전산 SMP 2주차

2015. 09. 22

CSE 12 김범수

bskim45@gmail.com

Special thanks to 박기석 (kisuk0521@gmail.com)

지난시간 복습

왜 배우나요?

대세임 ㅋ

“프로그래밍”과 **“문제해결”**

강좌의 목적은 좋은 프로그래머가 되는 것이 아니고

컴퓨터의 **작동 원리** 전반에 대해 이해하고,

컴퓨터의 힘을 빌려 **문제를 해결하는 능력**을 기르는 것

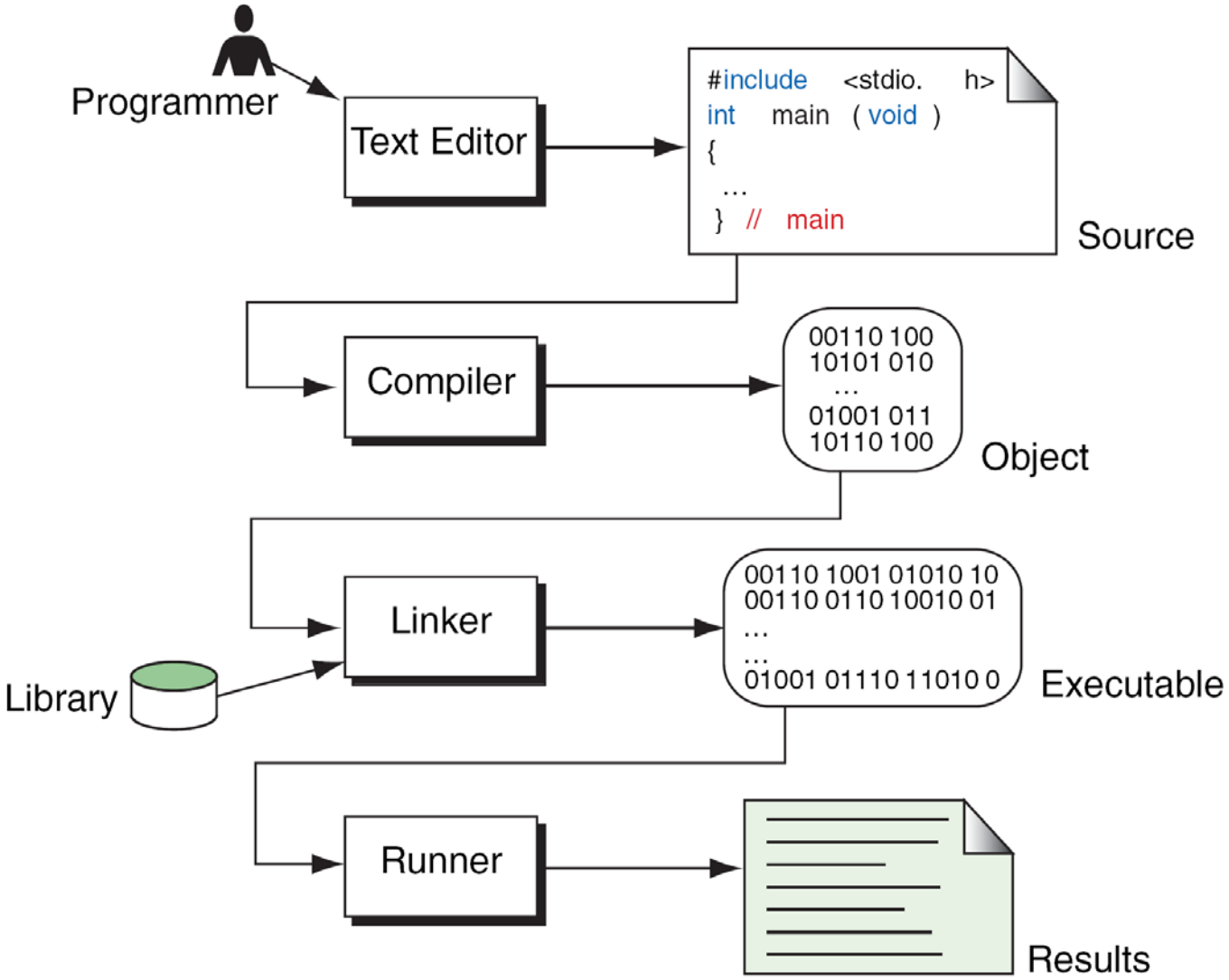
High-level vs Low-level

- 인간과 컴퓨터 중 누구에게 더 가까운가?

고급(High Level)

저급(Low Level)

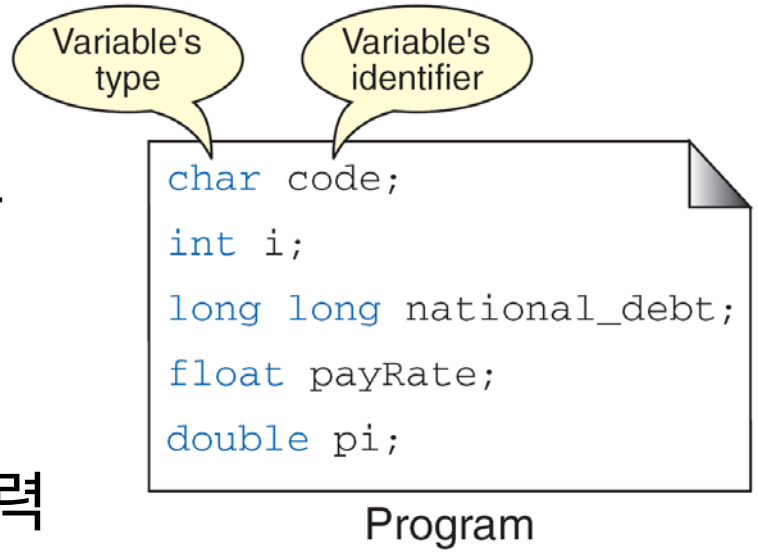
C 언어에서 프로그램(Executable)으로



변수(Variable)란?

- 값을 저장할 수 있는 메모리 공간에 붙여진 이름
- 변수를 선언하면 메모리 공간이 할당되고 그 공간에 이름이 붙는다.
- 일반 변수와 포인터 변수
- const 키워드: 변수를 상수화시킨다.

```
int num; // int : 정수의 저장을 위한 메모리 공간의 할당
         num : 할당된 메모리 공간의 이름(declare)
num=20; //변수 num에 접근하여 20을 저장(assign)
printf("%d", num); // num의 값을 10진 정수로 출력
```



#include

- `#include` 는 다른 헤더파일을 불러와 사용하기 위해 쓰임
- `#include <stdio.h>`: Standard I/O
- `stdio.h`에 들어있는 함수(`scanf, printf`, …) 들을 자유롭게 사용할 수 있게 된다.
- 다양한 헤더 파일 존재: stdlib.h, time.h, math.h, string.h …
- http://www.cplusplus.com/reference/clibrary/

#include

- 직접 헤더파일을 만들 수도 있다.

- #include "header.h"

- 사용자 정의 헤더파일 안에 들어있는 함수와 변수 등을 사용할 수 있게 된다.

#define

- 전처리(Preprocessor Directive) 부분에 선언
- **컴파일** 시 자동으로 대체(substitution)되어 들어간다.

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

#define square(x) (x * x)
…
int a = square(2)       // int a = 2*2;
```

Type Summary

Category	Type	C Implementation
Void	Void	*void*
Integral	Boolean	*bool*
	Character	*char, wchar_t*
	Integer	*short int, int, long int, long long int*
Floating-Point	Real	*float, double, long double*
	Imaginary	*float imaginary, double imaginary, long double imaginary*
	Complex	*float complex, double complex, long double complex*

기본 자료형(Primitives)의 종류와 데이터의 표현 범위

자료형		크기	값의 표현범위
정수형	char	1byte	$-128 \sim 127\ (-2^7 \sim 2^7-1)$
	unsigned char	1byte	$0 \sim 255\ (0 \sim 2^8-1)$
	short	2byte	$-2^{15} \sim 2^{15}-1$
	unsigned short	2byte	$0 \sim 2^{16}-1$
	int	4byte	$-2^{31} \sim 2^{31}-1$
	unsigned int	4byte	$0 \sim 2^{32}-1$
	long	4byte	$-2^{31} \sim 2^{31}-1$
	unsigned long	4byte	$0 \sim 2^{32}-1$
	long long	8byte	$-2^{63} \sim 2^{63}-1$
	unsigned long long	8byte	$0 \sim 2^{64}-1$
실수형	float	4byte	$\pm 3.4 \times 10^{-37} \sim \pm 3.4 \times 10^{+38}$
	double	8byte	$\pm 1.7 \times 10^{-307} \sim \pm 1.7 \times 10^{+308}$
	long double	8byte이상	double이상의 표현범위

형 변환 (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 number)
  - 최종적으로 값 대입(assing)시 저장되는 공간의 타입에 따라 결정
  - `int a = 3.14;  // a = ??`

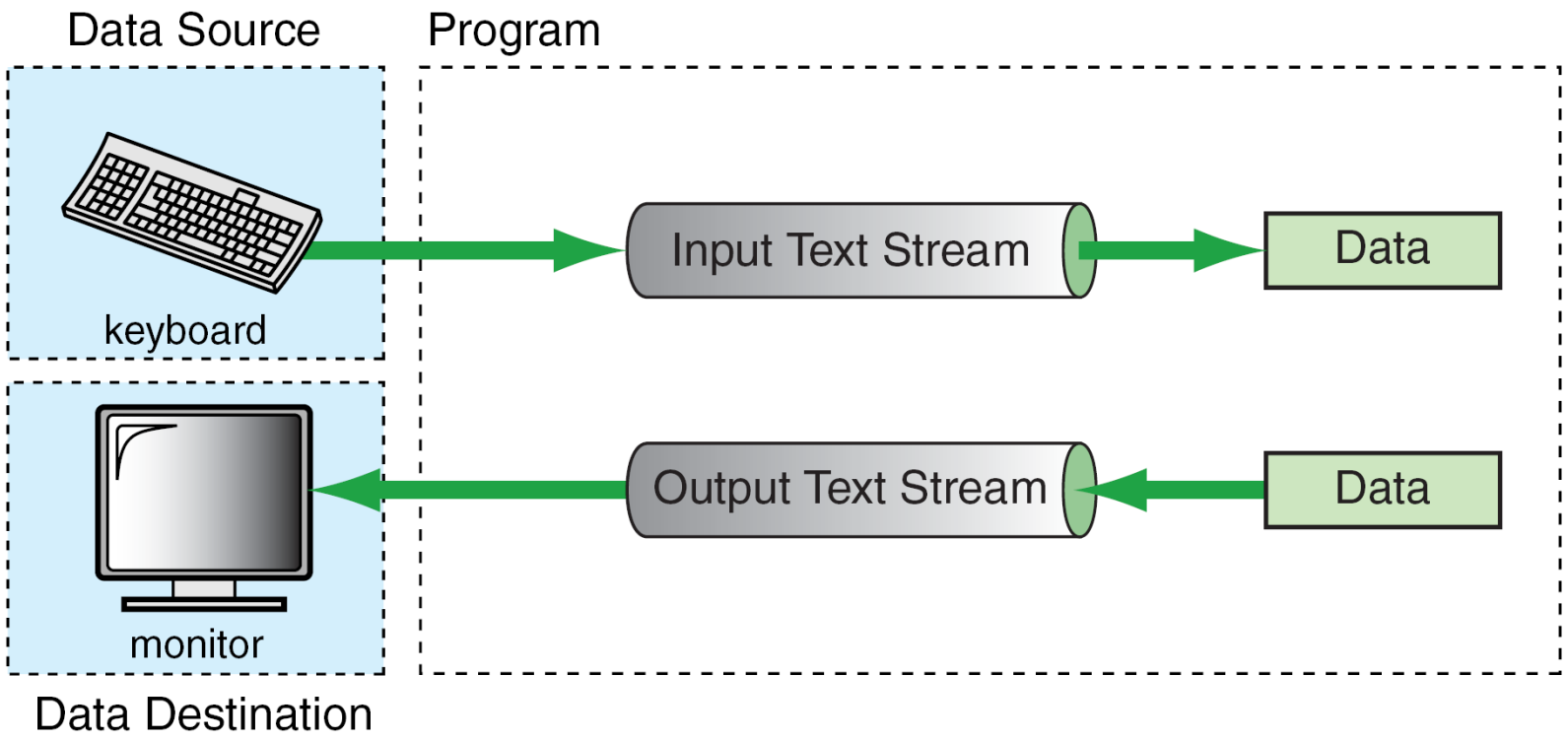
- **Explicit Conversion(casting)**
  - 프로그래머가 인위적으로 타입을 바꿈. 사용시 주의!
  - `double a = (double) 4/3  // a = ??`
  - `double b = (double) 4 / (double) 3  // b = ??`

강제(explicit) 형 변환

```
#include <stdio.h>

int main(void) {
	int num1 = 3;
	int num2 = 4;
	double result = num1/num2;

	printf(“Result: %f \n", result); // 0.000000
}
```

Standard I/O Stream

- Header **`<stdio.h>`**: Standard input/output libraries



printf

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

printf(“%d  %d  %d\n”, a, b, c);
```

서식문자(Format String) : 데이터의 출력 형태를 지정

ex. %d : 부호 있는 10진 정수로 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

```
int a;
char b;

scanf(“%d %c”, &a, &b);
```

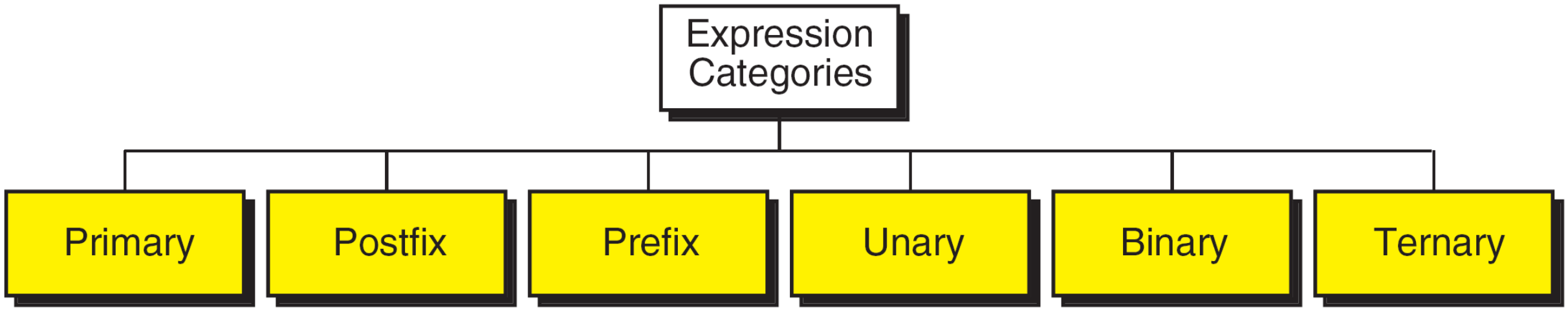
scanf로 값을 받기 위해서는 받는 위치, 즉 주소값을 인자로 주어야 한다.
이유는 다음 시간에. 일단 외우자.

& 가 변수 이름 앞에 붙을 시 그 변수가 가리키는 주소를 의미한다.

A : 변수 이름 / &a : 변수 a 의 주소

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

연산자


Expression Categories
Primary
Postfix
Prefix
Unary
Binary
Ternary

Primary Expressions

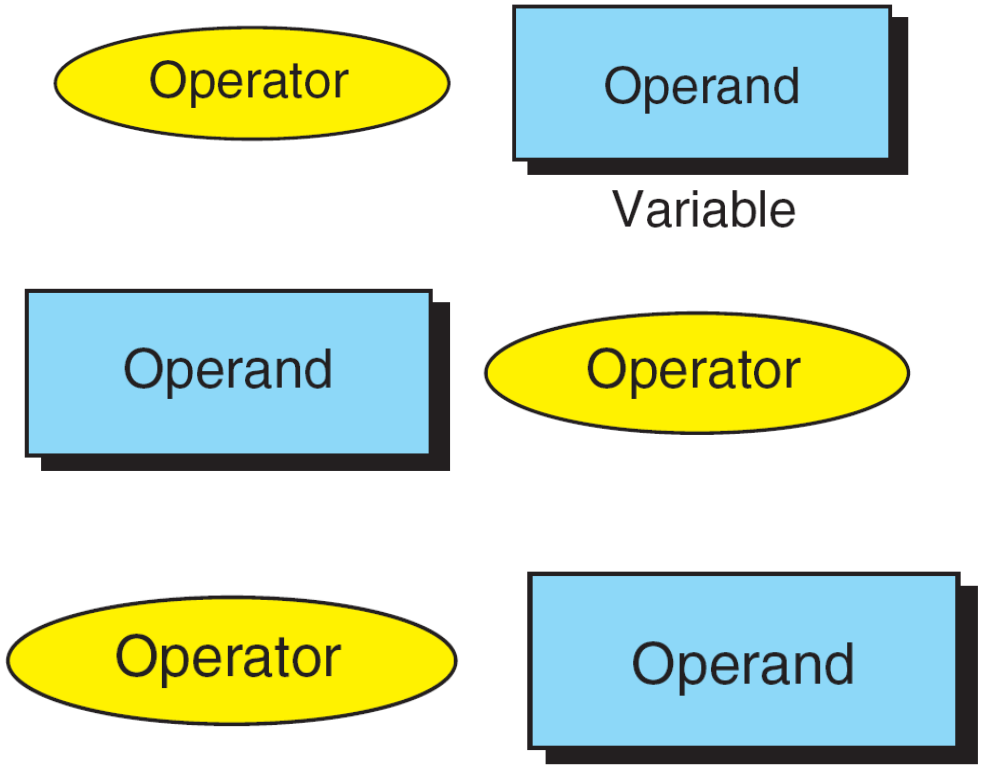
- Names
  - a, b12, price, INT_MAX, SIZE
- Literal constants
  - 5, 123.98, ‘A’, “Welcome”
- Parenthetical expressions
  - (2 * 3 + 4), (a = 23 + b * 6)

Prefix/Postfix/Unary

Prefix
- 증가/감소 연산자: --a, ++a

Postfix
- 증가/감소 연산자: ++a, --a
- 함수 호출

Unary
- 부호 반전: -a



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
−	− a	변수 a의 부호를 변경

```
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

```
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

```
int a = 10, b = 12;
printf("result1: %d \n", a==b);  // result1 : 0
printf("result2: %d \n", a!=b);  // result2 : 1
printf("result3: %d \n", a>=b);  // result3 : 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

```
int a=10, b=12;
printf("result1: %d \n", (a==10 && b==12);      // 1
printf("result2: %d \n", (a<12 || b>12);        // 1
printf("result3: %d \n", !a);                   // 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(void) {
        int a=0, b=0;
        while(a<3){
                printf("a=%d  b=%d\n", a++, ++b);
}
```

증감 연산자 예제

```
3         Date:
4   */
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
```

증감 연산자 예제

```
#define square(x) (x * x)
…
int a = 1;
int b = square(a++);        // int b = ??
```

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("결과: %d %d %d \n", c, d, e, f);  // 4, 30, 26 , -15
}
```

비트 연산자 – Shift

```
#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

    printf("%d \n", a<<2);  // 60
    printf("%d \n", b>>2);  // -4
    printf(" %d \n", c>>2);  // 3
    return 0;
}
```

sizeof 연산자

sizeof : 변수 또는 자료형의 크기를 byte수로 알려준다.

```c
#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));
}
```

```
변수 ch의 크기: 1
변수 a의 크기: 4
int의 크기: 4
float의 크기: 4
double의 크기: 8
'A'의 크기: 4
"ABC"의 크기: 4
3의 크기: 4
1.234의 크기: 8
```

sizeof 연산자

```
#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));
}
```

삼항 연산자

- (contition) ? 실행1 : 실행2

조건이 참이면 실행 1, 조건이 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

```
#include <stdio.h>


int main(void) {
   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

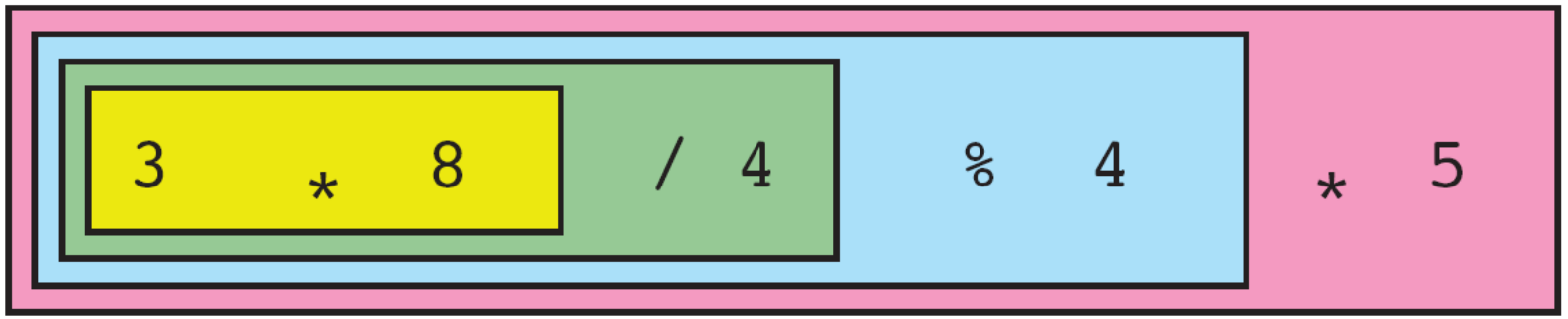
- A+B*C/D-E
    1. *
    2. /
    3. +
    4. -
- 단, 대입 연산자의 경우 오른쪽에서 왼쪽으로
- A = b = C+4
- +먼저 계산하고 b에 c+4를 먼저 대입하고, 그 다음에 A에 b를 대입

```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

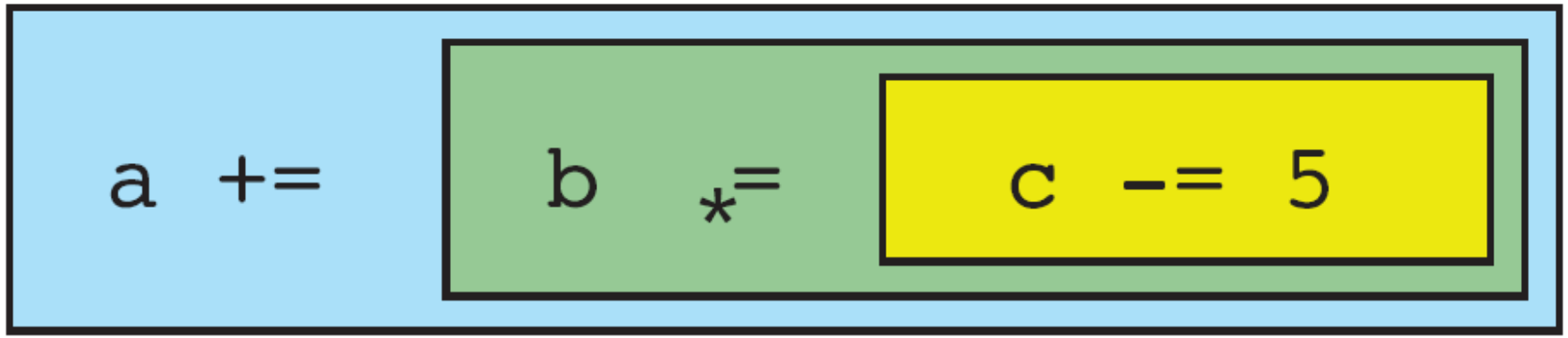
- a + b + c + d
- a * b * c * d



Left-to-Right Associativity

연산자 결합순서

- a = b = c = d = 10



Right-to-Left Associativity

Expressions and Side Effects

Expressions and Side Effects

- `x = a * 4 + b / 2 – c * b;`
- `y = a * 4 + b++ / 2 – c * --b;`
- `z = --a * (3 + b) / 2 - c++ * b;`

- `x, y, z?`

```c
#include <stdio.h>

int main(void) {
	int a = 3, b = 4, c = 5, x, y, z;

	printf("Initial values: \n")
	printf("a = %d\tb = %d\tc = %d\n\n", a, b, c);

	x = a * 4 + b / 2 – c * b
	y = a * 4 + b++ / 2 – c * --b
	z = --a * (3 + b) / 2 - c++ * b;

	printf("x = %d\ty = %d\tz = %d\n\n", x, y, z)
	printf("Values of the variables are now: ")
	printf("a = %d\tb = %d\tc = %d\n\n", a, b, c);
}
```

함수 인자와 증감 연산자

```
#include <stdio.h>

int main(void) {
	int i=1;
	printf("%d %d %d\n", i++, i++, i);

	return 0;
}
```

함수 인자와 증감 연산자

```
int i=1;
printf("%d %d %d\n", i++, i++, i);
```

- 2 1 3 - using g++ 4.2.1 on Linux.i686
- 1 2 3 - using SunStudio C++ 5.9 on Linux.i686
- 2 1 3 - using g++ 4.2.1 on SunOS.x86pc
- 1 2 3 - using SunStudio C++ 5.9 on SunOS.x86pc
- 1 2 3 - using g++ 4.2.1 on SunOS.sun4u
- 1 2 3 - using SunStudio C++ 5.9 on SunOS.sun4u

Parameter evaluation order of a function call in C

- Order of evaluation of function arguments is unspecified, from C99 §6.5.2.2p10:

- **The order of evaluation** of the function designator, the actual arguments, and subexpressions within the actual arguments **is unspecified**, but there is a sequence point before the actual call.

http://stackoverflow.com/questions/376278/parameter-evaluation-order-before-a-function-calling-in-c

마치기 전에

Assn #1

1. 이니셜 암호화하기
   - 암호화의 기본 아이디어: 인코딩(encoding)
   - 너무 쉬워서…

2. 속도와 거리로 걸린시간 구하기
   1. float 실수 scanf 입력(%f)
   2. #include <math.h>
   3. 공식대로 풀면 됨

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.**

다음 시간

- 포인터 맛보기
- 함수