1. 데이터 타입

데이터 타입	바이트
int	4 byte(32 비트 = 2^32 개를 나타낼 수 있음)대략 42 억 9 천만
short	2 byte(16 비트 = 2^16(65536)개에 해당)
char	1 byte(8 비트 = 2^8(256)개에 해당)
float	4 byte
double	8 byte(2^64)
long double	long double
· 데이터 타입에 unsigned 가 붙으면 음수가 존재하지 않음 · unsigned 가 없으면 음수값이 존재하게됨	

2. 음수 빨리 만드는 방법

· 1-1 = 0(?)

⇒ 0이 아닐 수 있다.

```
0000 0001       +1
1000 0001       우리의 잘못된 -1
--------------------------------
1000 0010       우리의 잘못된 0
```

· 2 의 보수

그러나 CPU를 설계하는 것이 아닌 이상 자세히 알 필요가 없다.

(CPU 설계쪽에선 매우 중요함)

```
 0000 0001      +1
 1111 1111      -1
------------------
10000 0000      0   (맨 앞의 1 은 버린다)
```

```
  0000 0101     +5
  1111 1011     -5
--------------------
1 0000 0000     0 (이하 동문)
```

```
  0000 1010     +10
  1111 0110     -10
--------------------
1 0000 0000     0 (이하 동문)
```

```
 0001 1100      +28
  1110 0100     -28
--------------------
1 0000 0000     0 (이하 동문)
```

3. 오버플로우

어떤 데이터 타입이 표현할 수 있는 최대값이 있고 이 범위를 벗어날 경우 오히려 맨 아래로 내려가는 현상

```
(예) 1000 0000
0x80 << 1  ⇒ 0000 0001
```

4. 언더플로우

표현할 수 있는 최소값에서 더 아래로 내려갈 경우 반대로 맨 위로 올라가는 현상

```
(예) 0000 0001
0x01>>1 ⇒ 1000 0000
```

ex) char 타입은 -128 ~ 127 이므로

```
127 + 1 = - 0
```

```
-128 - 1 = -128
```

```
127 + 2 = -1
```

5. 아스키(ASCII) 코드의 중요성

ASCII Table - 아스키 코드표 입니다.

제어 문자 | 공백 문자 | 구두점 | 숫자 | 알파벳

10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(	72	0x48	H	104	0x68	h
9	0x09	HT	41	0x29	)	73	0x49	I	105	0x69	i
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j
11	0x0B	VT	43	0x2B	+	75	0x4B	K	107	0x6B	k
12	0x0C	FF	44	0x2C	,	76	0x4C	L	108	0x6C	l
13	0x0D	CR	45	0x2D	-	77	0x4D	M	109	0x6D	m
14	0x0E	SO	46	0x2E	.	78	0x4E	N	110	0x6E	n
15	0x0F	SI	47	0x2F	/	79	0x4F	O	111	0x6F	o
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B	ESC	59	0x3B	;	91	0x5B	[	123	0x7B	{
28	0x1C	FS	60	0x3C	<	92	0x5C	₩	124	0x7C	|
29	0x1D	GS	61	0x3D	=	93	0x5D	]	125	0x7D	}
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL

- 컴퓨터에서 문자는 숫자로 치환(암호화에 적용)
  (예) SDR을 통해 무선상의 데이터를 취득하는데 이때 데이터안전을 위해 암호화가 필요

- 암호화의 역사
  · 가장 최초로 암호화를 사용한 사람은 로마의 시저 장군
    (암호화 시키는데 모든 문자에 + 3을 했고 이를 시저 암호라고 부름)

6. 기본 산술 연산자

기호	내용
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈
%	나머지 연산

7. printf 포맷

형식지정자	내용
%c	문자
%d, %i	부호가 있는 정수로서 10진수
%e, %E	지수를 가지는 형태의 실수 (지수부 표기에 %e는 소문자 e를 사용하고 %E는 대문자 E를 사용한다)
%f	실수로서 10진수
%o	부호가 없는 정수로서 8진수
%s	문자열
%u	부호가 없는 정수로서 10진수
%x, %X	부호가 없는 정수로서 16진수 (16진수 표현에 %x는 소문자를 사용하고 %X는 대문자를 사용한다)
%p	포인터(변수의 선두주소)를 16진수로 출력한다.
%n	관련된 인수는 그때까지 출력된 문자들의 수를 저장할 정수형 변수의 포인터
%%	%부호를 출력

8. 전위, 후위 연산

연산	내용	기타
p++	후치연산, 계산 후 증가	(+) 증가
++p	전위연산, 증가 후 계산	(-) 감소

9. 비트 연산자

연산	내용	사용예
AND	서로 참으로 같을 때만 참	10 & 3 1010 0011 AND ------- 0010 --> 2
OR	둘 중 하나만 참이면 참	10 | 16

		01010 10000 OR ------- 11010 --> 26
XOR	서로 달라야만 참	10 ^ 5 1010 0101 XOR ------ 1111 --> 15 10000 - 1 = 16 - 1 = 15
NOT	반전	~10 --> ~1010 00000000000001011 X = 11 11111111111....0101 -X = -11 ∴ 양수의 2의 보수 = ~양수+1

10. 쉬프트 연산

· 쉬프트 연산은 비트를 이동시키는 연산(2 의 승수로 곱 또는 나누는 연산)

ex.1	10 << 1 100 = 4
ex.2	1010 << 1 0001 0100 = 20
ex.3	1010 << 3 1010000 = 80
ex.4	10 >> 2 1000 = 8 (데이터 타입이 한 바이트일 경우)

* 주의 사항
0000 0001 >> 2
0100 0000 = 64
쉬프트 연산에 있어 overflow와 underflow에 따른 값 변동에 주의!

11. 관계 연산자

연산자	내용
<	작다
>	크다
<=	작거나 같다
>=	크거나 같다
==	같다
!=	다르다

12. 논리 연산자

연산자	내용
&&	AND 비교
||	OR 비교
!	NOT

13. 숏컷의 이점

기본적으로 if 문을 사용하면 mov, cmp, jmp 형식의
3 개의 어셈블리 코드가 만들어진다.
shortcut 을 사용하면 비교하는 cmp 가 사라진다.
특히 ARM 으로 구현할 때 더더욱 이득을 볼 수 있다.
(코드 최적화 용도로 사용하는 기법임)

14. '=' 연산자를 볼 때 주의할 점

- C 언어 코드 분석 시
  num1 = num2 = num3; 는
  num3 의 값을 num2, num1 에 셋팅하는것과 동일
  (오른쪽에서 왼쪽으로 보도록 함)

- '=' 연산자는 어셈의 mov 와 동일한 역할
- 단일 메인함수를 통한 코드작성 금지(가독성과 유지보수 위해 함수 사용)

(1) 입력을 6 을 주고 num << 4 를 수행하는 함수를 작성하고 결과를 출력

```
#include <stdio.h>

int shift_func(int num)
{
	return num << 4;
}

int main(void)
{
	int num = 6, res;
	res = shift_func(num);
	printf("res = %d\n", res);
	return 0;
}
```

(2) 입력에 55 를 넣고 num >> 3 을 수행하는 함수 작성

```
#include <stdio.h>

int shift_right(int num)
{
	return num >> 3;
}

int main(void)
{
	int num = 55;
	printf("result = %d\n", shift_right(num));
	return 0;
}
```

(3) 입력에 char num1 = 21, num2 = 31 을 넣고
AND, OR, XOR 를 수행하는 함수를 각각 만든다.
(함수가 총 3 개 만들어짐 main 포함 4 개)

```
char and_func(char n1, char n2)
```

```
{
        return n1 & n2;
}

char or_func(char n1, char n2)
{
        return n1 | n2;
}

char xor_func(char n1, char n2)
{
        return n1 ^ n2;
}

int main(void)
{
        char n1 = 21, n2 = 31;
        printf("n1 and n2 = %d\n", and_func(n1, n2));
        printf("n1 or n2 = %d\n", or_func(n1, n2));
        printf("n1 xor n2 = %d\n", xor_func(n1, n2));
        return 0;
}
```

15. scanf 사용법

- printf 와 쌍을 이룸
- scanf 의 첫 번째 입력은 "%d" 혹은 "%f", "%lf" 등이 올 수 있음
- 두 번째 입력은 결과 값을 가진 주소값(&변수명)

16. if 문

- if 문은 조건을 지정하고 싶을 때 사용
- 조건절 안에 연산자, 조건 연산자 등등이 올 수 있음
- if, else if, else 등의 조건문을 생성 가능
  (단, 참과 거짓의 지옥은 만들지 않도록 함)

```
(ex)
        if(조건문)
```

```
{
}
else if(조건문)
{
}
else
{
}
```

17. switch 문

- if, else if, else 문과 같은 조건절을 대체함
- 기타 조건절과는 달리 가독성이 좋음
  (컴파일러를 switch 문을 써서 만듬)

```
(ex)
switch(변수)
{
 case a:
   break;
 case b:
   break;
 default:
}
```

18. while 문

- 반복문의 대표적 형식문
- 조건이 만족되는 동안 반복

```
(ex)
while(조건문)
{


}
```