

C Language

C프로그래밍및실습

담당 교수: 최희식
dali3054@ssu.ac.kr

목차

- 1) 수식과 연산자 개요
- 2) 산술 연산자
- 3) 대입 연산자
- 4) 관계 연산자
- 5) 논리 연산자
- 6) 그 외 연산자
- 7) 연산자 우선순위와 결합 수칙

1) 수식과 연산자 개요

■ 연산자

- 데이터를 가공하고 처리하기 위한 가장 기본 도구
- 연산 종류에 따른 분류: 산술 연산자, 관계 연산자, 논리 연산자, 증감 연산자, 비트 연산자, 대입 연산자, 조건 연산자 등
- 피연산자 개수에 따른 분류: 단항 연산자, 이항 연산자, 삼항 연산자

■ 수식

- 피연산자들과 연산자의 조합으로 어떠한 값을 갖는 요소

`num = 10;`

⇒ 상수인 10이 수식

`num1 = num;`

⇒ 변수인 num이 수식

`num2 = num + 1;`

⇒ 연산식인 num+1이 수식

2) 산술 연산자

■ 산술 연산자 종류

- 사칙 연산자(+, -, *, /)와 나머지 연산자(%)가 있음
- **+** (더하기), **-** (빼기), ***** (곱하기)

```
int math = 99, korean = 90, science = 94;

// 과목의 총합을 변수 total에 저장
int total = math + korean + science;

printf("총점 : %d\n", total);

return 0;
```

2) 산술 연산자

- 나누기와 나머지 연산자
 - 피연산자의 자료형에 따라 계산 결과 다름

연산자	정수 연산	부동소수 연산
/	몫	실수 나눗셈
%	나머지	(정의 되지 않음)

11 / 4 ⇒ 연산 결과: 2

11 % 4 ⇒ 연산 결과: 3

11.0 / 4.0 ⇒ 연산 결과: 2.75

11.0 % 4.0 ⇒ 컴파일 오류

2) 산술 연산자

- 나머지 연산자 활용예: 정수의 자릿수 구하기

- 일의 자릿수 계산

```
int d = 2715 % 10;  
printf("일의 자릿수: %d\n", d);
```

실행 결과

일의 자릿수: 5

- 백의 자릿수 계산

```
int d = 2715 / 100 % 10;  
printf("백의 자릿수: %d\n", d);
```

실행 결과

백의 자릿수: 7

✓ 계산 과정

✓ $2715 / 100 \rightarrow 27$ (백미만 자릿수 제거)

✓ $27 \% 10 \rightarrow 7$ (일의 자릿수 추출)

2) 산술 연산자

- [예제 4.1]

- 다음 연산의 결과를 예측해보고, 프로그램을 작성하여 확인해보자.

✓ $5 / 2 * 3$ [결과 :]

✓ $3 * 5 / 2$ [결과 :]

2) 산술 연산자

- 사용자로부터 초를 입력받아, 산술 연산자를 활용하여 시간, 분, 초를 환산하여 출력하는 프로그램을 작성하시오.

[실행 결과]

초를 입력하세요:5000
입력한 초 5000는 1시 23분 20초 입니다.

2) 산술 연산자

▪ 산술 연산과 자료형

- 연산 결과도 자료형이 정해져 있어야 함
- 산술 연산의 경우 피연산자의 자료형에 따라 연산 결과값의 자료형이 결정됨
 - ✓ 정수형과 정수형 → 정수형
 - ✓ $5 / 2 \rightarrow$ 정수 2
 - ✓ 부동소수형과 부동소수형 → 부동소수형
 - ✓ $5.0 / 2.0 \rightarrow$ 부동소수 2.5
 - ✓ 정수형과 부동소수형 → 부동소수형 (정보 손실 방지)
 - ✓ $5.0 / 2 \rightarrow$ 부동소수 2.5
 - ✓ $5 / 2.0 \rightarrow$ 부동소수 2.5

2) 산술 연산자

- 피연산자가 모두 정수형인데, 부동소수 연산을 하고 싶으면?
 - 명시적 형변환 이용
 - ✓ 아래 예에서 각 x의 결과는?
 - ✓ 형 변환이 적용되는 범위에 주의

```
int a = 5, b = 2;
```

```
double x;
```

```
x = (double) a / b;    → a의 자료형을 변환
```

```
x = (double) (a / b); → a/b의 결과 값의 자료형 변환
```


목차

- 1) 수식과 연산자 개요
- 2) 산술 연산자
- 3) **대입 연산자**
- 4) 관계 연산자
- 5) 논리 연산자
- 6) 그 외 연산자
- 7) 연산자 우선순위와 결합 수칙

3) 대입 연산자

- 대입 연산자 =

- 연산자 오른쪽 수식의 값을 왼쪽 변수에 대입
(수학의 등호와 완전히 다른 의미)


변수 = 수식

✓ $a = c;$ ➔ '변수 a'에 '변수 c의 값' 대입

✓ $a = a + 1;$ ➔ '변수 a'에 '변수 a의 값에 1 더한 값' 대입

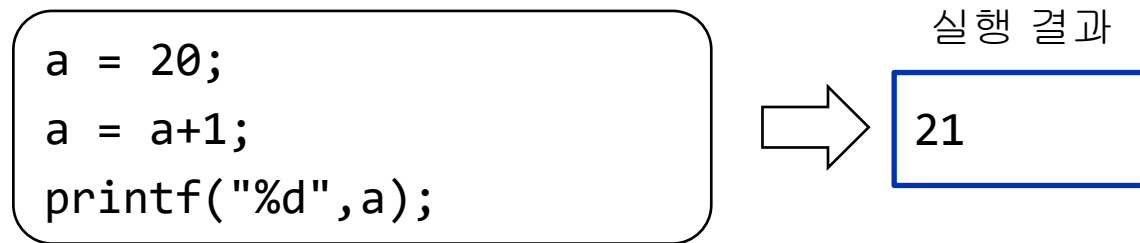
- 다음은 컴파일 에러 발생 (의미적으로 불가능)

✓ $200 = x;$

✓ $x + 2 = 0;$

3) 대입 연산자

- `a = a + 1;`
 - "a와 a+1이 같다"는 뜻이 아니라
 - "변수 a에 a+1의 값을 저장하라($a \leftarrow a+1$)"는 뜻
- 동작 과정
 - ✓ 대입문 수행 전에 변수 a에 20이 저장되어 있었다면
 - ✓ `a = a+1` \rightarrow `a = (a에 저장되어 있던)20 + 1`



3) 대입 연산자

- 복합 대입 연산자: 대입 연산자와 산술 연산자의 결합
 - $a \text{ += } x \rightarrow a$ 의 값을 x 만큼 증가시킴

복합 대입 연산	동일 대입문
$a \text{ += } x$	$a = a + (x)$
$a \text{ -= } x$	$a = a - (x)$
$a \text{ *= } x$	$a = a * (x)$
$a \text{ /= } x$	$a = a / (x)$
$a \text{ %= } x$	$a = a \% (x)$

3) 대입 연산자

- 복합 대입 연산자: 대입 연산자와 산술 연산자의 결합 수행 결과는?

```
int a = 3;  
int b = 2;  
a += 5; // a = a+5; 결과: [      ]  
a /= b; // a = a/b; 결과: [      ]  
a %= 3; // a = a%3; 결과: [      ]
```

3) 대입 연산자

증감 연산자

- 변수의 값을 **1씩 증가(++)** 혹은 **감소(--)** 시키는 단항 연산자
- 변수의 앞에 오느냐 뒤에 오느냐에 따라 수식의 해석이 달라진다.

증감 연산	의미
++a	a의 값 1 증가 → a의 값 사용
--a	a의 값 1 감소 → a의 값 사용
a++	a의 값 사용 → a의 값 1 증가
a--	a의 값 사용 → a의 값 1 감소

```
a = 1;  
b = ++a;  
printf("a: %d\n", a);  
printf("b: %d\n", b);
```

```
a = 1;  
b = a++;  
printf("a: %d\n", a);  
printf("b: %d\n", b);
```


목차

- 1) 수식과 연산자 개요
- 2) 산술 연산자
- 3) 대입 연산자
- 4) **관계 연산자**
- 5) 논리 연산자
- 6) 그 외 연산자
- 7) 연산자 우선순위와 결합 수칙

4) 관계 연산자

■ 관계 연산자

- 왼쪽과 오른쪽의 **대소 관계를 비교**하는 연산자
- 연산의 결과는 참 아니면 거짓으로, 참이면 **1**이고 거짓이면 **0**
 - ✓ (참고) C언어에서는 0이 아닌 값은 모두 참으로 간주

관계연산	의미
$x == y$	x 의 값과 y 의 값이 같다
$x != y$	x 와 y 가 같지 않다
$x < y$	x 가 y 보다 작다
$x <= y$	x 가 y 보다 작거나 같다
$x > y$	x 가 y 보다 크다
$x >= y$	x 가 y 보다 크거나 같다

```
a = (4 < 5);  
printf("a: %d\n", a);
```

```
a = (4 == 5);  
printf("a: %d\n", a);
```

4) 관계 연산자

■ 실습 예제

- 다음 소스 코드의 실행 결과를 예상해보고, 코드를 작성하여 확인해 보시오.

```
int a = 3;
printf("%d\n", a > 4);    결과: [      ]
printf("%d\n", a < 4);    결과: [      ]
printf("%d\n", a == 5);   결과: [      ]
printf("%d\n", a != 3);   결과: [      ]
printf("%d\n", 2 >= a);   결과: [      ]
printf("%d\n", a <= a+1); 결과: [      ]
```

목차

- 1) 수식과 연산자 개요
- 2) 산술 연산자
- 3) 대입 연산자
- 4) 관계 연산자
- 5) 논리 연산자
- 6) 그 외 연산자
- 7) 연산자 우선순위와 결합 수칙

5) 논리 연산자

■ 논리 연산자

- 논리 연산 값으로 참이면 1이고 거짓이면 0
(0을 제외한 모든 값은 참으로 간주)

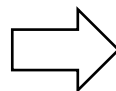
논리 연산	의미	연산 결과
<code>!x</code>	논리부정 (NOT)	x가 참이면 거짓, 거짓이면 참
<code>x && y</code>	논리곱 (AND)	x, y가 둘 다 참이면 참이고, 그렇지 않으면 거짓
<code>x y</code>	논리합 (OR)	x, y 중 하나라도 참이면 참이고, 그렇지 않으면 거짓

```
int x = 1, y = 0;

printf("%d\n", !x);
printf("%d\n", x&&x);
printf("%d\n", x&&y);
printf("%d\n", x||y);
printf("%d\n", y&&y);
```

실행 결과

```
0
1
0
1
0
```



5) 논리 연산자

■ 실습

- 다음 소스 코드의 실행 결과를 예상해보고, 코드를 작성하여 확인해 보시오.

```
int a = 3;
```

```
int b = 5;
```

```
printf("%d\n", (a>=3)&&(b<6));    결과: [      ]
```

```
printf("%d\n", (a!=3)&&(a>2));    결과: [      ]
```

```
printf("%d\n", (b!=5)|| (a==1));  결과: [      ]
```

```
printf("%d\n", (a!=!b)|| (b==2)); 결과: [      ]
```

5) 논리 연산자

▪ [예제 4.2]

- 다음 논리 연산을 계산해보고 프로그램을 작성하여 확인해보자.

```
int a = 3, b = 5;
```

```
printf("%d\n", a>=3 && b<6 );      결과: [      ]
```

```
printf("%d\n", a!=3 && a>2 );      결과: [      ]
```

```
printf("%d\n", b!=5 || a==1 );     결과: [      ]
```

```
printf("%d\n", a!=b || b==2 );     결과: [      ]
```

```
printf("%d\n", !(a!=b) || b==2 );  결과: [      ]
```

목차

- 1) 수식과 연산자 개요
- 2) 산술 연산자
- 3) 대입 연산자
- 4) 관계 연산자
- 5) 논리 연산자
- 6) 그 외 연산자**
- 7) 연산자 우선순위와 결합 수칙

6) 조건 연산자

- 조건 연산자를 사용하여 최대값을 구하시오.
- `int x=20, y=40, z=15;`

가장 큰 값=40

6) 조건 연산자

- 사용자로부터 양의 정수 하나를 입력받아 홀수이면 "odd", 짝수이면 "even"을 출력하는 조건 연산자를 이용하여 구하시오.

[실행 결과]

양의 정수 입력: 24
even

목차

- 1) 수식과 연산자 개요
- 2) 산술 연산자
- 3) 대입 연산자
- 4) 관계 연산자
- 5) 논리 연산자
- 6) 그 외 연산자
- 7) 연산자 우선순위와 결합 수칙

감사합니다.

