What is C Programming

Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

C언어는 왜 배울까?

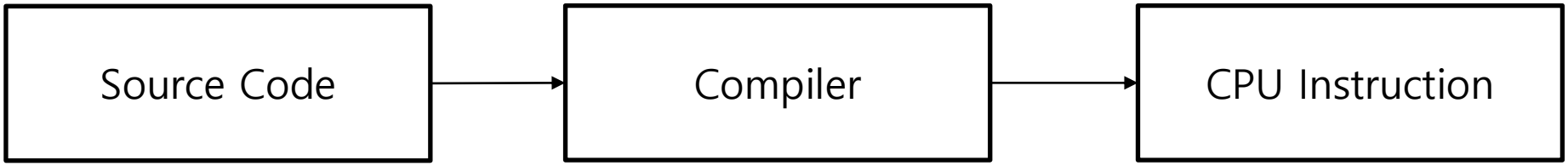
인간은 소통의 언어가 있다.
기계도 기계어라는 소통의 언어가 있다.

인간이 배우기에 기계어는 너무 어렵다.

그래서, 중간에 소통하는 녀석이 필요하다.

그것이 바로 '_____'이다.

그럼, Compiler는 뭘하는 걸까?

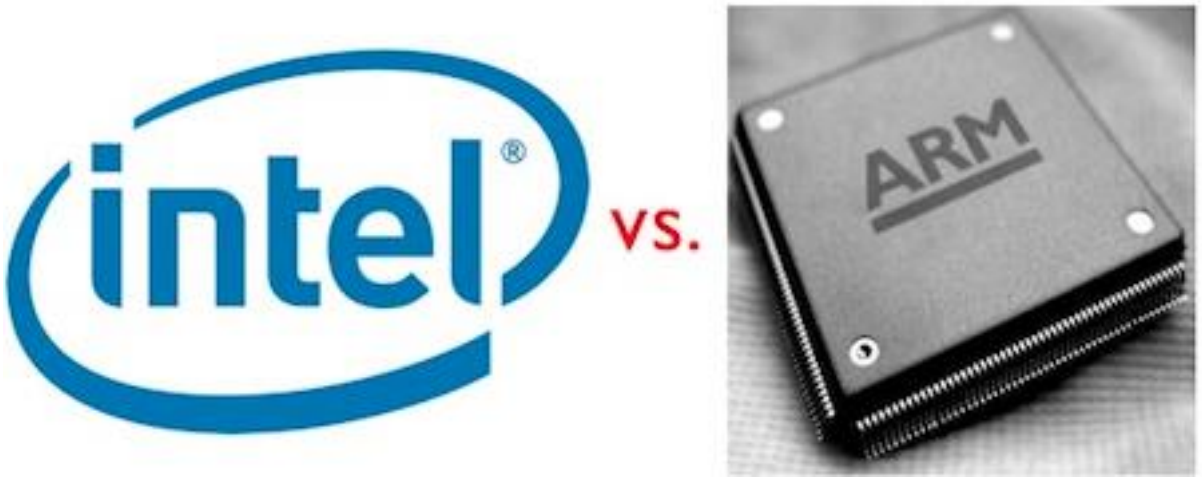


CPU Instruction이란 ?

각 CPU가 사용하는 기계어 방식을

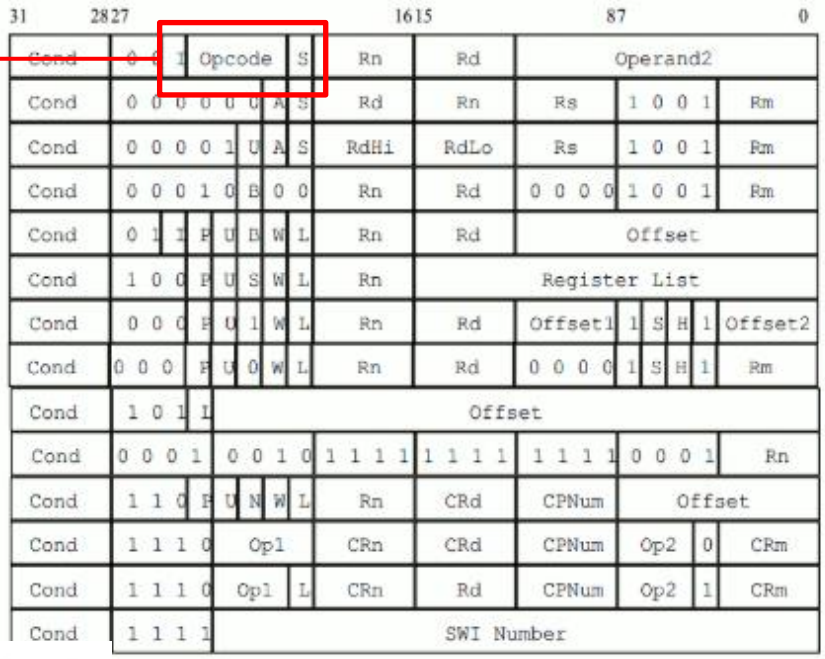
CPU Instruction이라고 하고

이를 ISA (Instruction Set Architecture) 라고 부른다.



ARM ISA EXAMPLE

ARM Instruction Set Format



Instruction type

- Data processing / PSR Transfer
- Multiply
- Long Multiply **(v3M / v4 only)**
- Swap
- Load/Store Byte/Word
- Load/Store Multiple
- Halfword transfer : Immediate offset **(v4 only)**
- Halfword transfer: Register offset **(v4 only)**
- Branch
- Branch Exchange **(v4T only)**
- Coprocessor data transfer
- Coprocessor data operation
- Coprocessor register transfer
- Software interrupt

Cond: Condition field

0000	EQ (EQual)
0001	NE (NEver)
0010	CS (Carry Set)
0011	CC (Carry Clear)
0100	MI (MInus)
0101	PL (PLus)
0110	VS (oVerflow Set)
0111	VC (oVerflow Clear)
1000	HI (HIgher)
1001	LS (Lower or Same)
1010	GE (Greater or Equal)
1011	LT (Less Than)
1100	GT (Greater Than)
1101	LE (Less than or Equal)
1110	AL (ALways)
1111	NV (NeVer)

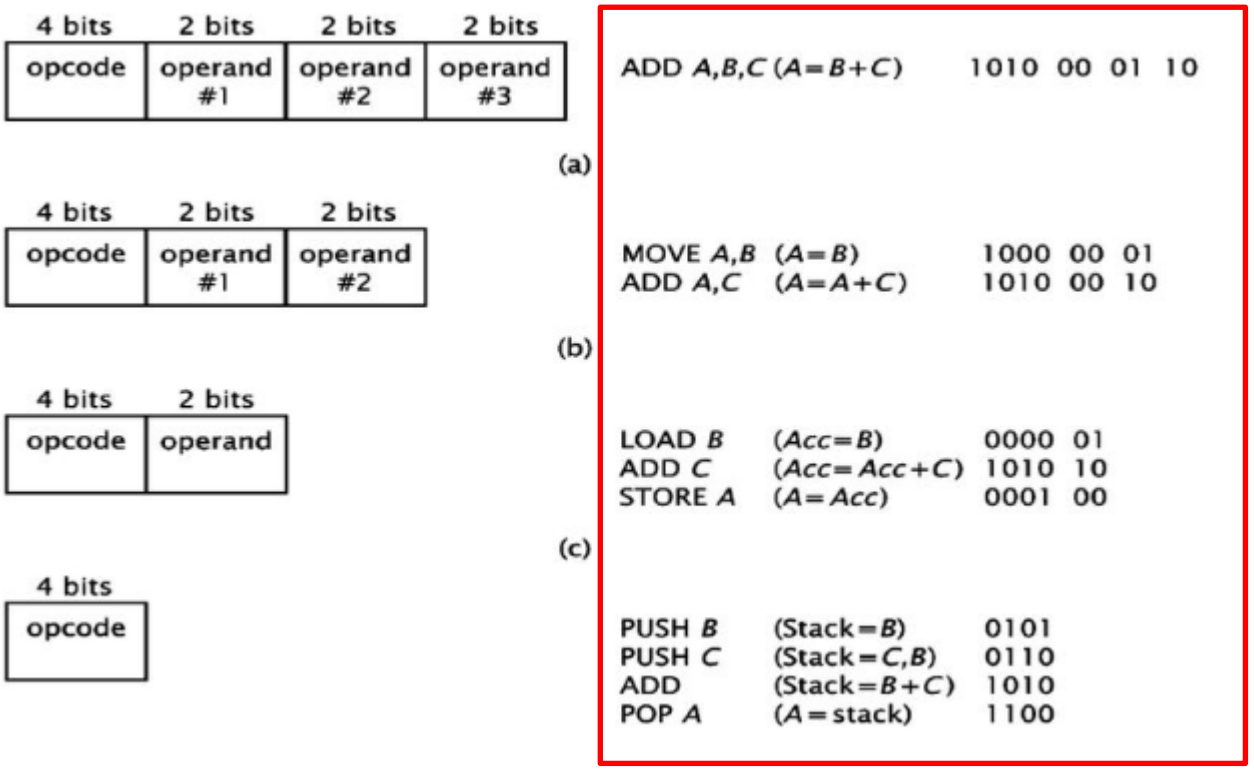
OpCode: Operation code

0000	AND
0001	EOR
0010	SUB
0011	RSB
0100	ADD
0101	ADC
0110	SBC
0111	RSC
1000	TST
1001	TEQ
1010	CMP
1011	CMN
1100	ORR
1101	MOV
1110	BIC
1111	MVN

ISA를 또 언어로 만든 게 어셈블리어?

기계어와 1:1 대응이 되는 언어이며,

각 CPU 마다 지원하는 방식이 다르다.



assembly EXAMPLE

x86	ARM
pop eax	pop {r0}
mov eax, ebx	mov r0, r1
add eax, ebx	add r0, r0, r1
add eax, 0x10	add r0, #16
mov eax, [ebx]	ldr r0, [r1]
mov [eax+0x10], ebx	str r1, [r0, #16]
call eax	blx r0
jmp eax	bx r0
call function	bl function (return address in lr)
ret	pop {pc} / bx lr
int 0x80	svc 0x80 / svc 0x0

아쉽지만, 우리가 지금 어셈블리를 공부할 건 아니니까... 여기까지

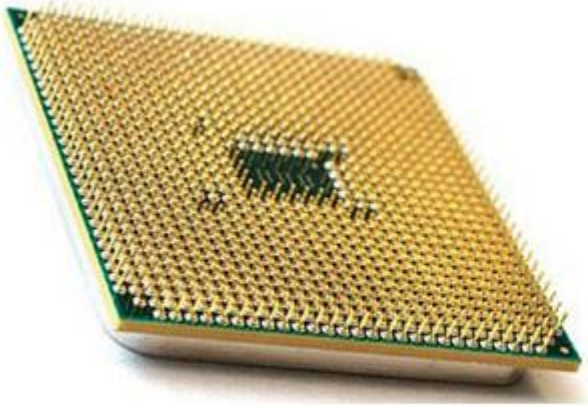
C언어가 왜 필요한지는 알았다.

임베디드 시스템의 기초가 되는

앞으로 배우게 될 Microcontroller는 뭘까?

뭐가 다르긴 다른거 같은데?

특정 목적?



VS

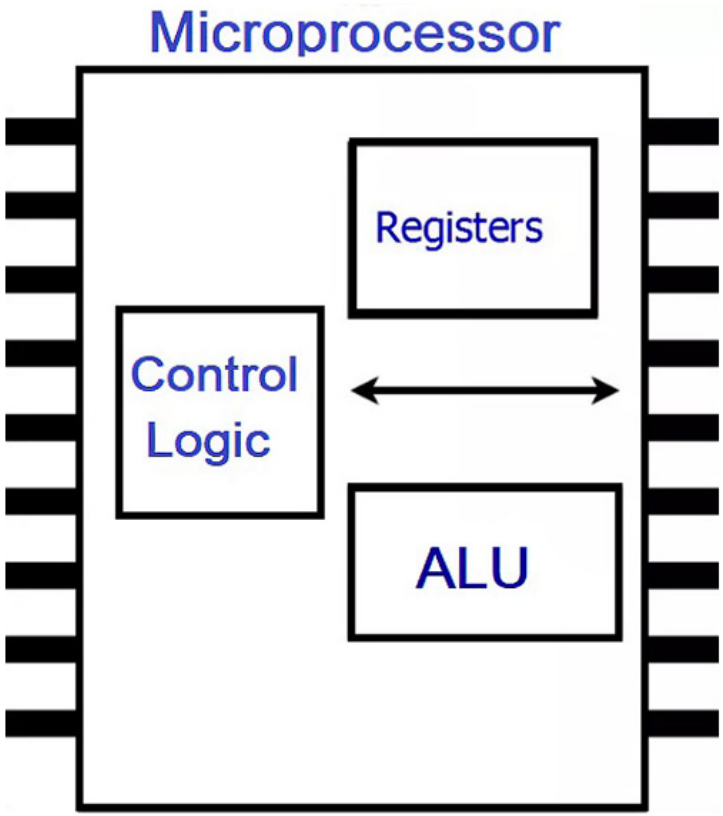
Microprocessor

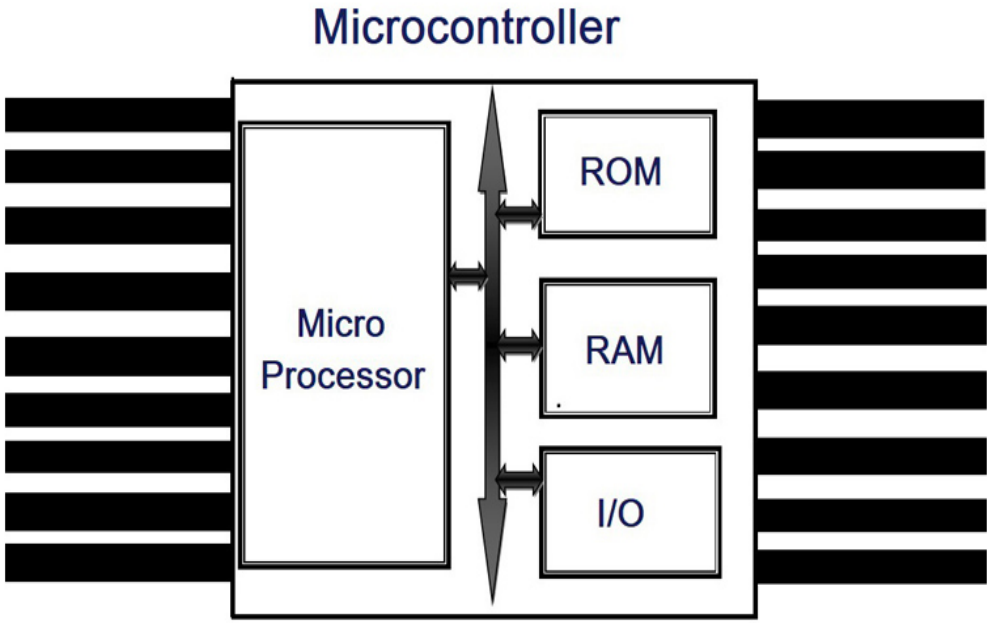
컴퓨터, 모바일, 비디오 게임, TV

Microcontroller

세탁기, 전자레인지, 청소기, 전자기기

내부 구조


Microprocessor
Registers
Control
Logic
ALU


Microcontroller
Micro
Processor
ROM
RAM
I/O

Today`s HW

1) 오늘 공부한 내용 정리

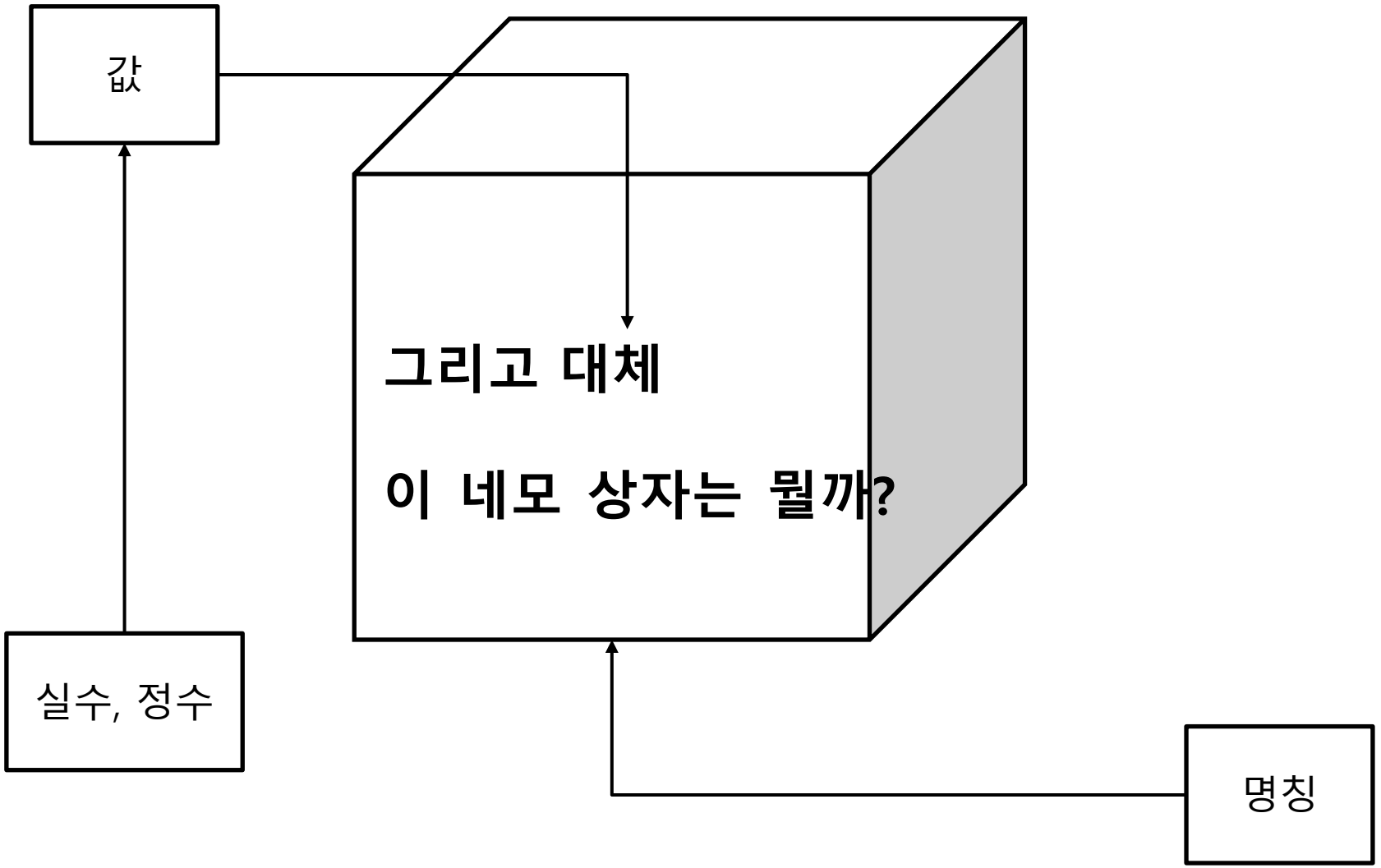
2) 폰노이만 구조와 하버드 구조에 대해서 공부해오기

C Programming
Variable

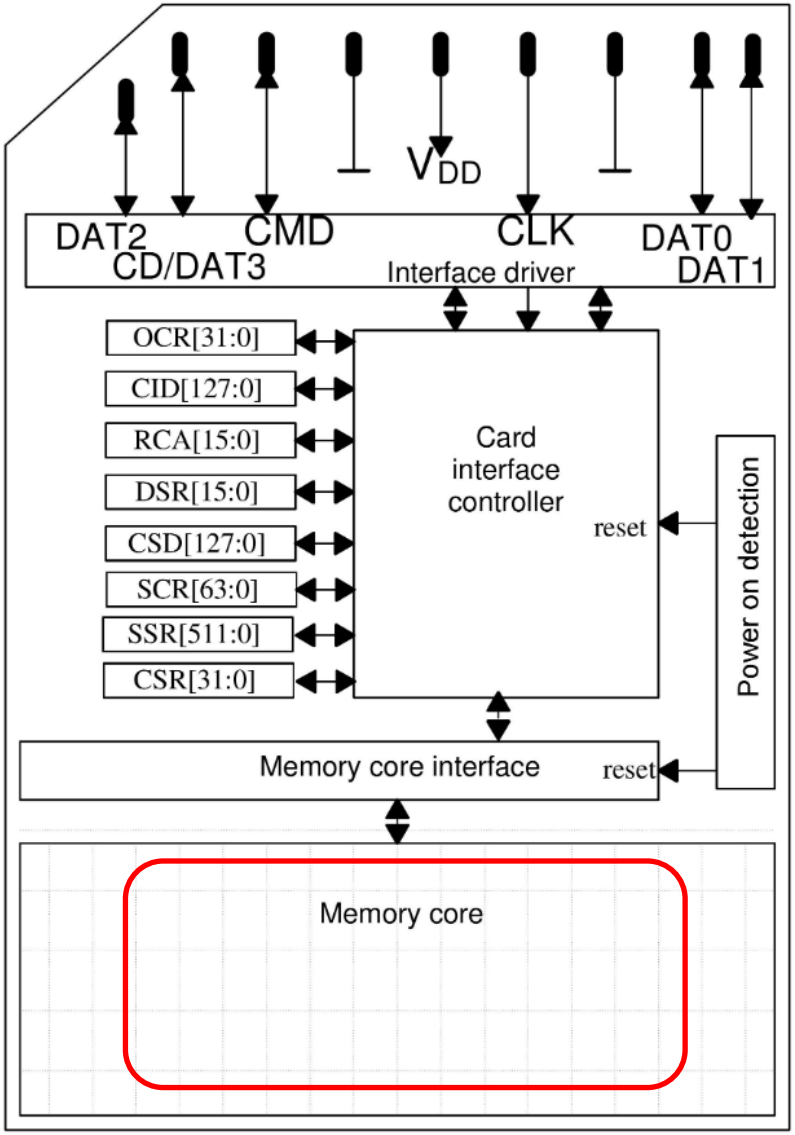
Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

변수는 왜 배워야 할까?



SD CARD 메모리

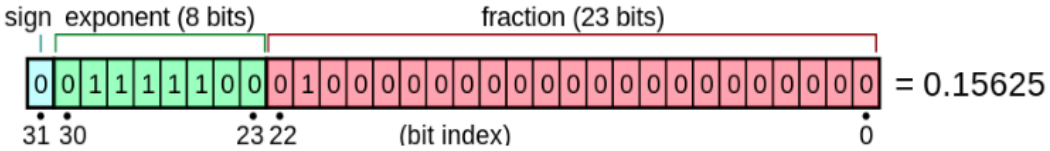

VDD
DAT2
CD/DAT3
CMD
CLK
DAT0
DAT1
Interface driver
OCR[31:0]
CID[127:0]
RCA[15:0]
DSR[15:0]
CSD[127:0]
SCR[63:0]
SSR[511:0]
CSR[31:0]
Card interface controller
reset
Power on detection
Memory core interface
reset
Memory core

사전적 변수 정의

: 우리가 쉽게 프로그램을 조작할 수 있도록 만든 메모리 영역의 지정된 이름일 뿐이다.
C의 각 변수에는 특정 유형이 있다. (Data Type)
이러한 유형은 변수 메모리의 크기와 레이아웃을 결정한다.
해당 메모리 내에 저장될 수 있는 값의 범위와 변수에 적용될 수 있는 일련의 연산에 대해 표시해주는 것이다.
변수의 이름은 문자, 숫자 및 밑줄 문자로 구성 될 수 있으며, 반드시 문자 나 밑줄로 시작 해야 한다.
C는 대소 문자를 구분하므로 대문자와 소문자를 구별해서 사용해야 한다.

Types & Description
• Char 1byte로 Integer type이다.
• int 가장 보편적인 기계어의 integer size type이다.
• float single-precision(32비트 표시형) 부동소수점 값
• double Double-precision (64비트 표시형) 부동소수점 값
• void 아무 Type도 아님을 표시

Single / Double Precision



The real value assumed by a given 32-bit *binary32* data with a given *sign*, biased exponent *e* (the 8-bit unsigned integer), and a *23-bit fraction* is

$$(-1)^{b_{31}} \times 2^{(b_{30}b_{29}\ldots b_{23})_2 - 127} \times (1.b_{22}b_{21}\ldots b_0)_2,$$

which yields

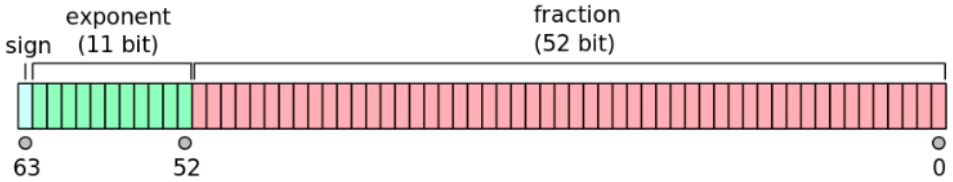
$$\text{value} = (-1)^{\text{sign}} \times 2^{(e-127)} \times \left(1 + \sum_{i=1}^{23} b_{23-i}2^{-i}\right).$$

In this example:

- $\text{sign} = b_{31} = 0$,
- $(-1)^{\text{sign}} = (-1)^0 = +1 \in \{-1, +1\}$,
- $e = b_{30}b_{29}\ldots b_{23} = \sum_{i=0}^{7} b_{23+i}2^{+i} = 124 \in \{1, \ldots, (2^8 - 1) - 1\} = \{1, \ldots, 254\}$,
- $2^{(e-127)} = 2^{124-127} = 2^{-3} \in \{2^{-126}, \ldots, 2^{127}\}$,
- $1.b_{22}b_{21}\ldots b_0 = 1 + \sum_{i=1}^{23} b_{23-i}2^{-i} = 1 + 1 \cdot 2^{-2} = 1.25 \in \{1, 1 + 2^{-23}, \ldots, 2 - 2^{-23}\} \subset [1; 2 - 2^{-23}] \subset [1; 2)$.

thus:

- $\text{value} = (+1) \times 2^{-3} \times 1.25 = +0.15625$.



기본적인 C언어 기본 골격

```
int main(void)
{
        return 0;
}
```

printf는 화면에 글자 출력

```
#include <stdio.h>

int main(void)
{

        printf("Hello C Programming₩n");
        return 0;
}
```

scanf는 키보드 글자 입력

```
int main(void)
{
        scanf( "%d", &변수);
}
```

기본 예제

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <float.h>


int main(void)
{
        int a = 0;
        char b = 0;
        double c = 0;
        float d = 1.1;

        printf("%d %c %f %f", a,b,c,d);

        scanf("%d %c %lf %f", &a,&b,&c,&d);

        printf("%d %c %f %f", a,b,c,d);

        return 0;
}
```

```
gcc -o main var.c
./main
```

Printf 포맷

서식문자	자료형	출력 형태
%d	char, int	부호 있는 10진수 정수
%hd	short	부호 있는 10진수 정수
%ld	long	부호 있는 10진수 정수
%lld	long long	부호 있는 10진수 정수
%u	unsigned int	부호 없는 10진수 정수
%o	unsigned int	부호 없는 8진수 정수
%x, %X	unsigned int	부호 없는 16진수 정수
%f	float, double	10진수 방식의 부동소수점 실수
%Lf	long double	10진수 방식의 부동소수점 실수
%e, %E	float, double	e, E 방식의 부동소수점 실수
%g, %G	float, double	값에 따라 %f와 %e 사이에서 선택
%c	char, short, int	값에 대응하는 문자
%s	char *	문자열
%p	void *	포인터의 주소 값

Scanf 포맷

서식문자	자료형	입력 형태
%d	char, int	부호 있는 10진수 정수
%hd	short	부호 있는 10진수 정수
%ld	long	부호 있는 10진수 정수
%lld	long long	부호 있는 10진수 정수
%u	unsigned int	부호 없는 10진수 정수
%o	unsigned int	부호 없는 8진수 정수
%x, %X	unsigned int	부호 없는 16진수 정수
%f, %e, %g	float	10진수 방식의 부동소수점 실수
%lf	double	10진수 방식의 부동소수점 실수
%Lf	long double	10진수 방식의 부동소수점 실수

HW1

C의 결과값은? 그리고 왜 126이 아닌 걸까?

```c
#include<stdio.h>

int main(void)
{
        char c = 125;
        c = c + 10;

        printf("%d\n",c);
        return -1;
}
```

```
gcc -o ovf overflow.c
./ovf
```

C Programming
Datatype / Type Casting

Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

Datatype

: 데이터타입은 변수의 유형에 따라
메모리에 차지하는 공간의 양과 저장된 비트 방식을 결정한다.

<integer types>

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	-9223372036854775808 to 9223372036854775807
unsigned long	8 bytes	0 to 18446744073709551615

<Floating-Point types>

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

<integer types>

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <float.h>

int main(int argc, char** argv) {

    printf("CHAR_BIT    :   %d\n", CHAR_BIT);
    printf("CHAR_MAX    :   %d\n", CHAR_MAX);
    printf("CHAR_MIN    :   %d\n", CHAR_MIN);
    printf("INT_MAX     :   %d\n", INT_MAX);
    printf("INT_MIN     :   %d\n", INT_MIN);
    printf("LONG_MAX    :   %ld\n", (long) LONG_MAX);
    printf("LONG_MIN    :   %ld\n", (long) LONG_MIN);
    printf("SCHAR_MAX   :   %d\n", SCHAR_MAX);
    printf("SCHAR_MIN   :   %d\n", SCHAR_MIN);
    printf("SHRT_MAX    :   %d\n", SHRT_MAX);
    printf("SHRT_MIN    :   %d\n", SHRT_MIN);
    printf("UCHAR_MAX   :   %d\n", UCHAR_MAX);
    printf("UINT_MAX    :   %u\n", (unsigned int) UINT_MAX);
    printf("ULONG_MAX   :   %lu\n", (unsigned long) ULONG_MAX);
    printf("USHRT_MAX   :   %d\n", (unsigned short) USHRT_MAX);

    return 0;
}
```

```
$gcc -o main *.c

$main

CHAR_BIT    :   8
CHAR_MAX    :   127
CHAR_MIN    :   -128
INT_MAX     :   2147483647
INT_MIN     :   -2147483648
LONG_MAX    :   9223372036854775807
LONG_MIN    :   -9223372036854775808
SCHAR_MAX   :   127
SCHAR_MIN   :   -128
SHRT_MAX    :   32767
SHRT_MIN    :   -32768
UCHAR_MAX   :   255
UINT_MAX    :   4294967295
ULONG_MAX   :   18446744073709551615
USHRT_MAX   :   65535
```

<Floating-Point types>

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <float.h>

int main(int argc, char** argv) {

    printf("Storage size for float : %d \n", sizeof(float));
    printf("FLT_MAX     :   %g\n", (float) FLT_MAX);
    printf("FLT_MIN     :   %g\n", (float) FLT_MIN);
    printf("-FLT_MAX    :   %g\n", (float) -FLT_MAX);
    printf("-FLT_MIN    :   %g\n", (float) -FLT_MIN);
    printf("DBL_MAX     :   %g\n", (double) DBL_MAX);
    printf("DBL_MIN     :   %g\n", (double) DBL_MIN);
    printf("-DBL_MAX     :  %g\n", (double) -DBL_MAX);
    printf("Precision value: %d\n", FLT_DIG );

    return 0;
}
```

```
$gcc -o main *.c

$main

Storage size for float : 4
FLT_MAX     :   3.40282e+38
FLT_MIN     :   1.17549e-38
-FLT_MAX    :   -3.40282e+38
-FLT_MIN    :   -1.17549e-38
DBL_MAX     :   1.79769e+308
DBL_MIN     :   2.22507e-308
-DBL_MAX     :  -1.79769e+308
Precision value: 6
```

Type Casting

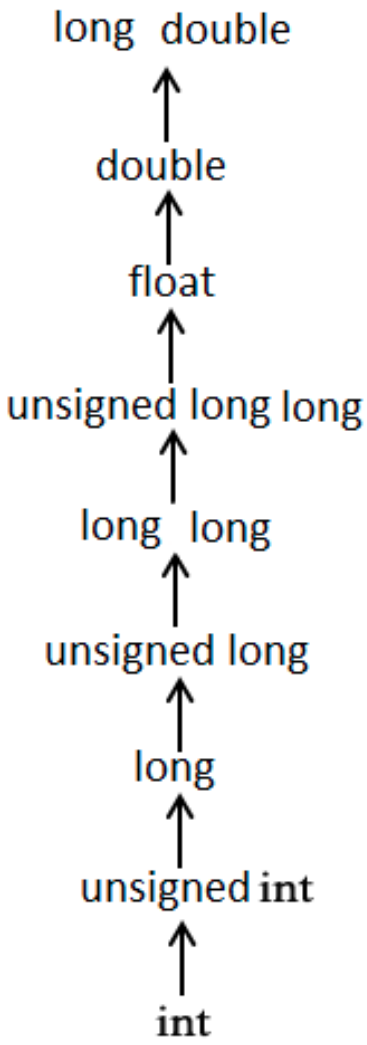
: 기존 데이터 타입을 다른 데이터 타입으로 바꾸는 것을 말한다.

<type casting integer to floating >

```c
#include <stdio.h>

int main() {

   int sum = 17, count = 5;
   double mean;

   mean = (double) sum / count;
   printf("Value of mean : %f\n", mean );

   return -1;
}

```

```
$gcc -o main *.c

$main

Value of mean : 3.400000
```

<usual arithmetic conversions>



<type casting character to integer >

```c
#include <stdio.h>

int main() {

   int  i = 17;
   char c = 'c'; /* ascii value is 99 */
   int sum;

   sum = i + c;
   printf("Value of sum : %d\n", sum );

   return -1;
}
```

```
$gcc -o main *.c

$main

Value of sum : 116
```

```c
#include <stdio.h>

int main() {

   int  i = 17;
   char c = 'c'; /* ascii value is 99 */
   float sum;

   sum = i + c;
   printf("Value of sum : %f\n", sum );

   return -1;

}
```

```
$gcc -o main *.c

$main

Value of sum : 116.000000
```

<stdint.h>

```
typedef signed char        int8_t;
typedef short              int16_t;
typedef int                int32_t;
typedef long long          int64_t;
typedef unsigned char      uint8_t;
typedef unsigned short     uint16_t;
typedef unsigned int       uint32_t;
typedef unsigned long long uint64_t;

typedef signed char        int_least8_t;
typedef short              int_least16_t;
typedef int                int_least32_t;
typedef long long          int_least64_t;
typedef unsigned char      uint_least8_t;
typedef unsigned short     uint_least16_t;
typedef unsigned int       uint_least32_t;
typedef unsigned long long uint_least64_t;

typedef signed char        int_fast8_t;
typedef int                int_fast16_t;
typedef int                int_fast32_t;
typedef long long          int_fast64_t;
typedef unsigned char      uint_fast8_t;
typedef unsigned int       uint_fast16_t;
typedef unsigned int       uint_fast32_t;
typedef unsigned long long uint_fast64_t;

typedef long long          intmax_t;
typedef unsigned long long uintmax_t;
```

```
#define INT8_MIN         (-127i8 - 1)
#define INT16_MIN        (-32767i16 - 1)
#define INT32_MIN        (-2147483647i32 - 1)
#define INT64_MIN        (-9223372036854775807i64 - 1)
#define INT8_MAX         127i8
#define INT16_MAX        32767i16
#define INT32_MAX        2147483647i32
#define INT64_MAX        9223372036854775807i64
#define UINT8_MAX        0xffui8
#define UINT16_MAX       0xffffui16
#define UINT32_MAX       0xffffffffui32
#define UINT64_MAX       0xffffffffffffffffui64
```

C Programming operators

Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

Operators

: operator는 컴파일러에게 특정 수학 또는 논리 기능을 수행하도록 하는 지시 기호입니다. 임베디드 개발을 할 때는 bit연산자 사용이 정말 많이 나옵니다. 그러므로 연산자는 반드시 숙달되어야 합니다.

<Arithmetic operator>

Operator	Description
+	Adds two operands.
–	Subtracts second operand from the first.
*	Multiplies both operands.
/	Divides numerator by de-numerator.
%	Modulus Operator and remainder of after an integer division.
++	Increment operator increases the integer value by one.
--	Decrement operator decreases the integer value by one.

<Logical operator>

Operator	Description
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.

<Relational operator>

Operator	Description
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.

<bitwise operator>

Operator	Description
&	Binary AND Operator copies a bit to the result if it exists in both operands.
|	Binary OR Operator copies a bit if it exists in either operand.
^	Binary XOR Operator copies the bit if it is set in one operand but not both.
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.

<Assign operator>

Operator	Description
=	Simple assignment operator. Assigns values from right side operands to left side operand
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.
<<=	Left shift AND assignment operator.
>>=	Right shift AND assignment operator.
&=	Bitwise AND assignment operator.
^=	Bitwise exclusive OR and assignment operator.
|=	Bitwise inclusive OR and assignment operator.

<Arithmetic operator>

```c
#include <stdio.h>

int main(void) {

   int a = 21;
   int b = 10;
   int c ;

   c = a + b;
   printf("Line 1 - Value of c is %d\n", c );

   c = a - b;
   printf("Line 2 - Value of c is %d\n", c );

   c = a * b;
   printf("Line 3 - Value of c is %d\n", c );

   c = a / b;
   printf("Line 4 - Value of c is %d\n", c );

   c = a % b;
   printf("Line 5 - Value of c is %d\n", c );

   c = a++;
   printf("Line 6 - Value of c is %d\n", c );

   c = a--;
   printf("Line 7 - Value of c is %d\n", c );

   return 0;
}
```

```
$gcc -o main *.c

$main

Line 1 - Value of c is 31
Line 2 - Value of c is 11
Line 3 - Value of c is 210
Line 4 - Value of c is 2
Line 5 - Value of c is 1
Line 6 - Value of c is 21
Line 7 - Value of c is 22
```

<Relational operator>

```c
#include <stdio.h>

int main(void) {

   int a = 21;
   int b = 10;
   int c ;

   if( a == b ) {
      printf("Line 1 - a is equal to b\n" );
   } else {
      printf("Line 1 - a is not equal to b\n" );
   }

   if ( a < b ) {
      printf("Line 2 - a is less than b\n" );
   } else {
      printf("Line 2 - a is not less than b\n" );
   }

   if ( a > b ) {
      printf("Line 3 - a is greater than b\n" );
   } else {
      printf("Line 3 - a is not greater than b\n" );
   }

   /* Lets change value of a and b */
   a = 5;
   b = 20;

   if ( a <= b ) {
      printf("Line 4 - a is either less than or equal to  b\n" );
   }

   if ( b >= a ) {
      printf("Line 5 - b is either greater than  or equal to b\n" );
   }

   return 0;
}
```

```
$gcc -o main *.c

$main

Line 1 - a is not equal to b
Line 2 - a is not less than b
Line 3 - a is greater than b
Line 4 - a is either less than or equal to  b
Line 5 - b is either greater than  or equal to b
```

<logical operator>

```c
#include <stdio.h>

int main(void) {

   int a = 5;
   int b = 20;
   int c ;

   if ( a && b ) {
      printf("Line 1 - Condition is true\n" );
   }

   if ( a || b ) {
      printf("Line 2 - Condition is true\n" );
   }

   /* lets change the value of  a and b */
   a = 0;
   b = 10;

   if ( a && b ) {
      printf("Line 3 - Condition is true\n" );
   } else {
      printf("Line 3 - Condition is not true\n" );
   }

   if ( !(a && b) ) {
      printf("Line 4 - Condition is true\n" );
   }

    return 0;
}
```

```
$gcc -o main *.c

$main

Line 1 - Condition is true
Line 2 - Condition is true
Line 3 - Condition is not true
Line 4 - Condition is true
```

<bitwise operator>

```c
#include <stdio.h>

int main(void) {

   unsigned int a = 60; /* 60 = 0011 1100 */
   unsigned int b = 13; /* 13 = 0000 1101 */
   int c = 0;

   c = a & b;       /* 12 = 0000 1100 */
   printf("Line 1 - Value of c is %d\n", c );

   c = a | b;       /* 61 = 0011 1101 */
   printf("Line 2 - Value of c is %d\n", c );

   c = a ^ b;       /* 49 = 0011 0001 */
   printf("Line 3 - Value of c is %d\n", c );

   c = ~a;          /*-61 = 1100 0011 */
   printf("Line 4 - Value of c is %d\n", c );

   c = a << 2;     /* 240 = 1111 0000 */
   printf("Line 5 - Value of c is %d\n", c );

   c = a >> 2;     /* 15 = 0000 1111 */
   printf("Line 6 - Value of c is %d\n", c );

   return 0;
}
```

```
$gcc -o main *.c

$main

Line 1 - Value of c is 12
Line 2 - Value of c is 61
Line 3 - Value of c is 49
Line 4 - Value of c is -61
Line 5 - Value of c is 240
Line 6 - Value of c is 15
```

<Assign operator>

```c
#include <stdio.h>

int main(void) {

   int a = 21;
   int c ;

   c =  a;
   printf("Line 1 - =  Operator Example, Value of c = %d\n", c );

   c +=  a;
   printf("Line 2 - += Operator Example, Value of c = %d\n", c );

   c -=  a;
   printf("Line 3 - -= Operator Example, Value of c = %d\n", c );

   c *=  a;
   printf("Line 4 - *= Operator Example, Value of c = %d\n", c );

   c /=  a;
   printf("Line 5 - /= Operator Example, Value of c = %d\n", c );

   c  = 200;
   c %=  a;
   printf("Line 6 - %= Operator Example, Value of c = %d\n", c );

   c <<=  2;
   printf("Line 7 - <<= Operator Example, Value of c = %d\n", c );

   c >>=  2;
   printf("Line 8 - >>= Operator Example, Value of c = %d\n", c );

   c &=  2;
   printf("Line 9 - &= Operator Example, Value of c = %d\n", c );

   c ^=  2;
   printf("Line 10 - ^= Operator Example, Value of c = %d\n", c );

   c |=  2;
   printf("Line 11 - |= Operator Example, Value of c = %d\n", c );

   return 0;
}
```

```
$gcc -o main *.c

$main

Line 1 - =  Operator Example, Value of c = 21
Line 2 - += Operator Example, Value of c = 42
Line 3 - -= Operator Example, Value of c = 21
Line 4 - *= Operator Example, Value of c = 441
Line 5 - /= Operator Example, Value of c = 21
Line 6 - %= Operator Example, Value of c = 11
Line 7 - <<= Operator Example, Value of c = 44
Line 8 - >>= Operator Example, Value of c = 11
Line 9 - &= Operator Example, Value of c = 2
Line 10 - ^= Operator Example, Value of c = 0
Line 11 - |= Operator Example, Value of c = 2
```

HW1

문제

(A+B)%C는 ((A%C) + (B%C))%C 와 같을까?

(A×B)%C는 ((A%C) × (B%C))%C 와 같을까?

세 수 A, B, C가 주어졌을 때, 위의 네 가지 값을 구하는 프로그램을 작성하시오.

첫째 줄에 (A+B)%C, 둘째 줄에 ((A%C) + (B%C))%C, 셋째 줄에 (A×B)%C, 넷째 줄에 ((A%C) × (B%C))%C를 출력한다.

예제 입력 1 복사

```
5 8 4
```

예제 출력 1 복사

```
1
1
0
0
```

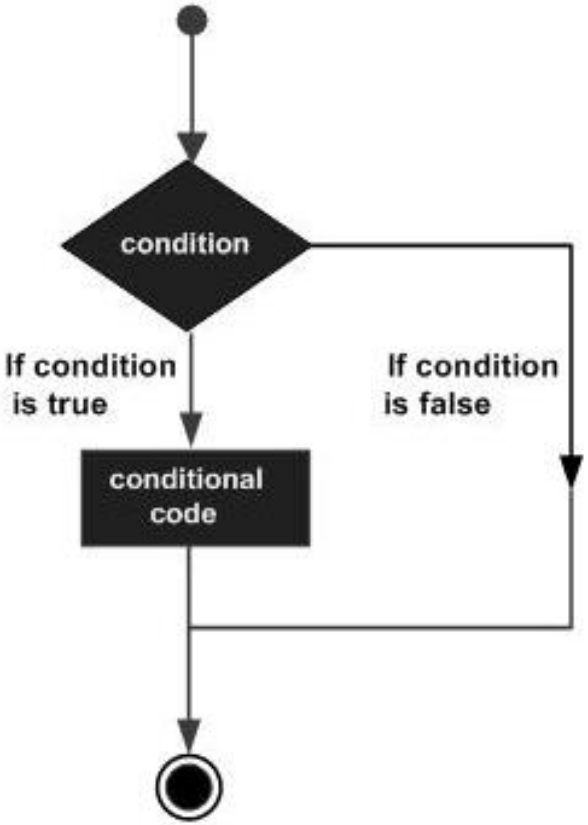
C Programming
Decision Making

Produced by donberja

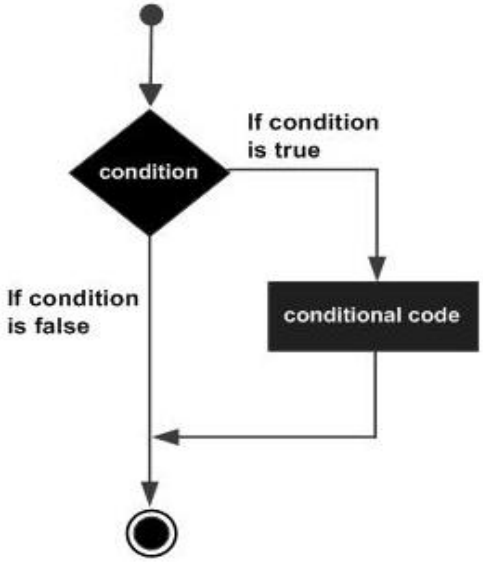
hyunwoo park(박현우)
phw820@gmail.com

Decision making

: 의사 결정은 특정 조건에 따라 문장의 실행 순서를 결정하거나 특정 조건이 충족될 때까지 문장 그룹을 반복하는 것이다. 아래 그림은 대부분의 프로그래밍 언어에서 볼 수 있는 일반적인 의사 결정 구조의 형태이다.



<if statement>



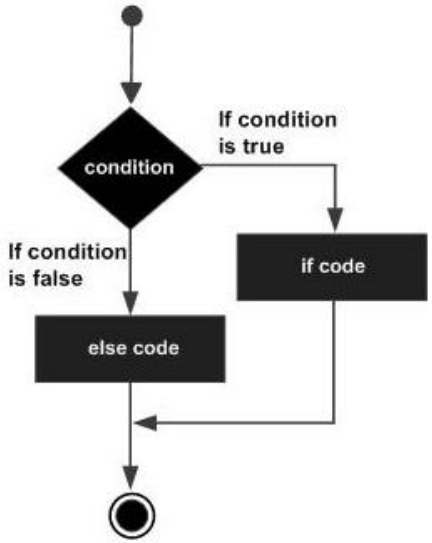

```c
#include <stdio.h>

int main (void) {

   /* local variable definition */
   int a = 10;

   /* check the boolean condition using if statement */

   if( a < 20 ) {
      /* if condition is true then print the following */
      printf("a is less than 20\n" );
   }

   printf("value of a is : %d\n", a);

   return 0;
}
```

```
$gcc -o main *.c

$main

a is less than 20
value of a is : 10
```

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 100;
   int b = 200;

   /* check the boolean condition */
   if( a == 100 ) {

      /* if condition is true then check the following */
      if( b == 200 ) {
         /* if condition is true then print the following */
         printf("Value of a is 100 and b is 200\n" );
      }
   }

   printf("Exact value of a is : %d\n", a );
   printf("Exact value of b is : %d\n", b );

   return 0;
}
```

```
$gcc -o main *.c

$main

Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200
```

<if ~else statement>



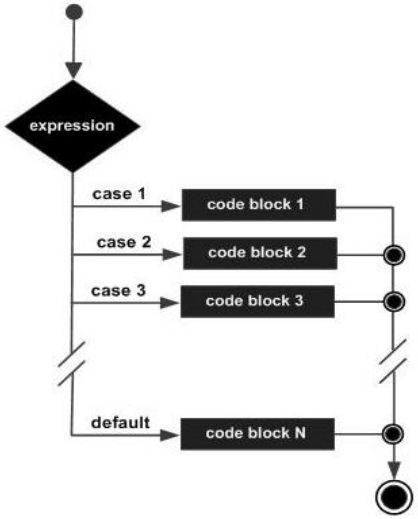

```c
#include <stdio.h>

int main (void) {

   /* local variable definition */
   int a = 100;

   /* check the boolean condition */
   if( a < 20 ) {
      /* if condition is true then print the following */
      printf("a is less than 20\n" );
   } else {
      /* if condition is false then print the following */
      printf("a is not less than 20\n" );
   }

   printf("value of a is : %d\n", a);

   return 0;
}
```

```
$gcc -o main *.c

$main

a is not less than 20
value of a is : 100
```

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 100;

   /* check the boolean condition */
   if( a == 10 ) {
      /* if condition is true then print the following */
      printf("Value of a is 10\n" );
   } else if( a == 20 ) {
      /* if else if condition is true */
      printf("Value of a is 20\n" );
   } else if( a == 30 ) {
      /* if else if condition is true  */
      printf("Value of a is 30\n" );
   } else {
      /* if none of the conditions is true */
      printf("None of the values is matching\n" );
   }

   printf("Exact value of a is: %d\n", a );

   return 0;
}
```

```
$gcc -o main *.c

$main

None of the values is matching
Exact value of a is: 100
```

<switch ~ case statement>




```c
#include <stdio.h>

int main (void) {

   /* local variable definition */
   char grade = 'B';

   switch(grade) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
         break;
      case 'D' :
         printf("You passed\n" );
         break;
      case 'F' :
         printf("Better try again\n" );
         break;
      default :
         printf("Invalid grade\n" );
   }

   printf("Your grade is  %c\n", grade );

   return 0;
}
```

```
$gcc -o main *.c

$main

Well done
Your grade is  B
```

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 100;
   int b = 200;

   switch(a) {

      case 100:
         printf("This is part of outer switch\n", a );

         switch(b) {
            case 200:
               printf("This is part of inner switch\n", a );
         }
   }

   printf("Exact value of a is : %d\n", a );
   printf("Exact value of b is : %d\n", b );

   return 0;
}
```

```
$gcc -o main *.c

$main

This is part of outer switch
This is part of inner switch
Exact value of a is : 100
Exact value of b is : 200
```

HW1

문제

시험 점수를 입력받아 90 ~ 100점은 A, 80 ~ 89점은 B, 70 ~ 79점은 C, 60 ~ 69점은 D, 나머지 점수는 F를 출력하는 프로그램을 작성하시오.

시험 성적을 출력한다.

예제 입력 1 복사

```
100
```

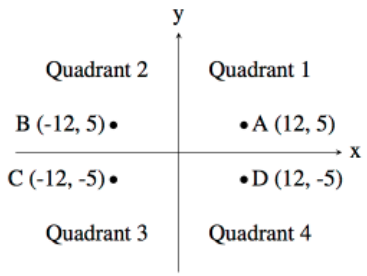
예제 출력 1 복사

```
A
```

HW2

문제

흔한 수학 문제 중 하나는 주어진 점이 어느 사분면에 속하는지 알아내는 것이다. 사분면은 아래 그림처럼 1부터 4까지 번호를 갖는다. "Quadrant n"은 "제n사분면"이라는 뜻이다.



예를 들어, 좌표가 (12, 5)인 점 A는 x좌표와 y좌표가 모두 양수이므로 제1사분면에 속한다. 점 B는 x좌표가 음수이고 y좌표가 양수이므로 제2사분면에 속한다.

점의 좌표를 입력받아 그 점이 어느 사분면에 속하는지 알아내는 프로그램을 작성하시오. 단, x좌표와 y좌표는 모두 양수나 음수라고 가정한다.

점 (x, y)의 사분면 번호(1, 2, 3, 4 중 하나)를 출력한다.

예제 입력 1 복사

```
12
5
```

예제 출력 1 복사

```
1
```

예제 입력 2 복사

```
9
-13
```

예제 출력 2 복사

```
4
```

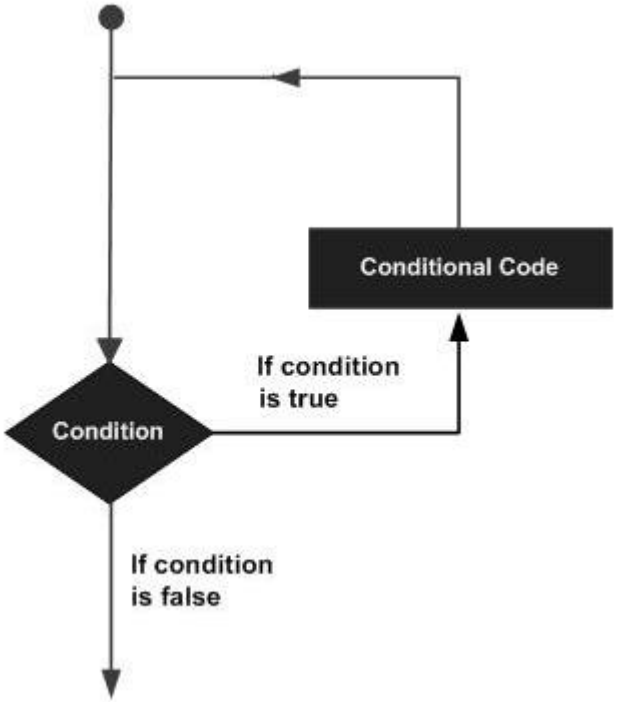
C Programming
Loop Control

Produced by donberja

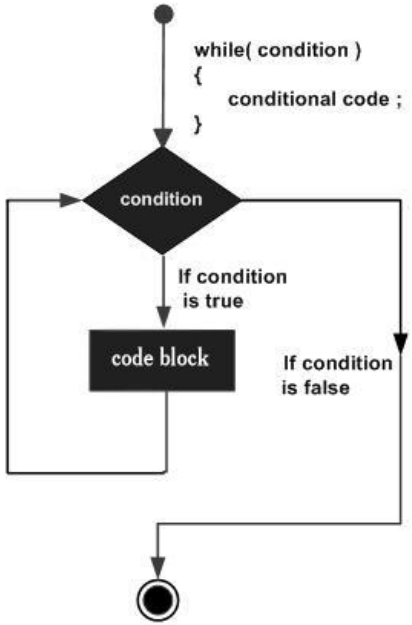
hyunwoo park(박현우)
phw820@gmail.com

Loop Control

: 루프는 특정 조건에 도달할 때까지 반복되는 명령어이다. 프로그래밍에서 코드 블록의 반복은 필수적으로 만나므로 이를 잘 활용해야 한다. 아래 그림은 대부분의 프로그래밍 언어에서 볼 수 있는 일반적인 loop control 형태이다.



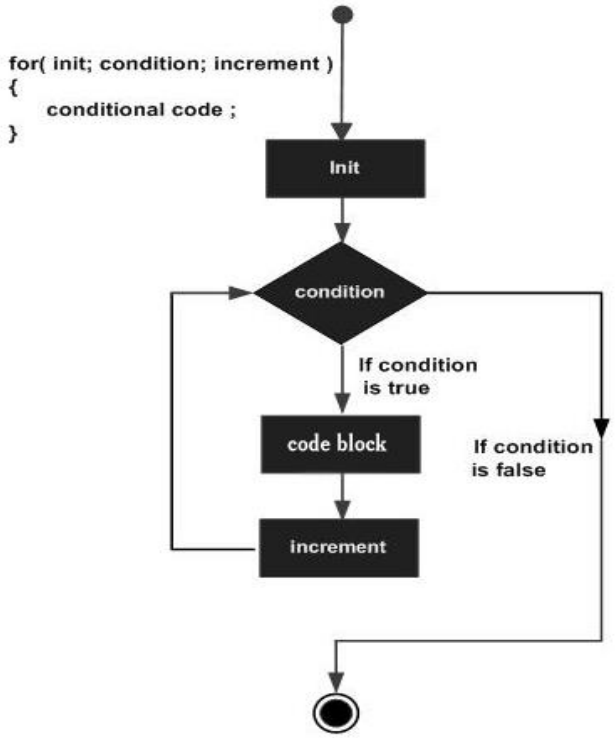
<while loop>




```c
#include <stdio.h>

int main (void) {

   /* local variable definition */
   int a = 10;

   /* while loop execution */
   while( a < 20 ) {
      printf("value of a: %d\n", a);
      a++;
   }

   return 0;
}
```

```
$gcc -o main *.c

$main

value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

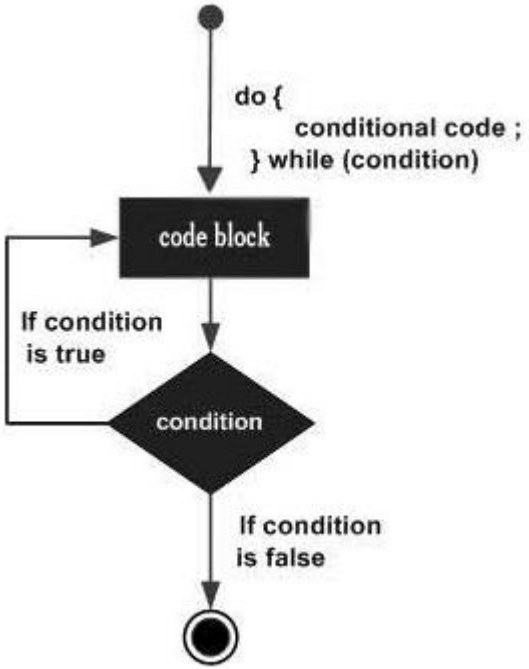
<for loop>




```c
#include <stdio.h>

int main (void) {

   int a;

   /* for loop execution */
   for( a = 10; a < 20; a = a + 1 ){
      printf("value of a: %d\n", a);
   }

   return 0;
}
```

```
$gcc -o main *.c

$main

value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

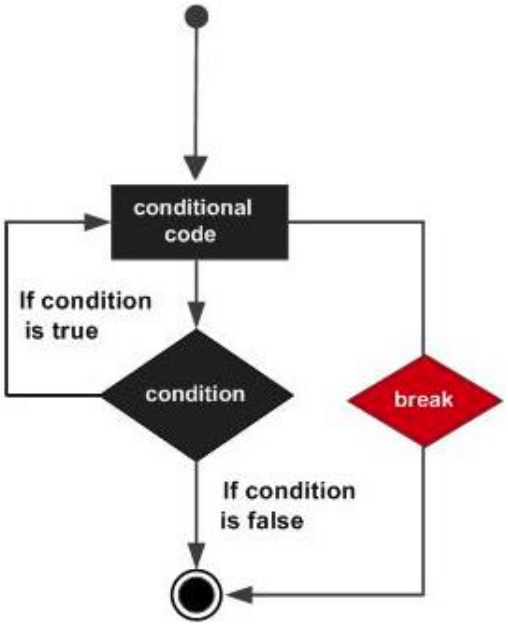
<do while loop>




```c
#include <stdio.h>

int main (void) {

   /* local variable definition */
   int a = 10;

   /* do loop execution */
   do {
      printf("value of a: %d\n", a);
      a = a + 1;
   }while( a < 20 );

   return 0;
}
```

```
$gcc -o main *.c

$main

value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

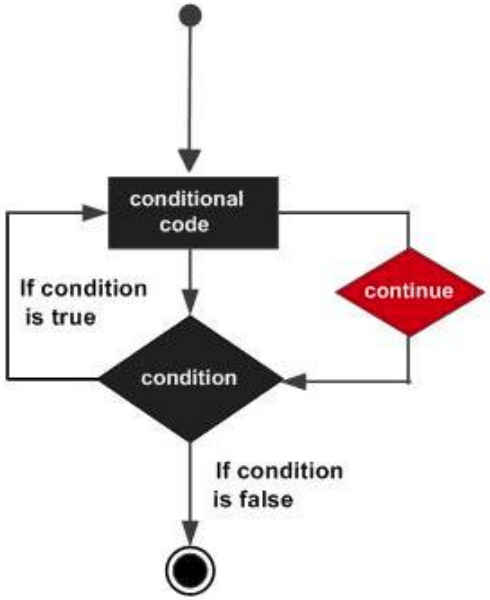
<break statement>




```c
#include <stdio.h>

int main (void) {

   /* local variable definition */
   int a = 10;

   /* while loop execution */
   while( a < 20 ) {

      printf("value of a: %d\n", a);
      a++;

      if( a > 15) {
         /* terminate the loop using break statement */
         break;
      }

   }

   return 0;
}
```

```
$gcc -o main *.c

$main

value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
```

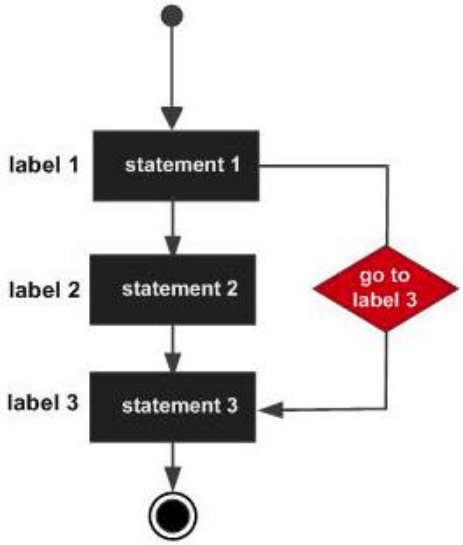
<continue statement>




```c
#include <stdio.h>

int main (void) {

   /* local variable definition */
   int a = 10;

   /* do loop execution */
   do {

      if( a == 15) {
         /* skip the iteration */
         a = a + 1;
         continue;
      }

      printf("value of a: %d\n", a);
      a++;

   } while( a < 20 );

   return 0;
}
```

```
$gcc -o main *.c

$main

value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

<goto statement>




```c
#include <stdio.h>

int main (void) {

   /* local variable definition */
   int a = 10;

   /* do loop execution */
   LOOP:do {

      if( a == 15) {
         /* skip the iteration */
         a = a + 1;
         goto LOOP;
      }

      printf("value of a: %d\n", a);
      a++;

   }while( a < 20 );

   return 0;
}
```

```
$gcc -o main *.c

$main

value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

HW1

문제

N을 입력받은 뒤, 구구단 N단을 출력하는 프로그램을 작성하시오. 출력 형식에 맞춰서 출력하면 된다.

출력형식과 같게 N*1부터 N*9까지 출력한다.

예제 입력 1 복사

```
2
```

예제 출력 1 복사

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
```

HW2

문제

첫째 줄에는 별 1개, 둘째 줄에는 별 2개, N번째 줄에는 별 N개를 찍는 문제

첫째 줄부터 N번째 줄까지 차례대로 별을 출력한다.

예제 입력 1 복사

```
5
```

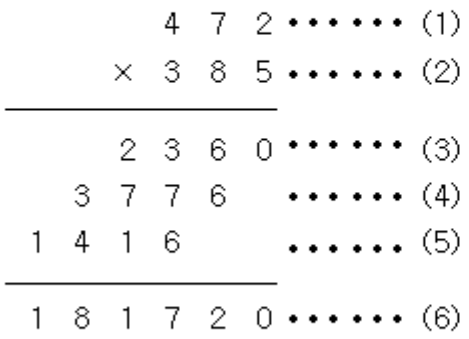
예제 출력 1 복사

```
*
**
***
****
*****
```

HW3

문제

(세 자리 수) × (세 자리 수)는 다음과 같은 과정을 통하여 이루어진다.



(1)과 (2)위치에 들어갈 세 자리 자연수가 주어질 때 (3), (4), (5), (6)위치에 들어갈 값을 구하는 프로그램을 작성하시오.

첫째 줄부터 넷째 줄까지 차례대로 (3), (4), (5), (6)에 들어갈 값을 출력한다.

예제 입력 1 복사

```
472
385
```

예제 출력 1 복사

```
2360
3776
1416
181720
```

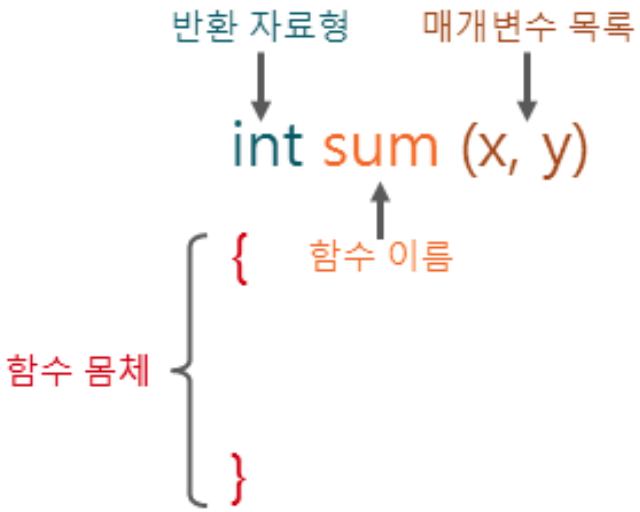
C Programming
Function

Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

Function

: 프로그래밍에서 함수란 하나의 특별한 목적의 작업을 수행하기 위해 독립적으로 설계된 프로그램 코드의 집합이다. 함수를 사용하면 반복적인 프로그래밍을 피할 수 있고 프로그램을 여러 개의 함수로 나누어 작성하면, 모듈화로 인해 전체적인 코드의 가독성이 좋아진다.



1. 반환자료형(return type) : 함수가 모든 작업을 마치고 반환하는 데이터 타입 명시
2. 함수이름 (function name) : 함수를 호출하기 위한 이름을 명시
3. 매개변수(parameters) : 함수 호출 시에 전달되는 인수의 값을 저장할 변수들을 명시
4. 함수 몸체 (function body) : 함수의 고유 기능을 수행하는 명령문의 집합.

```c
#include <stdio.h>

/* global variable declaration */
int a = 20;

int sum(int a, int b);

int main () {

  /* local variable declaration in main function */
  int a = 10;
  int b = 20;
  int c = 0;

  printf ("value of a in main() = %d\n",  a);
  c = sum( a, b);
  printf ("value of c in main() = %d\n",  c);

  return 0;
}

/* function to add two integers */
int sum(int a, int b) {

   printf ("value of a in sum() = %d\n",  a);
   printf ("value of b in sum() = %d\n",  b);

   return a + b;
}
```

```
$gcc -o main *.c

$main

value of a in main() = 10
value of a in sum() = 10
value of b in sum() = 20
value of c in main() = 30
```

```c
#include <stdio.h>

/* function declaration */
int max(int num1, int num2);

int main (void) {

   /* local variable definition */
   int a = 100;
   int b = 200;
   int ret;

   /* calling a function to get max value */
   ret = max(a, b);

   printf( "Max value is : %d\n", ret );

   return 0;
}

/* function returning the max between two numbers */
int max(int num1, int num2) {

   /* local variable declaration */
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

```
$gcc -o main *.c

$main

Max value is : 200
```

왜 값이 바뀌지 않지?

```c
#include <stdio.h>

/* function declaration */
void swap(int x, int y);

/* function definition to swap the values */
void swap(int x, int y) {

   int temp;

   temp = x; /* save the value of x */
   x = y;    /* put y into x */
   y = temp; /* put temp into y */

   return;
}

int main () {

   /* local variable definition */
   int a = 100;
   int b = 200;

   printf("Before swap, value of a : %d\n", a );
   printf("Before swap, value of b : %d\n", b );

   /* calling a function to swap the values */
   swap(a, b);

   printf("After swap, value of a : %d\n", a );
   printf("After swap, value of b : %d\n", b );

   return 0;
}
```

```
$gcc -o main *.c

$main

Before swap, value of a : 100
Before swap, value of b : 200
After swap, value of a : 100
After swap, value of b : 200
```

이건 왜 값이 바뀌고 int 형 변수 앞에 *는 뭘까?

```c
#include <stdio.h>

/* function declaration */
void swap(int *x, int *y);

/* function definition to swap the values */
void swap(int *x, int *y) {

   int temp;
   temp = *x;    /* save the value at address x */
   *x = *y;      /* put y into x */
   *y = temp;    /* put temp into y */

   return;
}

int main () {

   /* local variable definition */
   int a = 100;
   int b = 200;

   printf("Before swap, value of a : %d\n", a );
   printf("Before swap, value of b : %d\n", b );

   /* calling a function to swap the values.
      * &a indicates pointer to a ie. address of variable a and
      * &b indicates pointer to b ie. address of variable b.
   */
   swap(&a, &b);

   printf("After swap, value of a : %d\n", a );
   printf("After swap, value of b : %d\n", b );

   return 0;
}
```

```
$gcc -o main *.c

$main

Before swap, value of a : 100
Before swap, value of b : 200
After swap, value of a : 200
After swap, value of b : 100
```

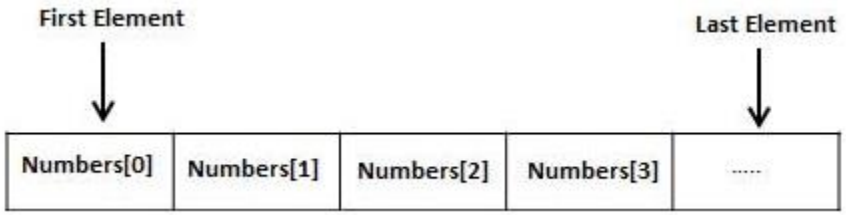
C Programming
Array

Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

Array

: 어떤 한가지 자료형을 연속적으로 나열하는 것을 말한다. 100명의 이름을 저장한다고 한다면, 100개의 변수를 선언해서 사용해야 하는데 배열을 하나 사용하면 아주 쉽게 100명의 이름을 저장할 수 있다.



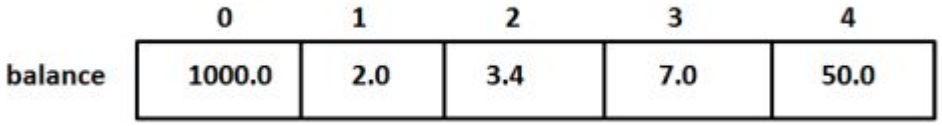
- **배열 선언**

type arrayName [arraySize] ;

ex) double balance[10];

- **배열 초기화**

ex) double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
ex) double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};



Multi- dimensional Array

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

- **배열 선언**

type arrayName [x] [y] ;

ex) int a[3][4];

- **배열 초기화**

ex) int a[3][4] = { {0, 1, 2, 3}, {4, 5, 6, 7}, {8, 9, 10, 11} };
ex) int a[][4] = { {0, 1, 2, 3}, {4, 5, 6, 7}, {8, 9, 10, 11} };

```c
#include <stdio.h>

int main () {

   int n[ 10 ]; /* n is an array of 10 integers */
   int i,j;

   /* initialize elements of array n to 0 */
   for ( i = 0; i < 10; i++ ) {
      n[ i ] = i + 100; /* set element at location i to i + 100 */
   }

   /* output each array element's value */
   for (j = 0; j < 10; j++ ) {
      printf("Element[%d] = %d\n", j, n[j] );
   }

   return 0;
}
```

```
$gcc -o main *.c

$main

Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

```c
#include <stdio.h>

int main () {

   /* an array with 5 rows and 2 columns*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;

   /* output each array element's value */
   for ( i = 0; i < 5; i++ ) {

      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }

   return 0;
}
```

```
$gcc -o main *.c

$main

a[0][0] = 0
a[0][1] = 0
a[1][0] = 1
a[1][1] = 2
a[2][0] = 2
a[2][1] = 4
a[3][0] = 3
a[3][1] = 6
a[4][0] = 4
a[4][1] = 8
```

```c
#include <stdio.h>

/* function declaration */
double getAverage(int arr[], int size);

double getAverage(int arr[], int size) {

   int i;
   double avg;
   double sum = 0;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }

   avg = sum / size;

   return avg;
}

int main () {

   /* an int array with 5 elements */
   int balance[5] = {1000, 2, 3, 17, 50};
   double avg;

   /* pass pointer to the array as an argument */
   avg = getAverage( balance, 5 ) ;

   /* output the returned value */
   printf( "Average value is: %f ", avg );

   return 0;
}
```

```
$gcc -o main *.c

$main

Average value is: 214.400000
```

HW

1. char str1[] = "angel", char str2[] = "devil" 2개의 문자열을 서로 바꿔보기.

2. 2 by 2 행렬의 곱셈을 계산하는 프로그램 만들기.

3. int arr[3][3] = {{1, 15, 4}, {8, 10, 16}, {2, 7, 20}} 에서 최대값을 찾는 함수 만들기.

4. int balance[] = {1000, 2, 3, 17, 50}를 오름차순으로 만드는 함수를 만들기.

C Programming
Pointer

Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

Pointer

: 포인터는 어렵게 생각할 필요없이 메모리 주소값을 담는 '변수'다.
먼저 포인터를 사용하기에 앞서, 메모리 주소를 명시하는 ampersand(&) operator를 봐야 한다.
그리고 (*)은 해당 변수가 가리키는 메모리 주소의 값을 명시한다.

```c
#include <stdio.h>

int main (void) {

   int  var1;
   char var2[10];

   printf("Address of var1 variable: %x\n", &var1  );
   printf("Address of var2 variable: %x\n", &var2  );

   return 0;
}
```

```
$gcc -o main *.c

$main

Address of var1 variable: b426ba9c
Address of var2 variable: b426ba92
```

```c
#include <stdio.h>

int main (void) {

   int  var = 20;   /* actual variable declaration */
   int  *ip;        /* pointer variable declaration */

   ip = &var;  /* store address of var in pointer variable*/

   printf("Address of var variable: %x\n", &var  );

   /* address stored in pointer variable */
   printf("Address stored in ip variable: %x\n", ip );

   /* access the value using the pointer */
   printf("Value of *ip variable: %d\n", *ip );

   return 0;
}
```

```
$gcc -o main *.c

$main

Address of var variable: 41678d74
Address stored in ip variable: 41678d74
Value of *ip variable: 20
```

<Incrementing a Pointer1>

```c
#include <stdio.h>

const int MAX = 3;

int main (void) {

   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have array address in pointer */
   ptr = var;

   for ( i = 0; i < MAX; i++) {

      printf("Address of var[%d] = %x\n", i, ptr );
      printf("Value of var[%d] = %d\n", i, *ptr );

      /* move to the next location */
      ptr++;
   }

   return 0;
}
```

```
$gcc -o main *.c

$main

Address of var[0] = a9f10c94
Value of var[0] = 10
Address of var[1] = a9f10c98
Value of var[1] = 100
Address of var[2] = a9f10c9c
Value of var[2] = 200
```

<Incrementing a Pointer2>

```c
#include <stdio.h>

int main () {

   /* an array with 5 elements */
   double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
   double *p;
   int i;

   p = balance;

   /* output each array element's value */
   printf( "Array values using pointer\n");

   for ( i = 0; i < 5; i++ ) {
      printf("*(p + %d) : %f\n",  i, *(p + i) );
   }

   printf( "Array values using balance as address\n");

   for ( i = 0; i < 5; i++ ) {
      printf("*(balance + %d) : %f\n",  i, *(balance + i) );
   }

   return 0;
}
```

```
$gcc -o main *.c

$main

Array values using pointer
*(p + 0) : 1000.000000
*(p + 1) : 2.000000
*(p + 2) : 3.400000
*(p + 3) : 17.000000
*(p + 4) : 50.000000
Array values using balance as address
*(balance + 0) : 1000.000000
*(balance + 1) : 2.000000
*(balance + 2) : 3.400000
*(balance + 3) : 17.000000
*(balance + 4) : 50.000000
```

<Return pointer form functions>

```c
#include <stdio.h>
#include <time.h>

/* function to generate and return random numbers. */
int* getRandom(void) {

   static int  r[10];
   int i;

   /* set the seed */
   srand( (unsigned)time( NULL ) );

   for ( i = 0; i < 10; ++i) {
      r[i] = rand();
      printf("%d\n", r[i] );
   }

   return r;
}

/* main function to call above defined function */
int main (void) {

   /* a pointer to an int */
   int *p;
   int i;

   p = getRandom();

   for ( i = 0; i < 10; i++ ) {
      printf("*(p + [%d]) : %d\n", i, *(p + i) );
   }

   return 0;
}
```

```
$main

1658718695
2091527415
320891267
767547476
870144118
1644865240
50638147
538935862
1196566281
907128490
*(p + [0]) : 1658718695
*(p + [1]) : 2091527415
*(p + [2]) : 320891267
*(p + [3]) : 767547476
*(p + [4]) : 870144118
*(p + [5]) : 1644865240
*(p + [6]) : 50638147
*(p + [7]) : 538935862
*(p + [8]) : 1196566281
*(p + [9]) : 907128490
```

<Passing pointers to function>

```c
#include <stdio.h>

/* function declaration */
double getAverage(int *arr, int size);

int main (void) {

   /* an int array with 5 elements */
   int balance[5] = {1000, 2, 3, 17, 50};
   double avg;

   /* pass pointer to the array as an argument */
   avg = getAverage( balance, 5 ) ;

   /* output the returned value  */
   printf("Average value is: %f\n", avg );
   return 0;
}

double getAverage(int *arr, int size) {

   int  i, sum = 0;
   double avg;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }

   avg = (double)sum / size;
   return avg;
}
```

```
$gcc -o main *.c

$main

Average value is: 214.400000
```

<pointer to pointer>

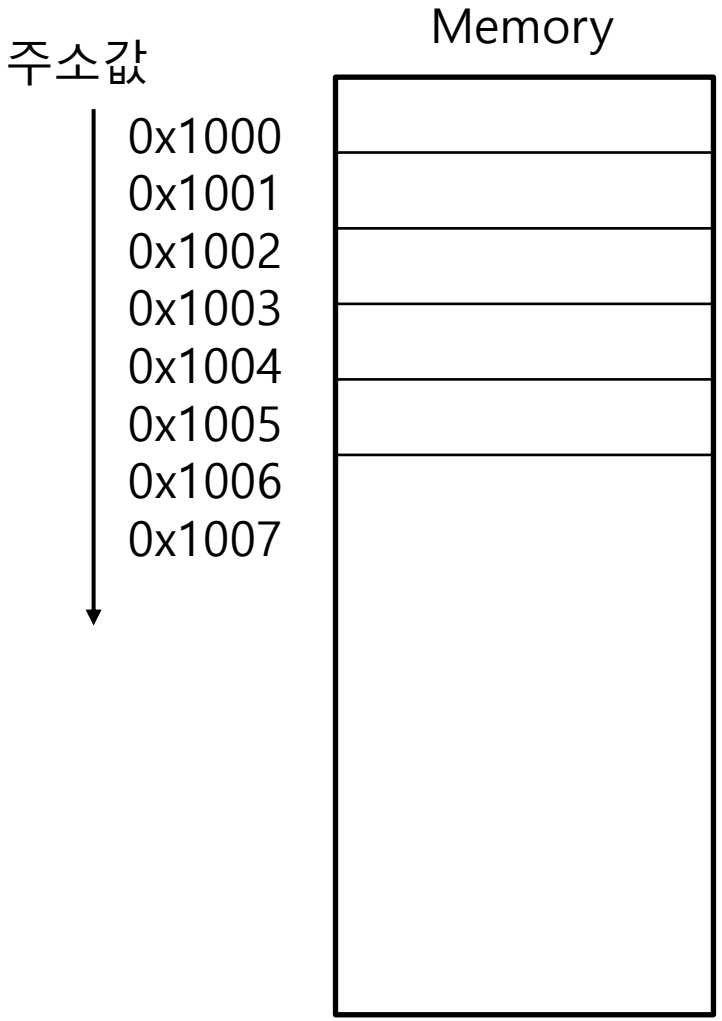

```c
#include <stdio.h>

int main (void) {

   int  var;
   int  *ptr;
   int  **pptr;

   var = 3000;

   /* take the address of var */
   ptr = &var;

   /* take the address of ptr using address of operator & */
   pptr = &ptr;

   /* take the value using pptr */
   printf("Value of var = %d\n", var );
   printf("Value available at *ptr = %d\n", *ptr );
   printf("Value available at **pptr = %d\n", **pptr);

   return 0;
}
```

```
$gcc -o main *.c

$main

Value of var = 3000
Value available at *ptr = 3000
Value available at **pptr = 3000
```

HW1

해당 main문을 보고 Memory를 완성하세요.

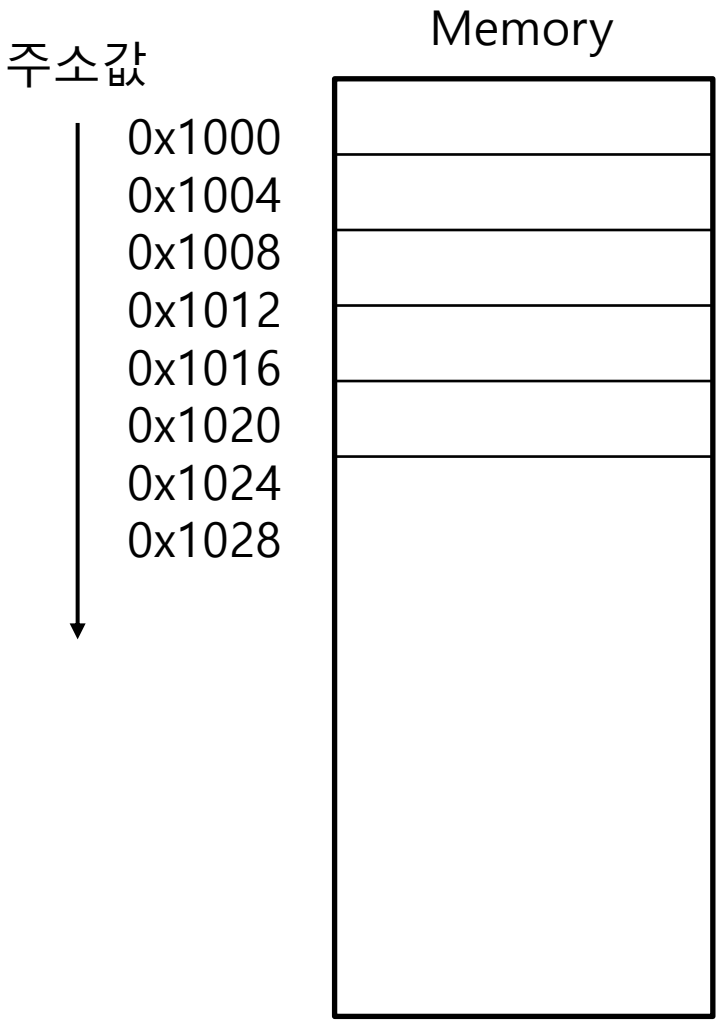
```
int main(void)
{

        char c = 'a';
        int n = 7;
        double d = 3.14;
}
```



HW2

해당 main문을 보고 Memory를 완성하세요.

```
int main(void)
{

        int a[2][3] = { {0,1,2} , {3,4,5} };
}
```

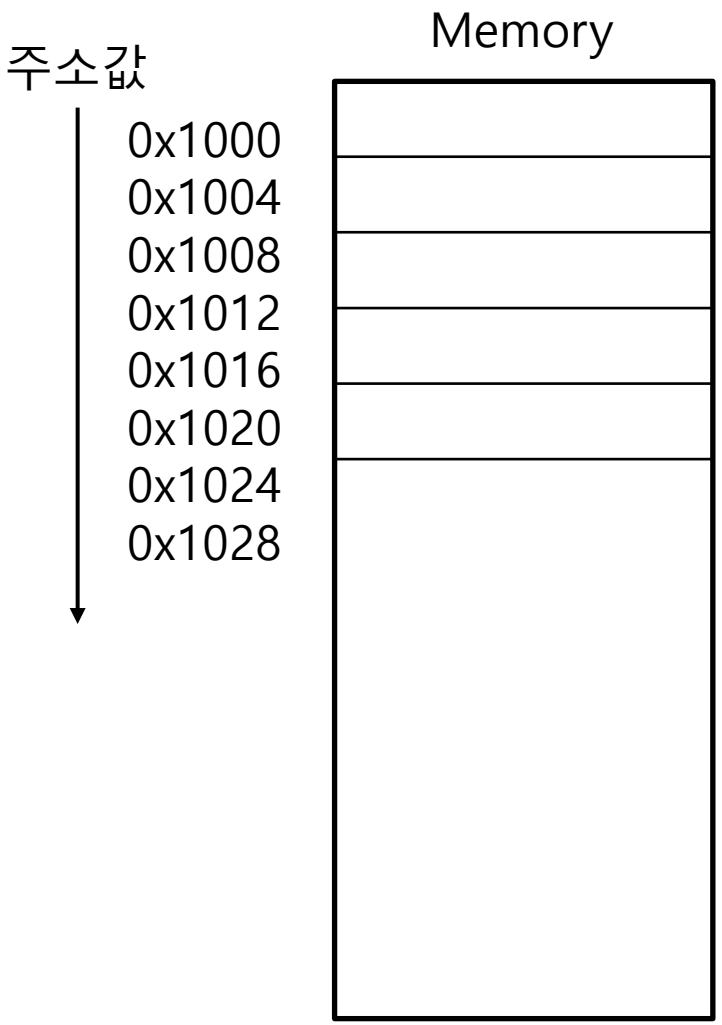


HW3

아래에 해당 하는 값을 넣으시오.

1) &a = ?
2) &a+1 = ?
3) a+1 = ?
4) *a = ?
5) *a+1 ?
6) **a = ?

```
int main(void)
{

        int a[2][3] = { {0,1,2} , {3,4,5} };
}
```



C Programming Structure

Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

Structure

: 구조체는 다양한 데이터 타입을 하나로 묶을 수 있다.

예) 도서관에서 책을 찾는다고 생각해보면, 제목, 저자, 도서 ID와 같은 정보를 하나로 묶어서 관리가 가능하다.

- **구조체 선언**

```
struct [structure tag]
{
        member definition;
        member definition;
} [struct name] ;

ex)
struct Books
{
        char title[50];
        char author[50];
        char subject[100];
        int book_id;
} book;
```

```c
#include <stdio.h>
#include <string.h>

struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

int main( ) {

   struct Books Book1;        /* Declare Book1 of type Book */
   struct Books Book2;        /* Declare Book2 of type Book */

   /* book 1 specification */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* print Book1 info */
   printf( "Book 1 title : %s\n", Book1.title);
   printf( "Book 1 author : %s\n", Book1.author);
   printf( "Book 1 subject : %s\n", Book1.subject);
   printf( "Book 1 book_id : %d\n", Book1.book_id);

   /* print Book2 info */
   printf( "Book 2 title : %s\n", Book2.title);
   printf( "Book 2 author : %s\n", Book2.author);
   printf( "Book 2 subject : %s\n", Book2.subject);
   printf( "Book 2 book_id : %d\n", Book2.book_id);

   return 0;
}
```

```
$gcc -o main *.c

$main

Book 1 title : C Programming
Book 1 author : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407
Book 2 title : Telecom Billing
Book 2 author : Zara Ali
Book 2 subject : Telecom Billing Tutorial
Book 2 book_id : 6495700
```

```c
#include <stdio.h>
#include <string.h>

struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

/* function declaration */
void printBook( struct Books book );

int main( ) {

   struct Books Book1;        /* Declare Book1 of type Book */
   struct Books Book2;        /* Declare Book2 of type Book */

   /* book 1 specification */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* print Book1 info */
   printBook( Book1 );

   /* Print Book2 info */
   printBook( Book2 );

   return 0;
}

void printBook( struct Books book ) {

   printf( "Book title : %s\n", book.title);
   printf( "Book author : %s\n", book.author);
   printf( "Book subject : %s\n", book.subject);
   printf( "Book book_id : %d\n", book.book_id);
}
```

```
$gcc -o main *.c

$main

Book title : C Programming
Book author : Nuha Ali
Book subject : C Programming Tutorial
Book book_id : 6495407
Book title : Telecom Billing
Book author : Zara Ali
Book subject : Telecom Billing Tutorial
Book book_id : 6495700
```

```c
#include <stdio.h>
#include <string.h>

struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

/* function declaration */
void printBook( struct Books *book );
int main( ) {

   struct Books Book1;        /* Declare Book1 of type Book */
   struct Books Book2;        /* Declare Book2 of type Book */

   /* book 1 specification */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* print Book1 info by passing address of Book1 */
   printBook( &Book1 );

   /* print Book2 info by passing address of Book2 */
   printBook( &Book2 );

   return 0;
}

void printBook( struct Books *book ) {

   printf( "Book title : %s\n", book->title);
   printf( "Book author : %s\n", book->author);
   printf( "Book subject : %s\n", book->subject);
   printf( "Book book_id : %d\n", book->book_id);
}
```

```
$gcc -o main *.c

$main

Book title : C Programming
Book author : Nuha Ali
Book subject : C Programming Tutorial
Book book_id : 6495407
Book title : Telecom Billing
Book author : Zara Ali
Book subject : Telecom Billing Tutorial
Book book_id : 6495700
```

C Programming
Union

Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

Union

: 구조체와는 달리 동일한 메모리 위치에 다른 데이터 유형을 저장할 수 있다.

- 공용체 선언

```
union [union tag]
{
        member definition;
        member definition;
} [struct name] ;

ex)
union Data
{
        int i;
        float f;
        char str[20];
} data;
```

```c
#include <stdio.h>
#include <string.h>

union Data {
   int i;
   float f;
   char str[20];
};

int main( ) {

   union Data data;

   printf( "Memory size occupied by data : %d\n", sizeof(data));

   return 0;
}
```

```
$gcc -o main *.c

$main

Memory size occupied by data : 20
```

```c
#include <stdio.h>
#include <string.h>

union Data {
   int i;
   float f;
   char str[20];
};

int main( ) {

   union Data data;

   data.i = 10;
   data.f = 220.5;
   strcpy( data.str, "C Programming");

   printf( "data.i : %d\n", data.i);
   printf( "data.f : %f\n", data.f);
   printf( "data.str : %s\n", data.str);

   return 0;
}
```

```
$gcc -o main *.c

$main

data.i : 1917853763
data.f : 412236058032779486045275999436 8.000000
data.str : C Programming
```

```c
#include <stdio.h>
#include <string.h>

union Data {
   int i;
   float f;
   char str[20];
};

int main( ) {

   union Data data;

   data.i = 10;
   printf( "data.i : %d\n", data.i);

   data.f = 220.5;
   printf( "data.f : %f\n", data.f);

   strcpy( data.str, "C Programming");
   printf( "data.str : %s\n", data.str);

   return 0;
}
```

```
$gcc -o main *.c

$main

data.i : 10
data.f : 220.500000
data.str : C Programming
```

C Programming
bit fields

Produced by donberja

hyunwoo park(박현우)
phw820@gmail.com

bit fields

: bit fields를 사용하면 메모리나 데이터 저장 공간을 효율적으로 사용이 가능하다.

- **bit fields 선언**

```
struct
{
        member definition : width;
        member definition : width;
} [struct name] ;

ex)
union Data
{
        int i;
        float f;
        char str[20];
} data;
```

```c
#include <stdio.h>
#include <string.h>

/* define simple structure */
struct {
   unsigned int widthValidated;
   unsigned int heightValidated;
} status1;

/* define a structure with bit fields */
struct {
   unsigned int widthValidated : 1;
   unsigned int heightValidated : 1;
} status2;

int main( ) {

   printf( "Memory size occupied by status1 : %d\n", sizeof(status1));
   printf( "Memory size occupied by status2 : %d\n", sizeof(status2));

   return 0;
}
```

```
$gcc -o main *.c

$main

Memory size occupied by status1 : 8
Memory size occupied by status2 : 4
```

```c
#include <stdio.h>
#include <string.h>

struct {
   unsigned int age : 3;
} Age;

int main( ) {

   Age.age = 4;
   printf( "Sizeof( Age ) : %d\n", sizeof(Age) );
   printf( "Age.age : %d\n", Age.age );

   Age.age = 7;
   printf( "Age.age : %d\n", Age.age );

   Age.age = 8;
   printf( "Age.age : %d\n", Age.age );

   return 0;
}
```

```
$gcc -o main *.c

main.c: In function ‘main’:
main.c:17:14: warning: large integer implicitly truncated to unsigned type [-Woverflow]
    Age.age = 8;
              ^


$main

Sizeof( Age ) : 4
Age.age : 4
Age.age : 7
Age.age : 0
```

임베디드 실제 개발 사례

HW

: USBCREGFlag는 어떻게 구성되어 있는지 분석 해오기!

```
typedef union
{
        UINT16 All;
            struct
            {
                UINT16 HiByte : 8;
                UINT16 LoByte : 8;
            } Byte;

            struct
            {
                UINT16 VAUXLow : 1;
                UINT16 VAUXoverCUR : 1;
                UINT16 CAN5VThShutdown : 1;
                UINT16 CAN5VUV : 1;
                UINT16 CAN5VOC : 1;
                UINT16 VSENSELow : 1;
                UINT16 VSUPUV : 1;
                UINT16 IDDOcNorm : 1;

                UINT16 : 3;
                UINT16 VDDThShutdown : 1;
                UINT16 : 1;
                UINT16 RSTLow : 1;
                UINT16 VSUPBatFail : 1;
                UINT16 IDDOcLp : 1;
            } Bit;
} USBCREGFlag;
```