전산 SMP 2주차

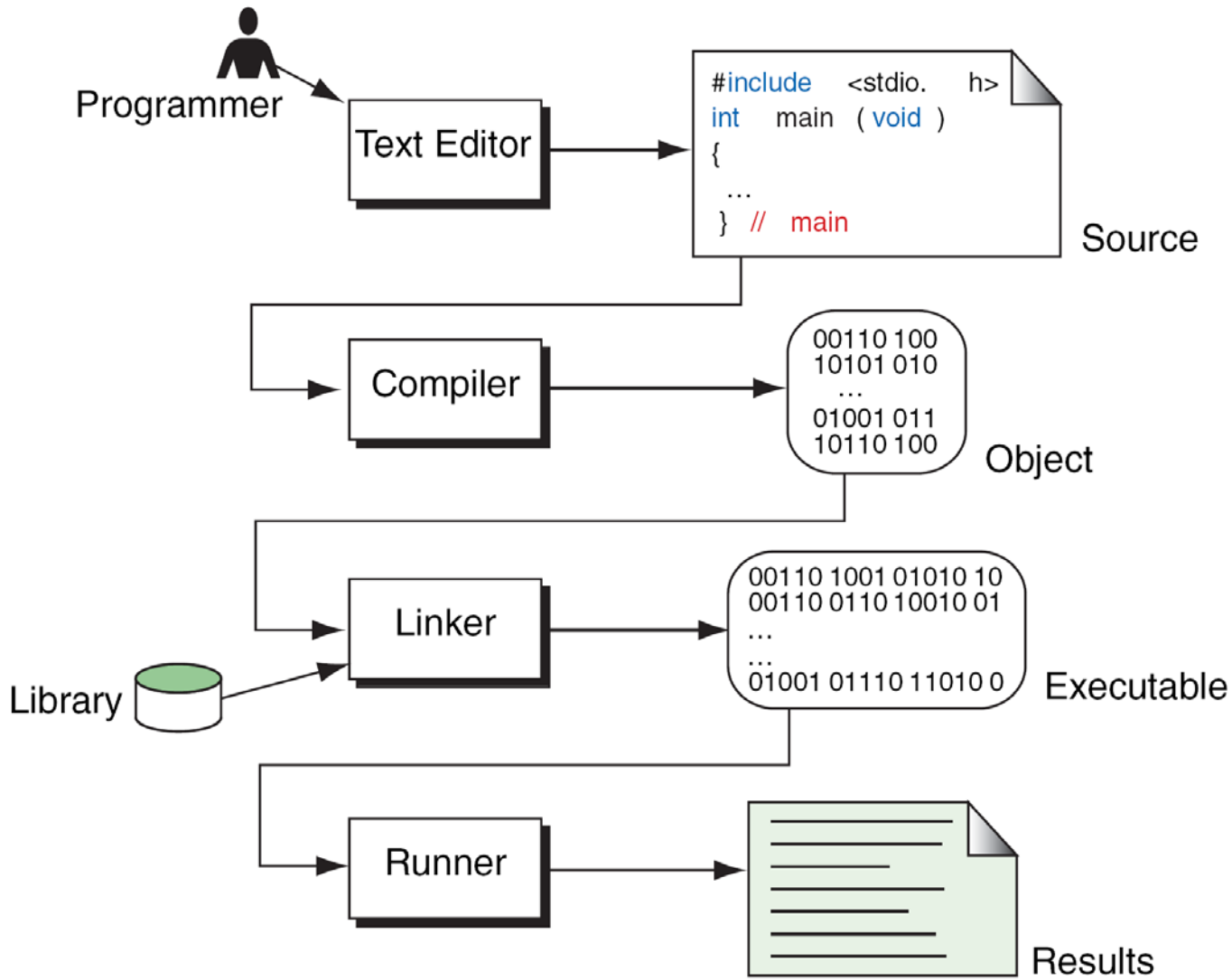
2014. 03. 21

김범수

bskim45@gmail.com

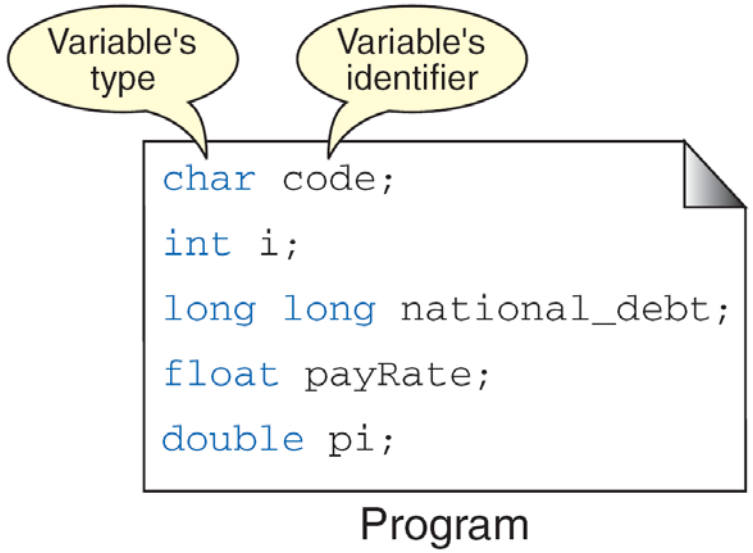
Special thanks to 박기석 (kisuk0521@gmail.com)

지난시간 복습


Programmer
Text Editor
#include <stdio. h>
int main (void)
{
…
} // main
Source
Compiler
00110 100
10101 010
…
01001 011
10110 100
Object
Linker
00110 1001 01010 10
00110 0110 10010 01
…
…
01001 01110 11010 0
Executable
Library
Runner
Results

변수란?

- 값을 저장할 수 있는 메모리공간에 붙여진 이름
- 변수라는것을 선언하면 메모리 공간이 할당되고 할당된 메모리 공간에 이름이 붙는다.
- 일반변수와 포인터 변수
- const키워드 사용 : 변수를 상수화시킨다.



int num; // int : 정수의 저장을 위한 메모리 공간의 할당
num : 할당된 메모리 공간의 이름은 num

num=20; //변수 num에 접근하여 20을 저장

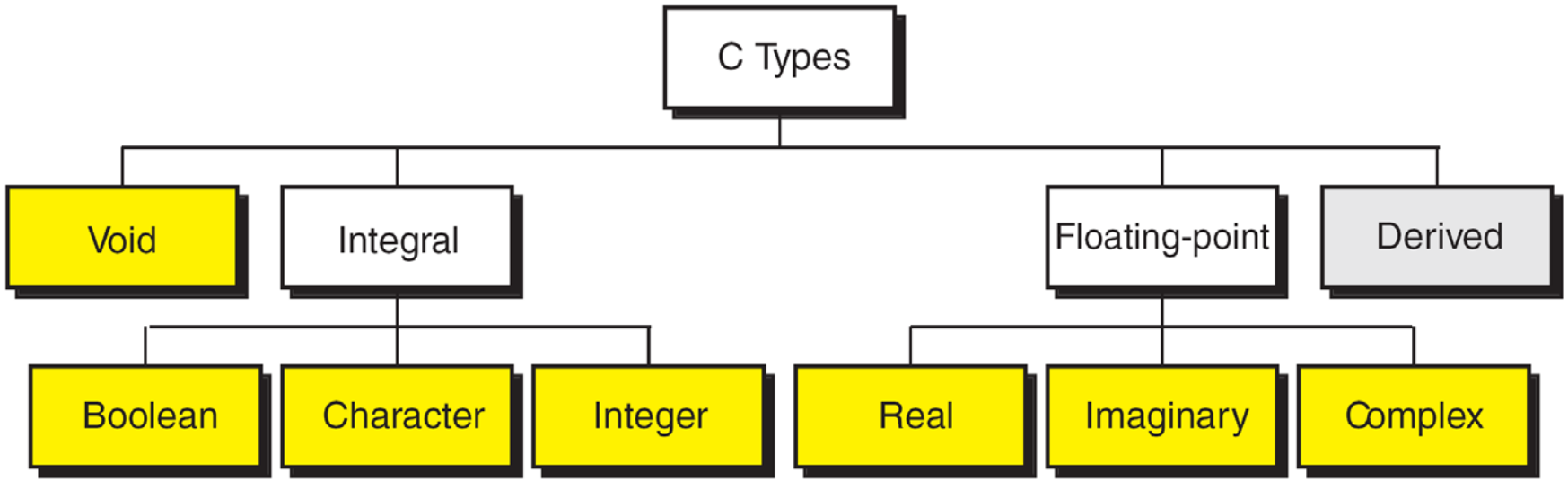
printf("%d", num); // num 변수안의 값을 정수형태로 출력

#define

- Preprocessor Directive 부분에 선언
- 컴파일 시 자동으로 샥샥 대체되어 들어간다.

```
#define PI 3.141592     //세미콜론 안 붙임
….
float pi = PI;          //float pi=3.141592;
```

자료형


C Types
Void
Integral
Floating-point
Derived
Boolean
Character
Integer
Real
Imaginary
Complex

기본 자료형의 종류와 데이터의 표현 범위

자료형		크기	값의 표현범위
정수형	char	1byte	−128 ~ 127
	unsigned char	1byte	0~255
	short	2byte	−32768~32767
	unsigned short	2byte	0~65535
	int	4byte	−2,147,483,648~2,147,483,647
	unsigned int	4byte	0~4,294,967,294
	long	4byte	−2,147,483,648~2,147,483,647
	unsigned long	4byte	0~4,294,967,294
	long long	8byte	−9,223,372,036,854,775,808~9,223,372,036,854,775,808
	unsigned long long	8byte	0~위 숫자의 2배
실수형	float	4byte	$\pm 3.4 \times 10^{-37}$ ~ $\pm 3.4 \times 10^{+38}$
	double	8byte	$\pm 1.7 \times 10^{-307}$ ~ $\pm 1.7 \times 10^{+308}$
	long double	8byte이상	double이상의 표현범위

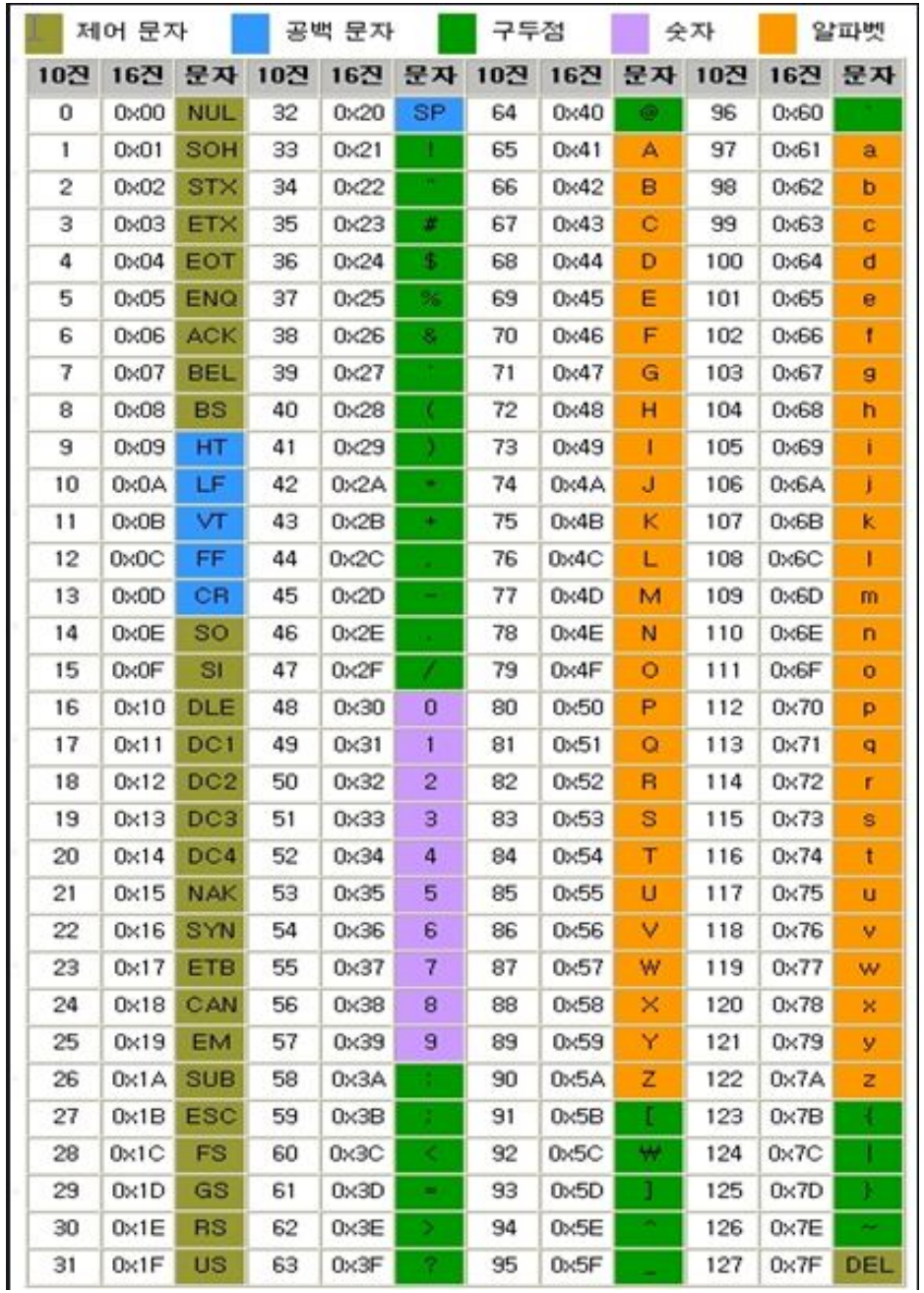
문자형(char)

- 1byte = 2^8 = 256개의 문자
- 각각의 문자가 아스키 코드에 대응!

아스키 코드

- C의 char에서는 ASCII코드를 이용해 나타냅니다. 문자 하나당 하나의 수로 대응.

- 소문자 a(97), b(98), c(99), …, z(112)
- 대문자 A(65), B(66), C(67), …, Z(90)
- 숫자 0(48), 1(49), 2(50), …, 9(57)
- 특수문자 &(38), *(42), +(43)



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

형변환 (Type Conversion)

- Implicit Conversion
  - 컴파일러에 의해 사용자 모르게 자동으로 이루어 지는 형변환
  - Promotion, Demotion : 연산시 우선순위 높은 거에 맞춰서 다른 데이터 변환 (bool 〈 char 〈 integer 〈 real)
  - 최종적으로 값 대입시 대입 받는 변수의 타입 따라 감.
- Explicit Conversion(cast)
  - 프로그래머가 인위적으로 타입을 바꿈. 사용시 주의할것!

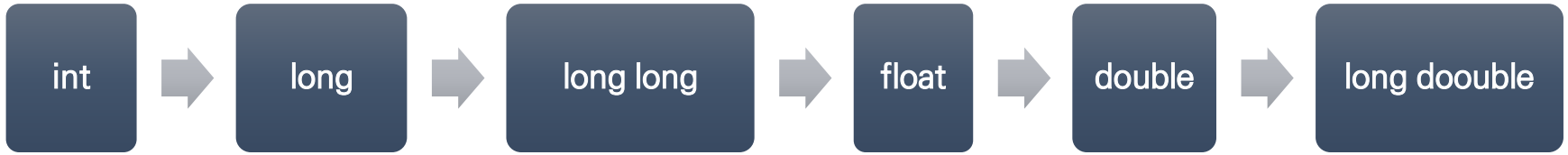
  double a = (double) 4/3

  double b = (double) 4 / (double) 3

  a=? b=?

자동 형 변환 (산술연산에서의 형 변환 규칙)

```
double a=1.7 + 30; // a=31.7
int b=1.7 + 30; // b=31

main()
{
        int a=-1;
        unsigned int b=100;
        if(a>b)
                printf("a가 크다");
        else
                printf("b가 크다");
}
```



printf

```
Int a = 1;
Int b = 2;
Int c = 3;

Printf("%d  %d  %d\n", a, b, c);
```

서식문자(Conversion Specifier) : 데이터의 출력 형태를 지정할 수 있다.
%d : 부호있는 10 진수로 출력하라!

scanf

```
int a;
char b;

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

Scanf로 값을 받기 위해서는 받는 위치, 즉 주소값을 인자로 주어야 한다.
& 가 변수 이름 앞에 붙을 시 그 변수가 가리키는 주소를 의미한다.

a : 변수 이름, 메모리 공간에 붙은 가상의 이름
&a : 변수 a 의 주소

서식문자 옵션은 precision, flag, size 값 줄 수 없다. 오직 width만 가능

scanf

- 첫번 째 인자의 서식문자와 뒤의 변수가 대응해야 한다.
  - 갯수, 타입
- 변수에 값을 넣기위해서는 변수의 주소! 를 주어야 한다.
  - &a
- 첫번째 인자(format string)에서 공백, 서식문자를 제외한 문자가 있으면 입력시 형태를 맞춰줘야 한다.
  - “%d-%d” -〉 1-2
- Format string 마지막에 공백이 있으면 안된다.!
  - “%d %d ” X -〉 “%d %d” O

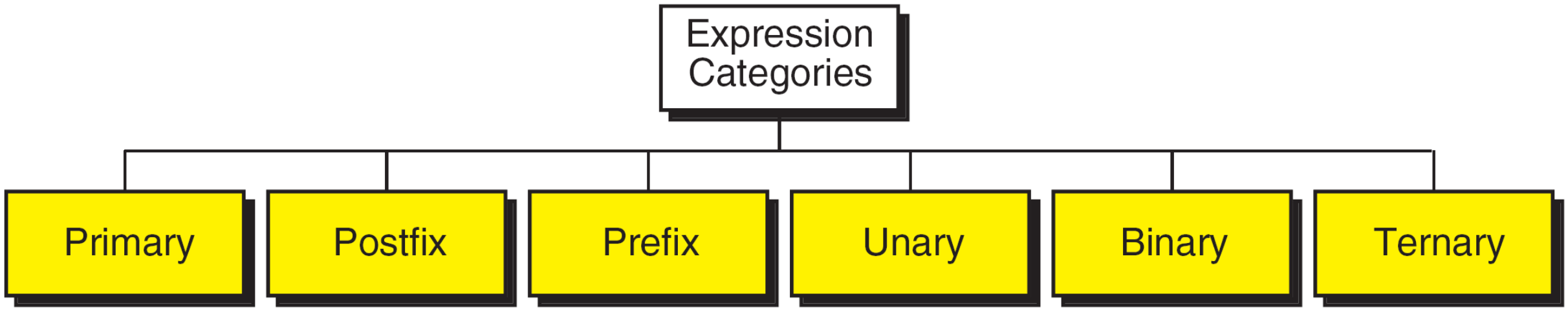
printf()

```
#include <stdio.h>

int main(void)
{
   printf(“%c %d %s \n”, ‘A’, ‘A’, “ABC”);
   printf(“%10d \n”,123); // 오른쪽정렬
   printf(“%-10d \n”,123); // 왼쪽 정렬
   printf(“%10.2f \n”,123.456789); // 왼쪽 정렬 ****123.46(*은 빈칸)
   printf(“%+f %+f\n”,123.45,-123,45); // +123.45 -123.45
   printf(“% f % f\n”, 123.45,-123,45); // *123.45 -123.45(*은 빈칸)
   printf(“%-6s %6s \n", “AAA", “BBB");
}
// 소수점 아래는 6자리까지만 문제가 없고 , 7자리부터 문제 발생
```

scanf()

```
#include <stdio.h>

int main(void)
{
   int num1, num2, num3;
   printf("세 개의 정수 입력: ");
   scanf("%d %o %x", &num1, &num2, &num3); //12 12 12

   printf("입력된 정수 10진수 출력: ");
   printf("%d %d %d \n", num1, num2, num3); // 12 10 8
   return 0;
}
```

연산자


Expression Categories
Primary
Postfix
Prefix
Unary
Binary
Ternary

Primary Expressions

- Names
  - a, b12, price, INT_MAX, SIZE
- Literal constants
  - 5, 123.98, ‘A’, “Welcome”
- Parenthetical expressions
  - (2 * 3 + 4), (a = 23 + b * 6)

Prefix/Postfix/Unary

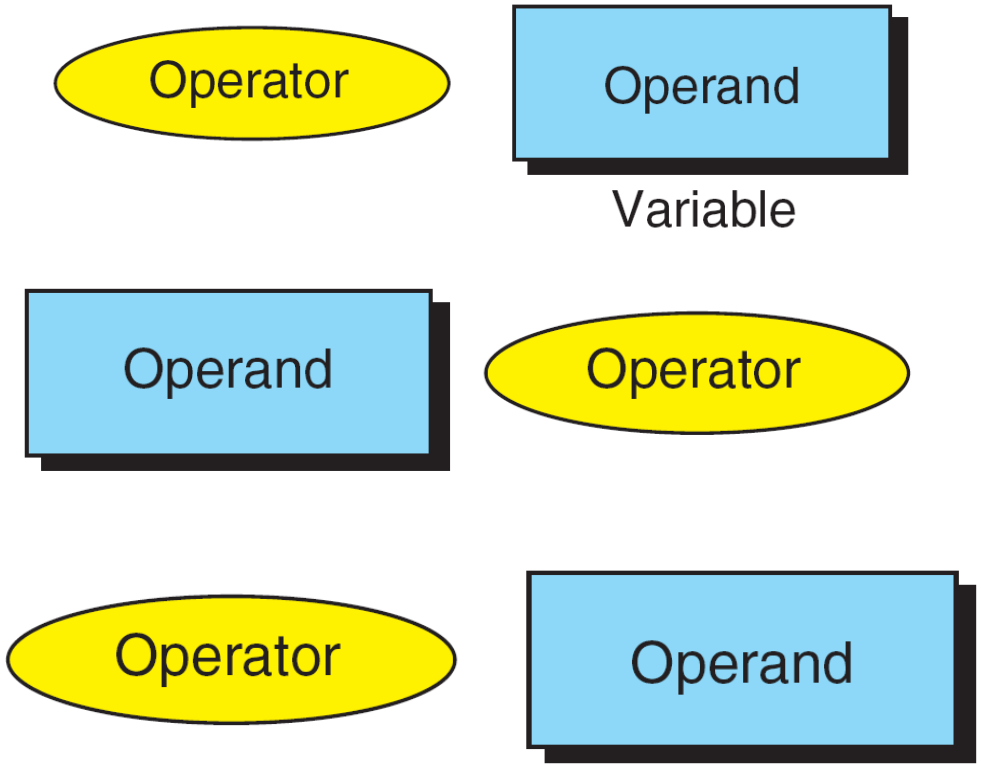
Prefix

- 증가/감소 연산자: --a, ++a

Postfix

- 증가/감소 연산자: ++a, --a
- 함수 호출

Unary



연산자의 종류

- 산술연산자 : +,−, *, /, %
- 대입연산자 : =, +=, −=, *=, /=, %=, 〈〈=, 〉〉=, &=, ^=, |=
- 관계(비교)연산자 : 〉, 〈, 〉=, 〈=, ,!=
- 논리연산자 : &&, ||, !
- 증가/감소 연산자 : ++, −−
- 비트연산자 : &, |, ^, ~
- 시프트 연산자 : 〈〈, 〉〉
- 주소 연산자 : &

산술연산자

종 류	사용 방법	설 명
+	a + b	두 개의 피연산자들을 덧셈
−	a − b	두 개의 피연산자들을 뺄셈
*	a * b	두 개의 피연산자들을 곱셈
/	a / b	변수 a를 변수 b로 나눗셈
%	a%b	변수 a를 변수 b로 나누어 나머지를 구함

```
#include <stdio.h>
int main(void)
{
    int a=22, b=5;
    printf("%d + %d = %d \n", a, b, a+b); // 22+5=27
    printf("%d - %d = %d \n", a, b, a-b); // 22-5=17
    printf("%d × %d = %d \n", a, b, a*b); // 22 x 5=110
    printf("%d ÷ %d의 몫 = %d \n", a, b, a/b);  // 22÷5=4
    printf("%d ÷ %d의 나머지 = %d \n", a, b, a%b);// 22÷5=2
}
```

복합 대입연산자

종 류	사용방법	설 명
a+=b	a=a+b	a+b 의 값을 a에 대입
a-=b	a=a-b	a-b 의 값을 a에 대입
a*=b	a=a*b	a*b 의 값을 a에 대입
a/=b	a=a/b	a/b 의 값(몫)을 a에 대입
a%=b	a=a%b	a%b 의 값(나머지)을 a에 대입

```
#include <stdio.h>
int main(void)
{
    int a=7, b=2,c=10;
    a+=3;
    b*=3;
    c%=3;
    printf("결과 : %d ,%d , %d \n", a, b, c); // 결과 : 10, 6 , 1
}
```

비교(관계) 연산자

종 류	사용방법	설 명
>	a>b	변수 a가 변수 b보다 크면 참, 작거나 같으면 거짓
<	a<b	변수 a가 변수 b보다 작으면 참, 크거나 같으면 거짓
>=	a>=b	변수 a가 변수 b보다 크거나 같으면 참, 작으면 거짓
<=	a<=b	변수 a가 변수 b보다 작거나 같으면 참, 크면 거짓
==	a==b	변수 a와 변수 b가 같으면 참, 같지 않으면 거짓
!=	a!=b	변수 a와 변수 b가 같지 않으면 참, 같으면 거짓

비교(관계) 연산자

```
#include <stdio.h>

int main(void)
{
    int a=10;
    int b=12;
    int result1, result2, result3;

    result1=(a==b);
    result2=(a!=b);
    result3=(a>=b);

    printf("result1: %d \n", result1);  // result1 : 0
    printf("result2: %d \n", result2);  // result2 : 1
    printf("result3: %d \n", result3);  // result3 : 0
    return 0;
}
```

논리연산자

종 류	사용 방법	설 명
&&	a && b	a와 b가 모두 참인 경우에만 참, 둘 중 하나라도 거짓이면 거짓
||	a || b	a와 b중에 하나만 참이거나 둘 모두 참인 경우에는 참, 둘 다 거짓인 경우에는 거짓
!	!a	a가 참인 경우에는 거짓, a가 거짓인 경우에는 참

논리연산자

```
#include <stdio.h>
int main(void)
{
    int a=10;
    int b=12;
    int result1, result2, result3;

    result1 = (a==10 && b==12);
    result2 = (a<12 || b>12);          //  result2 = (a<(12 || b)>12);  은 결과 0
    result3 = (((!a)));
    printf("result1: %d \n", result1);  // 1
    printf("result2: %d \n", result2);  // 1
    printf("result3: %d \n", result3);  // 0
    return 0;
}
```

증가/감소 연산자

종 류	사용방법	설 명
++	++a	값을 1증가 후 나머지 연산 진행
	a++	++이외의 연산을 먼저 진행 후 값을 1증가
--	--a	값을 1감소 후 나머지 연산 진행
	a--	--이외의 연산을 먼저 진행 후 값을 1감소

```
#include <stdio.h>

int main()
{
        int a=0, b=0;
        while(a<3){
                printf("a=%d  b=%d\n", a++, ++b);
}
```

증감 연산자 예제

```
 5  #include <stdio.h>
 6  int main (void)
 7  {
 8  // Local Declarations
 9     int a;
10
11  // Statements
12     a = 4;
13     printf("value of a     : %2d\n", a);
14     printf("value of ++a   : %2d\n", ++a);
15     printf("new value of a: %2d\n", a);
16     return 0;
17  }  // main
```

```
Results:
value of a     :  4
value of ++a   :  5
new value of a:  5
```

```
 5  #include <stdio.h>
 6  int main (void)
 7  {
 8  // Local Declarations
 9     int a;
10
11  // Statements
12     a = 4;
13     printf("value of a     : %2d\n", a);
14     printf("value of a++   : %2d\n",   a++);
15     printf("new value of a: %2d\n\n", a);
16     return 0;
17  }  // main
```

```
Results:
value of a     :  4
value of a++   :  4
new value of a:  5
```

비트 연산자

연산자	종류	사용방법	설 명
비트 논리 연산자	&	a&b	변수 a와 변수b의 비트들을 비교하여 각 비트가 모두 1인 경우에만 1, 그렇지 않으면 0
	|	a|b	변수 a와 변수b의 비트들을 비교하여 각 비트 중 하나라도 1인 경우에만 1, 그렇지 않으면 0
	^	a^b	변수 a와 변수 b의 비트가 다른 경우에는 1, 같으면 0
	~	~a	변수a의 비트가 1인 경우에는 0, 0인 경우는 1
시프트 연산자	〈〈	a〈〈3	a의 비트들을 왼쪽으로 3비트만큼 이동 1비트만큼 이동할 때마다 a*2만큼 증가
	〉〉	a〉〉3	a의 비트들을 오른쪽으로 3비트만큼 이동 1비트만큼 이동할 때마다 a/2만큼 감소

Shift 대상		공백 메우기
〈〈 (Left Shift)	signed	0으로 채워짐
	unsigned	
〉〉 (Right Shift)	signed	Sign bit의 값으로 채워짐
	unsigned	0으로 채워짐

비트 연산자–1

```
#include <stdio.h>

int main(void)
{
   int a = 14;         // 00000000 00000000 00000000 00001110
   int b = 20;         // 00000000 00000000 00000000 00010100
   int c = a & b;
   int d = a | b;
   int e = a^b;
   int f = ~a;
   printf("연산의 결과: %d %d %d \n", c, d, e, f);  // 4, 30, 26 , -15
}
```

비트 연산자–2

```
#include <stdio.h>

int main(void)
{
    int a=15;      // 00000000 00000000 00000000 00001111
    int b=-15;     // 11111111 11111111 11111111 11110000
    int c=15;      // 11111111 11111111 11111111 11110000

    int result = a<<2;   // a의 비트 열을 왼쪽으로 2칸씩 이동
    printf("%d \n", result);  // 60
    printf("%d \n", b>>2);  // -4
    printf(" %d \n", c>>2);  // 3
    return 0;
}
```

sizeof 연산자

- **sizeof : 변수 또는 자료형의 크기를 byte수로 알려준다.**

```
#include <stdio.h>
int main(void)
{
    char ch='A';
    int a=987;
    double b=3.1415;
    printf("변수 ch의 크기: %d \n", sizeof(ch));
    printf("변수 a의 크기: %d \n", sizeof(a));

    printf("int의 크기: %d \n", sizeof(int));
    printf("float의 크기: %d \n", sizeof(float));
    printf("double의 크기: %d \n", sizeof(double));

    printf("'A'의 크기: %d \n", sizeof('A'));
    printf("\"ABC\"의 크기: %d \n", sizeof("ABC"));
    printf("3의 크기: %d \n", sizeof(3));
    printf("1.234의 크기: %d \n", sizeof(1.234));
}
```

sizeof 연산자

```
#include <stdio.h>

int main(void)
{
    char c1=1, c2=2, result1=0;
    short s1=300, s2=400, result2=0;
    float f1=1.2, f2=3.4;
    printf("c1 & c2: %d, %d \n", sizeof(c1), sizeof(c2)); // 1,1
    printf("s1 & s2: %d, %d \n", sizeof(s1), sizeof(s2)); // 2,2
    printf("f1 & f2: %d, %d \n", sizeof(f1), sizeof(f2));  // 4,4

    printf("sizeof(c1+c2): %d \n", sizeof(c1+c2));  // 4
    printf("sizeof(c1)+sizeof(c2): %d \n", sizeof(c1)+sizeof(c2));
    // 2
    printf("sizeof(s1+s2): %d \n", sizeof(s1+s2));  // 4
    printf("sizeof(s1)+sizeof(s2): %d \n", sizeof(s1)+sizeof(s2)); //
    4
    printf("sizeof(f1+f2): %d \n", sizeof(f1+f2));  // 4
    printf("sizeof(f1)+sizeof(f2): %d \n", sizeof(f1)+sizeof(f2));  //
    8

    result1=c1+c2;
    result2=s1+s2;
    printf("result1 & result2: %d, %d \n", sizeof(result1),
    sizeof(result2)); // 1,2
}
```

삼항 연산자

- 조건식 ? 실행1 : 실행2

조건식이 참이면 실행1 문장을,

조건식이 거짓이면 실행 2문장을 수행

```
#include <stdio.h>

int main(void)
{
	int num=1, res;

	res = (num==1) ? ++num : --num;
	printf("결과: %d ₩n", res);  // 2
}
```

Try

- Int a = 2;
- Int b = 4;
- Int num = 3;

- a *= (num−−) == 4 || a+2 〈= 5−1 && ++b 〉=3;

- a=? num=? b=?

연산자 우선순위와 결합순서

우선 순위	연산 기호	연산자	결합방향
1위	()	함수호출	→
	[]	인덱스	
	->	간접지정	
	.	직접지정	
2위	++	증가	←
	--	감소	
	sizeof	바이트 단위 크기 계산	
	~	비트단위 NOT	
	!	논리 NOT	
	-,+	부호 연산(음수, 양수)	
	&	주소 연산	
	*	간접 지정 연산	
3위	(casting)	자료형 변환	←
4위	*,/,%	곱셈, 나눗셈,나머지	→
5위	+,-	덧셈, 뺄셈	→
6위	<<,>>	비트이동	→
7위	<,>,<=,>=	대소비교	→
8위	==,=!	동등비교	→
9위	&	비트 AND	→
10위	^	비트 XOR	→
11위	|	비트 OR	→
12위	&&	논리 AND	→
13위	||	논리 OR	→
14위	? :	조건 연산	←
15위	=,+=,-=,*,/=,%=,<,<=,>>=,&=,^=,|=	대입 연산	←
16위	,	콤마연산	→

연산자 우선순위

- 기본적으로 왼쪽에서 오른쪽으로

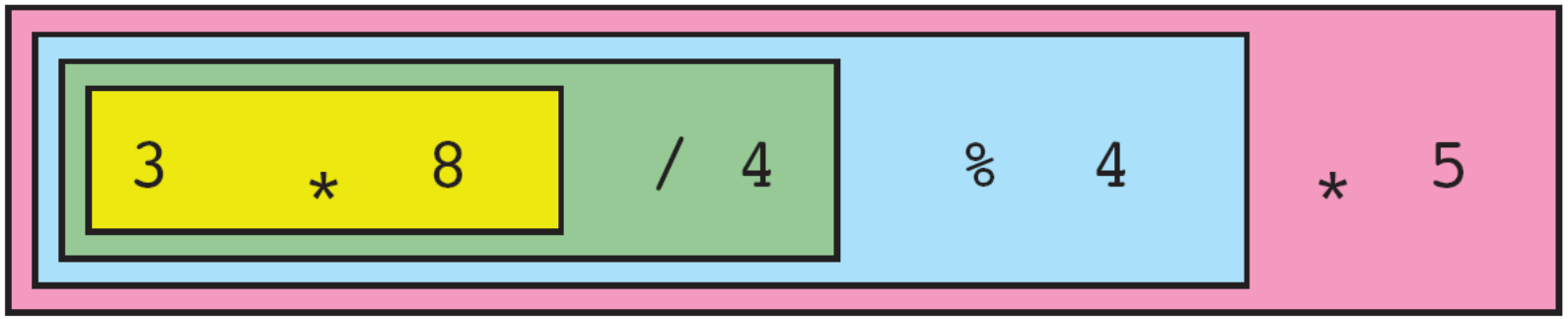
- A+B*C/D-E
  1. *
  2. /
  3. +
  4. -
- 단, 대입 연산자의 경우 오른쪽에서 왼쪽으로
- A = b = C+4
- +먼저 계산하고 b에 c+4를 먼저 대입하고, 그 다음에 A에 b를 대입합니다.

```c
#include <stdio.h>

int main (void)
{
// Local Declarations
   int a = 10;
   int b = 20;
   int c = 30;

// Statements
   printf ("a *  b + c  is: %d\n", a *  b + c);
   printf ("a * (b + c) is: %d\n", a * (b + c));
   return 0;
}  // main
```

```
Results:
a *  b + c  is: 230
a * (b + c) is: 500
```

연산자 결합순서

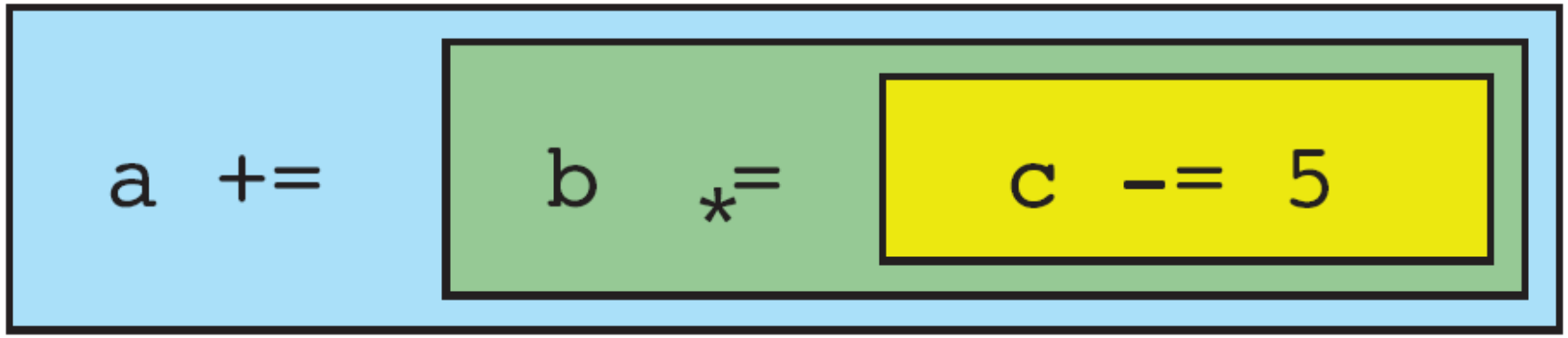
- a + b + c + d
- a * b * c * d



Left-to-Right Associativity

연산자 결합순서

- a = b = c = d = 10

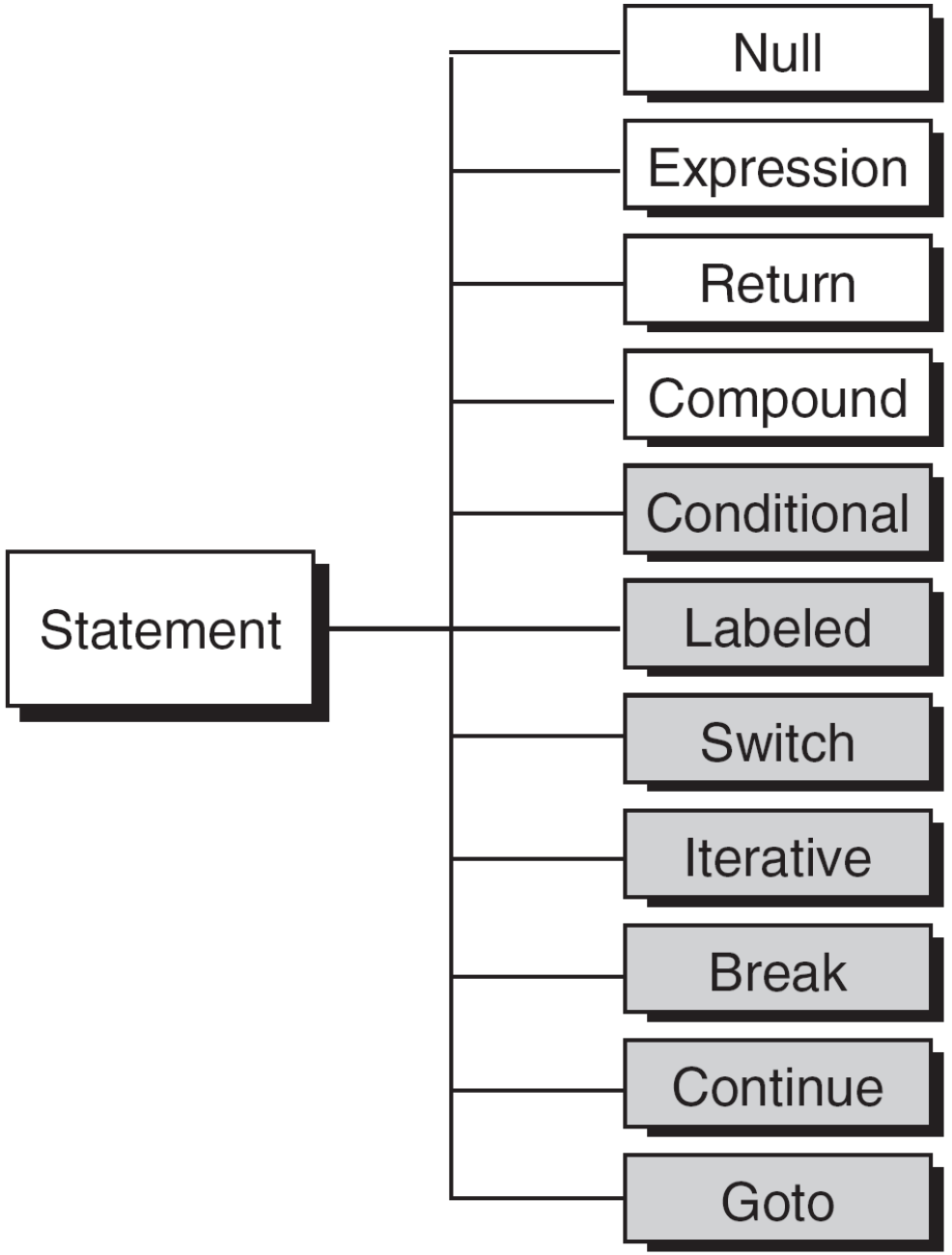


Right-to-Left Associativity

Statement(명령어)

Statement

- Null statement
- Expression statement
- Return statement
- Compound statement



Null Statement

- Just a semicolon
- Example

```
;                    // null statement
```

- They do nothing, but are still valid syntactical objects.

Expression Statement

- A statement consisting of an expression
- Example
  expression;

  a = 2;

  b = c = 3;

Return Statement

- A return statement terminates a function.

- Example

  return;

  return expression;


```
{
// Local Declarations
   int  x;
   int  y;
   int  z;

// Statements
   x  =  1;
   y  =  2;
   …
}  // End Block
```

The compound statement does not need a semicolon.

The Role of the Semicolon

- Every declaration in C is terminated by a semicolon
- Most statements in C are terminated by a semicolon

Statements and Defined Constants

```
#define TAX_RATE 0.825

tax = TAX_RATE * amount;
```

조건문

조건문 (Selection Statement)

- 조건 (Condition)에 따라서 선택적으로 프로그램을 진행
- 프로그램의 Flow를 조종

2-Way Selection
  - 두 가지 선택지 중에 한 가지

Multi-Way Selection
  - 여러 가지 선택지 중에 한 가지(혹은 그 이상)

- 조건문 안에 얼마든지 또 조건문을 넣을 수 있다. (nested)

복잡한 조건들 – 논리 연산자

- ! – NOT, && – AND, || – OR

- Truth Table

not

x	!x
false	true
true	false

and

x	y	x&&y
false	false	false
false	true	false
true	false	false
true	true	true

or

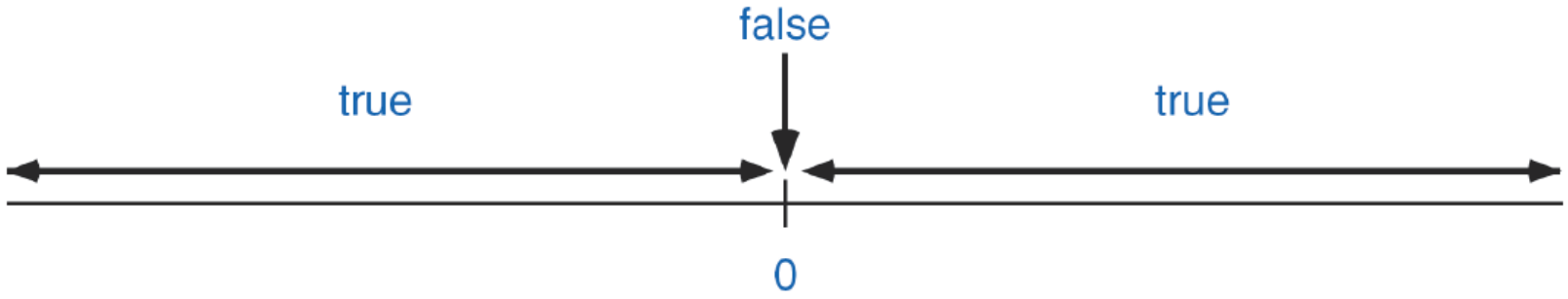
x	y	x||y
false	false	false
false	true	true
true	false	true
true	true	true

Ex)

- if(a == 0 && a 〉 c || !(b == c)) –〉 if((a==0 && a 〉 c) || !(b==c))
- Q) 만약 a=1, b=2, c=20이면?

조건 Condition (TRUE / FALSE)

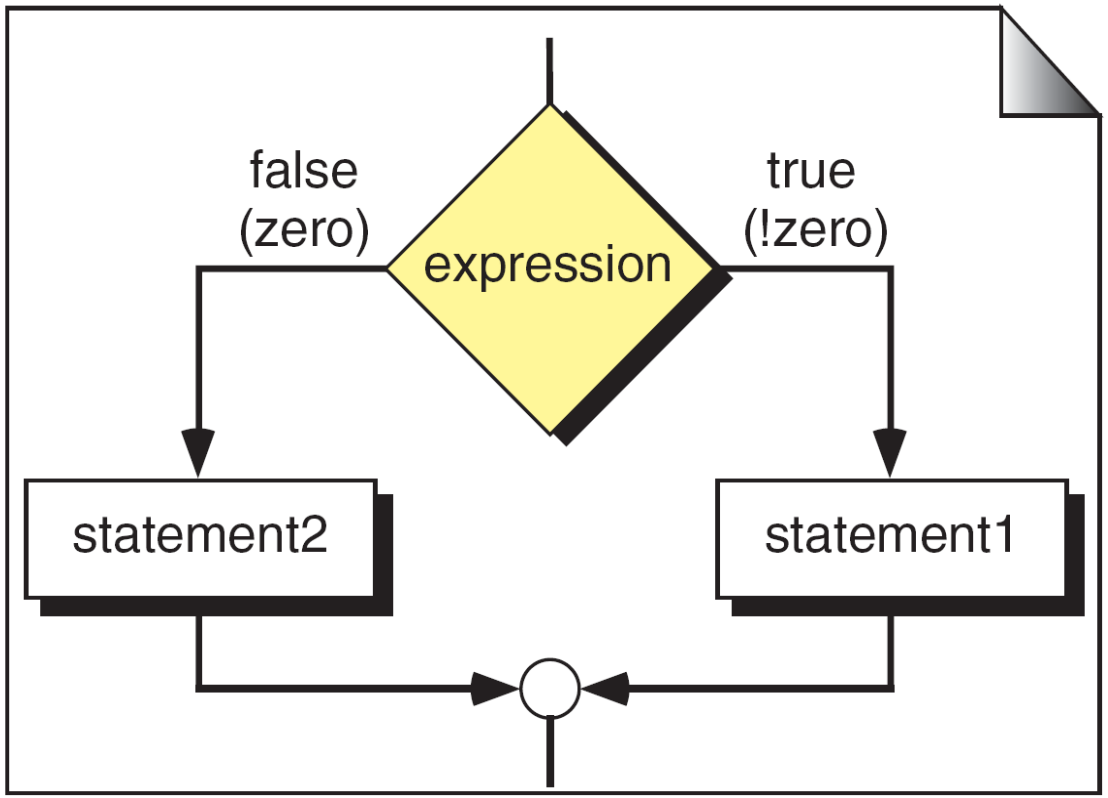
- 조건문과 반복문 등에서 '조건 검사'를 할 때 사용
- C에서는 0을 FALSE, 0이 아니면 TRUE
- ex. if(0.4)는 TRUE로 취급



2-Way Selection

- if … else … 구문

```
if (조건문)
{
        //조건문이 참(True)일 때 실행할 부분
}
else
{
        //조건문이 거짓(False)일 때 실행할 부분
}
```



(a) Logical Flow

```
if (expression)
    statement1
else
    statement2
```

(b) Code

if … else … Example

```
int x = 1;
int y = 0;
if (x || y)
{
    printf(“x||y is true\n”);
}
else
{
    printf(“x||y is false\n”);
}
```

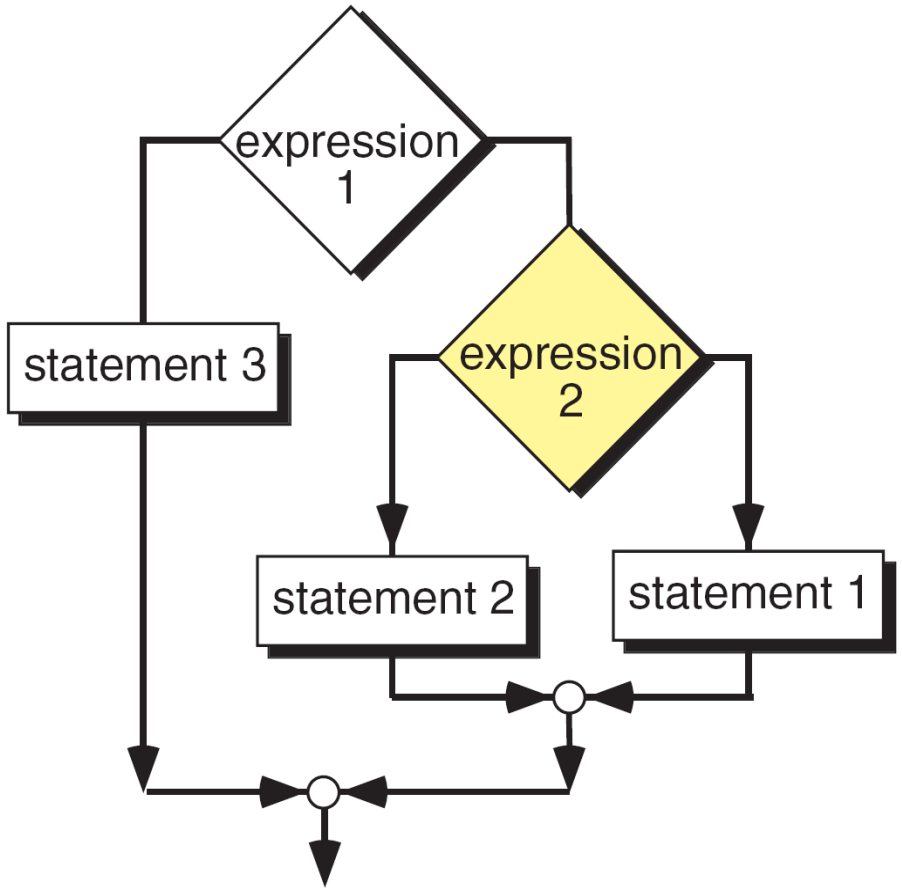
괄호 안 조건문의 참/거짓에 따라 실행할 부분을 선택
다른 부분은 실행되지 않고 넘어간다.

if … else … 특징

If 문만 만들 수도 있다. (else 가 필요 없을 시)

```
if ( 조건문 )
{
        //코드 입력
}
```

Nested *if* Statements



(a) Logic flow

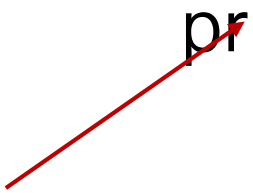
```
if (expression 1)
    if (expression 2)
        statement 1
    else
        statement 2
else
    statement 3
```

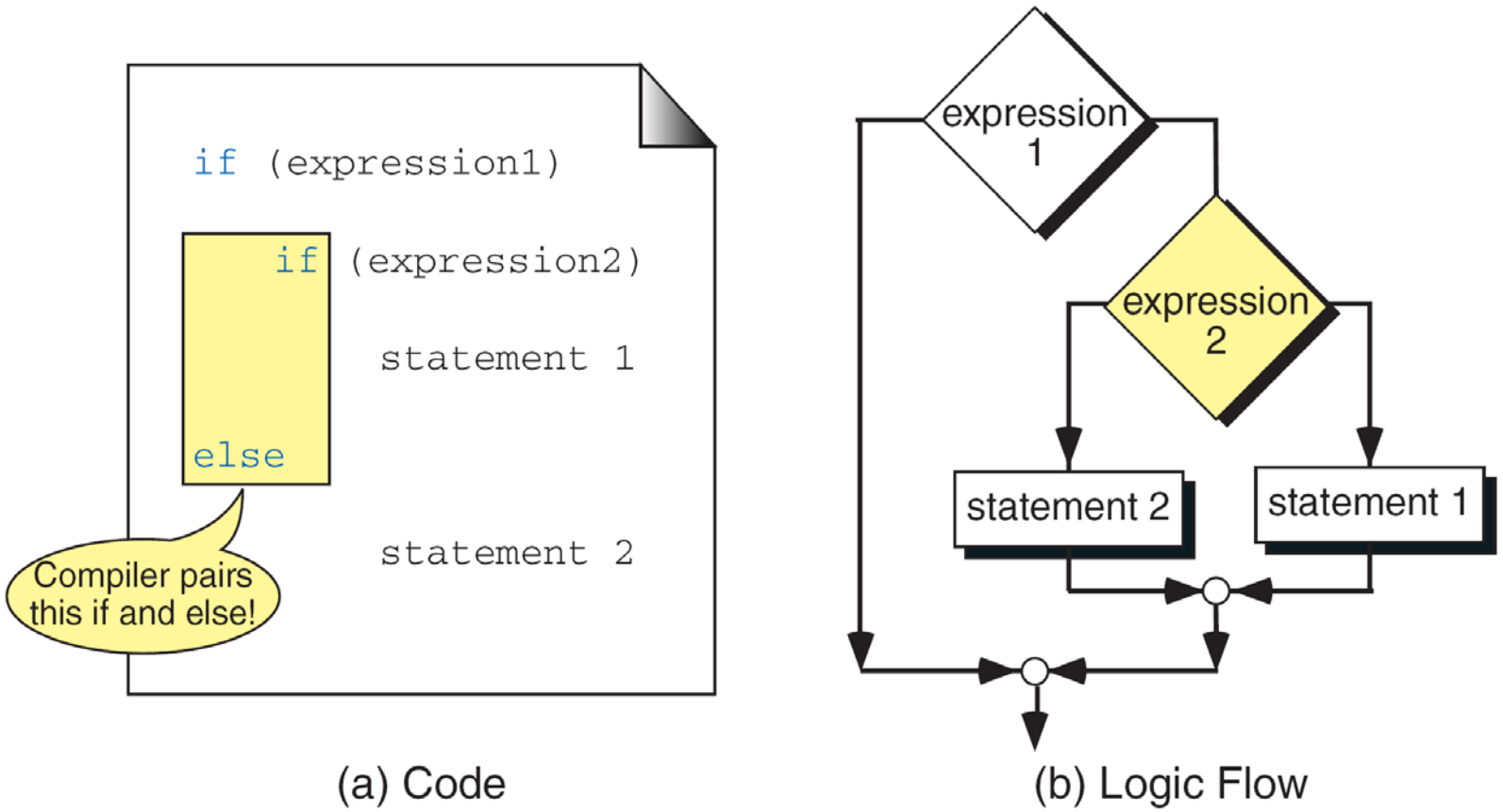
(b) Code

If else 주의 사항

- If, else다음에 실행할 명령이 하나이면 중괄호 필요 없음.
- 근데….

```
If (condition1)
        if(condition2)
                printf(“Hello”);
else
        printf(“Hi”);
```





(a) Code

(b) Logic Flow

***else* is always paired with the most recent unpaired *if*.**

Multi-Way Selection

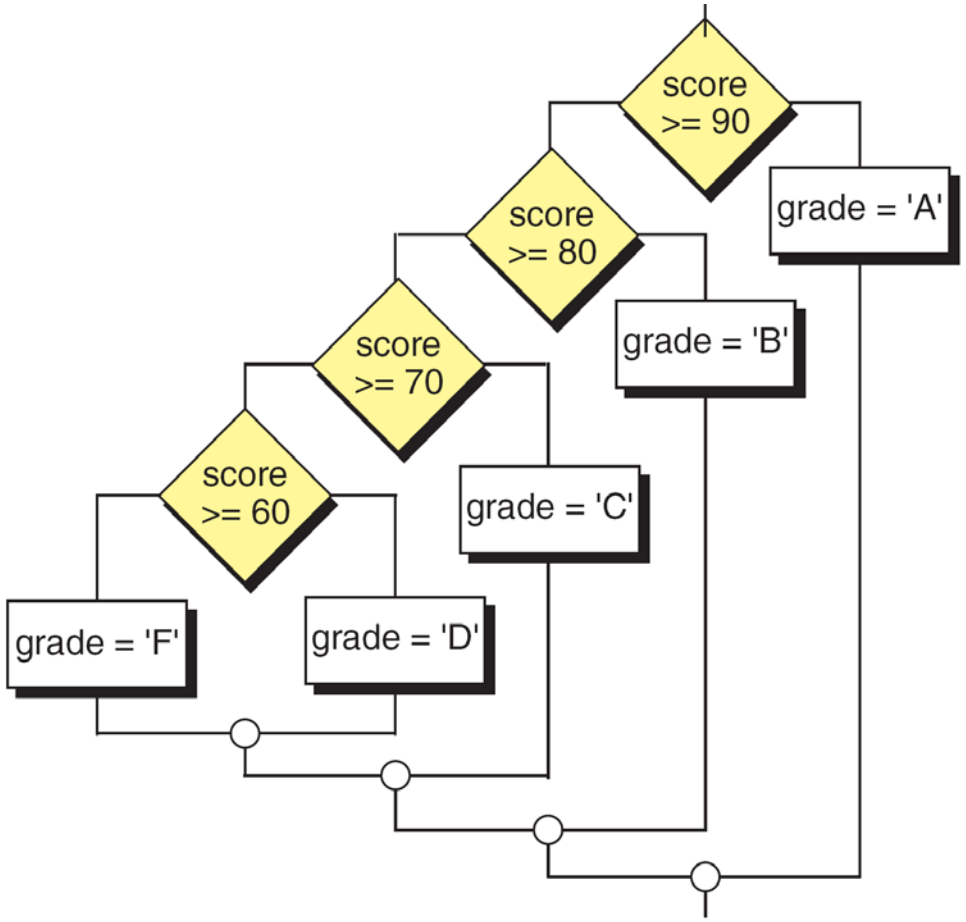
- 여러가지 선택지 중에서 하나 (혹은 그 이상)을 선택해야 할 때

- 종류
  - else if 구문
  - switch 구문

else if

- if…else… 구문의 확장형

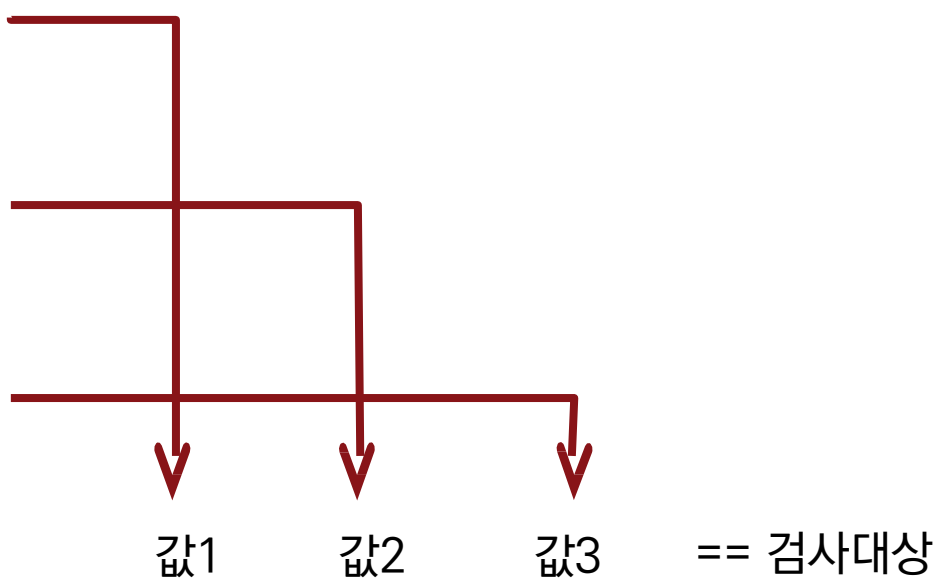
```
if( 조건문 )
{
        //조건문1 이 참일 때
}
else if(조건문2)
{
        //조건문1 이 거짓이고 조건문2가 참일 때
}
else
{
        //앞의 모든 조건문이 거짓일 때
}
```

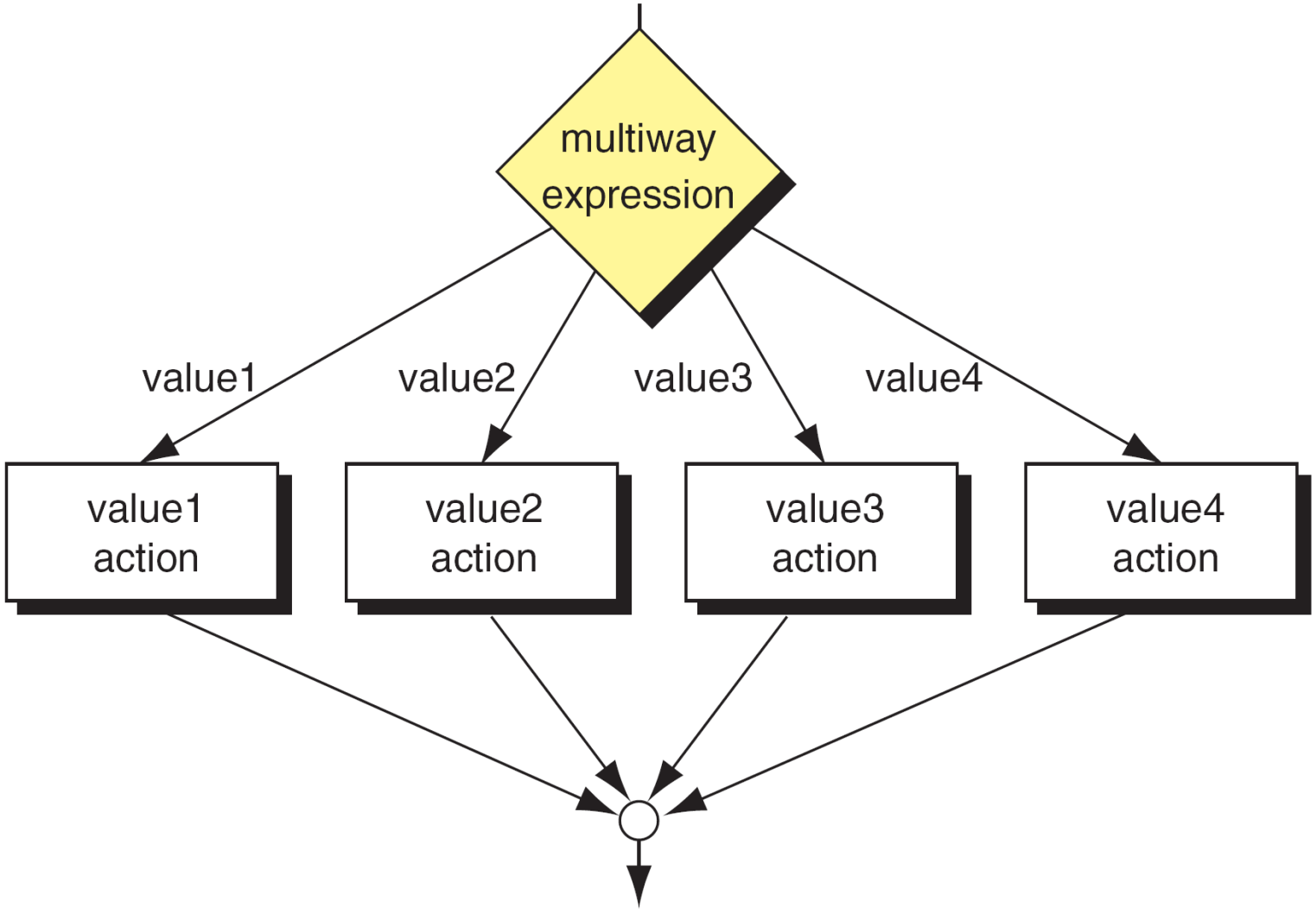


switch 문

- 검사하고자 하는 대상의 값(변수, 함수의 리턴 값)에 여러 가지 선택지가 있을 때

```
Switch (검사대상 )
{
      case 값1:
            statement1_1;
            statement1_2;
      case 값2 :
            statement2_1;
            statement2_2;
      case 값3 :
            statement3_1;
            statement3_2;
}
```




multiway
expression
value1
value2
value3
value4
value1
action
value2
action
value3
action
value4
action

break;

- switch, while, for, do while 같은 조건문 & 반복문을 빠져나가는 명령어
- if 문은 안된다!
- 가장 가까운 하나만 빠져나갈 수 있다.
- 사용 용도
  - 특정한 순간에 조건문을 나가고 싶다.
  - 특정한 순간에 반복문을 나가고 싶다.

switch 문

```
switch(num) {
case 0:
        Statement0;
        break;

case 1:
        Statement1;
        break;
...
default:
        StatementN;
        break;
}
```

- num에는 int형 정수
- num이 0이면 statement0 실행
- num이 1이면 statement1 실행
- num이 n이면 case n에 있는 statement 실행
- 만약 해당하는 case가 없을 경우에는 default에 있는 statement 실행

switch 문 주의사항

```
switch(num) {
case 0:
        Statement0;

case 1:
        Statement1;
        break;
...
default:
        StatementN;
        break;
}
```

- case 0에 break;를 안 쓰면 어떻게 될까?
- num이 0 – case 0의 statement0가 실행될 때 break가 없어 case 1의 statement 1까지 실행된다!
- 의도한 경우가 아니라면 case마다 break;를 꼭 쓰자!

예제

- 학점 계산기 만들기

- 60점 미만이면 F
- 60점 이상 70점 미만 D
- 70점 이상과 80점 미만 C
- 80점 이상 90점 미만 B
- 90점 이상 A

학점계산기 만들기

```
#include <stdio.h>

int main(void) {
        int score;
        printf("Input the score: ");
        scanf("%d", &score);
        switch(score/10) {
        case 9:
                printf("A grade\n");
                break;
        case 8:
                printf("B grade\n");
                break;
        case 7:
                printf("C grade\n");
                break;
        case 6:
                printf("D grade\n");
                break;
        default:
                printf("F grade\n");
                break;
        }

        return 0;
}
```

마치기 전에

다음 시간

- 반복문(for, while, do~while)
- 함수(function)

Tip

- 수업시간 자료 – 초록색 박스에 있는 키워드와 설명
- 책에 나오는 예시 코드 꼭 보기!
- 시험에서 코드를 주고 틀린 부분을 고치거나, 함수를 비워놓고 손코딩 하는 문제들이 나온다
- 중간중간 나오는 실수하기 쉬운 예시들 기억해두기

- **이런것도 될까? 싶을 때는 해보는게 최고!**
- **책에 있는 예제 코드를 한번씩 직접 코딩해 보자.**