C언어 이론 및 실습 1

마이크로프로세서

HRI 연구실

김동한

Human-Robot Interaction Laboratory

KYUNG HEE UNIVERSITY

C 프로그램

>> C와 프로그래밍 언어의 시대

- 1세대 : 기계어 (숫자 코드로 구성된 프로그램)
- 2세대 : 어셈블리어 (기호로 된 명령어 사용)
- 3세대 : 포트란(FORTRAN), 코볼(COBOL) 등
- 4세대 : 데이터 중심 언어 (SQL 등)
- 5세대 : 논리적이고 기능적인 언어 (LISP 등)
- C언어는 고급 언어와 저급 언어의 특징을 모두 가지고 있음

C 프로그램

>> 첫 번째 C 프로그램

```
#include <stdio.h>                    ①
int main(void)                        ②
{                                     ③
    printf("Hello, World!₩n");        ④
    return 0;                         ⑤
}
```

[출력결과]

Hello, World!

C 언어의 창시자인 Kernighan과 Ritchie가 C 언어를 발표하면서 소개한 프로그램

C언어 실습용 온라인 C 컴파일러 (실습과 숙제용)

https://www.onlinegdb.com/online_c_compiler

C 프로그램

첫 번째 C 프로그램

- # 부호로 시작하는 명령어는 모두 전처리기를 위한 지시문
  - stdio.h라는 헤더 파일을 C 프로그램 안으로 불러들이는 역할
  - C 컴파일러는 다수의 헤더파일을 제공함
  - 사용자가 임의로 자신의 고유한 헤더파일을 만들 수 있음
  - 관례적으로 .h라는 확장자를 붙임

- 함수라고 불리는 프로그램의 단위
  - 함수는 함수 이름과 뒤에 붙은 소괄호()로 구별
  - main 함수는 모든 C 프로그램에서 반드시 있어야 함
  - void는 main() 함수가 매개변수를 전혀 가지고 있지 않음을 의미함

첫 번째 C 프로그램

- 중괄호([])는 여러 개의 C 문장으로 구성된 블록을 의미

- printf라는 이름을 가진 함수를 호출하는 것
  - printf 함수는 stdio.h에 정의되어 있음
  - printf 함수는 큰 따옴표(" ")로 둘러싸인 글자를 그대로 화면에 출력함
  - ₩n은 줄 바꿈 문자로 커서를 다음 줄로 옮겨줌
  - 모든 C언어의 문장은 세미콜론(;)으로 끝남

- main() 함수의 호출자에게 0을 넘겨주고 함수 실행을 종료함
  - 시스템의 운영체제가 main() 함수의 호출자임
  - 넘겨주는 값이 정수이므로 int main()으로 사용함

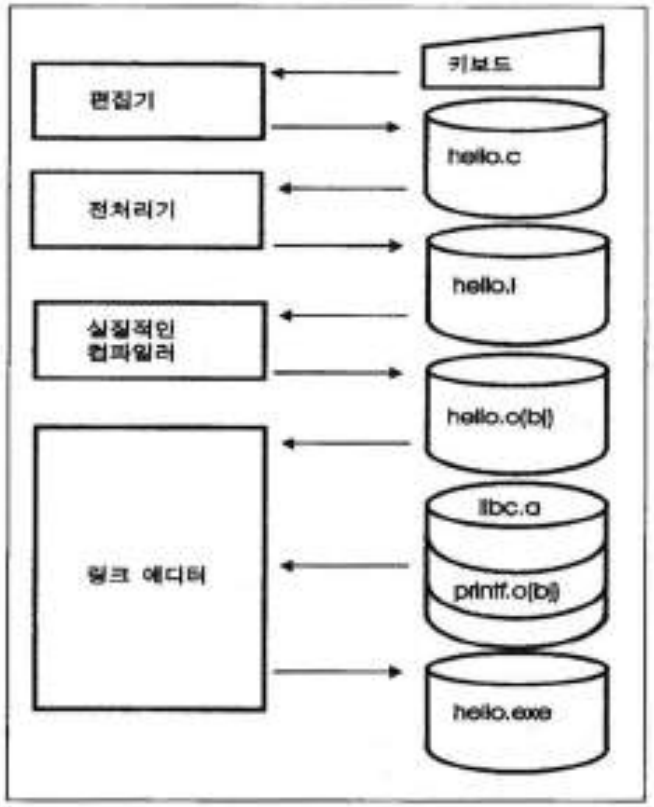
C 프로그램

>> C 컴파일러

- 원시파일(source file) : ASCII 파일 형대로 저장된 프로그램
- 일반적으로 파일의 확장자는 .c를 사용
- 일반적인 에디터 프로그램 및 전용 개발도구 사용 가능
  - 예) Microsoft Visual Studio / Visual Studio Code
- 컴파일러는 원시파일을 목적파일(object file)로 바꿈

C 컴파일러

- 링크 에디터(link editor)는 원시파일의 목적파일과 라이브러리의 목적 코드를 결합하여 실행 프로그램(execute program)을 만듦



실행 프로그램을 생성하는 과정

```
#include <stdio.h>

int main (void)          ①    /* sum.c */
{
    int s,               ②    /* sum variable */
        i ;                   /* counter variable */
    s = 0;               ③
    i = 1;
    while ( i <= 100)    ④    /* loop header */
    {
        s = s + i;       ⑤
        i = i + 1;       ⑥
    }
    printf ("sum of 1 to 100: %d₩n", s);        ⑦
    return 0;
}
```

[출력결과]

```
sum of 1 to 100: 5050
```

C 프로그램 예제

보충 프로그램

① 주석 : /* 과 */ 으로 둘러싸인 문장
- 프로그램의 어느 위치에나 사용할 수 있고, 프로그램의 실행에 아무런 영향을 미치지 않음
- cf) // : // 이후 해당 줄 끝 까지만 주석으로 처리 (C++)

② 선언문 : 정수(int) 타입의 변수 두 개(s, i)를 선언함

③ 대입문 : s에 0을 대입하고, i에는 1을 대입함

④ 반복문 : 괄호석의 조건이 참이면 블록을 반복함
- (변수 i의 값이 100보다 작거나 같으면 블록 내 문장 수행)

⑤ 대입문 : s와 i의 값을 더한 후, 다시 변수 s에 대입

⑥ 대입문 : i에 1을 더한 후, 다시 변수 j에 대입

⑦ 포맷 기술자 : &d (정수(int) 변수의 값을 화면에 표시) 이 위치에 변수 s의 값이 출력됨

데이터 타입과 입출력

>> 변수, 상수, 데이터 타입

- 변수의 성질
  ① 이름이 있음
  ② 길이가 정의되어 있음
  ③ 값이 저장되는 내부 형식이 있음

- 예제
  ① s = 0; //s(변수), 0(상수)
  ② s = 13; //s(변수), 13(상수)
  ③ Int s; //int(데이터 타입), s(변수)

- C의 데이터 타입은 기본적으로 두 가지 형태로 분류
  ① 정수 데이터 타입 또는 소수점 이하가 없는 고정소수점 수
  ② 부동소수점 수로 알려진 10진수 데이터 타입

데이터 타입과 입출력

상수

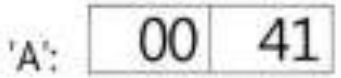
- int
  - ① 정수 상수이며 부호의 유무로 구별됨
  - ② 운영체제에 따라서 다르며 도스에서는 2바이트, 유닉스는 4바이트로 표현됨
  - ③ Int 상수를 8진수로 표현할 시에는 0으로 시작해야 하며, 0과 7 사이에 있는 수만 사용할 수 있음
  - ④ Int 상수를 16진수로 표현할 시에는 0X나 0x로 시작해야 하며, 0~9,A~F, a~f 사이에 있는 수만 사용할 수 있음

- long ing 또는 long
  - ① 길이가 두 배인 int 상수이며, 보통 int 상수와는 달리 끝이 대문자 L이나 소문자 l로 끝남
    - 예) 13L, -1074l, 0L, -0x3FFFFF000L
  - ② 내부 표현 방식은 일반적으로 4바이트 정수로 되어있음

데이터 타입과 입출력

상수

- 문자 상수
  - 작은 따옴표로 둘러싸인 문자이며, 그 문자가 지닌 ASCII 코드 값과 일치함
    - 예) 'A', 'm', '!'



- 줄 바꿈 문자, 쪽전진 문자, 탭 문자와 같은 문자는 다음과 같이 표현함
  - '₩n' (new-line character)
  - '₩f' (form feed character)
  - '₩t' (tabulator character)

데이터 타입과 입출력

>> 상수

- 문자열 상수
  - ① 문자열은 큰 따옴표로 표현함
    - 예) "Hieroglyphics"
  - ② 문자열 안에 있는 각 문자는 한 바이트를 차지하며, ASCII 코드로 표현되며, ASCII 제로로 불리는 '₩0' 문자는 자동적으로 끝에 붙음

H	i	e	r	o	g	l	y	p	h	i	c	s	\0

  - ③ 이 문자열을 16진수 ASCII 코드로 표현하면 다음과 같음

48	69	65	72	6F	67	6C	79	70	68	69	63	73	00

  - ④ 문자열 상수는 컴파일러가 특정한 메모리 위치에 그 값을 저장하는 유일한 상수임. 프로그램 문맥에서 문자열 상수는 그것이 저장된 주기억장치의 시작주소로 표현됨. 문자열의 끝은 마지막에 붙는 '₩0'로 알 수 있음

데이터 타입과 입출력

>> 상수

- double
  - ① 소수점이 포함된 10진수로 된 상수이며, 소수점은 점(.)으로 표시함. 10의 지수는 심볼 E나 e를 사용하여 표현함
    - 예) 3.141592, 0.(=0.0), .5, -2.7e23(= -2.7*10^23)
  - ② 내부 표현은 8바이트 부동소수점 수로 되어있음

- float
  - 일반적으로 4바이트 부동소수점 수이며 double 상수와 구분하기 위해 끝에 대문자 F나 소문자 f를 붙임
    - 예) 1.305f, -5.39e-12F

- long double
  - 일반적으로 10바이트 부동소수점 수이며 double 상수와 구분하기 위해 끝에 대문자 L이나 소문자 l을 붙임
    - 예) 1.305L, -5.39e-1200l

데이터 타입과 입출력

>> 변수

- ANSI 표준 : sizeof(short) ≤ sizeof(int) ≤ sizeof(long)

데이터 타입	크기	값의 범위
char unsigned char	1 바이트	-128~+127 0~255
short [int] unsigned short [int]	2 바이트	-32768~+32767 0~65535
int unsigned int	2 또는 4 바이트	short 또는 long 타입과 같음
long [int] unsigned long [int]	4 바이트	-2147483648~+2147483647 0~+4294967295
float	4 바이트 8-비트 지수 23-비트 가수	$-3.4*10^{+38}$~$3.4*10^{38}$
double	8 바이트 11-비트 지수 52-비트 가수	$-1.7*10^{+308}$~$1.7*10^{308}$
long double	10 바이트 16-비트 지수 63-비트 가수	$-3.4*10^{+4932}$~$1.1*10^{4932}$

C 데이터 타입의 크기의 값과 범위

데이터 타입과 입출력

>> 입출력 루틴

- 변수 값을 화면으로 출력하는 표준 함수 printf()

- 문자열 상수를 출력하여 포맷 기술자는 다음과 같이 표현됨

```
int x = 13;
double w = 3.1416;
printf("Integer:%d, decimal number:%lf \n",x,w);
```

```
출력 결과:
Integer:13, decimal number: 3.141600
```

① printf()는 그 다음 인수인 변수 x를 읽음
② %d를 만났기 때문에 변수 x의 값을 정수로 해석함
③ 고정소수점 수인 x의 값을 10진수로 된 문자열로 변환함
④ %d 대신에 이 변환된 문자열을 출력함

데이터 타입과 입출력

입출력 루틴

- 키보드를 이용하여 데이터를 입력하는 표준함수 scanf()
- scanf()의 구문은 데이터가 읽혀질 변수 앞에 연산자 &를 첨가해야 하는 것을 제외하고 printf()와 유사함

데이터 타입과 입출력

>> 문자 단위의 입출력

- 문자 단위로 입출력 할 수 있는 표준 도구는 getchar()와 putchar() 임
  - getchar() 함수 : 키보드로부터 한 문자를 읽어들여 실행중인 프로그램에 전달해 줌
  - putchar() 함수 : 프로그램에서 한 문자를 받아 화면에 표시해 줌

연산자와 표현식

>> 예약어

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

ANSI 표준에서 정의한 32개의 명령어

연산자와 표현식

>> 연산자

우선순위	연산자	타입	결합
1	() [] . ->	주	왼쪽
2	+ - ~ ! * & ++ -- sizeof(type)	단항	오른쪽
3	* / %	이항	왼쪽
4	+ -	이항	왼쪽
5	<< >>	이항	왼쪽
6	< > <= >=	이항	왼쪽
7	== !=	이항	왼쪽
8	&	이항	왼쪽
9	^	이항	왼쪽
10	|	이항	왼쪽
11	&&	이항	왼쪽
12	||	이항	왼쪽
13	?:	삼항	오른쪽
14	= *= /= %= += -= <<= >>= &= ^= |=	이항	오른쪽
15	,	이항	왼쪽

C 의 연산자

연산자와 표현식

연산자

- 우선순위가 높을수록 먼저 계산됨
- 연산자 타입은 요구되는 피연산자의 개수를 의미함
- 결합은 식 내에서 우선순위가 같은 경우의 계산순서를 표시
- 표현식 : 연산자와 피연산자로 구성됨
- 모든 표현식은 계산의 결과값을 가짐

연산자와 표현식

>> 산술 연산자

- *(곱셈), /(나눗셈), %(나머지 연산), +(덧셈), -(뺄셈)

- 산술연산자의 피연산자는 어떤 데이터 타입도 사용가능
  - 예외1 : %는 정수형 피연산자만 사용가능
  - 예외2 : 포인터 연산에는 +와 -연산자만이 제한적으로 사용가능

- 데이터 타입이 다른 피연산자가 함께 사용될 때
  - 크기가 다르면 작은 타입이 큰 타입으로 변환됨
  - unsigned 데이터 타입은 이에 상응하는 signed 데이터 타입보다 크다고 간주
  - 모든 char 타입과 short 타입의 피연산자는 계산에 앞서 항상 int로 변환됨

연산자와 표현식

>> 비교 연산자

- <(작다), >(크다), <=(작거나 같다), >=(크거나 같다), ==(같다), !=(같지 않다)
- 두 피연산자의 데이터 타입이 다를 경우 산술 연산자와 같은 타입변환 규칙이 적용됨
- 비교 연산식의 결과는 int 타입이며 0과 1중 하나가 됨
  - 1 : 논리적으로 참
  - 0 : 논리적으로 거짓

연산자와 표현식

>> 대입 연산자

- ++(증가), --(감소), =(대입), += -+ *= /= %= <<= >>= &= ^= != (대입 조합)
- 모든 대입연산자는 오른쪽 결합을 함
- 다중 대입이 가능함
  - 예) x = y = z = 5
- 일반 규칙
  - =의 왼쪽에는 반드시 변수가 와야 함
  - =의 오른쪽에는 어떠한 표현식도 가능함
- 타입 변환
  - 항상 L값의 데이터 타입으로 변환됨

연산자와 표현식

>> 대입 연산자

```
double x = 3.5;
int z;
z = 3 * x;
printf ("%dWn", z);
printf ("%dWn", 3 * x);
x = z;
printf ("%dWn", x);
```

출력결과:

```
10
10.5
10.0
```

대입조합	동일표현	비고
z += 5;	z = z + 5;	산술연산
z -= 5;	z = z - 5;	
z *= 5;	z = z * 5;	
z /= 5;	z = z / 5;	
z %= 5;	z = z % 5;	
z <<= 5;	z = z << 5;	비트연산
z >>= 5;	z = z >> 5;	
z &= 5;	z = z & 5;	
z ^= 5;	z = z ^ 5;	
z |= 5;	z = z | 5;	

산술 연산자와 대입 연산자의 조합

연산자와 표현식

>> 대입 연산자

- 증가와 감소 연산자
  - ① ++(증가), --(감소)
  - ② 전위 표기법과 후위 표기법

```
int x = 3, z;
z = ++x;          /* 증가된 후 식에 값을 할당 */
printf ("%d - %d₩n", x, z);
z = x++;          /* 증가되기 전 식에 값을 할당 */
printf ("%d - %d₩n", x, z);
```

[출력 결과]
4 - 4
5 - 4

  - ③ 일반 규칙
    - 후위 연산자가 붙은 변수가 식에 나오면 같은 식 안에서 그 변수가 다시 나오지 않도록 함
    - ++x 또는 x++ 는 L값이 아님

연산자와 표현식

>> 논리 연산자

- &&(AND, 논리곱), ||(OR, 논리합) : 이항연산자
  !(NOT, 논리부정) : 단항연산자

- 논리연산의 진리표

op1	op2	op1 && op2	op1 || op2	!op1
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

연산자와 표현식

>> 논리 연산자

- 연산규칙
  - 피연산자는 표현식이며 그 값이 0이면 거짓, 0이 아니면 참
  - 논리연산의 결과는 참이면 1, 거짓이면 0
  - 단축계산 : 논리식에서 op1이 먼저 평가되고, op2의 평가 결과가 논리연산의 결과에 영향을 미치지 않을 때, op2는 평가되지 않음

연산자와 표현식

>> 논리 연산자

int x = 19; printf ("%d₩n", x); printf ("%d₩n", !x); printf("%d₩n", !!x);	[출력결과] 19 0 1

int x = 19;
int y;
y = x && x < 10;
printf ("%d₩n", y);
	[출력결과]
	0

int x = 4, y = 17, z;
z = x > 5 && (y = 8);
printf("%d : %d : %d ₩n", x, y, z);
	[출력결과(단축계산)]
	4 : 17 : 0

논리 연산자 예제

연산자와 표현식

>> 비트 연산자

- &(AND), |(OR), ^(XOR), ~(보수), &=, ^=, !=, >>=, <<= (대입 연산자와의 조합)
- 비트 연산자의 진리표

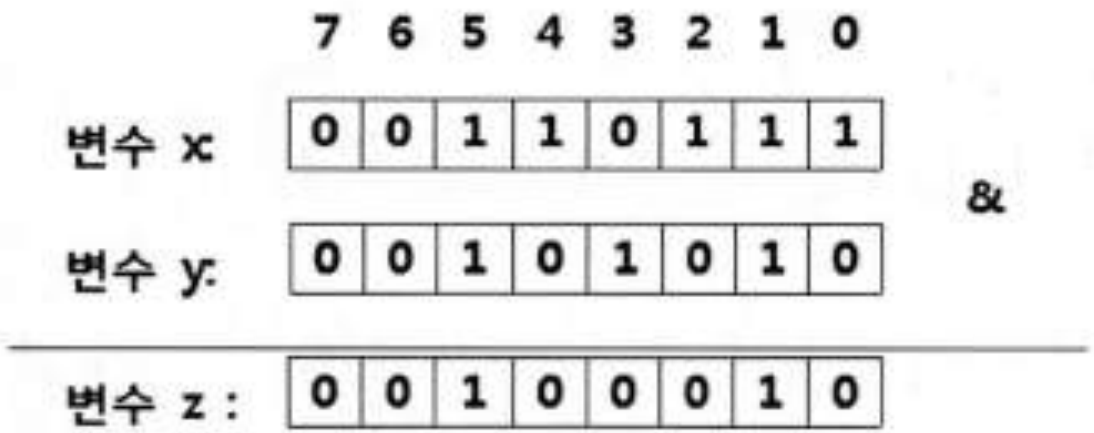
B1	B2	B1 & B2	B1 | B2	B1^B2	~B1
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

연산자와 표현식

>> 비트 연산자

```
unsigned char x = 0x37, y = 0x2A, z;
z = x & y;
printf ("%X : %d₩n", z, z);
```

[출력결과]
22 : 34



비트 연산자 예제 (비트 단위 &)

연산자와 표현식

>> 비트 연산자

- 비트이동 연산자
  - >> (비트 오른쪽 이동), << (비트 왼쪽 이동)
  - x << n 은 정수형 변수 x의 비트 패턴을 왼쪽으로 n비트만큼 이동시킴
  - 논리적 비트 이동과 산술적 비트 이동

```
short x = 0xFFA7, y;
unsigned short ux = 0xFFA7;
y = ux >> 4;
printf ("%X₩n", y);
y = x >> 4;
printf ("%X₩n", y);
x = 0x7FA7
y = x << 8;
printf ("%X₩n", y);
```

[출력결과]

0FFA

FFFA

A700

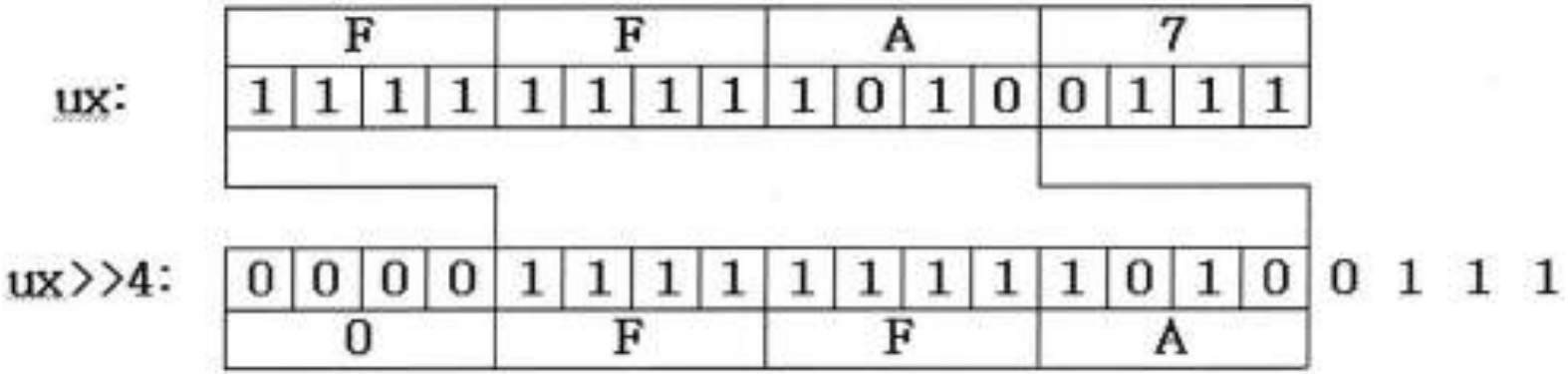
비트이동 연산자 예제

KYUNG HEE UNIVERSITY

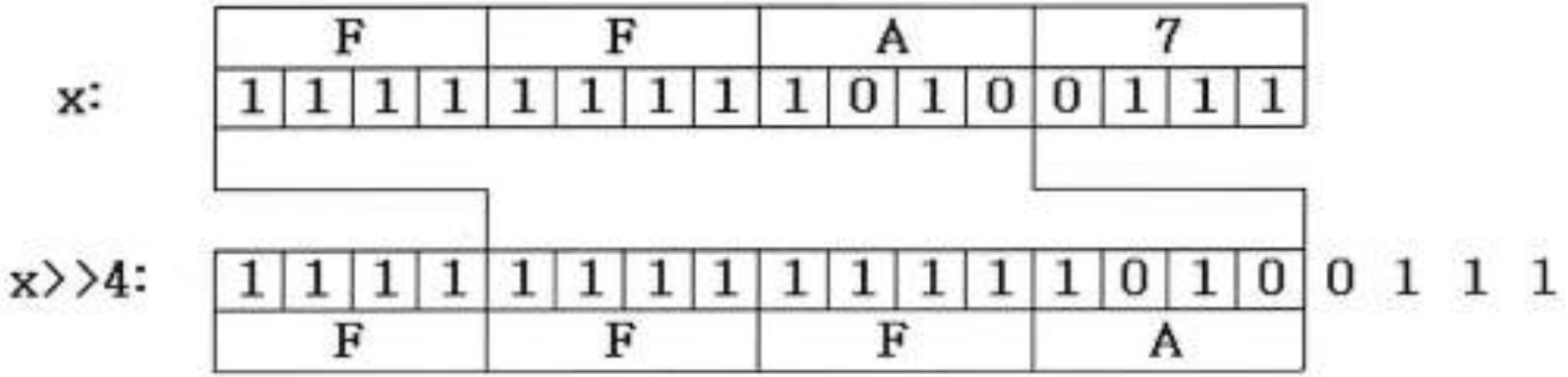
연산자와 표현식

>> 비트 연산자

- 비트 오른쪽 이동 (unsigned, 논리적 비트이동)



- 비트 오른쪽 이동 (signed, 산술적 비트이동)

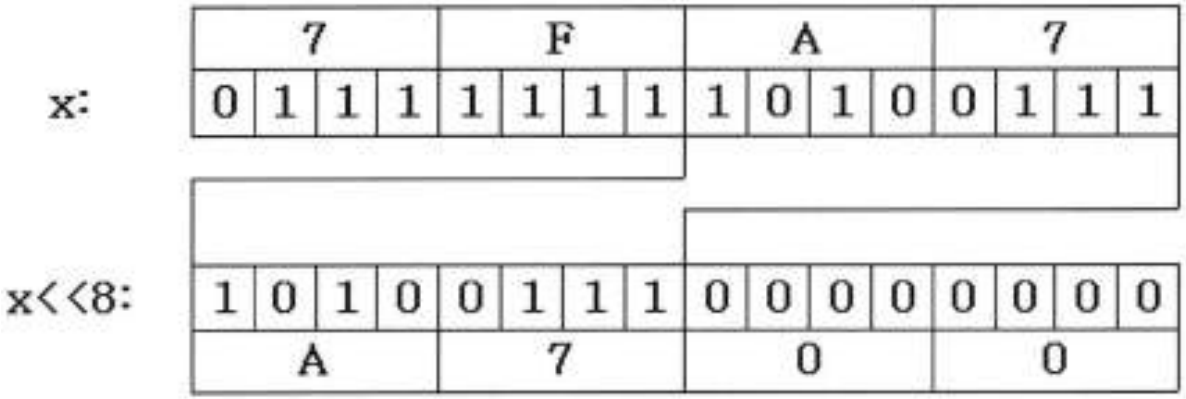


KYUNG HEE UNIVERSITY

연산자와 표현식

>> 비트 연산자

- 비트 왼쪽 이동



- 비트이동 연산의 수학적 의미
  - x >> n은 정수 x를 2n으로 나눈 것과 같은 효과
  - x << n은 정수 x에 2n을 곱한 것과 같은 효과

연산자와 표현식

기타 연산자

- 음의 부호 - (단항 연산자)
  ① 모든 signed 데이터 타입에 적용 가능
  ② 값의 부호를 바꿈

- 조건부 계산 (C의 유일한 삼항 연산자)
  ① condition ? exp1 : exp2
  ② condition이 참이면 exp1, 거짓이면 exp2가 계산됨

연산자와 표현식

>> 기타 연산자

```
int x = 137, y;
y = x % 2 == 0 ? x / 2 : x / 2 + 1;
printf ("%d₩n", y);

x % 2 == 0 ? (y = x / 2) : (y = x / 2 + 1);
printf ("%d₩n", y);

printf ("%d₩n", x % 2 == 0 ? x / 2 : x / 2 + 1);
```

[출력결과]

```
69
69
69
```

조건부 계산 예제

연산자와 표현식

>> 기타 연산자

- sizeof() 연산자 (단항 연산자)
  - ① sizeof(exp) 또는 sizeof exp
  - ② exp의 값을 저장하기 위해 필요한 메모리의 바이트 수 계산
  - ③ exp는 임의의 표현식 또는 데이터 타입이 될 수 있음

int x = 137;	**[출력 결과]**
printf ("%u₩n", sizeof x);	?(2 or 4)
printf ("%u₩n", sizeof 15L);	4
printf ("%u₩n", sizeof (x * 1.5));	8
printf ("%u₩n", sizeof (int));	?(2 or 4)

sizeof() 연산자 예제

연산자와 표현식

>> 기타 연산자

- cast 연산자 (단항 연산자)
  - ① (type) exp
  - ② exp에 할당된 값을 type에 명시된 데이터 타입으로 변환
  - ③ 대입연산이 수행 될 때는 자동타입변환이 일어남
  - ④ 타입변환 시, 소수점 아래 절단(truncation)에 유의

```
double x = 3.5;
int y = 7, z;
z = (int)x * y;
printf ("%d₩n", z);
double z = 3.26;
printf ("%.1lf₩n", (double)(int)(10. * (z + 0.05)) / 10.);
```

[출력결과]

21

3.3

cast 연산자 예제

연산자와 표현식

>> 기타 연산자

- 순차 연산자 (이항 연산자)
  - ① exp1, exp2
  - ② 먼저 exp1이 계산되고 나서 exp2가 계산됨

```
int x = 1, y = 2, z = 3;
while ((--x, --y, --z) > 0)
        printf ("%d, %d, %d₩n", x, y, z);
printf ("After the loop: %d, %d, %d₩n", x, y, z);
```

[출력결과]

```
0, 1, 2
-1, 0, 1
After the loop: -2, -1, 0
```

순차 연산자 예제