Lecture 02

변수, 데이터형, 연산자

리뷰

- C 언어는 빠른 속도, 작은 메모리, 다용도, 플랫폼 독립 프로그래밍 언어
- 컴파일러, 운영체제, 마이크로컨트롤러 등의 시스템 프로그래밍에 많이 쓰임

"*C is quirky, flawed, and an enormous success.*"

Ritchie

리뷰

- 변수 선언

```
int i; float f;
```

- 초기화

```
char c = 'a'; int x = y = 100;
```

- 연산자

```
+, -, *, /, %
```

- 표현

```
int x, y, z; x = y/2+z*5;
```

- 함수

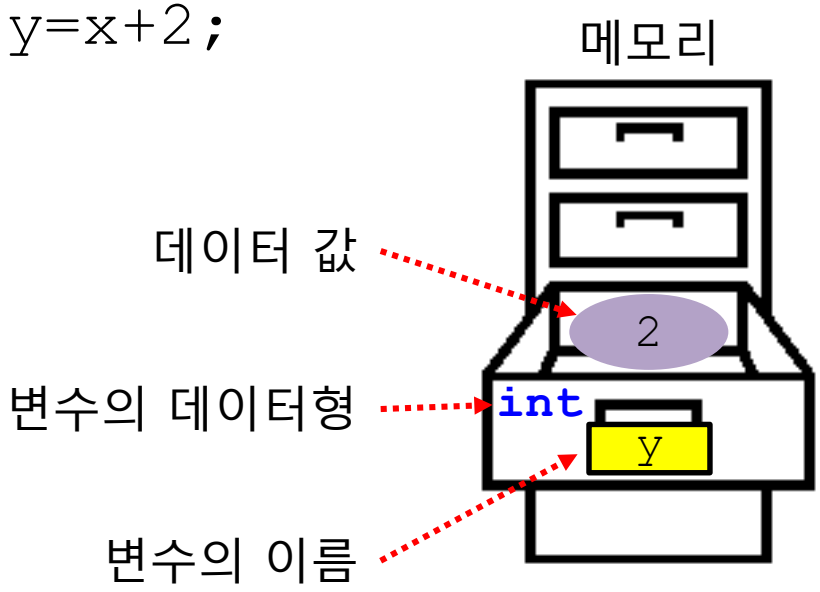
```
int square(int a);
```

정의

- 데이터형
  - 해당 객체는 메모리에 어떤 형식으로 저장되고, 어떤 값으로 할당되고, 어떻게 처리되어야 할 지를 명시적으로 알려줌
  - C는 약한 타입 언어(weakly typed language)이므로 연관이 없는 데이터형 간에 암시적 형변환을 허용함
    예, `char c = 'X'; int a = 1; // c+a의 값:'Y'`
- 연산자
  - 해당 객체는 어떻게 처리되어야 할 지를 알려줌
  - 연산자 종류
    - Unary : `-, ++`
    - Binary : `+, -, *, /, %`
    - Ternary : `?:`

정의

- 표현
  - 상수, 변수, 연산자, 함수 등을 포함하는 프로그램문
- 변수
  - 시스템 메모리에 저장되는 값 혹은 계산할 수 있는 표현과 연계되는 이름
- 예, `int x=0, y=0; y=x+2;`
  - `x, y`는 변수
  - `y=x+2`는 표현
  - `+`는 연산자



변수명

- 변수명 규칙
  - 문자, 숫자, 밑줄은 포함될 수 있음
  - 문자로 시작되어야 함
  - 키워드(**int**, **float**, **for**, **while**, …) **사용 금지**
  - 대소문자를 구분함
    예, int x; int X; // 변수 x와 변수 X는 다름
- 맞다/틀리다

```
int student_count;      맞다
int long;               틀리다: 키워드 사용 안 됨
int 2025class;          틀리다: 숫자로 시작 안 됨
int class2025;          맞다
int shopping_c@rt;      틀리다: 특수 문자 사용 안 됨
```

데이터형과 크기

- C의 기본적인 데이터형
  - 숫자 데이터형: **int**, **float**, **double**
  - 문자형: **char**
  - 사용자 정의(user-defined) 데이터형 또는 자료형: **struct**, **union**
- 문자(**char**)나 정수(**int**)에는 부호형(signed) 또는 무부호형(unsigned) 한정사(qualifier)가 있음
  - 무부호형 수는 항상 양수이며, 범위는 $[0, 2^n - 1]$ (n: 데이터형 크기)
    예, `unsigned char a; // 0 ≤ a ≤ 255`
  - 부호형 수는 음수가 될 수 있으며, 범위는 $[2^{-(n-1)}, 2^{n-1} - 1]$
    예, `char a; // -128 ≤ a ≤ 127`
- 문자형은 항상 1 byte를 차지하나, 숫자형은 하드웨어나 컴파일러에 따라 크기가 다를 수 있음

숫자 데이터형

	signed	unsigned
short	`short int x; short y;`	`unsigned short x;` `unsigned short int y;`
default	`int x;`	`unsigned int x;`
long	`long x;`	`unsigned int x;`
float	`float x;`	NA
double	`double x;`	NA
char	`char x; signed char x;`	`unsigned char x;`

- 부호형과 무부호형 문자는 산술 표현에 사용될 때만 차이가 보일 수 있음

Big endian vs. Little endian

- 데이터형 크기는 하드웨어나 컴파일러에 따르지만 다음과 같은 조건이 보증됨

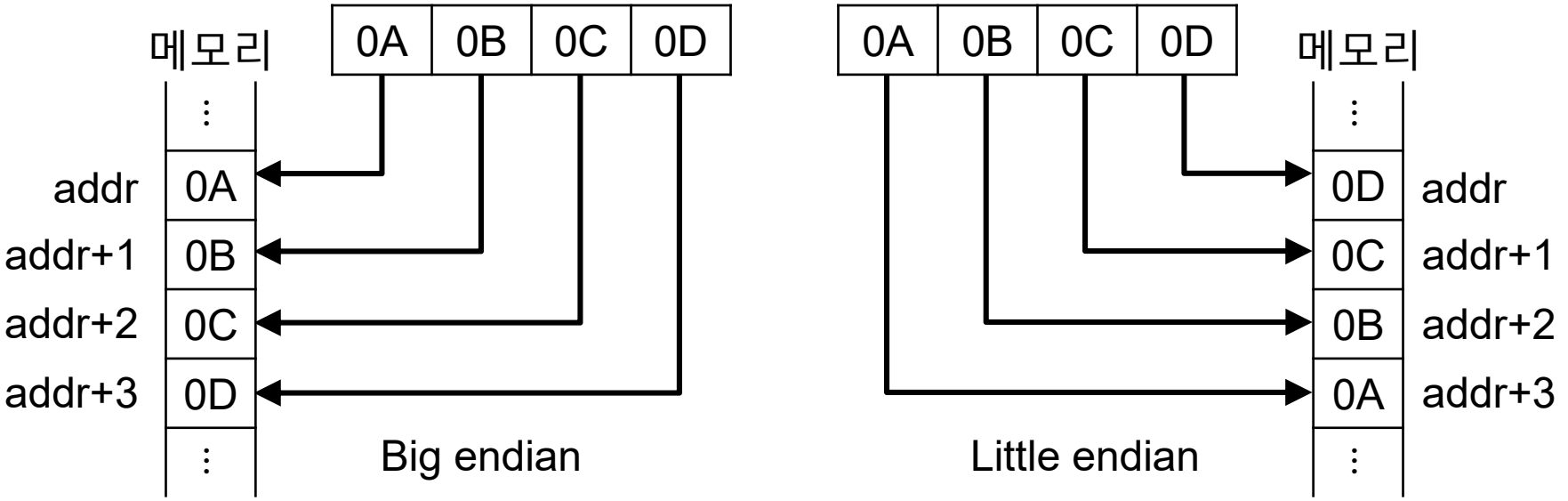
`sizeof(char)` < `sizeof(short)` ≤ `sizeof(int)` ≤ `sizeof(long)` and
`sizeof(char)` < `sizeof(short)` ≤ `sizeof(float)` ≤ `sizeof(double)`

- 문자형(**`char`**)는 1 byte : `sizeof(char) = 1`
- 숫자형(**`int`**, **`float`**, **`double`**)는 1 byte 이상을 차지함
  - 데이터형 크기는 여러 byte인 데 메모리에 어떻게 저장될까?
  - Big endian: 최상위 byte는 낮은 메모리 주소에 저장됨 (PowerPC 프로세서에 쓰임)
  - Little endian: 최하위 byte는 낮은 메모리 주소에 저장됨 (x86 프로세서에 쓰임)

Big endian vs. Little endian

- 예, `unsigned int x = 168496141;`

168496141 (10진수) = 0A0B0C0D (16진수)



상수

- 변수에 저장되는 값을 말하거나 수식 표현에 사용되는 값을 말함

데이터형	예	의미
정수	`int x=3;`	정수
	`long x=3;`	long 정수
	`unsigned long x=3UL;`	무부호 long 정수
	`int x=0xA;`	16진수
	`int x=012;`	8진수
실수	`float x=3.14159;`	부동소수점
	`float x=3.141F;`	
	`double x=3.141592653589793284L;`	배정도 부동소수점

상수

데이터형	예	의미
문자	`char x='A';`	문자
	`char x='\x41';`	16진수로 선언
	`char x='\0101';`	8진수로 선언
문자열	`char x[]="hello world";`	hello world 문장
	`char x[]="hello ""world";`	
열거 (enumeration)	`enum bool {NO, YES};`	NO=0, YES=1
	`enum COLOR {R=1,G,B,Y=10,P};`	R=1, G=2, B=3, Y=10, P=11

선언문

- 일반적인 포맷

  데이터형 [space] 변수이름 [=초기값];

  생략 가능

- 예,

```
char x; // 초기화하지 않음
char x='A'; // 'A'로 초기회함
char x='A', y='B'; // 여러 변수를 선언하고 초기화함
char x=y='A'; // 여러 변수를 선언하고 초기화함
```

퀴즈

- x>y? : `int x=0x19; int y=20;`
- 맞는가? : `unsigned char uc=-1;`
- 맞는 것 고르세요 : `printf("hello"+"world");` 및 `printf("hello""world");`
- XL=? : `enum sz {S=0,L=3,XL};`
- XL=? : `enum sz {S=0,L=-10,XL};`

산술 연산자

연산자	의미	예
+	더하기	`int x=3+2;`
		`int x=y+z;`
		`float x=y+2.513;`
–	빼기	`int x=3-2;`
		`int x=y-z;`
		`float x=y-z-4.369;`
*	곱하기	`int x=3*2;`
		`int x=y*z;`
		`float x=y*z*5.3;`

산술 연산자

연산자	의미	예
/	나누기	`int x=3/2; // 결과 1`
		`float x=3/2; // 결과 1.0`
		`float x=3.0/2; // 결과 1.5`
%	나머지	`int x=3%2; // 결과 1`
		`int x=7; int y=x%4; // 결과 3`
		`int x=7; int y=x%10; // 결과 7`

- 나머지는 float나 double에는 적용될 수 없으며, 음수에 적용할 때의 결과는 컴파일러에 따라 다름

관계 연산자

- 두 값을 비교하여 참(true) 또는 거짓(false)을 반환하는 연산자임
  - 참(true): 0이 아닌 모든 값
  - 거짓(false): 0을 의미함

연산자	의미	예
>	왼쪽 값이 더 큰지 비교	`3>2; // 참(1)`
		`2.67>3; // 거짓(0)`
>=	왼쪽 값이 크거나 같은지 비교	`3>=3; // 참(1)`
		`2.99>=3; // 거짓(0)`
<	왼쪽 값이 더 작은지 비교	`'A'<'B'; // 참(1)`
		`3.1<3; // 거짓(0)`

관계 연산자

- 등호 연산자를 사용할 때 **유의 사항**
  - 등호 연산자(==)와 대입 연산자(=)가 다름
  - 부동소수점에 등호 연산자를 사용하지 않으면 더 나음
    예, `float x=6.7; x==6.7; // 거짓(0)`
    `double x=6.7; x==6.7; // 참(1)`

연산자	의미	예
<=	왼쪽 값이 더 작거나 같은지 비교	`3<=2; // 거짓(0)`
		`2.67<=3; // 참(1)`
==	두 값이 같은지 비교	`3==3; // 참(1)`
		`'A'=='a'; // 거짓(0)`
!=	두 값이 다른지 비교	`'A'!='B'; // 참(1)`
		`3!=3; // 거짓(0)`

논리 연산자

- 관계 연산자로 연결된 수식은 왼쪽에서 오른쪽으로 계산됨
- 참 또는 거짓이 판정되면 곧바로 검사를 마침
  - `(3==3) || ((c=getchar())=='A');` → 계산 진행되지 않음
  - `(0) && ((x=x+1)>0);` → 계산 진행되지 않음

연산자	의미	예
&&	AND	`(3==3)&&('A'=='a');` // 거짓(0)
		`(6/2==3)&&(4*2==8);` // 참(1)
||	OR	`3==3 || 'A'=='a';` // 참(1)
		`2>=5 || 0;` // 거짓(0)
!	NOT(부정)	`!(2.87>=3);` // 참(1)
		`!(5<9);` // 거짓(0)

증가/감소 연산자

- 증가 연산자 ++는 오퍼랜드에 1을 증가시킴
- 감소 연산자 --는 오퍼랜드에 1을 감소시킴
- 증가/감소 연산자는 변수의 앞에도 쓸 수 있고, 뒤에도 쓸 수 있음
  - `++x; // x이 사용되기 전에 증가시킴`
  - `x++; // x이 사용 후에 증가시킴`
- 예, `x=5;`

```
y = x++; // y=5
y = ++x; // y=6
y = x--; // y=5
y = --x; // y=4
```

비트 연산자

연산자	의미	예
&	AND	0x77&0x03; // 결과 0x03
		0x77&0x00; // 결과 0x00
|	OR	0x77|0x03; // 결과 0x77
		0x700|0x003; // 결과 0x703
^	XOR	0x770^0x773; // 결과 0x003
		0x55^0x55; // 결과 0x00
<<	왼쪽 시프트	0x01<<2; // 결과 0x04
		1<<4; // 결과 16
>>	오른쪽 시프트	0x10>>4; // 결과 0x01
		8>>2; // 결과 2

지정 연산자

- C에는 다음과 같은 수식 표현을 많이 보임

```
x = x+1; x = x*5; x = x/2; x = x%3; …
```

- 다음의 압축형을 사용할 수 있음

```
x+=1; x*=5; x/=2; x%=3; …
```

- 대부분 binary 연산자에 해당하는 지정 연산자가 있음

```
+ - * / % << >> & ^ |
```

- 지정 연산자를 사용할 때 **유의 사항**

`x*=y+1`는

`x=x*(y+1)`을 의미함

`x=x*y+1`을 의미하는 것이 아님

즉, `*=`은 `+`보다 우선순위가 낮음

조건문

- 조건문은 거의 모든 프로그래밍 언어에 많이 쓰인 것임

```
if (조건) {
    프로그램문;
} else {
    프로그램문;
}
```

```
if (조건)
    프로그램문;
else
    프로그램문;
```

- **if else** 속에 프로그램문이 하나이면 괄호 생략 가능
  - 이 경우 조건문을 **ternary 연산자(?:)**로 대체할 수 있음

```
z = (a>b) ? a : b;
if (a>b)
    z = a;
else
    z = b;
```

```
sign = (a>0) ? 1 : -1;
if (a>0)
    sign = 1;
else
    sign = -1;
```

형변환

- 형이 다른 대상에 연산을 하면 형변환이 일어남
- 자동변환
  - 정수 → 실수

```
int i; float f; f = i+3.1; //더하기 전에 i는 float로 변환됨
```

  - 실수 → 정수: 손실이 일어날 수 있음

```
int i; float f; i = f+1.2; //더하기 결과는 int로 변환됨
```

  - 문자형 → 정수형: 문자 또는 문자열을 처리할 때 편리함

```
isupper = (c>='A' && c<='Z') ? 1 : 0;
if (!isupper)
    c = c-'a'+'A';
```

우선순위와 계산순서

연산자	연산순서
() [] -> .	좌에서 우
! ~ ++ -- + - * & (type) sizeof	우에서 좌
* / %	좌에서 우
+ -	좌에서 우
<< >>	좌에서 우
< <= > >=	좌에서 우
== !=	좌에서 우
&	좌에서 우
^	좌에서 우
|	좌에서 우
&&	좌에서 우
||	좌에서 우
?:	우에서 좌
= += -= *= /= %= &= ^= |= <<= >>=	우에서 좌
,	좌에서 우

우선순위를 기억하지 말고 **괄호를 사용**하는 것을 권장드림