Pointers in C



```
1 #include <stdio.h>
2
3 int main()
4 {
5     int x = 5, y = 2;
6
7     ++x;
8     printf("x = %d\n", x);
9
10    y++;
11    printf("y = %d\n", y);
12
13    --x;
14    printf("x = %d\n", x);
15
16    y--;
17    printf("y = %d\n", y);
18
19    return 0;
20 }
21
```

OUTPUT

```
└─(kitton㉿Kitton)-[/mnt/d/boostup-xi/example]
$ ./main.exe
x = 6
y = 3
x = 5
y = 2

└─(kitton㉿Kitton)-[/mnt/d/boostup-xi/example]
$
```

จงหาตัวแปร x จากนิพจน์ต่อไปนี้ โดยกำหนดให้ $a = 2$, $b = 3$,
 $c = 4$, $d = 5$, $e = 6$ และ $f = 7$

1. $x = a + e / f * c$
2. $x = (f - e) * (c / a)$
3. $x = a * d / a + e / b$
4. $x = a * (d / (a + e)) / b$
5. $x = a * b - c + e * d$

มีค่าเท่ากับ

มีค่าเท่ากับ

มีค่าเท่ากับ

มีค่าเท่ากับ

มีค่าเท่ากับ

Relational Operator

Relational Operators

Operator	Context
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
==	equal to
!=	not equal to

Relational Operations on Pointers in C



```
1 #include <stdio.h>
2
3 int main()
4 {
5     int x = 5, y = 2, z = 5;
6
7     printf("x == y: %d\n", x == y);
8     printf("x == z: %d\n", x == z);
9
10    printf("x != y: %d\n", x != y);
11    printf("x != z: %d\n", x != z);
12
13    printf("x > y: %d\n", x > y);
14    printf("y > z: %d\n", y > z);
15
16    printf("x >= y: %d\n", x >= y);
17    printf("x >= z: %d\n", x >= z);
18
19    printf("x < y: %d\n", x < y);
20    printf("y < z: %d\n", y < z);
21
22    printf("x <= y: %d\n", x <= y);
23    printf("x <= z: %d\n", x <= z);
24
25    return 0;
26 }
27
```

OUTPUT

```
└─(kitton㉿Kitton)-[~/mnt/d/boostup-xi/example]
└─$ ./main.exe
x == y: 0
x == z: 1
x != y: 1
x != z: 0
x > y: 1
y > z: 0
x >= y: 1
x >= z: 1
x < y: 0
y < z: 1
x <= y: 0
x <= z: 1

└─(kitton㉿Kitton)-[~/mnt/d/boostup-xi/example]
└─$
```

Logical Operator

Operator	Name	Description
&&	Logical and	Returns true if both statements are true
	Logical or	Returns true if one of the statements is true
!	Logical not	Reverse the result, returns false if the result is true

Truth Table of Logic Operators

In C++ boolean **True** is 1 and **False** is 0

a	b	a && b	a b	! a
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int a = 1;
6     int b = 0;
7     int c = 1;
8     int result;
9
10    result = a && b;
11    printf("a && b = %d\n", result);
12
13    result = a || b;
14    printf("a || b = %d\n", result);
15
16    result = !a;
17    printf("!a = %d\n", result);
18
19    result = a == c;
20    printf("a == c = %d\n", result);
21
22    result = a != b;
23    printf("a != b = %d\n", result);
24
25    return 0;
26 }
27
```

OUTPUT

```
└─(kitton㉿Kitton)-[~/mnt/d/boostup-xi/example]
└$ ./main.exe
a && b = 0
a || b = 1
!a = 0
a == c = 1
a != b = 1

└─(kitton㉿Kitton)-[~/mnt/d/boostup-xi/example]
└$
```

จงหาผลลัพธ์การดำเนินการทางคณิตศาสตร์ของบิพจน์ต่อไปนี้ โดยกำหนดตัวแปรให้ดังนี้

int x = 4;

float y = -1.2;

char z = "A";

1. $(x < y) \parallel (z \geq y)$
2. $(y * 100) \&\& (x \leq z)$
3. $(z > x) \parallel (x < y) \&\& (y \leq x)$
4. $(x * y) \&\& (z > y) \parallel (y < x)$
5. $(y == z) \parallel (x > y)$

ผลลัพธ์ที่ได้ คือ
ผลลัพธ์ที่ได้ คือ
ผลลัพธ์ที่ได้ คือ
ผลลัพธ์ที่ได้ คือ
ผลลัพธ์ที่ได้ คือ

THANK YOU



borntoDev