

01000100010101010001110

0101010101

.....



● 윤철희 ●

블록코딩

7주차. 자바스크립트를 통한 기본 문법 이해
기본 문법

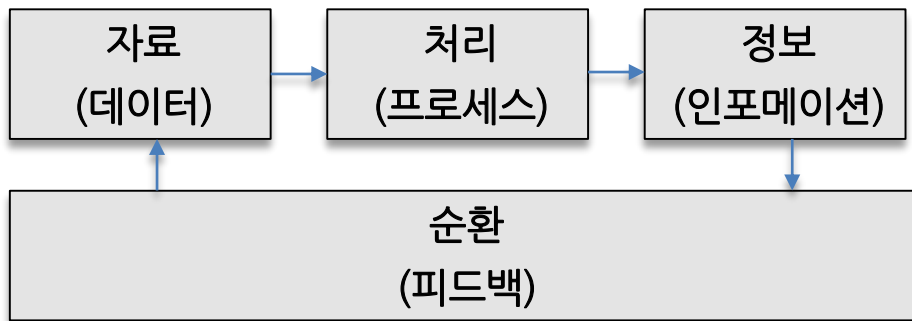
1010101

.....



✓ 자료와 정보

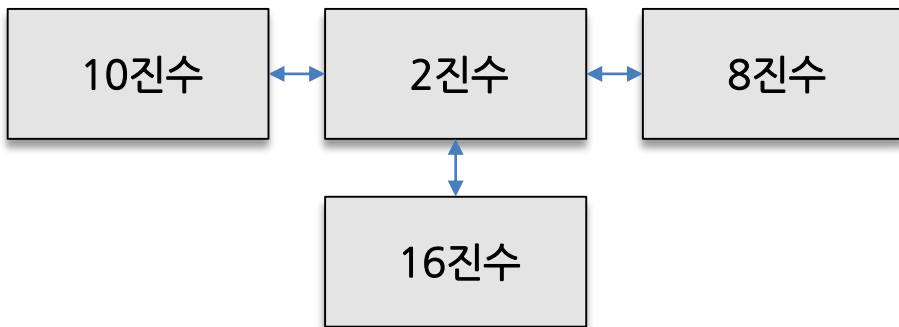
- 자료 : 처리 이저 상태의 문자나 수치, 컴퓨터에 입력이 되는 기본 자료
- 정보 : 자료를 처리한 결과





✓ 진법과 수의 구성

- 10진법 : 0~9까지 사용하며, 10을 한 자리의 기본 단위로 하는 진법
- 2진법 : 0과 1의 조합으로 숫자를 표시하는 방법
- 8진법 : 0~7까지 수로 표시하는 것이 8진법
- 16진법 : 0~9까지 그리고 A~F까지를 사용하여 표시하는 진법





✓ 수치 데이터의 표현

- **비트(bit)** : 컴퓨터에서 사용하는 최소의 단위로서 0, 1을 나타냄.
- **바이트(byte)** : 영문 1 글자를 나타내는 단위로 8비트로 이루어짐.
- **워드(word)** : 워드의 크기는 컴퓨터의 종류에 따라 2바이트, 4바이트, 8바이트 등이 있는데 통상 4바이트를 말함
- **필드(Field)** : 파일 구성의 최소 단위로, 아이템 또는 항목이라고 함.
DB에서는 열을 나타냄
- **레코드(Record)** : 하나이상의 필드들이 모여서 구성된 자료의 단위.
DB에서는 행을 나타냄
- **파일(File)** : 여러개의 레코드가 모여 구성되며, 디스크의 저장 단위로 사용함.



✓ 문자 데이터의 표현

- **아스키 코드 (American Standard Code for Information Interchange : ASCII)** : 미국 정보 교환 표준 코드,
 - 미국 표준 협회가 제정한 데이터 처리 및 통신시스템 상호 간의 정보 교환용 표준 코드 구성 되어 있음
- **BCD 코드(Binary-Coded Decimal Code)** : 6비트를 사용하여 하나의 문자를 표시하는 방식으로 기억 장치의 단어 길이가 6의 배수로 설계된 컴퓨터에 적합함. 자료 구조는 존 필드와 디지털 필드로 나뉘어 있으며 하나의 문자를 표현함
- **유니코드(UNICODE)** : 전세계 언어를 하나의 코드 체계 안으로 통합하려는 컴퓨터 업체들의 합의에 의해 만들어진 코드임. 2바이트를 사용하여 각 국가의 언어를 표시할 수 있으며, 유니코드를 지원하는 프로그램이면 한글 등에 대한 별도의 처리 필요 없음



자바스크립트 작성 기초



▪ JavaScript



▪ Node.js

〈출처 : www.sharedit.co.kr, 웹페이지 검색〉



자바스크립트?



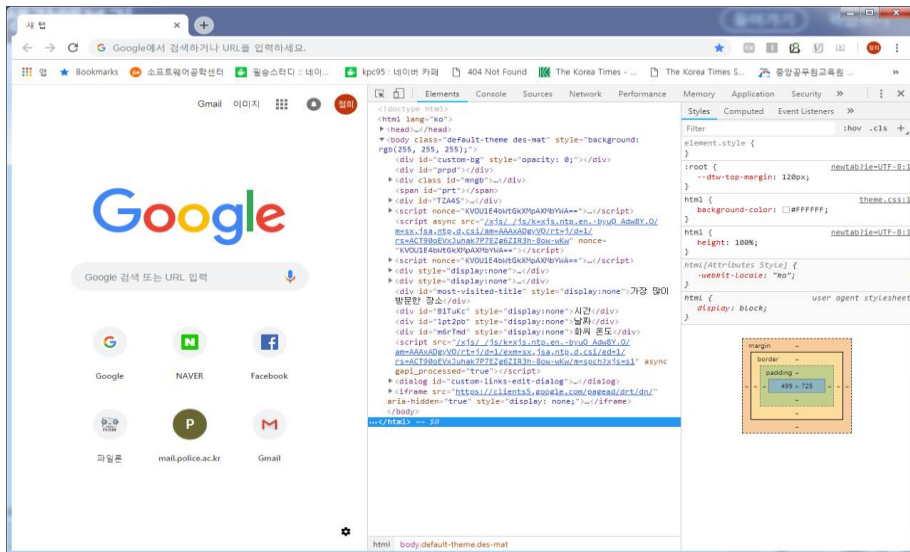
- 부수적인 프로그래밍 언어로 취급?
- 풍부한 경험을 제공하는 인터넷 애플리케이션으로 부각
 - 웹 클라이언트 애플리케이션 개발
 - 웹 브라우저에서 실행되는 웹 클라이언트 애플리케이션 개발이 목적임
 - 웹 브라우저에서 실행할 수 있는 유일한 프로그래밍 언어
 - 웹 서버 개발
 - 기존에 웹 개발은 두 가지 이상의 프로그래밍 언어가 필요했음
 - ✓ 웹 클라이언트, 웹 서버를 다른 언어로 개발
 - 웹 서버도 자바스크립트로 개발 가능(Node.js)



크롬 ? 에디터 ?

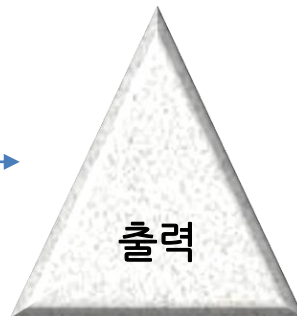


- 자바스크립트 코드를 실행할 수 있으며, 오류 확인이 쉬운 크롬



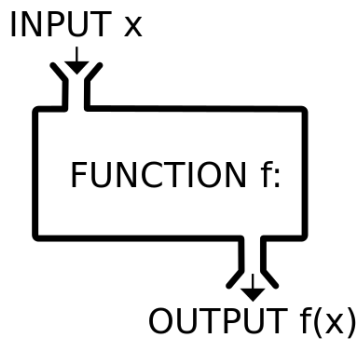


함수란?





함수란?



- 하나의 큰 프로그램을 여러 부분으로 분리함으로써 **구조적 프로그래밍**이 가능하다.
- 같은 코드를 계속 쓰지 않음으로서 프로그램의 용량을 줄일 수 있고, 다른 부분이나 다른 프로그램에서 같은 코드를 사용할 수 있다.
- 함수의 기능과 내부 구현을 분리하는 **캡슐화**가 이루어진다.

<출처 : 위키백과, 함수란?>

1. 자료의 표현

1) 자료와 숫자의 표현

- 자료의 표현

- 컴퓨터에 저장되는 모든 데이터는 2진수로 저장
- 2진수는 0 또는 1로 구성
- 비트(bit) : 2진수 0 또는 1

- 숫자의 표현

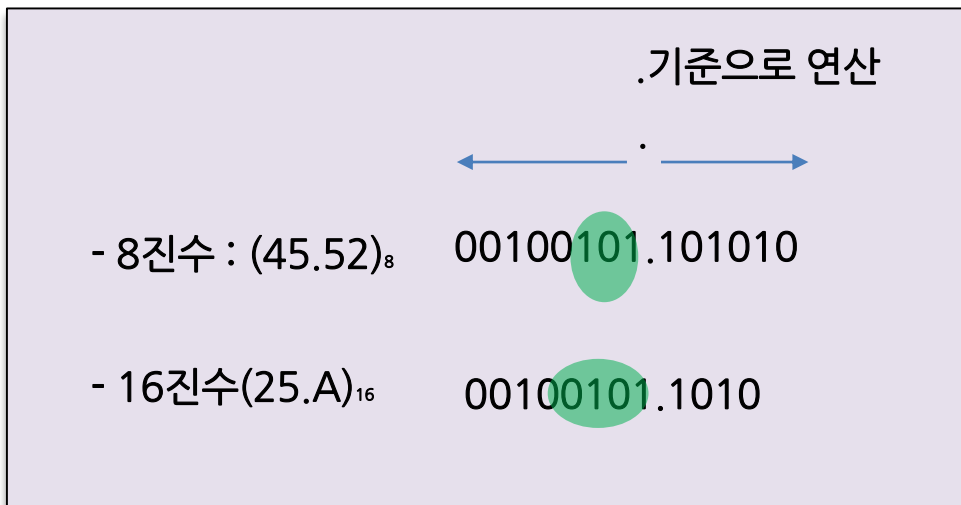
- 컴퓨터에서 숫자 데이터의 연산 시 2진수로 연산
- 예) $5 + 3$

0 1 0 1	5
+ 0 0 1 1	3
1 0 0 0	8

1. 자료의 표현

1) 자료와 숫자의 표현

- 2진수, 8진수, 16진수의 상호 변환 관계



1. 자료의 표현

2) 진법의 변환

- 10 진수 \rightarrow 2진수 변환

10진수를 계속 2로 나누고 나머지를 역순으로 표기

$$\begin{array}{r|l}
 2 & 25 \\
 \hline
 2 & 12 \dots 1 \\
 2 & 6 \dots 0 \\
 2 & 3 \dots 0 \\
 & 1 \dots 1
 \end{array}
 \quad
 (25)_{10} \rightarrow (11001)_2$$

- 10 진수 \rightarrow 8진수 변환

10진수를 계속 8로 나누고 나머지를 역순으로 표기

$$(25)_{10} \rightarrow (31)_8$$

1. 자료의 표현

2) 진법의 변환

- 2 진수 → 10진수 변환

2진수 각 자리 수에 가중치를 곱한 값을 더한다.

(예) 2진수 11001 를 10진수로 변환

$$\begin{aligned} & 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ & = 16 + 8 + 0 + 0 + 1 \\ & = 25 \end{aligned}$$

- 8진수 → 10진수 변환

$$(11001)_2 \rightarrow (25)_{10}$$

(예) 8진수 31 을 10진수로 변환

$$\begin{aligned} & 3 \times 8^1 + 1 \times 8^0 \\ & = 24 + 1 = 25 \end{aligned}$$

$$(31)_8 \rightarrow (25)_{10}$$

2. 보수

1) 1의 보수와 2의 보수

절대치	최상위 1비트를 양수는 0, 음수는 1로 표현하고 나머지 비트는 절대치로 표현함
1의 보수	0은 1로, 1은 0으로 변환함
2의 보수	오른쪽 끝자리에 1을 더함

2. 보수

2) 1의 보수와 2의 보수의 표현

- 1의 보수(1's Complement)와 2의 보수(2's Complement)의 표현

(예) 2진수 11001 의 1의 보수(1's Complement)

11001

00110 ← 1의 보수

(예) 2진수 11001 의 2의 보수(1's Complement) : 1의 보수 + 1

11001

00111 ← 2의 보수

(예) 10111000 의 1의 보수와 2의 보수를 구하시오.

10111000

01000111 ← 1의 보수

01001000 ← 2의 보수

3. 음수의 표현 방법

✓ 양의 정수를 음수로 만드는 방법

- 2의 보수

- (예) 00100000(32)를 2의 보수법에 따라 음수로 인코딩한 값은?
= 11100000(-32)

형식	부호	정수부						
부호화절대치	0	0	1	0	0	0	0	0
1의 보수	1	1	0	1	1	1	1	1
2의 보수	1	1	1	0	0	0	0	0

최상위 비트는 부호(Sign) 비트 : 0 - 양수, 1 - 음수

4. 문자와 숫자 자료의 표현 방법

1) 문자 자료 :

- 문자(character) : 한 글자를 나타내는 문자와 단일 인용부호(' ')로 표시
- 문자열(string) : 여러 글자로 구성되며 큰 따옴표(" ")로 표시

구분	표현 형태
문자(character)	'a' '1' '+' 'k'
문자열(string)	"star-7" "hope" "Dream 01" "Love"

- 문자(character) 표현 기호 : ' '
- 문자열(string) 표현 기호 : " "

2) 숫자 자료 :

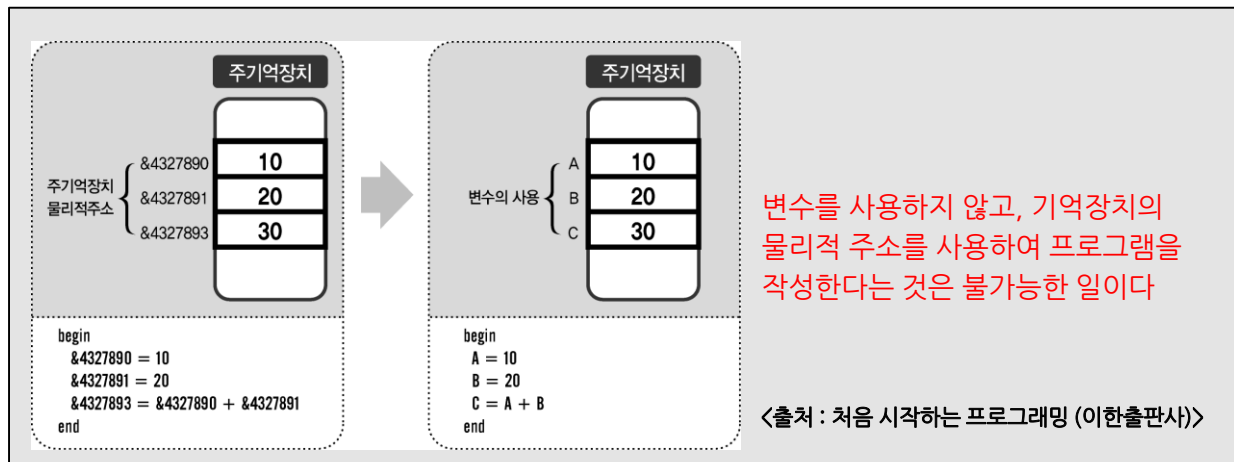
- 표현할 숫자 크기에 따라 형을 구분하며, 저장되는 메모리 크기도 다르다.
- 정수형과 실수형으로 구분

구분	표현 형태
정수형 (integer)	230 369 17 9
실수형 (float, double)	25,1 30, 120508,3692 0,05678

1. 프로그램의 실행

1) 프로그램의 입출력에 사용되는 모든 자료는 저장될 메모리 주소가 필요

- 자료가 저장된 메모리의 주소를 숫자로 표현하면 기억하기 어렵다.
→ 프로그램을 작성하기 힘들다.
- 변수명 : 데이터를 저장하기 위한 메모리 주소를 기호화(symbolic)한 것

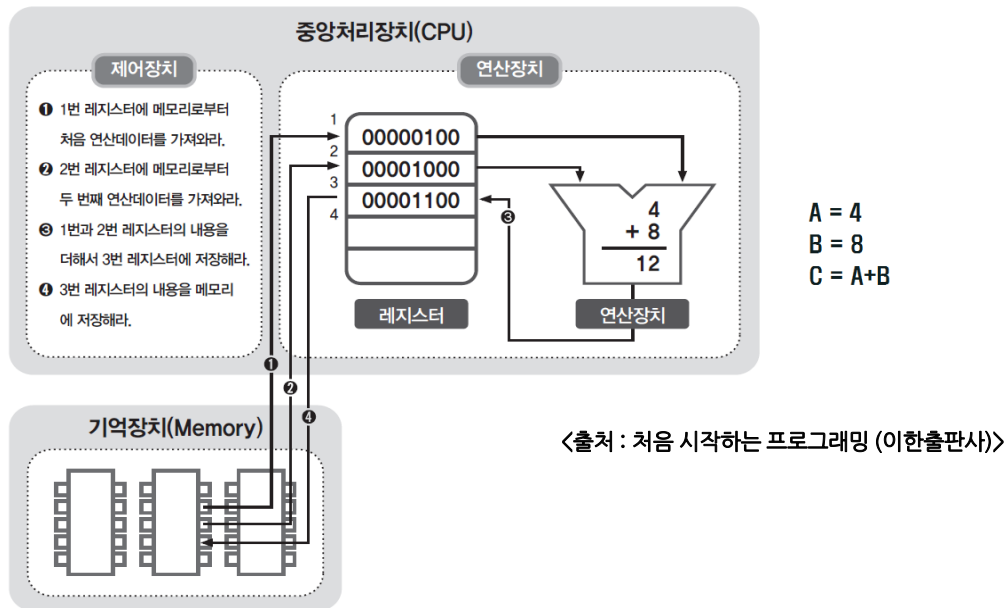


<주기억 장치의 물리적 주소를 사용하는 프로그램과 변수를 사용하는 프로그램>

1. 프로그램의 실행

2) 주기억장치(메모리) 적재

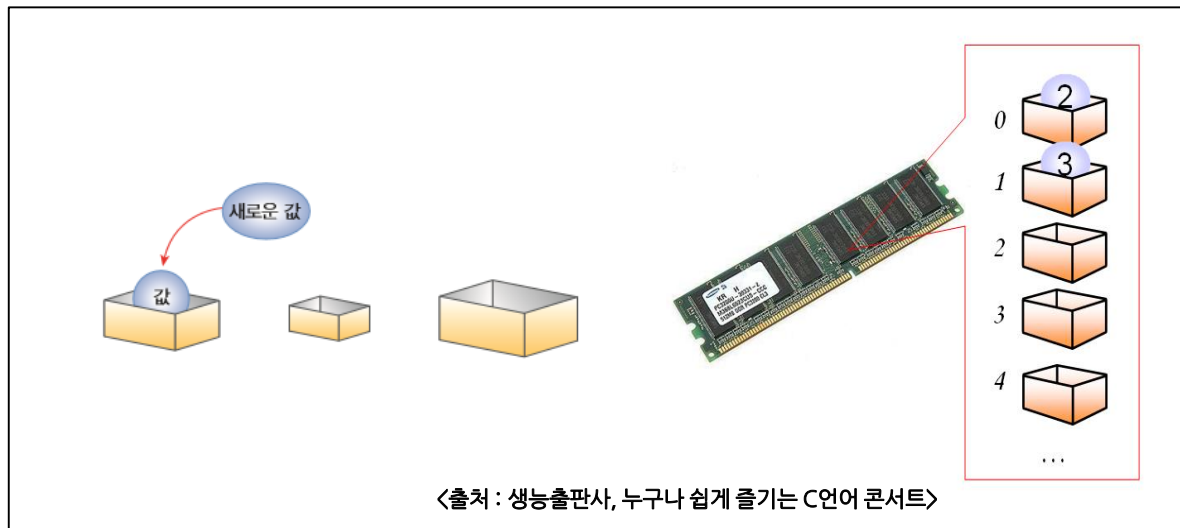
- 적재된 주기억 장치의 주소를 중앙처리장치에 전달하면
중앙처리장치는 그 주소에 있는 데이터를 하나씩 가져다 처리한다.



2. 변수

1) 변수의 정의

- 변수 (variable)란?
 - 프로그램에서 일시적으로 데이터를 저장하는 공간
- 변수는 왜 필요한가?
 - 데이터가 입력되면 어딘가에 저장해야만 사용할 수 있음



2. 변수

2) 변수명 부여 규칙

- 변수명의 첫 글자는 영문자로 시작
 - 두 번째 글자 부터는 숫자, 언더스코어(_) 사용 가능.
 - 특수문자는 사용할 수 없다
- 변수명의 길이가 제한된다.
- 예약어(reserved word)는 변수명으로 사용불가

2. 변수

2) 변수명 부여 규칙

- 사용할 자료를 연상하기 쉬운 기호로 부여하는 것이 좋다.
- 변수명의 첫 글자는 영문자를 사용하는 것이 좋다.

자료 내용	자료형	변수명
상여금	integer	bonus
이름	string	name
평균	float	average

2. 변수

3) 변수 초기화

형식	자료형 변수이름 = 초기값
----	-------------------------

사용 예	Int a = 9
------	--

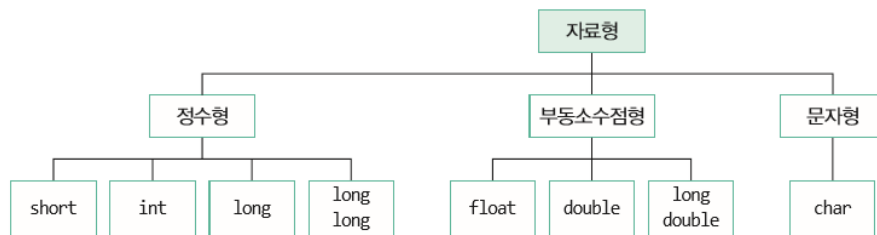
2. 변수

4) 자료형

- 변수에 저장될 자료의 형에 따라 구분한다.
- 대부분의 프로그래밍 언어에서는 다음과 같은 자료형을 갖는다.

(자바스크립트는 제외)

- 문자형 : char
- 정수형 : int, long
- 실수형 : float, double
- 논리형 : boolean



<출처 : 생능출판사, 누구나 쉽게 즐기는 C언어 콘서트>

2. 변수

4) 자료형

- bool
 - 참과 거짓
 - 보통은 1과 0을 사용
- char
 - 글자 하나를 나타냄
 - 1byte -> 8bits -> 256가지 글자 표현
 - 그러면 한글은 어떻게 보이나요?
- char*
 - 글자가 쭉 이어진 문자열을 나타냄
 - string (자세한 내용은 나중에)

2. 변수

4) 자료형

- structure
 - 관련이 있는 여러 개의 정보를 나타내고 싶을 때
 - 같은 이름을 사용함으로써 직관적
 - 프로그램의 효율성 상승
- array
 - structure와 비슷한 점과 다른 점
 - 동일한 타입만을 가져야 함
- char*
 - 글자가 쭉 이어진 문자열을 나타냄
 - 가장 흔히 쓰이는 것이지만 사용이 어려움
 - JAVA에서는 string형으로 쓰기 쉽게 정의

2. 변수

5) 변수 사용 예

C 언어

```
int count;  
double frequency = 10.9;  
char ch = 'a';
```

Java 언어

```
int count=100 ;  
float f =3.14159 ;  
char cc = 'K' ;  
boolean flag = false;
```

자바스크립트 언어 (변수의 형이 없으며, 모든 변수는 var로 선언)

```
var count ;  
var ch = 'kim' ;  
var f = 3.14159;  
var count = 20;
```

특수하게, 자바스크립트의 경우에만,
변수 형을 구분하지 않고 사용한다

3. 상수

1) 상수(constant)란?

- 변수와 달리 메모리에 저장되면 프로그램 실행이 종료될 때 까지 그 값이 변경되지 않는 자료형.
- 원주율이나 특정 공식 등 변하지 않는 값을 필요로 할 때 상수로 설정 사용

(예) `const pi = 3.141592;`

(잘못된 사용 예)

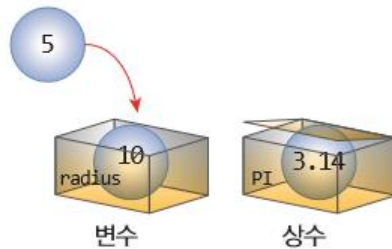
```
const rate = 10;    // 상수로 선언된 rate의 값은 변경 불가능
```

```
.....
```

```
rate = 20;
```

3. 상수

1) 상수(constant)란?



<출처 : 생능출판사, 누구나 쉽게 즐기는 C언어 콘서트>

3. 상수

2) 상수(constant)의 종류

- 정수형 상수
 - 10진 상수, 8진 상수, 16진 상수, Long형 상수
 - 10진 상수 : 0이외의 숫자로 시작하는 10진 숫자열
- 부동소수점 상수
 - 10진 상수, 지수형 상수
 - 10진 상수 : 실수 10진수 표시

3. 상수

2) 상수(constant)의 종류

- 문자 상수
 - 한 문자만 표시
 - + 제어문자 : 'w'를 붙여서 사용
 - wn , ww, w', w", w0
- 문자열상수
 - string : "안녕하세요"
 - 마지막에 문자열의 끝을 표시해 주어야 함
 - "Hi" -> 'H' + 'i' + 'w0'
- 기호상수
 - #define 기호상수 내용

1. 연산자

1) 연산자의 종류(대표적 연산자)

- 산술연산자(Arithmetic Operator) : 수치 계산에 대한 산술적 연산 처리
덧셈(+), 뺄셈(-), 곱셈(*), 나눗셈(/), 나머지(%) 등
- 관계연산자(Relational Operator) : 두 자료에 대한 대소관계를 비교하는 연산 처리
- 크다(>), 크거나 같다(>=), 작다(<), 작거나 같다(<=), 같다(==),
같지 않다(!=)등
- 논리연산자(Logical Operator) : 진리 값에 따라 참, 거짓의 논리적 연산 처리
논리부정(!), 논리곱(&&), 논리합(||) 등

1. 연산자

2) 연산자의 이해

- 산술연산자(Arithmetic Operator) : 수치 계산에 대한 산술적 연산 처리
덧셈(+), 뺄셈(-), 곱셈(*), 나눗셈(/), 나머지(%) 등

단항 산술 연산자 : +, -

이항 산술 연산자 : +, -, *, /, %

```
x = -3 ;  
x = -(-3) ;  
x = -(3-2) ;
```

1. 연산자

2) 연산자의 이해

- 관계연산자

- 두 수의 값의 대소 관계를 비교하기 위한 연산자
- 주로 선택문이나 반복문의 조건식에서 참/거짓을 판단할 때 사용
- 관계연산자가 포함된 조건식의 결과는 항상 참 또는 거짓

연산자	사용 예	설명
>	op1 > op2	op1이 op2보다 큰 경우
>=	op1 >= op2	op1이 op2보다 크거나 같은 경우
<	op1 < op2	op1이 op2보다 작은 경우
<=	op1 <= op2	op1이 op2보다 작거나 같은 경우
==	op1 == op2	op1이 op2보다 같은 경우
!=	op1 != op2	op1이 op2보다 같지 않은 경우

1. 연산자

2) 연산자의 이해

- 논리연산자
 - 피연산자에 대한 진리표에 따라 논리 연산을 수행하며 결과는 참, 거짓
 - 연산자의 우선순위 : NOT > AND > OR > XOR

입력		논리연산자		
피연산자1 (op1)	피연산자2 (op2)	논리NOT (! op1)	논리AND (op1 && op2)	논리OR (op1 op2)
true	true	false	true	true
true	false	false	false	true
false	true	true	false	true
false	false	true	false	false

1. 연산자

2) 연산자의 이해

- 배정 연산자(Assignment Operator)

- 배정 연산자는 '=' 기호를 사용하여, '=' 오른쪽의 수식 처리 결과 또는 자료 값을 '=' 기호의 왼쪽에 지정된 변수(주소) 번지로 저장하라는 의미이다.

예	name = "정 약용"	// 문자열 "정 약용"을 name 변수에 저장
	grade = 'A'	// grade 변수에 문자 'A'를 저장
	total = 99	// total 변수에 정수 99를 저장

- 오퍼랜드(Operand, 피연산자)

- 연산의 대상이 되는 값을 피연산자(operand)라 하고, 연산기호를 연산자(operator)라 함.



1. 연산자

3) 그외 연산자의 종류

연산자 (자바)	증감 연산자: ++ --
	비트 연산자: & ^ ~ << >> >>>
	조건 연산자: ?:
	배정 연산자: = += -= *= /= %= &= ^= = >>= <<= >>>=
	캐스트 연산자: (자료형)
	배열 연산자: []
	메소드 연산자: () .
	instanceof 연산자: instanceof

1. 연산자

3) 그외의 연산자의 종류

- 증감연산자

증감 연산자는 피연산자로부터 더하거나 빼는 등 단항 연산을 위한 연산자이다. 증가 연산자와 감소 연산자로 나누며, 이 연산자들은 명령형 프로그래밍언어에 구현되어 공통적으로 있다. C와 같은 언어들은 각 연산자마다 의미를 달리하는 전치와 후치 연산 기능이 있다.

```
int x;
int y;

// 증가 연산자
x = 1;
y = ++x;    // 이 경우 x는 2이고, y도 2이다. (전치)
y = x++;    // 이 경우 x는 3이지만 y는 그대로 2이다. (후치)

// 감소 연산자
x = 3;
y = x--;    // 이 경우 x는 2이고 y는 3이다. (후치)
y = --x;    // 이 경우 x는 1이고 y도 1이다. (전치)
```

<출처 : 위키백과, 증감연산자>

1. 연산자

3) 그외의 연산자의 종류

- 증감연산자

- 연산자

```
(a + b)++ // error
```

- ☞ ++, --

- ☞ 변수가 아닌 식에는 사용 못함

- ☞ 실수형 적용 안됨

- 전위 연산자

```
n = 1;  
x = ++n; // x=2, n=2
```

`x = ++n;` 라는 코드는
`n = 1`을 더한 `x`라고 볼 수 있음.
따라서 `x`는 2,
최종적으로 `x=2, y=2`

- 후위 연산자

```
n = 1;  
x = n++; // x=1, n=2
```

`x = n` 이 먼저 나온 뒤, 1을
더하는 ++연산자가 나옵니다.
따라서 `x`에는 처음의 `n`값인 1이
들어감. `n=2`가 됨

1. 연산자

3) 그외의 연산자의 종류

- 비트 연산자

- 비트 단위로 연산 --- 기억장소 절약(자바의 저급언어적 특성)
- 종류 --- &, |, <<, >>, >>>, ^, ~
- 피연산자는 반드시 정수형

- 우선순위

연산자	우선순위
~ << >> >>> & \wedge 	(높음) ↕ (낮음)

2. 연산자 우선순위

1) 연산자의 우선순위

우선 순위	연산자	의미	사용 예
1	()	괄호	$8 * (5 + 2)$ $(width * height) / 2$
2	단항연산자 -	음수부호	$-7 + 5$
3	*, /, %	곱셈, 나눗셈, 나머지	$4 * 10$, $width * height$ $20 / 5$, $sum / count$ $20 \% 3$, $dividend \% divisor$
4	+, -	덧셈, 뺄셈	$10 + 20$, $num1 + num2$ $10 - 20$, $num1 - num2$
5	=	배정연산자	$avr = sum / n$, $sum = kor + eng + mat$

2. 연산자 우선순위

1) 연산자의 우선순위

연산자	우선순위	의미
()	1	괄호
+, -, !	2	양수 음수 부호 논리 연산자 NOT
*, /, %	3	*: 곱하기 /: 나누기 %: 나머지
+, -	4	+: 더하기 -: 빼기
<, <=, >, >=	5	관계 연산자 < : 작다 <= : 작거나 같다 > : 크다 >= : 크거나 같다
==, !=	6	관계 연산자 == : 같다 != : 같지 않다
&&	7	논리 연산자 AND
	8	논리 연산자 OR
=	9	배정 연산자

산술식에서 동일한 우선순위가 여러 개 있을 경우에는 왼쪽부터 오른쪽으로 차례대로 수행한다.

2. 연산자 우선순위

1) 연산자의 우선순위

- 연산자 우선순위

괄호, 배열, 구조체, 공용체 멤버를 지시하는 연산자 $()$, $[]$, $->$, $.$

하나의 피연산자를 가지고 있는 연산자 $!$, $++$, $-$

$*$, $/$, $+$, $-$

대입연산자

나열연산자

- 연산자 결합방향

대입연산자, 단항연산자는 오른쪽에서 왼쪽으로 결합

그 외 연산자는 왼쪽에서 오른쪽으로 결합

$X = y += z$ $X = (y += z)$

$A = x + y + z$ $A = (x + y) + z$

2. 연산자 우선순위

1) 연산자의 우선순위

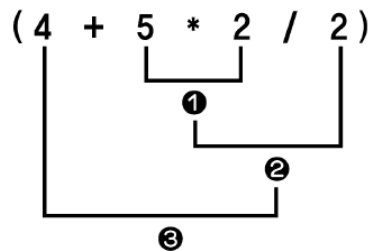
- $y = x + y - z;$ // 좌측 결합
- $y = -x;$ // 우측 결합
- $y = -x++;$ // x의 값에 단항 - 연산을 적용한 후
y에 배정하고 x를 증가
- $y = -++x;$ // x를 증가한 후 x의 값에 단항
- 연산을 적용한 후 y에 배정
- $y = -x + z;$ // x의 값에 단항 - 연산한 후 z를 더하고 그 결과를
y에 배정

2. 연산자 우선순위

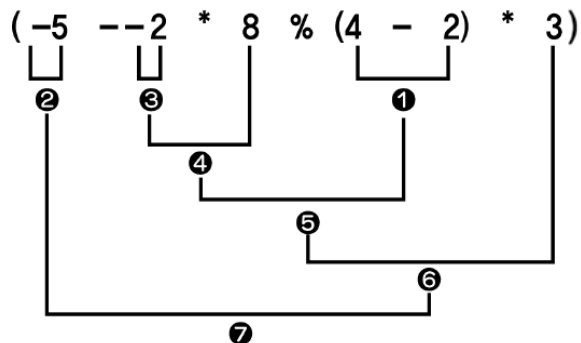
2) 연산자의 우선순위 표시

- 다음과 같은 수식의 우선순위를 표시하시오.

$$(4 + 5 * 2 / 2)$$



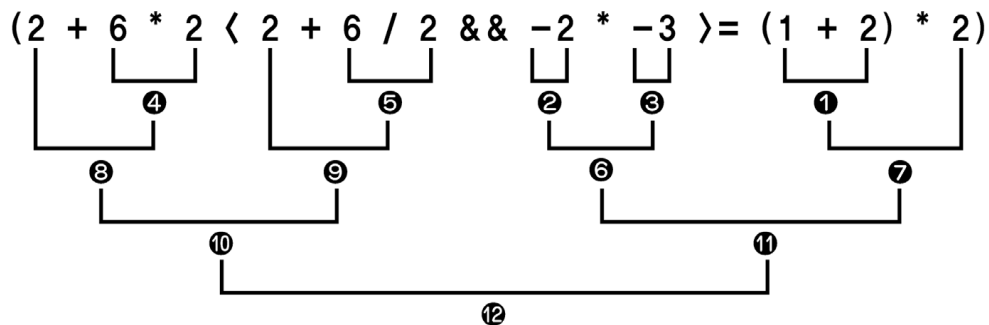
$$(-5 - -2 * 8 \% (4 - 2) * 3)$$



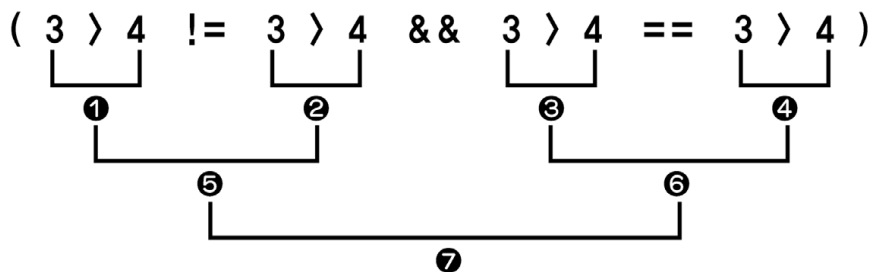
2. 연산자 우선순위

2) 연산자의 우선순위 표시

$$(2 + 6 * 2 < 2 + 6 / 2 \&\& -2 * -3 > = (1 + 2) * 2)$$



$$(3 > 4 != 3 > 4 \&\& 3 > 4 == 3 > 4)$$



3. 복합조건식

1) 복합 조건식

- 관계연산자와 논리연산자가 연결된 복합적인 조건식의 사용
 - 키가 160보다 크고, 180보다 작은 사람
 - 국어, 영어 점수 모두 60점 이상인 사람
 - 국어, 영어 과목 중 하나라도 60점 미만인 사람
- 키(height)가 160부터 180이하인 조건식
 - ~~(X) $160 \leq \text{height} \leq 180$~~
 - (O) $(\text{height} \geq 160) \ \&\& \ (\text{height} \leq 180)$
- 국어(kor), 영어(eng) 두 과목 점수 모두 60점 이상인 조건식
 - $((\text{kor} \geq 60) \ \&\& \ (\text{eng} \geq 60))$

3. 복합조건식

2) 복합 조건식 적용

- 학년(grade)이 1학년이면서 평균 평점(score) 3.5이상을 만족하는 조건식

`(grade == 1) && (score >= 3.5)`

- 성별(gender)이 "남자" 이면서 나이(age)가 21세 이상인 경우를 만족하는 조건식

`((gender == "남자") && (age >= 21))`

- 주소(addr)가 "경기도"이거나 "서울"인 사람을 만족하는 조건식

`((addr == "경기도") || (addr == "서울"))`

- 국어(kor), 영어(eng) 두 과목 모두 80점 이상이거나 두 과목의 합이 160점 이상을 만족하는 조건식

`((kor >= 80 && eng >= 80) || ((kor + eng) >= 160))`

1) 자바스크립트 특징

■ 자바스크립트?

- 썬마이크로 시스템과 넷스케이프 커뮤니케이션에서 공동 제작한 웹 브라우저용 언어
- 객체지향 프로그램 언어
- 독립적으로 실행되지 않고, 주로 HTML 언어에 포함되어 실행(프론트페이지 제작에 주목적)

■ 자바스크립트의 특징

- 브라우저의 인터프리터가 소스코드를 실행. 번역기(Compiler) 불필요
- 다른 언어로 작성된 프로그램(Java, PHP, HTML 등)들에 포함되어서 실행
- 클라이언트에서 실행되어 N/W을 통한 데이터 전송 없이 작업 처리 가능
- 변수를 선언할 때 자료형 구분 없이 예약어 var로 선언

1) 자바스크립트 특징

- 자바스크립트

- 웹 문서를 동적으로 제어하기 위해 고안된 프로그래밍 언어

- 웹 프로그래밍 핵심 언어

- HTML: 모델 담당

- CSS: 뷰 담당

- 자바스크립트: 제어 담당



2) 자바스크립트 역할

▪ 자바스크립트의 역할

- 필요 item 의 추가 및 삭제
- HTML, CSS 스타일 변경, form 유효성 검증
- 마우스와 키보드 이벤트에 대한 스크립트 실행
- 웹 클라이언트 (웹 브라우저) 제어
- 웹 서버와의 통신(AJAX) : Asynchronous JavaScript and XML, 자바 스크립트와 Xml을 이용하여 비동기 통신을 하는 기능
(페이지 전환없이 동적으로 변화하는 모습 구현 가능)

3) 자바스크립트 기본형식

- HTML문서 안에서 <script>태그를 사용하여 코딩

```
<BODY>  
  <SCRIPT>  
    자바스크립트 코딩  
  </SCRIPT>  
</BODY>
```

- 사용 스크립트가 javascript임을 명시적으로 선언

```
<BODY>  
  <SCRIPT LANGUAGE="JavaScript">  
    자바스크립트 코딩  
  </SCRIPT>  
</BODY>
```

3) 자바스크립트 기본형식

- 자바스크립트 기본 형식 (외부의 자바스크립트 파일)
 - JavaScript 소스 코드를 외부에 파일(*.js)로 저장해 두고 HTML문서에 포함시켜 사용할 것임을 선언

```
<BODY>  
  <SCRIPT LANGUAGE="JavaScript" SRC=파일명>  
  
  </SCRIPT>  
</BODY>
```

4) 자바스크립트 작성 방법

- 대소문자 구분하여 작성
- 문장은 세미콜론(;)으로 구분

바른 예	<pre>var age=25 document.write("당신의 나이는 " + age + "입니다.")</pre>
	<pre>var age=25; document.write("당신의 나이는 " + age + "입니다.");</pre>
	<pre>var age=25; document.write("당신의 나이는 " + age + "입니다.");</pre>
잘못된 예	<pre>var age=25 document.write("당신의 나이는 " + age + "입니다.")</pre>

- 큰따옴표(“)와 작은따옴표(‘ ’)를 구분하여 사용

바른 예	<pre>document.write("<div style='color: red;'> 자바스크립트 학습 </div>");</pre>
	<pre>document.write("<div style='color: red;'> 자바스크립트 학습 </div>");</pre>
잘못된 예	<pre>document.write("<div style='color: red;'> 자바스크립트 학습 </div>")</pre>

4) 자바스크립트작성 방법

- BODY내 <SCRIPT>태그에서 JavaScript 직접 작성하는 예제

```
<html>
  <head>
    <title>BODY에서 자바스크립트 태그 사용</title>
  </head>
  <body>
    <h1>BODY에서 자바스크립트 태그 사용 실습</h1>
    <SCRIPT>
      document.write("<H2>BODY에서 자바스크립트 태그 출력 결과</H2>");
      document.write("자바스크립트 사용 성공<BR>");
    </SCRIPT>
  </body>
</html>
```



[실행 결과]

BODY에서 자바스크립트 태그 사용 실습

BODY에서 자바스크립트 태그 출력 결과

자바스크립트 사용 성공

4) 자바스크립트작성 방법

- HTML 문서에 외부의 JavaScript 파일(*.js)을 사용할 것을 선언하는 예제

```
<html>
  <head>
    <title>BODY에서 자바스크립트 외부파일 선언</title>
  </head>
  <body>
    <h1>BODY에서 자바스크립트 파일 선언 실습</h1>
    <SCRIPT LANGUAGE = "JavaScript" SRC="test1.js">
    </SCRIPT>
  </body>
</html>
```

```
document.write("<H2>test1.js 파일 출력 결과</H2>");
document.write("외부 스크립트 연동 성공<BR>");
```

[test1.js 파일]

[실행 결과]

BODY에서 자바스크립트 파일 선언 실습
test1.js 파일 출력 결과
외부 스크립트 연동 성공

1) 자바스크립트 자료형

- 변수에 저장되는 자료형은 정수, 실수, 문자, 논리형 등으로 구분

구분		자료 표현	비고
정수	십진수	0 125 -30	
	16진수	0x125 0X369	0x 또는 0X 뒤에 숫자 표시
실수	십진수	65.8 -75.123	소수점 포함
	지수	0.2E3 1.89e5 2E-3	0.2×10^3 1.89×10^5 2^{-3}
문자	문자	'a' 'b' 'c'	' '로 표현
	문자열	"gift" "sunday"	" "로 표현
논리 형		true, false	true, false의 논리 값 지정

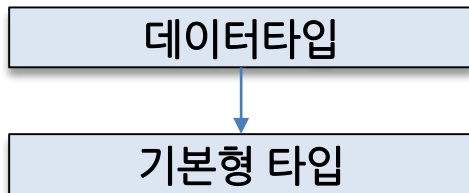
- 자료형 변환

- 변수(실수형) = 정수형 + 실수형
- 변수(문자형) = 정수형 + 문자형
- 변수(문자형) = 실수형 + 문자형

예) `var sum;`
`sum = 20 + 5.5; //sum = 25.5`
`sum = 20 + 'A'; //sum = "20A"`
`sum = 20.1 + 'b'; //sum = "20.1b"`

실수형 변환
 문자로 인식하여 연결
 문자로 인식하여 연결

2) 자바스크립트 데이터 타입



number	정수 ,실수
string	문자열, 문자
boolean	참, 거짓
array	배열
object	속성과 값으로 이루어진 집합
null	객체 값이 없음
undefined	데이터 값이 정해지지 않음

3) 자바스크립트 데이터 타입 확인

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8"/>
</head>
<body>
  <script>
    var num;    // 변수 값 undefined
    var obj=null; // 객체 값 없음
    document.write(typeof 100+"<br>");
    document.write(typeof 10.5+"<br>");
    document.write(typeof "홍길동"+"<br>");
    document.write(typeof true+"<br>");
    document.write(typeof [1,2,3]+"<br>");
    document.write(typeof {name:'홍길동', age:25}+"<br>");
    document.write(typeof num+"<br>");
    document.write(typeof obj+"<br>");
  </script>
</body>
</html>
```

number
number
string
boolean
object
object
undefined
object

4) 자바스크립트 변수명 작성 규칙

- 영어 대소문자 구별('변수C'와 '변수c'는 서로 다른 변수)
- 문자, 밑줄(_), 달러 기호(\$)로 시작 가능함
- 예약어는 변수명으로 사용 불가

- Abstract
- Arguments
- Boolean
- Break
- Byte
- Case
- Catch...

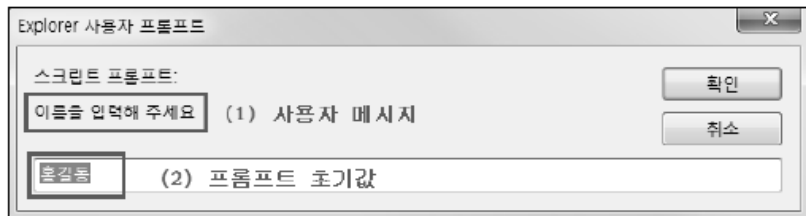
1) 자료의 입력

- 키보드로 자료를 입력 받을 때 : prompt()함수

```
var name;  
name = prompt("이름을 입력해 주세요");
```

- prompt() 함수

- 형식 : var 변수명 = prompt("메시지", "초기값");
- prompt()함수는 문자열 형태의 자료 입력만 받게 되므로 숫자를 입력하여도 문자열로 처리된다.



1) 자료의 입력

- 키보드로 자료를 입력 받을 때 : prompt() 함수

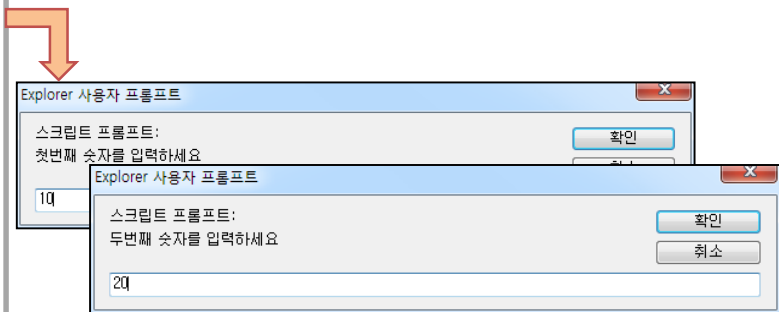
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="Generator" content="EditPlus®">
  <meta name="Author" content="">
  <meta name="Keywords" content="">
  <meta name="Description" content="">
  <title>prompt</title>
</head>
<body>
  <script>
    var name = prompt("이름을 입력해주세요", "윤철희");
  </script>

</body>
</html>
```


1) 자료의 입력

실습) 키보드로 두 수를 입력 받아 합을 출력하는 예제

```
<html>
<head>
  <title>prompt()함수 실습(1)</title>
</head>
<body>
  <SCRIPT>
    var num1, num2, sum;
    num1 = prompt("첫번째 숫자를 입력하세요");
    num2 = prompt("두번째 숫자를 입력하세요");
    sum = num1 + num2;
    document.write("두 수의 합 = " + sum);
  </SCRIPT>
</body>
</html>
```



두 수의 합 = 1020

왜 이런 결과가 나왔을까???

1) 자료의 변환 처리

- parseInt() : 지정된 변수나 문자열을 정수로 변환 시키는 함수

▶ 형식

```
(1) parseInt(변수명);  
(2) parseInt("문자열");
```

```
score = prompt("점수를 입력하세요"); // 변수 score에는 문자열이 저장
```

```
n = parseInt("100"); // 문자열 "100"을 정수 100으로 변환하여 변수 n에 저장
```

```
score = parseInt(prompt("점수를 입력하세요"));
```

```
// 괄호 안의 prompt()함수를 먼저 실행하여 입력 받은 값을 문자열로 입력 받은 후,  
// parseInt() 함수를 실행하여 문자열을 정수로 변환한 후 변수 score에 저장
```

2) 자료의 변환 처리 실습

- parseInt() 사용 예

```
var score1, score2;  
score1 = prompt("점수를 입력하세요");  
score2 = parseInt(score1);
```

- 사용자가 숫자 80을 입력하면 변수 score1에 문자열 "80"이 저장되고,
함수 parseInt(score1)에서 변수 score1에 저장된 문자열 "80"을 숫자 80으로 변환하여
변수 score2에 저장한다.

- 위 문장은 아래와 같이 간단히 표현할 수 있다.

```
var score;  
score = parseInt(prompt("점수를 입력하세요"));
```

2) 자료의 변환 처리 실습

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="Generator" content="EditPlus®">
  <meta name="Author" content="">
  <meta name="Keywords" content="">
  <meta name="Description" content="">
  <title>prompt</title>
</head>
<body>
  <script>
    var score;
    score = parseInt(prompt("점수를 입력하세요"));

    document.write(score);
  </script>

</body>
</html>
```

Explorer 사용자 프롬프트

스크립트 프롬프트:
점수를 입력하세요

확인

취소

undefined

120

2) 자료의 변환 처리 실습

- `parseFloat()` : 지정된 변수나 문자열을 실수로 변환 시키는 함수

▶ 형식

```
(1) parseFloat(변수명);  
(2) parseFloat("문자열");
```

```
var weight = prompt("체중(Kg)를 입력하세요.");
```

예를 들어 사용자가 체중을 73.5를 입력하였다면, 변수 `weight`에 "73.5"가 저장된다.
문자열은 산술연산이 불가능하게 된다.



```
var weight = parseFloat(prompt("체중(Kg)를 입력하세요.));
```

따라서, `parseFloat("73.5")`는 문자열 "73.5"를 실수 73.5로 변환 시킨다.

1) 자료의 출력

- document.write() 함수 : 화면에 값을 출력할 때 사용
 - document.write() 함수는 문자나 숫자, 변수의 값을 화면에 출력한다
 - 출력할 숫자는 괄호() 안에 그냥 기술하고, 문자열은 큰 따옴표 안에 반드시 표시한다.
 - 출력할 문자열들을 문자열 결합 연산자인 '+' 연산자를 사용하여 서로 연결하여 표현 한다

형식

```
document.write("문자열");
```

```
예) document.write(55);  
document.write("test");  
document.write(10 + "&nbsp;" + 20);  
document.write(10 + 20);  
document.write(10 + 20 + "<br>");  
document.write("test" + 10 + "program" + 20 + "<br>");
```

1) 자료의 출력

- document.write() 함수 : 화면에 값을 출력할 때 사용
 - document.write() 함수는 문자나 숫자, 변수의 값을 화면에 출력한다
 - 출력할 숫자는 괄호() 안에 그냥 기술하고, 문자열은 큰 따옴표 안에 반드시 표시한다.
 - 출력할 문자열들을 문자열 결합 연산자인 '+' 연산자를 사용하여 서로 연결하여 표현 한다.

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>출력</title>
</head>
<body>
```

```
<SCRIPT>
```

```
document.write(55);
document.write("test");
document.write(10 + "&nbsp;" + 20);
document.write(10+20);
document.write(10+20+ "<br>");
document.write("test" + 10 + "program" +
```


```
20 + "<br>");
</SCRIPT>
</body>
</html>
```

test10 203030
test10program20

2) 자료의 출력 실습

실습) 두 수의 합을 출력하는 예제이다. 출력 결과를 예측해 보세요.

```
<html>
<head>
  <title>두 수의 합을 출력하는 예제 실습</title>
</head>
<body>
  <SCRIPT>
    document.write("결과1 = " + "100 200");
    document.write("<BR>");
    document.write("결과2 = " + (100 + 200));
  </SCRIPT>
</body>
</html>
```



```
결과1 = 100200
결과2 = 300
```

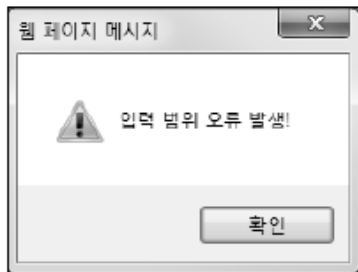

2) 자료의 출력 실습

- `alert()` 함수 : 경고 창에 메시지를 출력할 때 사용
- 경고 창에 표시할 메시지의 내용을 문자열 매개변수로 받음
- “메시지” : 경고 창에 표시할 내용

형식

```
alert("메시지");
```

예 `var name = alert("입력 범위 오류 발생!");`



1) 연산자와 수식

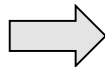
▪ 산술 연산자의 종류

연산자	사용 예	설명
+	<code>sum = 30 + 4;</code>	덧셈 (sum = 34)
-	<code>sub = 30 - 4;</code>	뺄셈 (sub = 26)
*	<code>mul = 30 * 4;</code>	곱셈 (mul = 120)
/	<code>div = 30 / 4;</code>	나눗셈 (div = 7.5)
%	<code>rem = 30 % 4;</code>	나머지 (rem = 2)
++	<code>var a = 30; a++;</code>	증가 (a = 31)
--	<code>var a = 30; a--;</code>	감소 (a = 29)

1) 연산자와 수식

▪ 산술 연산자 실습

```
<script>
  var a, b, sum, sub, mul, div, rem;
  a = 30;
  b = 4;
  sum = a + b;
  sub = a - b;
  mul = a * b;
  div = a / b;
  rem = a % b;
  document.write("덧셈 = " + sum + "<BR>");
  document.write("뺄셈 = " + sub + "<BR>");
  document.write("곱셈 = " + mul + "<BR>");
  document.write("나눗셈 = " + div + "<BR>");
  document.write("나머지 = " + rem + "<BR>");
</script>
```



```
덧셈 = 34
뺄셈 = 26
곱셈 = 120
나눗셈 = 7.5
나머지 = 2
```

1) 연산자와 수식

- 증가연산자(increment operator) : ++

(예1)

```
var a;  
a = 10;  
a++;  
document.write("a = " +a);
```

```
var a;  
a = 10;  
++a;  
document.write("a = " +a);
```

위의 두 프로그램은 변수 a에 저장된 값(10)에 1을 증가하는 연산을 수행한다.
따라서 아래와 같은 프로그램이라 할 수 있다.

```
<script>  
var a;  
a = 10;  
a = a + 1;  
document.write("a = " +a);  
</script>
```



a = 11

1) 연산자와 수식

■ 증가연산자(increment operator)

- 증가(++ 연산자가 다른 연산자와 같이 사용하면 증가연산자의 위치에 따라 연산 우선순위가 다르다.

(예2)

```
var a, b;  
a = 10;  
b = a++;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b);
```



```
var a, b;  
a = 10;  
b = a;  
a = a + 1;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b);
```

b = a++; 문장은 먼저 a의 값을 b에 저장한 후, a++; 문장을 수행한다.

(결과)

```
a = 11  
b = 10
```

1) 연산자와 수식

(예3)

```
var a, b;  
a = 10;  
b = ++a;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b);
```



```
var a, b;  
a = 10;  
a = a + 1;  
b = a;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b);
```

b = ++a; 문장은 먼저 a의 값을 1 증가한 후, a의 값을 b에 저장한다.

(결과)

```
a = 11  
b = 11
```

1) 연산자와 수식

▪ 증가연산자 실습

- 아래 두 프로그램의 출력 결과는 어떻게 되겠는가?

```
var a, b;  
a = 10;  
b = a++ + 20;  
document.write("a = " +a + "<BR>");  
document.write("b = " +b + "<BR>");
```

a = ?

b = ?

a = 11
b = 30

```
var a, b;  
a = 10;  
b = ++a + 20;  
document.write("a = " +a + "<BR>");  
document.write("b = " +b + "<BR>");
```

a = ?

b = ?

a = 11
b = 31

1) 연산자와 수식

- 감소연산자(decrement operator) : --

(예1)

```
var a;  
a = 10;  
a--;  
document.write("a = " + a);
```

```
var a;  
a = 10;  
--a;  
document.write("a = " + a);
```

위의 두 프로그램은 변수 a에 저장된 값(10)에 1을 감소하는 연산을 수행한다.
따라서 아래와 같은 프로그램이라 할 수 있다.

```
var a;  
a = 10;  
a = a - 1;  
document.write("a = " + a);
```


1) 연산자와 수식

▪ 감소연산자(decrement operator)

- 감소(--)연산자가 다른 연산자와 같이 사용하면 감소연산자의 위치에 따라 연산 우선순위가 다르다.

(예2)

```
var a, b;  
a = 10;  
b = a--;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b);
```



```
var a, b;  
a = 10;  
b = a;  
a = a - 1;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b);
```

b = a--; 문장은 먼저 a의 값을 b에 저장한 후, a--; 문장을 수행한다.

(결과)

```
a = 9  
b = 10
```

1) 연산자와 수식

(예3)

```
var a, b;  
a = 10;  
b = --a;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b);
```



```
var a, b;  
a = 10;  
a = a - 1;  
b = a;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b);
```

$b = --a$; 문장은 먼저 a 의 값을 1 감소한 후, a 의 값을 b 에 저장한다.

(결과)

```
a = 9  
b = 9
```

1) 연산자와 수식

■ 감소연산자 실습

- 아래 두 프로그램의 출력 결과는 어떻게 되겠는가?

```
var a, b;  
a = 10;  
b = a-- + 20;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b + "<BR>");
```

a = ?

b = ?

a = 9
b = 30

```
var a, b;  
a = 10;  
b = --a + 20;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b + "<BR>");
```

a = ?

b = ?

a = 9
b = 29

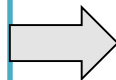
2) 연산자의 종류

▪ 복합 연산자의 종류

연산자	사용 예	설명
<code>+=</code>	<code>a += b;</code>	<code>a = a + b;</code>
<code>-=</code>	<code>a -= b;</code>	<code>a = a - b;</code>
<code>*=</code>	<code>a *= b;</code>	<code>a = a * b;</code>
<code>/=</code>	<code>a /= b;</code>	<code>a = a / b;</code>
<code>%=</code>	<code>a %= b;</code>	<code>a = a % b;</code>
<code><<=</code>	<code>a <<= b;</code>	<code>a = a << b;</code>
<code>>>=</code>	<code>a >>= b;</code>	<code>a = a >> b;</code>

3) 복합연산자 실습

```
<script>  
  var sum, n;  
  sum = 30;  
  n = 4;  
  
  sum += n;  
  document.write("sum = " +sum + "<BR>");  
  sum -= n;  
  document.write("sum = " +sum + "<BR>");  
  sum *= n;  
  document.write("sum = " +sum + "<BR>");  
  sum /= n;  
  document.write("sum = " +sum + "<BR>");  
  sum %= n;  
  document.write("sum = " +sum + "<BR>");  
</script>
```



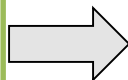
```
sum = 34  
sum = 30  
sum = 120  
sum = 30  
sum = 2
```

4) 논리 연산자

- 논리연산자(logical operator)

- 피연산자에 대한 연산 결과로 true 또는 false의 논리 값을 반환한다.

```
var a, b, c;  
a = !true;  
b = (10 != 20) || false;  
c = 100 > 50;  
d = true && false;  
document.write("a = " + a + "<BR>");  
document.write("b = " + b + "<BR>");  
document.write("c = " + c + "<BR>");  
document.write("d = " + d + "<BR>");
```



```
a = false  
b = true  
c = true  
d = false
```

5) 조건 연산자

▪ 조건연산자

- 변수 = (조건식) ? 식1 : 식2;
- 조건식이 참이면 식1을, 거짓이면 식2를 변수에 할당한다.

(예) `max = (a > b) ? a : b;`

변수 a, b에 저장된 값을 비교하여, a가 b보다 크다면 a의 값을 max에 할당하며, a가 b보다 크지 않다면 작은 값인 변수 b에 저장된 값을 max에 저장한다.

따라서 위 문장은 변수 a, b에 저장된 값 중 큰 수를 max에 저장하는 문장이다.

```
var num1, num2, max;  
num1 = parseInt(prompt("첫 번째 수를 입력하시오"));  
num2 = parseInt(prompt("두 번째 수를 입력하시오"));  
max = (num1 > num2) ? num1 : num2;  
document.write("maximum value : " + max + "<BR>");
```



maximum value : 30

1) 함수의 정의

- 특정 기능을 수행하는 단위 프로그램

- 한 프로그램 안에서 동일한 과정에 대한 반복 수행이 필요하다면, 반복 처리 과정에 대한 함수 명을 부여해서 한 번만 코딩한 후, 필요 시 마다 호출해서 사용

```
Function 함수명 ( [매개변수 목록])  
{  
    함수에서 처리할 명령문;  
  
    [return 변수명;]  
}
```


2) 함수의 선언과 호출

- 함수명 : 함수 이름
- 인자 : 함수를 호출할 때 전달하는 입력값
- 매개 변수 : 함수 호출문에서 전달한 인자를 받기 위해 선언된 변수
- function : 함수를 선언할 때 사용하는 키워드
- return : 함수에서 수행한 결과값을 반환할 때 사용하는 키워드

```
function 함수명(매개 변수1, 매개 변수2, ...) { // 함수 선언
    실행 문장;
    return 반환값;
}
함수명(인자1, 인자2, ...); // 함수 호출
```

2) 함수의 선언과 호출

▪ 함수 호출

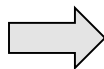
- 함수명 ();
- 함수명 ([실인수 목록]);
- 변수명 = 함수명([실인수 목록]);

```
<script>
  var q1="함수 선언 전 호출";
  var q2="함수 선언 후 호출";

  message(q1);      // 함수 선언 하기 전 호출

  function message(a) { // 함수를 선언함
    document.write("함수 : " + a + "<br>");
  }

  message(q2);      // 함수 선언 후 호출
  message(q1);      // 함수 재 호출
</script>
```



함수 : 함수 선언 전 호출
함수 : 함수 선언 후 호출
함수 : 함수 선언 전 호출

3) 함수 호출 실습

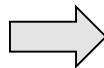
▪ Sum 함수 사용

```
<pre>
    <h2> Sum함수 사용<hr>

    <script>
        function add(a,b) // 합계 처리
        {
            var s = a+b;
            return s;
        }

        function sub(a,b) // 뺄셈 처리
        {
            var t = a-b;
            return t;
        }

        var tot1, tot2, n = 5, m = 10;
        tot1 = add(n,m); //add 함수 호출
        tot2 = sub(n,m); //sub 함수 호출
        document.write("tot1 =" + tot1
        + "&nbsp;" + "tot2 =" + tot2);
    </script>
```



Sum함수 사용

tot1 =15 tot2 =-5

1) 함수 호출 실습

- 형식

<form>

<input type = “button” value = “ ” onClick = “함수명”>

</form>

Type : 폼에 출력할 입력 양식인 버튼을 지정

Value : 버튼에 표시할 문자열

onClick : 버튼을 클릭하면 실행할 함수명을 지정

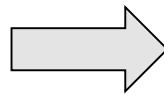
1) 함수 호출 실습

```
<body>
  <pre>

  <script>
    function first(name, age) { // 함수 선언
      document.write("교수 이름 : " + name +
" <br>");
      document.write("교수 나이 : " + age + " <br>");
    }
  </script>

  <button type= "button" onclick= "first('윤철희', 44)">
교수 정보</button>

</body>
```



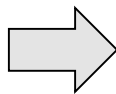
교수 정보

교수 이름 :윤철희
교수 나이 :44

1) 반환값 출력

```
Function 함수명(매개변수 1, 2,3) {  
    실행 문장;  
    Return 반환값;  
}  
  
Result = 함수명(인자1,인자2,인자3);
```

```
<script>  
var result;  
function add(name, n) {  
    document.write(name + " 학생이 1부터 " + n + "까지 덧셈 수행  
<br>");  
    var sum=0;  
    for(var i=1; i<=n; i++) {  
        sum+=i; // sum=sum+i  
    }  
    return sum;  
}  
result=add('공쥐', 10);  
document.write("결과 : " + result + "<p/>");  
result=add('팔쥐', 100);  
document.write("결과 : " + result + "<p/>");  
</script>
```



공쥐 학생이 1부터 10까지 덧셈 수행
결과 : 55

팔쥐 학생이 1부터 100까지 덧셈 수행
결과 : 5050

2) 출력 실습

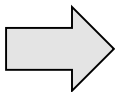
- 정수와 실수값을 입력 받아, 두수에 대한 덧셈과 뺄셈을 수행
 - 그 결과를 alert 창에 출력하는 함수를 정의한다.

```
<body>
  <h2> 실습 </h2>
  <hr>
  <form>
    <input type = "button" value = "덧셈" onClick = "add(n,m)">
    <input type = "button" value = "뺄셈" onClick = "sub(n,m)">
  </form>
  <script>
    function add(a, b)
    {
      var s = a+b;
      alert("덧셈 : " + a + "+" + b + "=" + s);
    }

    function sub(a, b) {

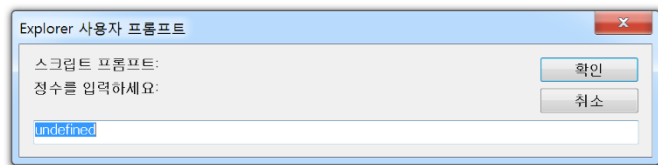
      var t = a-b;
      alert("뺄셈 : " + a + "-" + b + "=" + t);
    }

    var n,m;
    n = parseInt(prompt("정수를 입력하세요: "));
    m = parseFloat(prompt("실수를 입력하세요: "));
  </script>
</body>
```



실습

덧셈 뺄셈

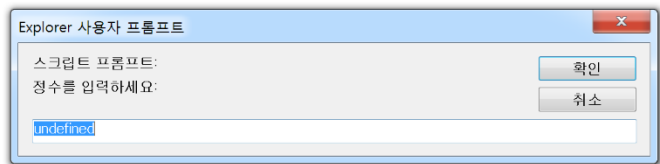


2) 출력 실습

- 정수와 실수값을 입력 받아, 두수에 대한 덧셈과 뺄셈을 수행
 - 그 결과를 alert 창에 출력하는 함수를 정의한다.

실습

덧셈 뺄셈



정수 : 35, 실수 13.5

