(2016.12.13, Tue)

$f(x) = x+5$: definition

function name

parameter

body

$f(3)$ // use, call $\quad f(3) \equiv 8$

$x+5 \rightarrow 3+5 = 8$

$g(x,y) = \{(\phi, \phi)$ 이외 (x,y) 실수 값 $\}$

$g(2,3) + 3$ $\quad f(3) + 3 = 8+3 = 11$

1, 1.ϕ integer, real number

$1 + 1.\phi = 2$

inf

$h(x,y) = f(x) + y$

header body.

수업내용.

math.h

stdio.h ← int printf(const char* fmt, ...);

stdlib.h

string.h

...

int x;

String (문자열) "..." = char* (characters pointer.)

@platform

Unix

Windows

hello world\n

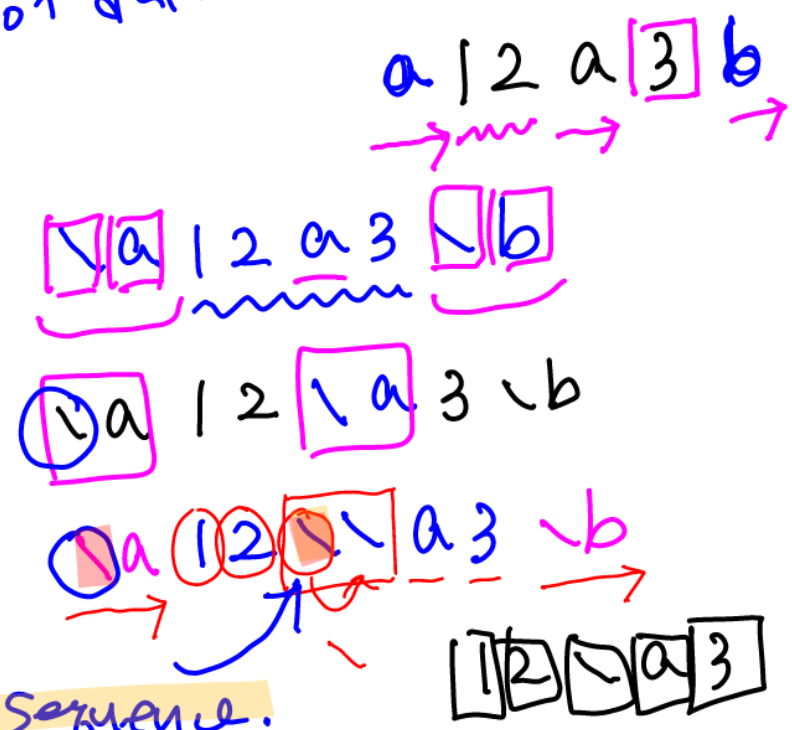
\n

hello world\r\n

printf("...\r\n");

@Escape Sequence

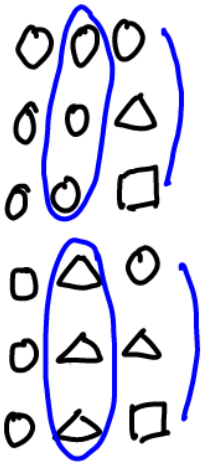
Protocol



@Verbose

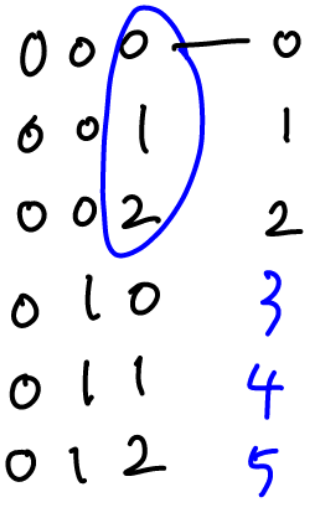
#include < vs. "

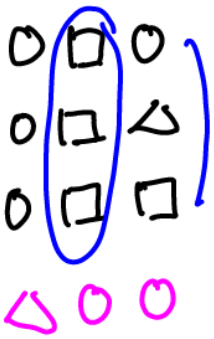
@hexadecimal

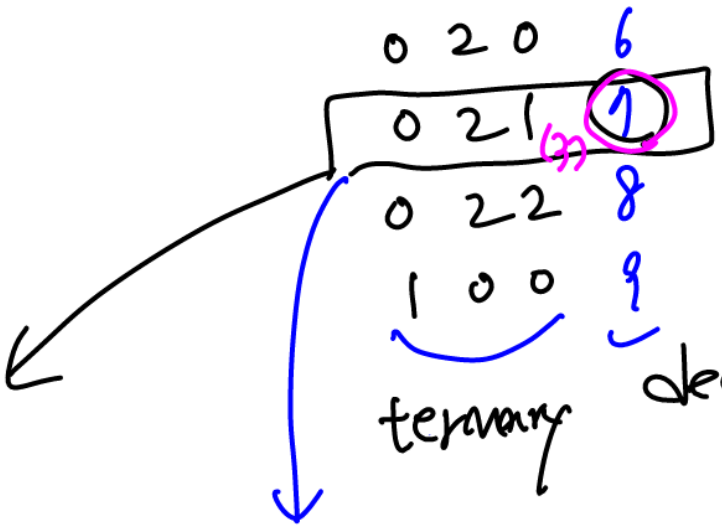


#of symbol, (0, △, □)

3진수, ternary







(page 4)

$123_{(10)}$

$1*10^2+2*10^1+3*10^0$

$=1*100+2*10+3*1$

$=100+20+3$

$021_{(3)}$

$0*3^2+2*3^1+1*3^0$

$=0+6+1$

$=7_{(10)}$

@ Binary Number System.

0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

$1*2^2+0*2^1+1*2^0$

$=4+0+1=5_{(10)}$

$2^4=16$

$2^3=8$

0 0 1 1 0 1 1 0 1 1 1 0

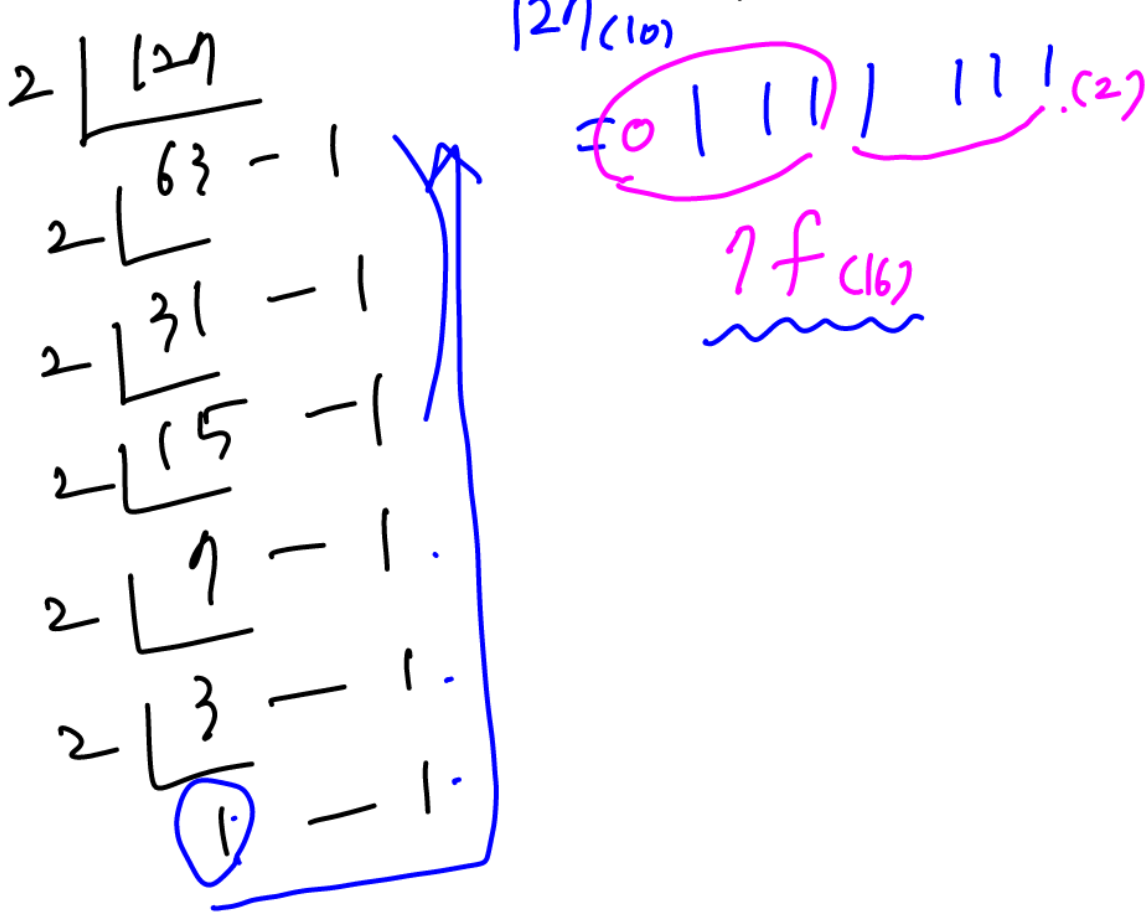
→ 36e

16진수 → 16개

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f

a = 10, b = 11, c = 12, d = 13, e = 14, f = 15

convert decimal → binary number



④ Overflow.

complementary number.

$r^n - m$: definition.

10^1 ① $r = 10$, $n = 1$, $m = 7$

$10^1 - 7 = 3$ 10's complement of 7

72 (10's complement)

$10^2 - 72 = 100 - 72 = 28$

ex) $\chi(-3) \Rightarrow 7 + 7 = 14$

carry

$= 4$

$(+) \rightarrow (-)$

ex) sign bit	decimal	
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1

2's complement of "100"

011

+ 1

100 → 4

101

010

+ 1

011 → 3

unsigned char

int

long long

$0 \sim 7$, $(0 \sim 2^3 - 1)$,

$-2^{3-1} \sim 2^{3-1} - 1$

$-4 \sim +3$

@Byte (8 bits)

@0000 0000

unsigned char

$0 \sim 2^8 - 1 \Rightarrow 0 \sim 255$

signed char

$-2^7 \sim +2^7 - 1 \Rightarrow -128 \sim +127$

char i = 129; → -127

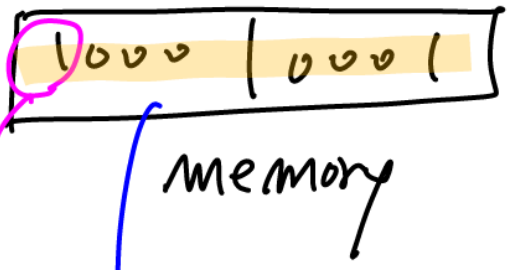
signed

$129_{(10)}$

$0\ 0\ 0\ 0\ 0\ 0\ 0\ 1_{(2)}$

$= 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1$

$8\ 1_{(16)}$



(sign) → −

2's complement

0111 1110

\+ 1

0111 1111

7 f

$= 7 \times 16^1 + 15 \times 16^0$

$= 127$

⇒ -127

* Real Number

$-14_{(10)}^{24}$ → 8bit

0000 . 0000

-14 . 24

fixed point notation.

-14.24

$-0.01424 * 10^{3}$

normalization.

$-0.1424 * 10^{2}$

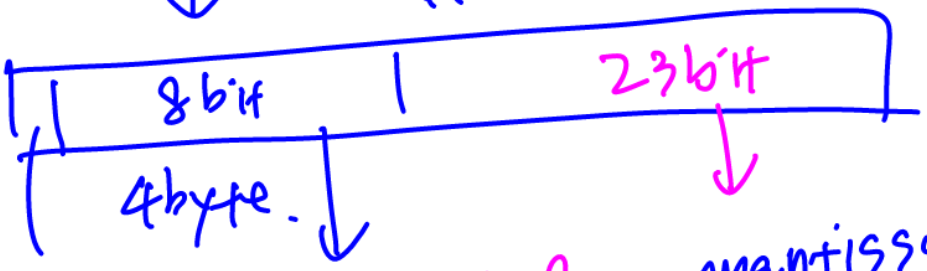
$-1.424 * 10^{1}$

$-14.24 * 10^{0}$

$-142.4 * 10^{-1}$

$-1424.0 * 10^{-2}$

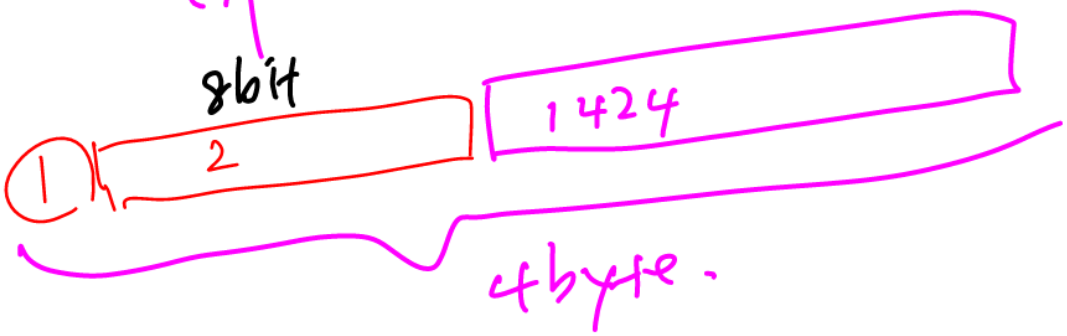
...



sign

exponential

mantissa



-14.24

14 (10) → 1110

2 | 14
2 | 7 ... 0
2 | 3 ... 1
2 | 1 ... 1
1 ... |

0.24 × 2 = 0.48 → 0
0.48 × 2 = 0.96 → 0
0.96 × 2 = 1.92 → 1
0.92 × 2 = 1.84 → 1
0.84 × 2 = ...

0011 1101 0111 0000 1010

14.24

1110. 0011 1101 0111 0000 1010

0.1110 ...

1) 1100 0111 × 2^3

3 = e - 127

e = 130

(-1)^s . (1.m) 2^(e-127)

-14.24

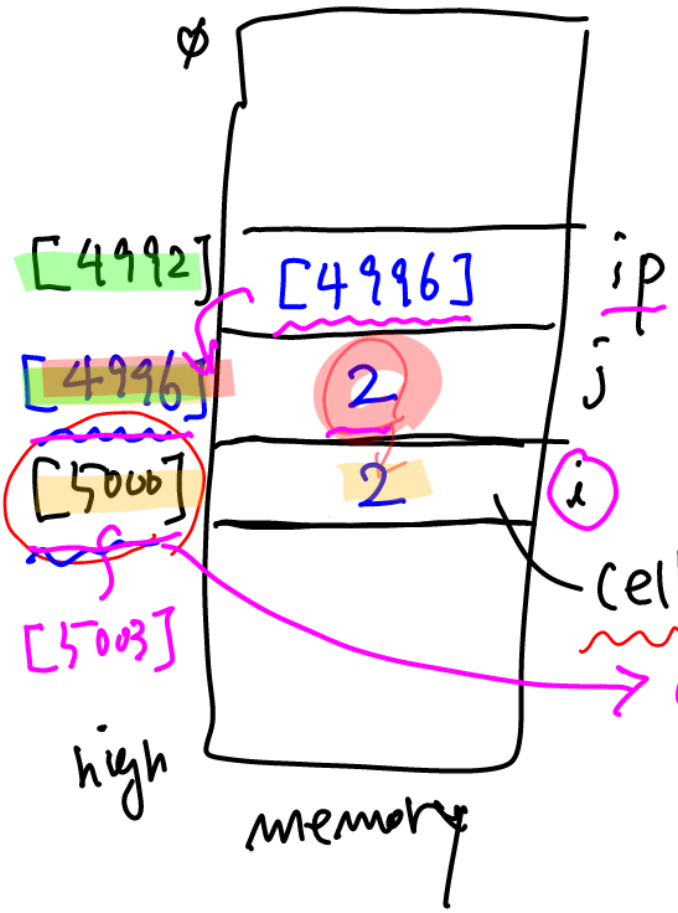
sign	exponential	mantissa
1	1000 0010	1100 0111 1010 1110 0001 010
1 bit	8 bits	23 bits
1	1	6 ...

C 163 D1 ΦA

floating point notation.

* Pointer.

RAM




```
int i; // garbage.
int j = 2;
```

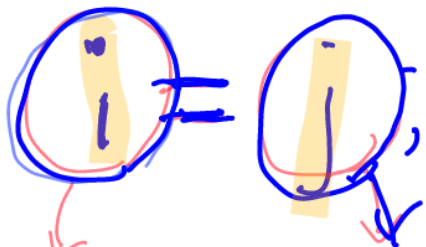
[4996] ~ [4999]

5000 ⇒ [5000]

```
i = j;
```

[5000] = *[4996];

```
printf("%i\n", i);
// 2
```



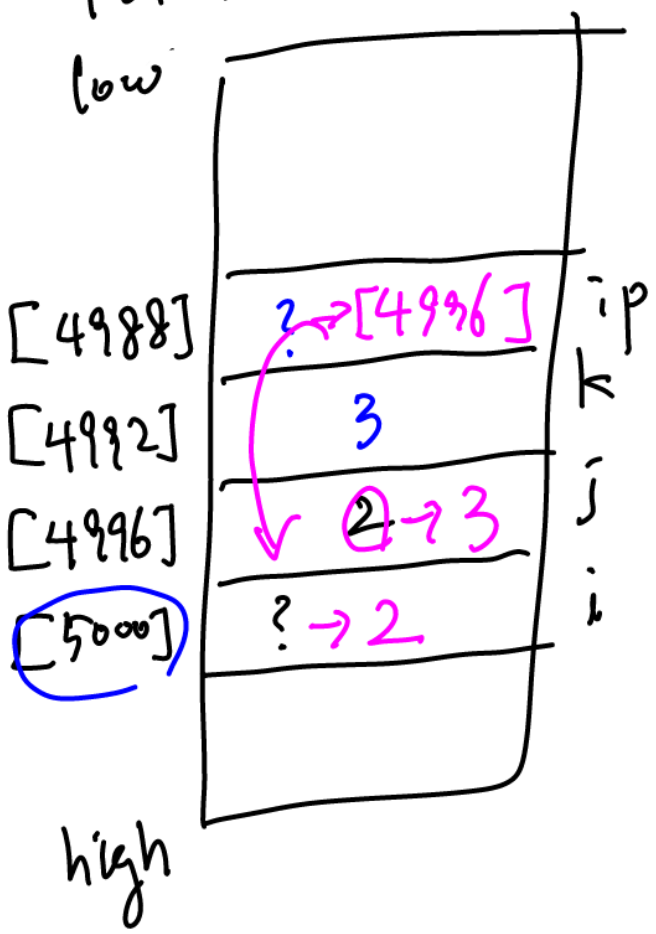
[5000] *[4996]

left-value (lvalue)

right value (rvalue)

ip = &j; ?

*Pointer2



i = j; // [5000] ← *[4996]

j = k; // [4996] ← *[4992]

ip = &j; // [4988] ← [4996]

printf("%i, %p, %p, %p"

*ip, ip, &ip, &j

*[4996] [4996] [4988] [4996]

⇒ 3

*Pointer3

~~string~~ string literal

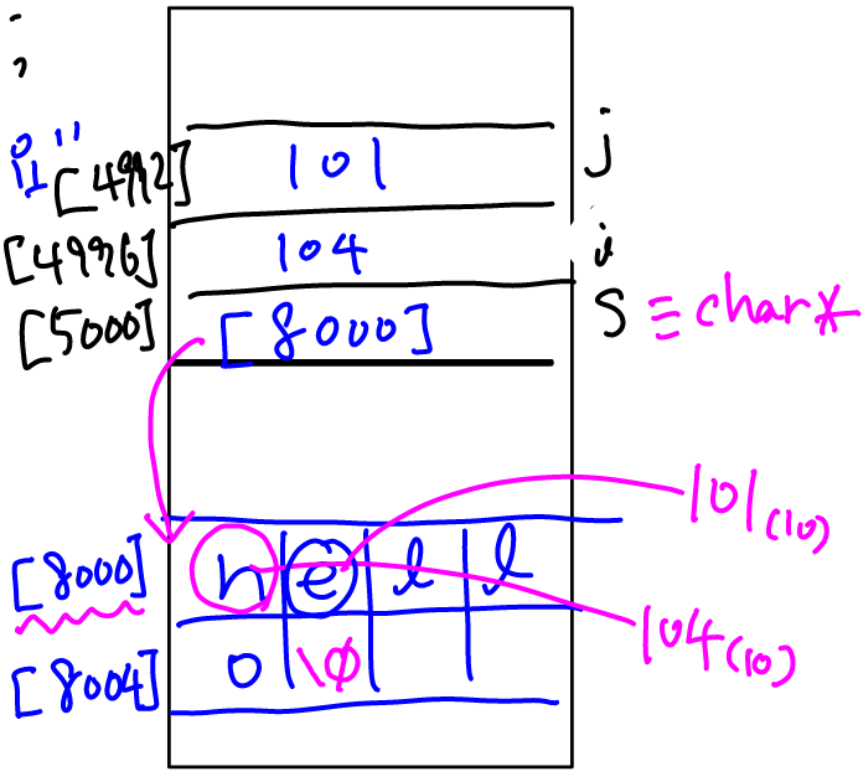
S = "hello";

// "안녕하세요"

char * S;

S = "hello";

// [8000]



*(S+n) // S: pointer, n: integer

= S[n]

*(n+S) → n[S]

[ ] operator: contents access, [ ]

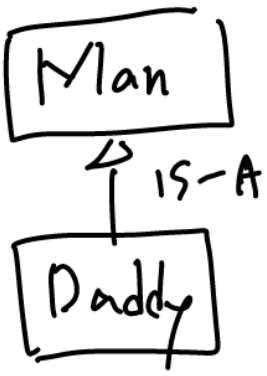
---

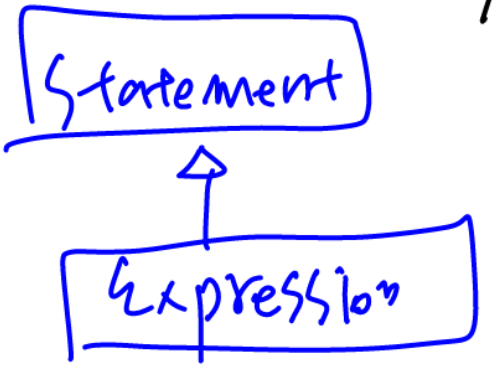
* Statement vs. Expression.

Man vs. daddy.

daddy is a Man.

~~Man is a daddy.~~





Expression is a Statement.

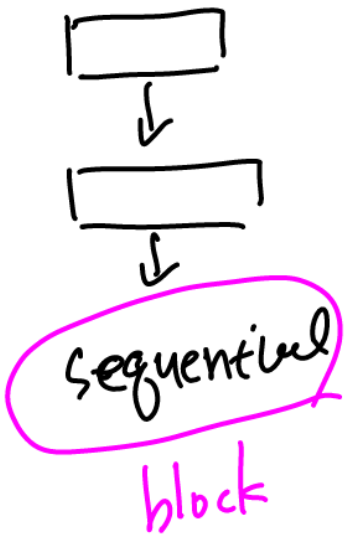
```
2+3 ;  // 5
2>3 ;  // false.
```

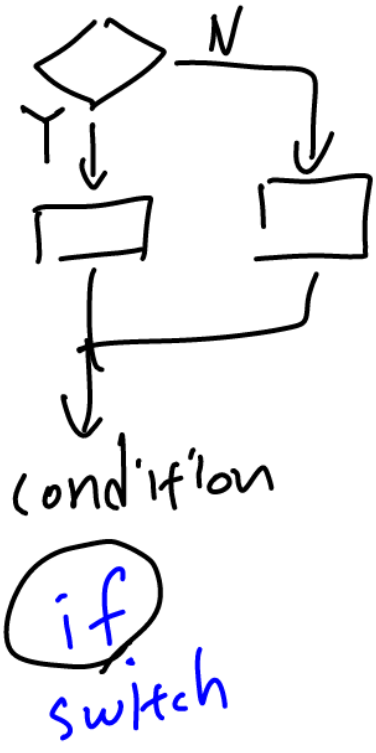
in C, no boolean expression

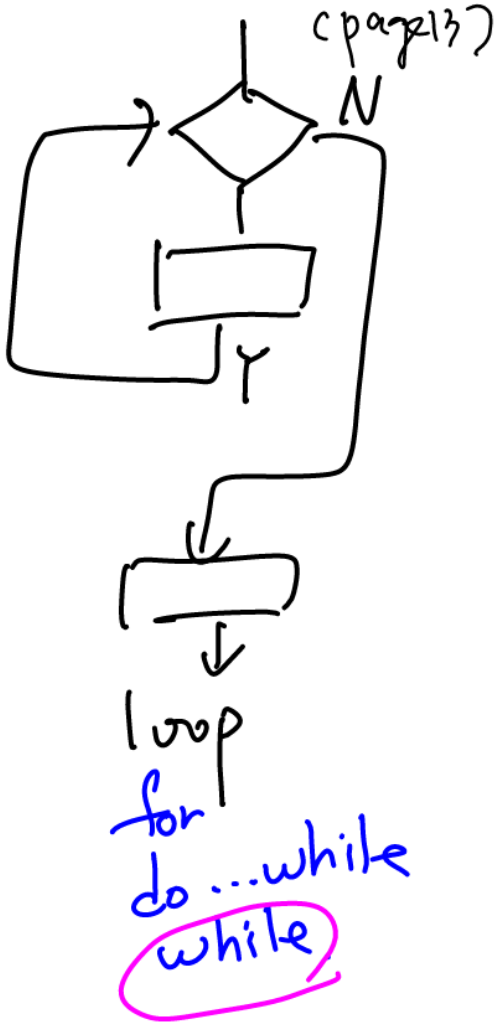
↓

c++ → integer expression (true, false)

* Control Structor.







* if. statement syntax.

```
if ( 조건1 )
    문장1;
[else if ( 조건2 )
    문장2; ][ ... ]
[else
    문장3; ]
```

* while statement

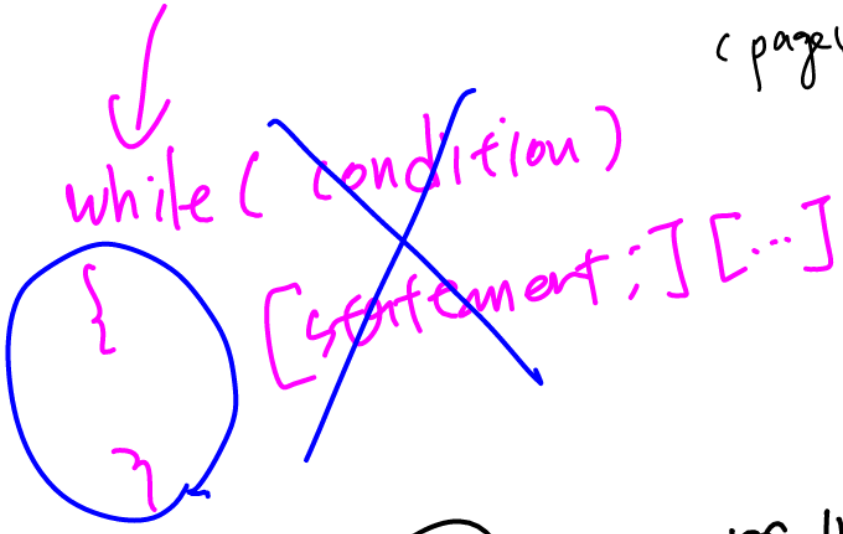
```
while ( condition )
    statement;
```

* sequential statement

```
statement1;
statement2;
statement3;
```

```
block {
    statement1;
    statement2;
    ...
}
```

1 statement



* Scope rule.

block: nearest block

refers

:: scope resolver

* arithemaric operator

bitwise op.

assignment op.

relational op

...

unary sign operator + -

binary operator + - * /

unary increment operator ++

decrement " --

modulo op. %.

```
i = i+(++j) - (k--) +(j++) +k;
  j=j+1; // j=1
i = i + j -k + j +k; // i = 2
```

k=k-1; // -1=k
j=j+1; // j=2

i=2, j=2, k=-1

* assignment operator

=
+=
-=
*=
/=
%=

i = j + remainder part

i += remainder part

* expression

2+3 ==5

i=1; // i=1

j = (i=1);

* comma ,

sizeof

i= expression1, exp2, exp3

* logical operator

(page16)

! not	
F	T
T	F

|| OR	F	T
F	f	t
T	t	t

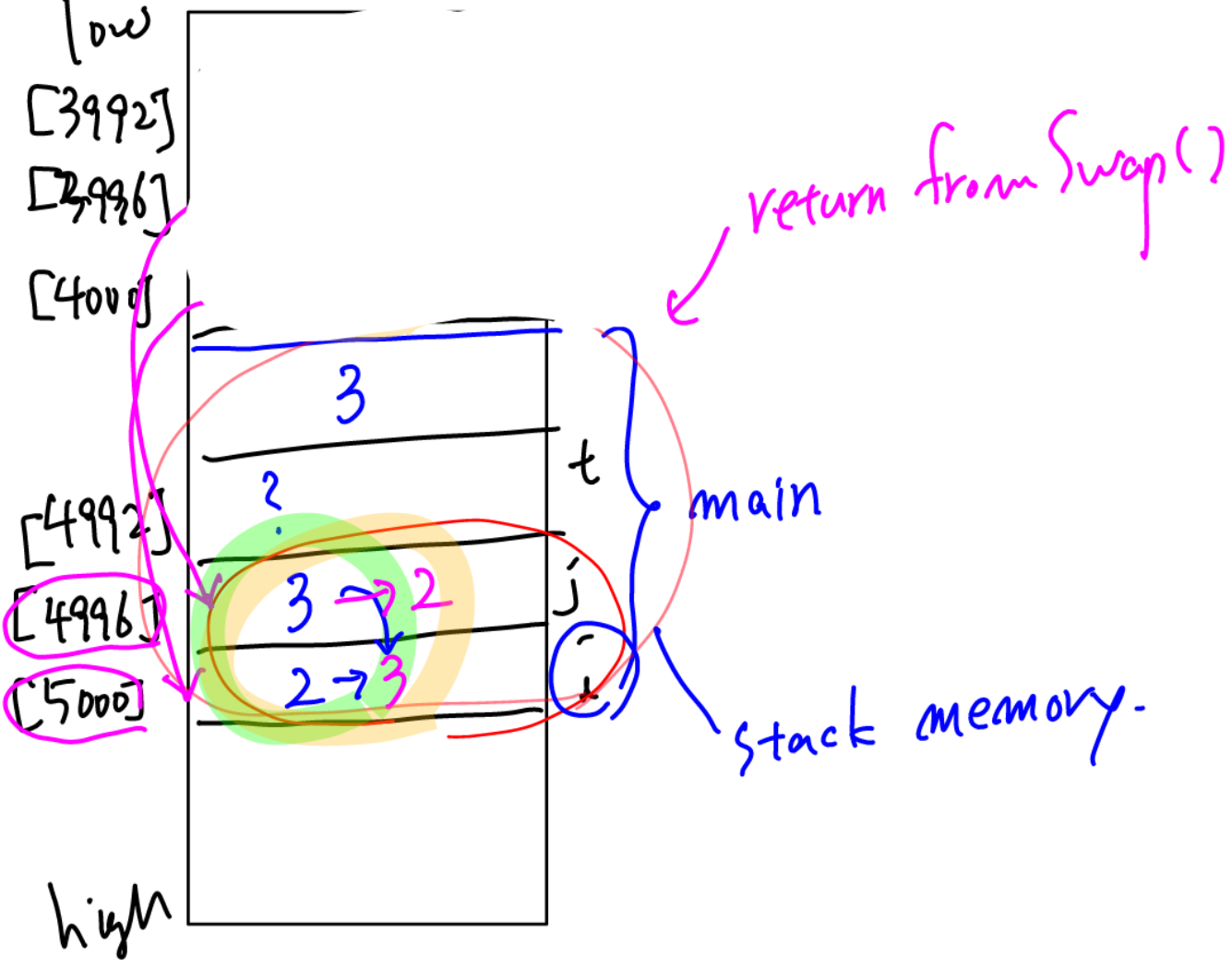
&& AND	f	t
f	f	f
t	f	t

* casting operator.

* & operator.

&: bitwise and operator.
: address-of op.

* Swap()



* meaning of pointer arithemetic

int* ip; //
... // [5000]

ip+1 ≡ [5004]

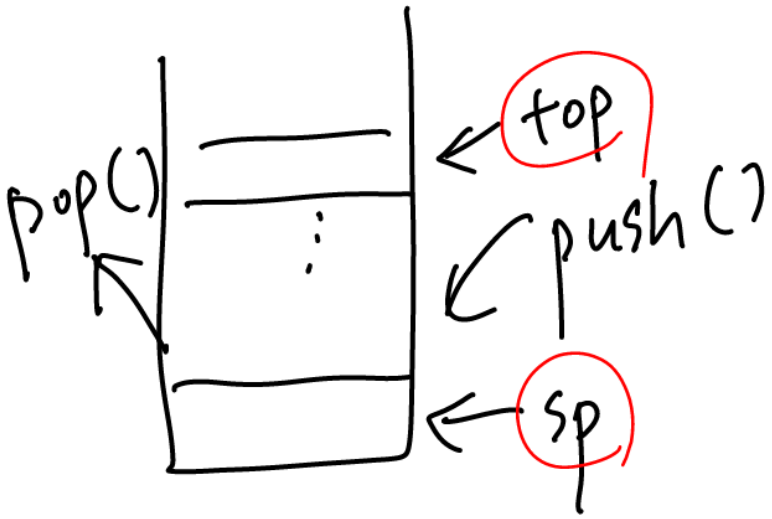
ip + n ⇒ ip + sizeof(*ip) x n

ip+2 →? [5000] + sizeof(*ip) x 2
4
+ 8 ⇒ [5008]

char* cp; //[5000] cp+1

int* ip; //[5000] ip+1

* stack. (data structure)

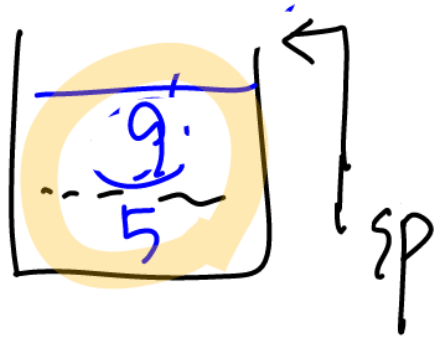


Last In First Out

(LIFO)

ex) push(5)
push(3)
pop()
push(9)

3, | 9 |
| 5 |



* Block

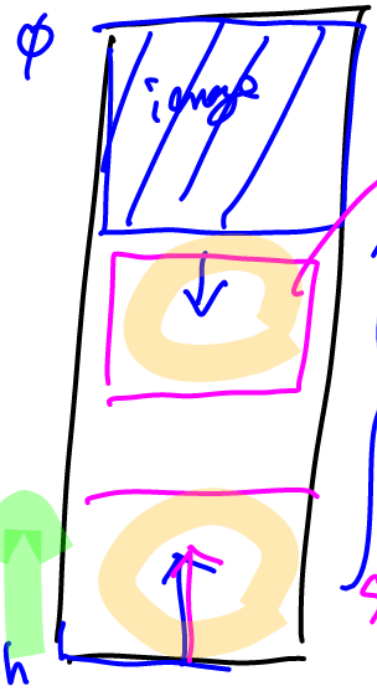
A {

int i;
int j;

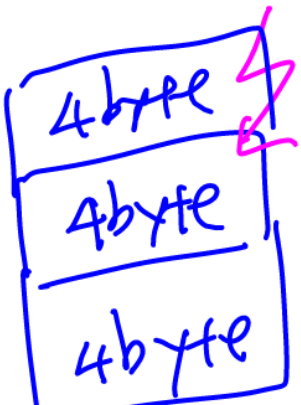
B { int i;
int k;
}

}

5000-4996
= 4



int i;
int j;



(2016.12.14)