C 프로그래밍 및 실습

4. 수식과 연산자

세종대학교

목차

1) 수식과 연산자의 개념

- **연산자**
  - 프로그램을 개발할 때 가장 기본이 되는 문법
  - 연산자의 예로는 산술 연산자, 관계 연산자, 논리 연산자, 증감 연산자, 비트 연산자, 대입 연산자, 조건 연산자 등이 있다.

- **수식**
  - 피연산자들과 연산자의 조합으로 어떠한 값을 갖는 요소

`int num = 10;`	상수인 **10**이 수식
`int num1 = num;`	변수인 **num**이 수식
`int num2 = num + 1;`	연산식인 **num+1**이 수식
`int num3 = strlen("abc");`	함수 호출의 리턴 값이 있는 **strlen("abc")**이 수식

- **연산식**
  - 연산자를 이용해서 만든 수식으로 연산자와 피연산자로 구성됨.

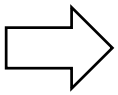
2) 산술 연산자

- **산술 연산자**
  - 산술 연산자에는 사칙 연산자와 나머지 연산자로 구성.
  - 나머지 연산자란 나눗셈의 몫을 의미하는 것이 아니라 나머지를 구하는 연산자이다.
  - 모든 산술 연산자는 두 개의 피연산자를 가지며 왼쪽에서 오른쪽으로 연산을 수행

- **덧셈/뺄셈 연산자**
  - A+B는 A에 B를 더하는 것이고 A-B는 A에서 B를 빼는 것이다.

소스 코드

```
int A = 3;
int B = 6;
printf("%d\n", A+B);
printf("%d\n", A-B);
```



실행 결과

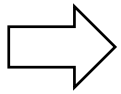
```
9
-3
```

2) 산술 연산자

- **곱셈/나눗셈 연산자**
  - A*B는 A에 B를 곱하는 것이고 A/B는 A를 B로 나누는 것이다.
  - 나눗셈의 경우 A와 B의 자료형에 따라 결과가 확연히 다르므로 주의.
- **나머지 연산자**
  - A%B는 A를 B로 나눈 나머지를 구한다.
  - 정수형만 가능
    - ✓ 정수간에는 /와 %는 **몫**과 **나머지** 연산자 역할

소스 코드

```
int A = 10;
int B = 3;
printf("%d\n", A * B);
printf("%d\n", A / B);
printf("%d\n", A % B);
```



실행 결과

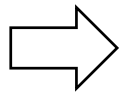
```
30
3
1
```

2) 산술 연산자

- **산술 연산 시 자료형 변환**
  - 다른 자료형끼리 연산 시 더 넓은 범위의 자료형으로 자동 형변환 된다.
    - ✓ 정수끼리 연산 → 정수
    - ✓ 실수끼리 연산 → 실수
    - ✓ 정수와 실수 연산 → 실수

소스 코드

```
printf("%f\n", 3 + 4.0f);
printf("%f\n", 10 / 3.0f);
```



실행 결과

```
7.000000
3.333333
```

2) 산술 연산자

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

```
printf("%d\n", 3*4);
printf("%f\n", 2.0f*3.0f);
printf("%f\n", 2.0f*4);
printf("%d\n", 10/2);
printf("%d\n", 10/3);
printf("%f\n", 10/3.0f);
printf("%d\n", 10%3);
printf("%f\n", 3.0f+3/2);
```

2) 산술 연산자

- **실습 예제**
  - 실행결과

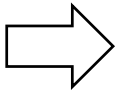
```
12
6.000000
8.000000
5
3
3.333333
1
4.000000
```

2) 산술 연산자

- **연산자 우선순위**
  - 여러 연산자가 공존할 경우 연산자 우선순위에 따라 실행 순서가 정해진다.
  - 곱셈/나눗셈/나머지 > 덧셈/뺄셈 > 대입연산 순으로 우선순위가 높음.
- **괄호 연산자**
  - 괄호를 사용하여 연산 순서를 조정할 수 있다.

소스 코드

```
printf("%d\n", 3+6*3);
printf("%d\n", (3+6)*3);
printf("%f\n", 3.0f+3/2);
printf("%f\n", (3.0f+3)/2);
printf("%f\n", 3.0f/(3+2));
```



실행 결과

```
21
27
4.000000
3.000000
0.600000
```

2) 산술 연산자

- **자동 형 변환**
  - 대입연산 시 오른쪽에 있는 값은 왼쪽에 있는 변수의 자료형으로 자동 형 변환이 이루어진다.
  - 수식연산 시 더 넓은 범위의 자료형으로 자동 형 변환이 이루어진다.

- **강제 형 변환**
  - (자료형)A는 A의 자료형을 명시한 자료형으로 강제로 변경하다.
  - 정수 → 실수, 실수 → 정수, int → char 으로도 변환 가능.
  - 강제 형 변환 시에는 정보가 유실되지 않도록 주의.

소스 코드

```
printf("%d\n", (int)3.81f);
```

실행 결과

```
3
```

```
printf("%d\n", 3.81f);  엉터리 수 출력
```

2) 산술 연산자

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

```
int a;
a = 3.3f;
printf("%d\n", a);
printf("%f\n",3.3f+5);
printf("%f\n", (float)a);
```

2) 산술 연산자

- **실습 예제**
  - 실행 결과

```
3
8.300000
3.000000
```

3) 대입 연산자

- **대입 연산자**
  - 수학의 등호와 완전히 다른 의미
  - 좌변의 변수가 나타내는 메모리에 우변의 값을 저장한다.
  - 기존에 저장되어 있던 값이 있다면 기존의 값에 덮어쓴다.

변수명 = 값

- **가능한 대입 연산자 형태**
  - 변수명 = 값;
  - 변수명 = 변수;
  - 변수명 = 수식;

```
x = 5;
y = x;
y = x+30;
```

✓ 200 = x; // 컴파일 에러
x+2 = 0; // 컴파일 에러

3) 대입 연산자

```
int sum=0, kor=90, eng=80;
sum = sum + kor ;
sum = sum + eng ;
```

- **변수에 값을 대입시키는 문장**
  - a = a + 1;
  - "a와 a+1이 같다"는 뜻이 아니라
  - "a+1의 값을 변수 a에 저장하라(a ← a+1)"는 뜻

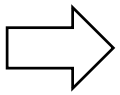
```
int prod=1 ;
prod = prod * 2;
prod = prod * 3;
```

- **대입문 동작 과정**
  - 대입문 수행 전에 변수 a에 20이 저장되어 있었다면

```
a = a+1;  // a = (a에 저장되어 있던)20 + 1
```

소스 코드

```
a = 20;
a = a+1;
printf("%d",a);
```



실행 결과

```
21
```

3) 대입 연산자

- **연속 대입**
  - 대입 연산 결과를 이용하면 연속 대입(cascading assignment) 연산을 수행할 수 있다.

  - `a = b = c = 3;` 이 문장의 의미는

    ```
    c = 3;  //오른쪽부터 수행함에 주의
    b = c;
    a = b;
    ```

    와 같다.

    ✓ `a = b+2 = c = d = 5` //컴파일 에러

    `b+2 = c` (X)

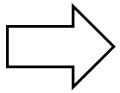
3) 대입 연산자

- **복합 대입 연산자**
  - 대입 연산자와 산술 연산자를 합쳐 놓은 연산자
  - 복합 대입 연산자를 사용하면 소스를 간결하게 만들 수 있다.

복합 대입 연산자	의미
x += y	x = x + y
x -= y	x = x – y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y

소스 코드

```
int a = 3;
int b = 2;
a += 5;
printf("%d\n", a);
a /= b;
printf("%d\n", a);
a %= 3;
printf("%d\n", a);
```



실행 결과

```
8
4
1
```

3-1) 증감 연산자

- **증감 연산자**
  - ++기호나 --기호를 사용하여 변수의 값을 1증가 혹은 감소시키는 연산.
  - 증감연산자는 단항 연산자이다.
  - 변수의 앞에 오느냐 뒤에 오느냐에 따라 수식의 해석이 달라진다.

증감 연산자	의미
++x	x의 값을 먼저 1 증가시킨 후 다른 연산에 사용한다.
x++	x의 값을 먼저 사용한 후 1 증가시킨다
--x	x의 값을 먼저 1 감소시킨 후 다른 연산에 사용한다.
x--	x의 값을 먼저 사용한 후 1 감소시킨다

```
int x = 5;
int y = x++;
printf("%d", x);
printf("%d", y);
```

```
int x = 5;
int y =++x;
printf("%d", x);
printf("%d", y);
```

3) 대입 연산자

- **다음 각각에 해당하는 C언어 수식을 작성하시오**

  (각 문제는 독립적이고, 문제마다 하나의 수식으로 작성할 것)

  1. 변수 sum에 1부터 5까지 더한 값을 대입
  2. 변수 x의 값에 2를 곱한 후 변수 y를 더한 값을 변수 x에 대입
  3. 변수 x에는 x+2의 값을, 변수 y에는 변수 x의 값을, 변수 z에는 변수 y의 값을 차례로 대입 (연속 대입 사용할 것)
  4. 변수 x의 값을 3만큼 감소 (복합대입연산자 사용할 것)
  5. 변수 x의 값을 3으로 나눈 나머지를 x에 대입 (복합대입연산자 사용할 것)
  6. 변수 x의 값을 1만큼 감소 (증감연산자 사용할 것)
  7. 변수 x의 값을 1만큼 감소시키고, 그 결과에 변수 y와 z의 합을 곱한 결과를 변수 z에 대입
  8. 변수 x의 값을 y배 시키고, y는 1만큼 증가

4) 관계 연산자

- **관계 연산자**
  - 왼쪽과 오른쪽의 관계를 비교하는 연산.
  - 연산의 결과는 항상 참 아니면 거짓이 된다.
  - 참의 논리는 결과가 **1**이고 거짓의 논리는 결과가 **0**이 된다.
  - C언어에서는 0이 아닌 값이면 모두 참이라고 한다.

관계 연산자	의미
<	왼쪽이 오른쪽 보다 작다.
<=	왼쪽이 오른쪽보다 작거나 같다.
==	왼쪽과 오른쪽이 같다.
>=	왼쪽이 오른쪽보다 크거나 같다.
>	왼쪽이 오른쪽보다 크다.
!=	왼쪽과 오른쪽이 같지 않다.

```
int x = 5;
int y = 6;
printf("%d", x>y);
printf("%d", x<=y);
```

4) 관계 연산자

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

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

4) 관계 연산자

- **실습 예제**
  - 실행 결과

```
0
1
0
0
0
1
```

4) 관계 연산자

- **다음 각각에 해당하는 C언어 수식을 작성하시오**

  (각 문제는 독립적이고, 문제마다 하나의 수식으로 작성할 것)

  1. 변수 x는 7과 같다.
  2. 변수 x와 y의 합은 3보다 크다.
  3. 변수 y는 x+1보다 작거나 같다.
  4. 변수 x는 3과 같지 않다.
  5. 변수 x는 y보다 크거나 같다
  6. 12를 5로 나눈 나머지는 x를 5로 나눈 나머지보다 작거나 같다.
  7. 변수 x에서 y를 뺀 값은 음수이다.
  8. x를 1만큼 증가시키고, 그 값은 y와 같지 않고, y를 1만큼 감소시킨다.

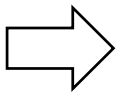
5) 논리 연산자

- **논리 연산자**
  - 복수개의 조건을 결합하여 전체의 논리가 참인지 거짓인지 판별하는데 사용됨.

논리 연산자	의미
!	논리의 부정 (NOT)
&&	논리 곱 (AND)
||	논리 합 (OR)

소스 코드

```
int a = 1, b = 0;
printf("%d\n", !a);
printf("%d\n", a&&a);
printf("%d\n", a&&b);
printf("%d\n", a||b);
printf("%d\n", b&&b);
```



실행 결과

```
0
1
0
1
0
```

5) 논리 연산자

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

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

printf("%d\n", (a>=3)&&(b<6));
printf("%d\n", (a!=3)&&(a>2));
printf("%d\n", (b!=5)||(a==1));
printf("%d\n", (a!=!b)||(b==2));
```

5) 논리 연산자

- **실습 예제**
  - 실행 결과

```
1
0
0
1
```

5) 논리 연산자

- **다음 각각에 해당하는 C언어 수식을 작성하시오**

  (각 문제는 독립적이고, 문제마다 하나의 수식으로 작성할 것)

  1. x가 참이거나 y가 참이다.
  2. x는 참이 아니다. (즉, x는 거짓이다.)
  3. x가 3보다 크거나 y가 4보다 작다.
  4. y는 3이 아니면서 x보다는 크거나 같다.
  5. 2 < x < 9
  6. x는 0보다 크고, 2로 나누어 떨어지고, 5로는 나누어 떨어지지 않는다. (2로 나누어 떨어진다 == 2로 나눈 나머지는 0이다)

```
int a; // >는 차례로 계산
a = 4 > 3 > 2 ?  44 : 33 ;
printf("%d", a);
```

```
        0 ?  printf("3") : printf("2") ;
       10 ?  printf("3") : printf("2") ;
4 > 3 > 2 ?  printf("3") : printf("2") ;
```

- **조건 연산자**
  - if ~ else 문(5장)을 대신하여 사용할 수 있는 연산자.
  - 피연산자 수가 3개가 되어 삼항 연산자라고도 불린다.

조건 ? A : B	조건이 true인 경우 A를 반환
	조건이 false인 경우 B를 반환

```
int a = 10;
int b = 5;
int c;
c = a<b ? a : b;
printf("%d",c);
```

실행 결과

```
5
```

아래와 같이 표현하는 것도 가능
(주의! 괄호 생략하면 구문 오류 발생)

```
a<b ? (c=a) : (c=b);
```

6) 조건 연산자

- **다음 각각에 해당하는 C언어 수식을 작성하시오**
  (각 문제는 독립적이고, 문제마다 하나의 수식으로 작성할 것)

  1. x가 y보다 크면 z에 3대입하고, 작거나 같으면 z에 2대입.

  2. x와 y중 큰 값을 반환

  3. x와 y가 같지 않고 y가 7보다 작으면, x값 1 증가, 그렇지 않으면 y값 1 감소

7) 연산자의 우선순위와 결합수칙

- **연산자 우선순위**
  - 여러 연산자가 함께 사용된 경우 우선순위에 의존.
  - 다른 순서로 연산을 하고 싶은 경우 괄호를 사용.

- **결합 수칙**
  - 연산의 순서를 나타냄.
  - 연산자 우선순위가 같은 경우 결합수칙에 의존.

  - 예) 5 / 2 * 4 의 결과는?
    - ✓ / 와 * 의 우선순위는 동일
    - ✓ / 와 * 의 결합수칙은 '왼쪽우선'

7) 연산자의 우선순위와 결합수칙

우선순위	연산자	결합수칙
1	( ) [ ] . ->	왼쪽 우선
2	*(간접지정) & ! ++ --	오른쪽 우선
3	*(곱셈) % /	왼쪽 우선
4	+ -	왼쪽 우선
5	<< >>	왼쪽 우선
6	< > <= >=	왼쪽 우선
7	== !=	왼쪽 우선
8	&	왼쪽 우선
9	^	왼쪽 우선
10	|	왼쪽 우선
11	&&	왼쪽 우선
12	||	왼쪽 우선
13	? :	오른쪽 우선
14	= += -= *= %= /= ^= <<= >>=	오른쪽 우선
15	,	왼쪽 우선

7) 연산자의 우선순위와 결합수칙

- **다음 각각에 해당하는 C언어 수식을 작성하시오**

  (각 문제는 독립적이고, 문제마다 하나의 수식으로 작성하고,
  불필요한 괄호는 사용하지 마시오)

  1. x의 값에 2를 더한 결과 값에 y를 곱한 값을 z에 대입
  2. x와 y를 각각 2로 나눈 나머지의 합을 z에 대입
  3. x를 3으로 나눈 나머지를 y에 대입하고 x를 1증가 시킨다.

  4. x는 0보다 작거나, 5 < x < 9 이다.
  5. x는 양수이면서, 2로 나누어 떨어지거나 5로 나누어 떨어진다.
     (즉, x에 해당하는 수는 2, 4, 5, 6, 8, 10, ...)
  6. 5번의 명제는 거짓이다.