3 연산자

학습목표 • 연산, 비교, 논리 등의 여러 종류의 연산자의 의미를 알고 프로그래밍에 적절하게 사용할 수 있다.

생각 펼치기

현재 사용하는 +, −, ×, ÷와 같은 사칙 연산자는 13세기경부터 사용되었다. 만약 70% 세일을 한다는 광고를 보고 20,000원짜리 물건을 사려고 한다면 우리들의 머릿속에서는 다음과 같이 연산이 이루어질 것이다.

20,000원 × 0.7 = 14,000원

이와 같이 연산은 우리가 원하는 결과를 얻기 위한 다양한 조작을 할 수 있다.

핵심 질문 ▶ C 언어에서 제공하는 연산, 비교, 논리 연산자의 종류와 기능은 무엇일까?

미션 이 단원을 학습하면서 해결해 보자.

정기 예금으로 은행에 맡긴 원금이 만기가 되었을 때 원리 합계가 얼마나 되는지 계산하고자 한다. 이때 입력되어야 할 항목은 다음과 같다.

	설명
예금액(원금)	예치할 예금액 입력
이율	예금의 이율 입력 예) 5.3
기간	예금한 기간을 연 단위로 입력

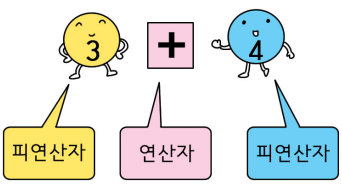
원금, 이율, 기간을 입력하였을 때, 원리 합계를 출력하는 프로그램을 작성해 보자.

원리 합계(단리법) = 원금 + 원금 × 이율(%) × 기간(년)

1 연산자에 대해 알아보자

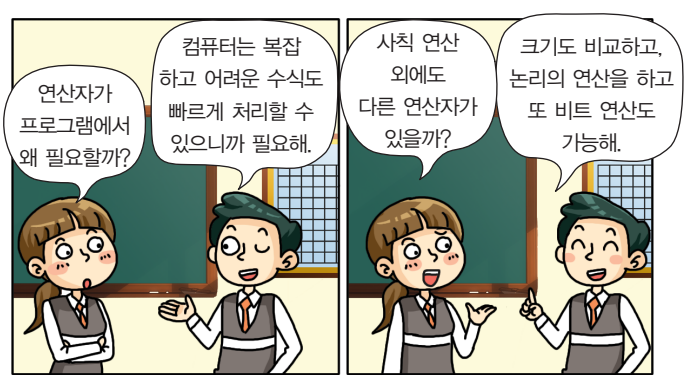
연산이란 식이 나타내는 일정한 규칙에 따라 계산하는 것을 의미한다. 컴퓨터는 수식을 이용하여 여러 가지 연산 처리를 한다. C 언어에서 수식은 연산자와 피연산자의 조합으로 구성된다.

수식
프로그래밍 작업을 할 때, 수학적 계산을 위한 문장을 수식이라고 한다. 연산자란 +, - 와 같은 기호를 말하며, 피연산자에는 상수, 변수 등이 올 수 있다.



[그림 Ⅱ - 1] 수식의 구성

C 언어에서는 우리가 흔히 사칙 연산에서 사용하는 산술 연산자뿐만 아니라 다양한 계산을 할 수 있도록 할당, 증감, 관계, 조건, 논리, 비트, 시프트 등 다양한 연산자(operator)를 제공하고 있다. 피연산자(operand)에는 연산에 참여하는 변수, 상수 등이 올 수 있다.



[표 Ⅱ - 6] 연산자의 종류

구분	연산자 설명	연산자 종류
산술	사칙 연산 및 나머지를 구하기 위한 연산자	+, -, *, /, %
할당	왼쪽 피연산자(변수)에 오른쪽 값을 할당하는 연산자	=
증감	변숫값을 증감(증가, 감소)하기 위한 연산자	++, --
관계	식의 내용을 크다(작다), 같다(같지않다)를 판단하는 연산자	<, >, <=, >=, ==, !=
조건	조건식의 참, 거짓을 판단하여 실행하는 식이 달라지는 연산자	(조건식) ? 문장 1: 문장 2;
논리	식의 논리곱(and), 논리합(or), 논리 부정(not)의 연산을 할 때 사용하는 연산자	&&, ||, !
비트	비트 단위 논리 연산을 수행하는 연산자	&, |, ~, ^
시프트	데이터를 비트 단위로 이동시켜 값을 증감시키는 연산자	≪, ≫

2 연산자 우선 순위에 대해 알아보자

하나의 수식이 여러 개의 연산자를 포함한 복잡한 식일 때에는 미리 정해진 연산자 우선 순위를 지켜야 정확한 값을 구할 수 있다. 일반적인 연산자의 우선 순위 원칙은 다음과 같다.

- 괄호 안의 내용이 우선 처리된다.
- 각 연산자의 상대적인 우선 순위에 의해 처리한다.
- 조건 연산자는 산술 연산자보다 나중에 계산한다.
- 우선 순위가 동일한 경우 결합성에 의해 처리한다.

다음은 연산자의 우선 순위 및 결합성을 나타내는 표이다.

[표 Ⅱ-7] 연산자의 우선 순위 및 결합성

우선 순위	연산자 종류			결합성
높음 ↑	( ) [ ] –>		괄호, 배열	좌 → 우
	!, ~, ++, --, *(포인터), &(주소), sizeof( ), (자료형)		단항 연산자	
	이항 연산자	*, /, %	산술 연산자	좌 → 우
		+, –		
		〈〈, 〉〉	시프트 연산자	
		〈, 〉, 〈=, 〉=	관계 연산자	
		==, !=		
		&	비트 연산자	
		^		
↓ 낮음		|		
		&&	논리 연산자	
		||		
	( )? :		조건 연산자	좌 ← 우
	=, +=, –=, *=, %=, &=, ^=, |=		할당 연산자	

예제 **연산자의 우선 순위에 따라 수식의 값을 계산하여 출력하는 프로그램을 작성해 보자.**

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int result = 4 + 3 * 5;
05      printf("%d", result);
06      return 0;
07  }
```

실행 결과

```
19
```

3 연산자 사용에 대해 알아보자

연산자를 사용할 때에는 각 연산자의 기능과 사용법을 정확하게 이해하고 있어야 수식의 처리 과정에서 발생할 수 있는 오류를 줄일 수 있다.

1 산술 연산자

산술 연산자는 덧셈, 뺄셈, 곱셈, 나눗셈 등의 연산을 하기 위한 연산자이다. 사칙연산 중에서 곱하기(×)는 *로, 나누기(÷)는 /로 표시한다. 나머지 연산자(%)는 나눗셈 연산 후 나머지 값을 구할 때 사용되는 연산자이다.

예제 산술 연산자를 사용하여 각 연산의 결괏값을 출력하는 프로그램을 작성해 보자.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i = 13, j = 5;
05      printf("%d %d %d %d %d", i+j, i-j, i*j, i/j, i%j);
06      return 0;
07  }
```

실행 결과

```
18 8 65 2 3
```

2 할당 연산자

C 언어에서 **할당 연산자**는 '=' 기호를 중심으로 오른쪽의 값이나 수식을 계산하여 왼쪽의 변수에 저장하라는 의미이다.

형변환
형변환이란 자료형을 다른 형태로 변경하는 것을 말한다.

예제 할당 연산자를 사용하여 값을 새로 할당하는 프로그램을 작성해 보자.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i = 13, j = 5;
05      i = j;          // 변수 j값을 변수 i에 할당
06      printf("%d\n", i);
07      return 0;
08  }
```

실행 결과

```
5
```

3 증감 연산자

증감 연산자는 변숫값을 1씩 증가 또는 감소시키는 연산자로 위치에 따라 변수 앞 또는 뒤에 올 수 있다.

증감 연산자를 사용하는 이유
- 프로그램을 간결하게 만들 수 있다.
- 가독성 좋은 보기 편한 코드를 작성하는 데 도움을 준다.

예제 **증감 연산자를 피연산자의 전위(앞), 후위(뒤)에 배치시켜 어떻게 연산되는지 알아보는 프로그램을 작성해 보자.**

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i = 13, j = 5, temp,
05          a = 10, b = 20, c = 30, d = 40;
06      temp = ++i; printf("%d %d\n", temp, i);
07      temp = i++; printf("%d %d\n", temp, i);
08      temp = --j; printf("%d %d\n", temp, j);
09      temp = j--; printf("%d %d\n", temp, j);
10      ++a; b++; --c; d--;
11      printf("%d %d %d %d", a, b, c, d);
12      return 0;
13  }
```

실행 결과

```
14 14
14 15
4 4
4 3
11 21 29 39
```

전위, 후위 증감 연산자

전위, 후위 증감 연산자의 연산자 우선 순위는 매우 높다. 우선 순위가 낮은 할당 연산자와 같이 사용되어 전위에서는 값을 증감하고 할당하며, 후위에서는 할당하고 증감을 실행한다.

프로그램 설명

06 변수 i값을 1 증가시키고 temp 변수에 저장

07 i 변숫값을 temp 변수에 저장하고, i값을 1 증가

temp	i		temp	i
14	14	⟶	14	15

08 j 변숫값을 1 감소시키고 temp 변수에 저장

09 j 변숫값을 temp 변수에 저장하고, j값을 1 감소

temp	j		temp	j
4	4	⟶	4	3

10 할당 연산자가 아닌 단독으로 증감 연산자가 사용될 때는 1 증가 또는 1 감소

4 관계 연산자

관계 연산자는 두 개 이상의 값을 서로 비교하는 연산자로 비교 결과는 1(참) 또는 0(거짓)으로 나타내며, 그 종류는 다음과 같다.

[표 Ⅱ-8] 관계 연산자의 종류

연산자	설명	연산자	설명
A > B	A는 B보다 크다.	A > = B	A는 B보다 크거나 같다.
A < B	A는 B보다 작다.	A < = B	A는 B보다 작거나 같다.
A == B	A와 B는 서로 같다.	A ! = B	A와 B는 같지 않다.

예제 관계 연산자를 사용하여 다음 프로그램을 작성해 보자.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i = 13, j = 5;
05      printf("%d %d %d %d %d %d",
            i<j, i>j, i<=j, i>=j, i!=j, i==j);
06      return 0;
07  }
```

실행 결과

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

5 조건 연산자

조건 연산자는 조건식에 따라 참과 거짓 중 하나를 수행한다. 조건식의 연산 결과가 참이면 문장 1을 실행하고, 거짓이면 문장 2를 실행한다.

형식
(조건식) ? 문장 1: 문장 2;

예제 조건 연산자를 이용하여 다음 프로그램을 작성해 보자.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i=13, j=5, result;
05      result = i < j ? 10 : 100;
06      printf("%d", result);
07      return 0;
08  }
```

i<j의 결과가 거짓이므로 100을 result에 할당함.

실행 결과

```
100
```

6 논리 연산자

논리 연산자는 두 개 이상의 값을 비교하여 참이면 1, 거짓이면 0을 나타낸다.

[표 Ⅱ-9] 논리 연산자의 종류

연산자	연산자 설명	비고
&&	두 개의 연산 값이 모두 참일 때만 결과가 참(1)이다.	논리곱(and)
||	두 개의 연산 값 중 하나 이상이 참일 때 결과가 참(1)이다.	논리합(or)
!	연산 값이 참이면 거짓(0), 거짓이면 참(1)이 된다.	논리 부정(not)

예제 논리 연산자를 사용하여 나타나는 결괏값을 알아보는 프로그램을 작성해 보자.

실행 결과

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

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i = 13, j = 5, temp;
05      printf("%d %d\n", i == 13 && j == 5, i == 13 && j == 4);
06      printf("%d %d\n", i == 13 || j == 4, i == 12 || j == 4);
07      temp = i; printf("%d ", !temp);
08      temp = 0; printf("%d", !temp);
09      return 0;
10  }
```

더 알아보기 논리 연산자의 단락 효과

논리 연산자의 단락 효과란 전 조건이 거짓이면 후 연산자를 계산하지 않는 것을 말한다. 다음 프로그램을 통해 논리 연산자의 단락 효과를 확인해 보자.

프로그램 논리 연산자의 단락 효과

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i = 13, j = 5;
05      if (i < 10 && ++j > 3) {}
06      printf("%d %d\n", i, j);
07      return 0;
08  }
```

i<10이 거짓이기 때문에 오른쪽의 비교 연산은 실행되지 않음. 따라서 j는 1 증가하지 않음.

실행 결과

```
13 5
```

7 비트 연산자

비트 연산자는 비트 단위 논리 연산을 수행한다.

[표 Ⅱ-10] 비트 연산자의 종류 및 실행 결과($a=1101_{(2)}$, $b=0101_{(2)}$)

연산자	연산자 설명	실행 결과
&	두 수에 대한 비트 단위 and 연산을 수행	$a\&b=0101_{(2)}$
¦	두 수에 대한 비트 단위 or 연산을 수행	$a¦b=1101_{(2)}$
^	두 수에 대한 비트 단위 xor 연산을 수행	$a\hat{}b=1000_{(2)}$
~	해당 값에 대한 1의 보수를 구함.	$\sim a=0010_{(2)}$

xor(배타적 논리합, Exclusive or)
2개의 값 중 하나만 1일 경우를 판단하는 논리 연산이다. 두 비교의 수가 0, 0과 1, 1의 경우에는 0을, 0, 1과 1, 0 일 경우에는 1을 리턴한다.

예제 비트 연산자를 사용하여 나타나는 결괏값을 알아보는 프로그램을 작성해 보자.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i = 13;
05      int j = 5;
06      printf("%d %d %d %d", i&j, i¦j, i^j, ~i);
07      return 0;
08  }
```

00000000 00000000 00000000 00001101

00000000 00000000 00000000 00000101

실행 결과

```
5 13 8 -14
```

8 시프트 연산자

시프트 연산자는 지정된 비트 수만큼 왼쪽 또는 오른쪽으로 이동하는 연산자이다. 왼쪽으로 시프트(<<) 하는 것은 2를 곱하는 결과와 같고, 오른쪽으로 시프트(>>) 하는 것은 2로 나눈 결과와 같다.

예제 시프트 연산자를 사용하여 나타나는 결괏값을 알아보는 프로그램을 작성해 보자.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int i = 13;
05      int j = 5;
06      printf("%d %d %d %d", i<<2, i>>2, j<<3, j>>3);
07      return 0;
08  }
```

00000000 00000000 00000000 00001101

00000000 00000000 00000000 00000101

실행 결과

```
52 3 40 0
```

미션 해결하기

원금, 이율, 기간을 입력하였을 때, 원리 합계를 출력하는 프로그램을 작성해 보자.

프로그램

```
01  #include<stdio.h>
02  int main( )
03  {
04      int money, duration;
05      double rate;
06      double result;
07      printf("정기 예금 금액을 입력해 주세요: ");
08      scanf("%d", &money);
09      printf("정기 예금 이율을 입력해 주세요: ");
10      scanf("%lf", &rate);
11      printf("정기 예금 기간을 입력해 주세요: ");
12      scanf("%d", &duration);
13      result = money + money * 0.01 * rate * duration;
14      printf("%d원을 %.1f%%이율로 %d년간 예치 후 원리 합계는 %.lf입니다.",
                money, rate, duration, result);
15      return 0;
16  }
```

실행 결과

```
정기 예금 금액을 입력해 주세요: 1000000 Enter
정기 예금 이율을 입력해 주세요: 5.2 Enter
정기 예금 기간을 입력해 주세요: 3 Enter
1000000원을 5.2%이율로 3년간 예치 후 원리 합계는 1156000.0입니다.
```

스스로 해결하기

실행 결과와 같이 출력될 수 있도록 프로그램을 완성해 보자.

알고리즘 설계

① scanf( ) 함수를 통해서 정숫값 2개를 입력받는다.
② 두 개의 정숫값에 대한 산술 연산의 결과를 실행 결과와 같이 출력한다.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int a, b;
05      printf("두 수를 입력하세요: ");
06      scanf("%d %d", &a, &b);
07
08
09
10
11
12      return 0;
13  }
```

실행 결과

```
두 수를 입력하세요: 16 5 Enter
16 + 5 = 21
16 - 5 = 11
16 * 5 = 80
16 / 5 = 3.000000
16 % 5 = 1
```

컴퓨팅 사고력을 키우는

수행 평가 연산자를 활용한 단위 변환

활동 목표 연산자를 활용하여 단위를 변환할 수 있다.

야구에서 사용하는 투구 속도(마일)를 km 단위로 환산하는 프로그램이다.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04      int mileSpeed;
05      double kmSpeed;
06      printf("볼의 속도 입력(마일): ");
07      scanf("%d", &mileSpeed);
08      
09      printf("%d마일은 %.3fkm입니다.", mileSpeed, kmSpeed);
10      return 0;
11  }
```

초급 1마일은 1.609 km일 때, 빈칸에 들어갈 코드를 채워 보자.

중급 위의 프로그램 구조를 응용하여 집의 면적을 변환하는 프로그램을 작성해 보자(단, 1제곱미터는 0.3025평이며, 제곱미터 값을 입력한다.).

스스로 평가하기

평가 항목	구분		
	그렇다	보통이다	그렇지 않다
• 주어진 문제에 대한 연산식을 표현할 수 있다.			
• 연산식을 프로그램에 적용할 수 있다.			

내 실력 확인하기

내용을 이해했나요?

- **연산**: 식이 나타내는 일정한 규칙에 따라 계산하는 것을 의미하며, C 언어에서 수식은 연산자와 피연산자의 조합으로 구성된다.
- **연산자**: 연산을 사용하기 위한 기호로, 산술, 할당, 증감, 관계, 조건, 논리, 비트, 시프트 연산자가 있다.

문제로 확인할까요?

1. 다음은 여러 명이 사용한 총 금액에서 각자 내야할 금액을 계산하는 프로그램이다. 입력값이 다음과 같을 때 결괏값을 예측하여 적어 보자.

```
01  #include <stdio.h>
02  int main( )
03  {
04      int price, person;
05      printf("총 금액: ");
06      scanf("%d", &price);
07      printf("총 인원: ");
08      scanf("%d", &person);
09      printf("%d명이 각각 %d원씩 지불", person, price/person);
10      return 0;
11  }
```

입력값

- 총 금액: 10,000원
- 총 인원: 10명

2. 만약 10,000원을 6명이 나누어야 할 때에는 한 사람당 1666.666… 원씩 계산해야 하지만, 현실에서는 소수점의 금액을 지불하지 못한다. '5명이 각각 1,666원씩 지불, 1명 1,670원 지불' 과 같이 출력되는 프로그램을 작성해 보자.

평가해 볼까요?

★다음 평가 항목에 따라 자신의 성취 척도를 스스로 점검해 보자.

영역	평가 항목	척도 1	2	3	4	5
이해	연산의 개념을 이해하고 있는가?					
적용	연산식을 프로그램에 적용할 수 있는가?					