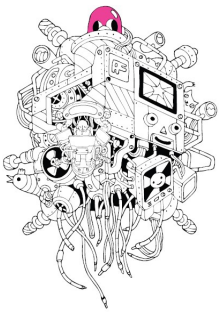


Chapter 03. 변수와 연산자

연산을 위한 연산자와 값의 저장을 위한 변수

printf 함수 사용에 대한 복습 : 이것만은 기억하자!

- printf 함수는 괄호 속 쌍 따옴표 (" ") 내부의 것을 출력한다.
  - printf("나의 이름은 홍길동입니다.");
- printf 함수의 " " 내에 %와 같은 특수기호 입력 가능
  - %는 %d, %f, %c, %ld, %lf 등으로 사용
  - %를 사용하면 " " 뒤에 % 개수만큼의 인자를 적어 주어야 함
  - printf("나의 나이는 %d입니다.", 20);
- printf 함수의 " " 내에 ₩ (또는 \ )와 같은 특수기호 입력 가능
  - ₩는 ₩n, ₩t 등으로 사용
  - printf("나의 이름은 홍길동입니다.₩n");
  - printf("나의 나이는 %d입니다.₩n", 20);
  - printf("나의 나이는 %d이고, 출생 년도는 %d입니다.₩n", 20, 1993);
  - printf("나의 나이는 %d이고,₩n출생 년도는 %d입니다.₩n", 20, 1993);

3

변수 (방) 생성 및 사용에 대한 복습 : 이것만은 기억하자!

- 변수는 메모리 상의 방을 의미한다.
- 방을 만들기 위해서 방의 타입과 이름이 필요하다.
  - int a; // 방의 타입은 int (정수), 이름은 a
  - int b, age, year; // 방 b, year, age 모두 타입은 int
- 방을 먼저 만들어야 그 방에 값을 저장할 수 있다.
  - int age, year; // age라는 정수 방을 만들라는 명령이다.
  - age = 20; // 앞에서 만든 age라는 방에 20을 저장하라는 명령이다.
  - year = 1993; // 앞에서 만든 year라는 방에 1993을 저장하라는 명령이다.
- printf 함수를 이용해서 방 안에 있는 값을 출력할 수 있다.
  - printf("나의 나이는 %d입니다.₩n", 20);
  - printf("나의 나이는 %d입니다.₩n", age);
  - printf("나의 나이는 %d이고, 출생 년도는 %d입니다.₩n", age, year);

4

덧셈 프로그램의 구현에 필요한 + 연산자

```
int main(void)
{
    3+4;    // 3과 4의 합을 명령함
    return 0;
}
```

SimpleAddOne.c

- 연산자: +
  - 컴파일 및 실행 시 문제가 발생하지 않으므로 인식 가능한 기호임이 확실함
  - 실제로 +는 덧셈의 의미를 갖는다. 따라서 실행으로 인해서 3과 4의 합이 진행 됨
  - +와 같은 기호를 가리켜 연산자라 함
- 연산의 결과는?
  - + 연산만 요구를 하였지 그 결과를 출력하기 위한 어떠한 코드도 삽입되지 않았음
  - 따라서 아무런 출력도 이뤄지지 않음
  - 연산의 결과를 저장해 두어야 원하는 바를 추가로 진행할 수 있음
  - 연산결과 또는 값의 저장을 위해서 C언어에서는 변수(variable)이라는 것을 제공함

5

변수를 이용한 데이터의 저장

- 변수란?
  - 값을 저장할 수 있는 메모리 공간 (방)에 붙여진 이름
  - 변수라는 것을 선언하면 메모리 공간이 할당되고 할당된 메모리 공간에 이름이 붙음
- 변수의 이름
  - 변수의 이름을 통해서 할당된 메모리 공간에 접근 가능
  - 값을 저장할 수도 있고 저장된 값을 참조 가능

```
int main(void)
{
    int num;
    num=20;
    printf("%d", num);
    . . . .
}
```

- int num;
  - int : 정수 저장 위한 메모리 공간 (방) 할당
  - num : 할당된 메모리 공간의 이름은 num
- num=20;
  - 변수 num에 접근하여 20을 저장
  - printf("%d", num); // num에 저장된 값을 참조(출력)

6

변수의 다양한 선언 및 초기화 방법

```
int main(void)
{
    int num1, num2;     // 변수 num1, num2의 선언
    int num3=30, num4=40;   // 변수 num3, num4의 선언 및 초기화

    printf("num1: %d, num2: %d \n", num1, num2);
    num1=10;    // 변수 num1의 초기화
    num2=20;    // 변수 num2의 초기화

    printf("num1: %d, num2: %d \n", num1, num2);
    printf("num3: %d, num4: %d \n", num3, num4);
    return 0;
}
```

VarDecAndInit.c

실행결과

```
num1: -858993460, num2: -858993460
num1: 10, num2: 20
num3: 30, num4: 40
```

- **int num1, num2;**
  - ◆ 변수를 선언만 할 수 있다.
  - ◆ 콤마를 이용하여 둘 이상의 변수를 동시에 선언할 수 있다.
  - ◆ 선언만 하면 값이 대입되기 전까지 쓰레기 값(의미 없는 값)이 채워진다.
- **int num3=30, num4=40;**
  - ◆ 선언과 동시에 초기화 할 수 있다.

7

변수선언 시 주의할 사항

```
int main(void)
{
    int num1;
    int num2;
    num1=0;
    num2=0;
    . . . .
}
```

컴파일 가능한 변수 선언

```
int main(void)
{
    int num1;
    num1=0;
    int num2;
    num2=0;
    . . . .
}
```

과거의 C 표준에서는 변수의 선언이 맨 앞에 올 것을 요구하였다. 그런데 지금도 그 표준을 따르는 컴파일러가 존재한다.

컴파일이 불가능할 수도 있는 변수선언

변수의 이름 규칙

- 첫째 변수의 이름은 알파벳, 숫자, 언더바(_)로 구성된다.
- 둘째 C언어는 대소문자를 구분한다. 따라서 변수 Num과 변수 num은 서로 다른 변수이다.
- 셋째 변수의 이름은 숫자로 시작할 수 없고, 키워드도 변수의 이름으로 사용할 수 없다(키워드에 대해서는 잠시 후 설명한다).
- 넷째 이름 사이에 공백이 삽입될 수 없다.

의미 있는 이름을 짓는 것이 가장 중요!

잘못된 이름들

```
int 7ThVal;       // 변수의 이름이 숫자로 시작했으므로
int phone#;       // 변수의 이름에 #과 같은 특수문자는 올 수 없다.
int your name;    // 변수의 이름에는 공백이 올 수 없다.
```

8

덧셈 프로그램의 완성

```
int main(void)
{
    int num1=3;
    int num2=4;
    int result=num1+num2;
    printf("덧셈 결과: %d \n", result);
    printf("%d+%d=%d \n", num1, num2, result);
    printf("%d와(과) %d의 합은 %d입니다.\n", num1, num2, result);
    return 0;
}
```

SimpleAddTwo.c

실행결과

```
덧셈 결과: 7
3+4=7
3와(과) 4의 합은 7입니다.
```

변수를 선언하여 덧셈의 결과를 저장했기 때문에 덧셈결과를 다양한 형태로 출력할 수 있다.

9

C언어의 다양한 연산자 소개

대입 연산자와 산술 연산자

연산자	연산자의 기능	결합방향
=	연산자 오른쪽에 있는 값을 연산자 왼쪽에 있는 변수에 대입한다. 예) num = 20;	←
+	두 피연산자의 값을 더한다. 예) num = 4 + 3;	→
-	왼쪽의 피연산자 값에서 오른쪽의 피연산자 값을 뺀다. 예) num = 4 - 3;	→
*	두 피연산자의 값을 곱한다. 예) num = 4 * 3;	→
/	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눈다. 예) num = 7 / 3;	→
%	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눴을 때 얻게 되는 나머지를 반환한다. 예) num = 7 % 3;	→

OperatorOne.c

```
int main(void)
{
    int num1=9, num2=2;
    printf("%d+%d=%d \n", num1, num2, num1+num2);
    printf("%d-%d=%d \n", num1, num2, num1-num2);
    printf("%d×%d=%d \n", num1, num2, num1*num2);
    printf("%d÷%d의 몫=%d \n", num1, num2, num1/num2);
    printf("%d÷%d의 나머지=%d \n", num1, num2, num1%num2);
    return 0;
}
```

함수호출 문장에 연산식이 있는 경우 연산이 이뤄지고 그 결과를 기반으로 함수의 호출이 진행된다.

```
9+2=11
9-2=7
9×2=18
9÷2의 몫=4
9÷2의 나머지=1
```

실행결과

11

연산자의 우선순위와 결합방향

- **연산자의 우선순위**
  - 연산의 순서에 대한 순위
  - 덧셈과 뺄셈보다는 곱셈과 나눗셈의 우선순위가 높다.
- **연산자의 결합방향**
  - 우선순위가 동일한 두 연산자 사이에서의 연산을 진행하는 방향
  - 덧셈, 뺄셈, 곱셈, 나눗셈 모두 결합방향이 왼쪽에서 오른쪽으로 진행

3+4*5/2-10

- 연산자의 우선순위에 근거하여 곱셈과 나눗셈이 먼저 진행된다.
- 결합방향에 근거하여 곱셈이 나눗셈보다 먼저 진행된다.

12

복합 대입 연산자


```
int main(void)
{
    int num1=2, num2=4, num3=6;
    num1 += 3;    // num1 = num1 + 3;
    num2 *= 4;    // num2 = num2 * 4;
    num3 %= 5;    // num3 = num3 % 5;
    printf("Result: %d, %d, %d \n", num1, num2, num3);
    return 0;
}
```

OperatorTwo.c

실행결과

```
Result: 5, 16, 1
```

13

부호의 의미를 갖는 + 연산자와 - 연산자

```
int main(void)
{
    int num1 = +2;
    int num2 = -4;

    num1 = -num1;
    printf("num1: %d \n", num1);
    num2 = -num2;
    printf("num2: %d \n", num2);
    return 0;
}
```

int num1 = 2; 와 동일한 문장!
+를 연산자의 범주에 포함시켰기 때문에 컴파일이 가능하다.

OperatorThree.c

실행결과

```
num1: -2
num2: 4
```

```
num1=-num2;    // 부호 연산자의 사용
num1-=num2;    // 복합 대입 연산자의 사용
```

두 연산자를 혼동하지 않도록 주의!

```
num1 = -num2;    // 부호 연산자의 사용
num1 -= num2;    // 복합 대입 연산자의 사용
```

혼동을 최소화 하는 띄어쓰기

14

증가, 감소 연산자

연산자	연산자의 기능	결합방향
++num	값을 1 증가 후, 속한 문장의 나머지를 진행(선 증가, 후 연산) 예) val = ++num;	←
num++	속한 문장을 먼저 진행한 후, 값을 1 증가(선 연산, 후 증가) 예) val = num++;	←
--num	값을 1 감소 후, 속한 문장의 나머지를 진행(선 감소, 후 연산) 예) val = --num;	←
num--	속한 문장을 먼저 진행한 후, 값을 1 감소(선 연산, 후 감소) 예) val = num--;	←

```
int main(void)
{
    int num1=12;
    int num2=12;
    printf("num1: %d \n", num1);
    printf("num1++: %d \n", num1++);   // 후위 증가
    printf("num1: %d \n\n", num1);
    printf("num2: %d \n", num2);
    printf("++num2: %d \n", ++num2);   // 전위 증가
    printf("num2: %d \n", num2);
    return 0;
}
```

OperatorFour.c

실행결과

```
num1: 12
num1++: 12
num1: 13

num2: 12
++num2: 13
num2: 13
```

15

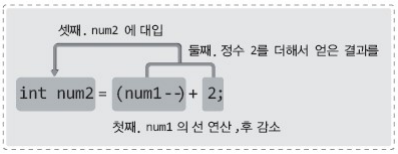
증가, 감소 연산자 추가 예제

```
int main(void)
{
    int num1=10;
    int num2=(num1--)+2;   // 후위 감소
    printf("num1: %d \n", num1);
    printf("num2: %d \n", num2);
    return 0;
}
```

OperatorFive.c

```
num1: 9
num2: 12
```

실행결과



연산의 과정

16

프로그램 연습 (Lab 1)

- 나눗셈을 수행하여 몫과 나머지를 구하는 프로그램
  1) 정수 값 20을 변수 a에 저장한다.
  2) 변수 a에 3을 더한다.
     - 단, 덧셈을 위해 복합 대입 연산자를 이용한다.
  3) a를 3으로 나눈 몫과 나머지를 변수 b와 c에 각각 저장한 후 출력한다.
  4) 위의 결과에 추가해서 몫은 ++ 연산을 사용하고, 나머지는 -- 연산을 사용하는데 아래의 두 번째 줄과 같이 출력되도록 한다.
     - 단, ++ 및 -- 연산은 printf 함수 내에서 수행한다.
  5) 최종적으로 a, b, c의 값을 아래의 세 번째 줄과 같이 출력한다.
  6) 단, printf 함수 호출문 (즉, printf ("...", ...))내에 어떠한 숫자도 보이지 않아야 한다.

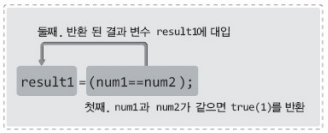
```
23을 3으로 나눈 몫은 7이고 나머지는 2입니다.
현재의 b의 값은 7이고 c의 값은 1입니다.
최종적으로 a는 23, b는 8, c는 1입니다.
```

17

관계 연산자

연산자	연산자의 기능	결합방향
<	예) n1 < n2 n1이 n2보다 작은가?	→
>	예) n1 > n2 n1이 n2보다 큰가?	→
==	예) n1 == n2 n1과 n2가 같은가?	→
!=	예) n1 != n2 n1과 n2가 다른가?	→
<=	예) n1 <= n2 n1이 n2보다 같거나 작은가?	→
>=	예) n1 >= n2 n1이 n2보다 같거나 큰가?	→

연산의 조건을 만족하면 true를 의미하는 1을 반환하고 만족하지 않으면 false를 의미하는 0을 반환하는 연산자들이다.



C언어는 0이 아닌 모든 값을 true으로 간주한다. 다만 1이 true을 의미하는 대표적인 값일 뿐이다.

OperatorSix.c

```
int main(void)
{
    int num1=10;
    int num2=12;
    int result1, result2, result3;
    result1=(num1==num2);
    result2=(num1<=num2);
    result3=(num1>num2);
    printf("result1: %d \n", result1);
    printf("result2: %d \n", result2);
    printf("result3: %d \n", result3);
    return 0;
}
```

실행결과

```
result1: 0
result2: 1
result3: 0
```

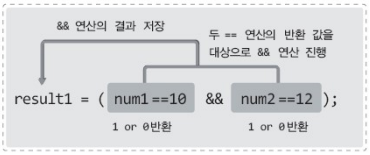
18

논리 연산자

연산자	연산자의 기능	결합방향
&&	예) A && B A와 B 모두 '참'이면 연산결과로 '참'을 반환(논리 AND)	→
||	예) A || B A와 B 둘 중 하나라도 '참'이면 연산결과로 '참'을 반환(논리 OR)	→
!	예) !A A가 '참'이면 '거짓', A가 '거짓'이면 '참'을 반환(논리 NOT)	←

```
int main(void)
{
    int num1=10;
    int num2=12;
    int result1, result2, result3;
    result1 = (num1==10 && num2==12);
    result2 = (num1<12 || num2>12);
    result3 = (!num1);
    printf("result1: %d \n", result1);
    printf("result2: %d \n", result2);
    printf("result3: %d \n", result3);
    return 0;
}
```

OperatorSeven.c



왼쪽 예제에서 num1은 0이 아니므로 참과 거짓의 관계로 본다면 거짓에 해당한다. 따라서 ! 연산의 결과로 참을 의미하는 1이 반환되는 것이다.

```
result1: 1
result2: 1
result3: 0
```

실행결과

19

콤마 연산자

- **콤마(,)**
  - **콤마도 연산자**
  - **둘 이상의 변수를 동시에 선언하거나 둘 이상의 문장을 한 행에 삽입하는 경우에 사용되는 연산자**
  - **둘 이상의 인자를 함수로 전달할 때 인자의 구분을 목적으로도 사용**
  - **콤마 연산자는 다른 연산자들과 달리 연산의 결과가 아닌 '구분'이 목적**

```
int main(void)
{
    int num1=1, num2=2;
    printf("Hello "), printf("world! \n");
    num1++, num2++;
    printf("%d ", num1), printf("%d ", num2), printf("\n");
    return 0;
}
```

CommaOp.c

실행결과

```
Hello world!
2 3
```

20

연산자의 우선순위와 결합방향

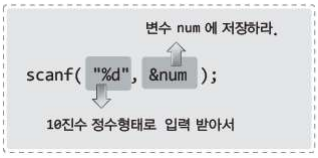
순위	연산기호	연산자	결합방향
1위	()	함수호출	→
	[]	인덱스	
	->	간접지정	
	.	직접지정	
	++ (postfix) -- (postfix)	후위증가 및 감소	
2위	++ (prefix) -- (prefix)	전위증가 및 감소	←
	sizeof	바이트 단위 크기 계산	
	~	비트 단위 NOT	
	!	논리 NOT	
	-, +	부호 연산(음수와 양수의 표현)	
	&	주소 연산	
	*	간접지정 연산	
3위	(casting)	자료형 변환	←
4위	*, /, %	곱셈, 나눗셈 관련 연산	→
5위	+, -	덧셈, 뺄셈	→
6위	<<, >>	비트이동	→
7위	<, >, <=, >=	대소비교	→
8위	==, !=	동등비교	→
9위	&	비트 AND	→
10위	^	비트 XOR	→
11위	|	비트 OR	→
12위	&&	논리 AND	→
13위	||	논리 OR	→
14위	? :	조건연산	←
15위	=, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, |=	대입연산	←
16위	,	콤마연산	→

21

키보드로부터의 데이터 입력과 C 언어의 키워드

키보드로부터의 정수 입력을 위한 scanf 함수의 호출

```
int main(void)
{
    int num;
    scanf("%d", &num);
    . . . .
}
```



- printf 함수에서의 %d는 10진수 정수의 출력을 의미한다.
- 반면 scanf 함수에서의 %d는 10진수 정수의 입력을 의미한다.
- 변수의 이름 num 앞에 & 를 붙인 이유는 이후에 천천히 알게 된다.

SimpleAddThree.c

```
int main(void)
{
    int result;
    int num1, num2;
    printf("정수 one: ");
    scanf("%d", &num1);    // 첫 번째 정수 입력
    printf("정수 two: ");
    scanf("%d", &num2);    // 두 번째 정수 입력
    result=num1+num2;
    printf("%d + %d = %d \n", num1, num2, result);
    return 0;
}
```

실행결과

```
정수 one: 3
정수 two: 4
3 + 4 = 7
```

23

프로그램 연습

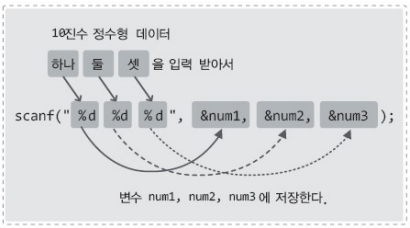
- scanf와 printf 함수를 이용한 프로그램
  1) 아래와 같은 형태로 입출력이 되도록 프로그래밍 수행
  2) 첫 line의 23은 keyboard로 입력
  3) 둘째 line의 23은 프로그램에서 printf 함수를 이용하여 출력한다. 단, printf 문장 내에 23이라는 숫자는 보이지 않도록 한다.

```
이번 학기에 수강하는 총 학점을 입력하세요 : 23
이번 학기에 수강하는 총 학점은 23 학점입니다.
```

24

입력의 형태를 다양하게 지정 가능

```
int main(void)
{
    int num1, num2, num3;
    scanf("%d %d %d", &num1, &num2, &num3);
    . . . .
}
```



한 번의 scanf 함수호출을 통해서 둘 이상의 데이터를 원하는 방식으로 입력 가능

```
int main(void)
{
    int result;
    int num1, num2, num3;
    printf("세 개의 정수 입력: ");
    scanf("%d %d %d", &num1, &num2, &num3);
    result=num1+num2+num3;
    printf("%d + %d + %d = %d \n", num1, num2, num3, result);
    return 0;
}
```

SimpleAddFour.c

실행결과

```
세 개의 정수 입력: 4 5 6
4 + 5 + 6 = 15
```

25

키보드로부터의 문자 입력을 위한 scanf 함수의 호출

- 키보드로부터 정수 값 하나를 입력하여 정수형 변수 a에 저장
  - int a;
  - scanf (“%d”, &a);
- 키보드로부터 문자 한 글자를 입력하여 문자형 변수 b에 저장
  - char b;
  - scanf (“%c”, &b);

```
#include <stdio.h>

int main (void)
{
    char abc;

    printf(“문자 한 개를 입력하세요: ”);
    scanf(“%c”, &abc);
    printf(“입력한 문자는 %c 입니다. ₩n”, abc);

    return 0;
}
```

26

C 언어의 표준 키워드

auto	_Bool	break	case
char	_Complex	const	continue
default	do	double	else
enum	extern	float	for
goto	if	_Imaginary	return
restrict	short	signed	sizeof
static	struct	switch	typedef
union	unsigned	void	volatile
while			

C 언어의 문법을 구성하는, 그 의미가 결정되어 있는 단어들!
이러한 단어들을 가리켜 키워드(keyword)라 한다.

27

연산자 모음

1. 대입 및 산술연산자

연산자
=
+
-
*
/
%

2. 복합대입연산자

3. 증감연산자

연산자
++num
num++
--num
num--

4. 관계연산자

연산자	연산자의 기능
<	예) n1 < n2 n1이 n2보다 작은가?
>	예) n1 > n2 n1이 n2보다 큰가?
==	예) n1 == n2 n1과 n2가 같은가?
!=	예) n1 != n2 n1과 n2가 다른가?
<=	예) n1 <= n2 n1이 n2보다 같거나 작은가?
>=	예) n1 >= n2 n1이 n2보다 같거나 큰가?

5. 논리연산자

연산자
&&
||
!
.....

28

프로그램 연습 (Lab 1)

- 나눗셈을 수행하여 몫과 나머지를 구하는 프로그램
  1) 정수 값 20을 변수 a에 저장한다.
  2) 변수 a에 3을 더한다.
     - 단, 덧셈을 위해 복합 대입 연산자를 이용한다.
  3) a를 3으로 나눈 몫과 나머지를 변수 b와 c에 각각 저장한 후 출력한다.
  4) 위의 결과에 추가해서 몫은 ++ 연산을 사용하고, 나머지는 -- 연산을 사용하는데 아래의 두 번째 줄과 같이 출력되도록 한다.
     - 단, ++ 및 -- 연산은 printf 함수 내에서 수행한다.
  5) 최종적으로 a, b, c의 값을 아래의 세 번째 줄과 같이 출력한다.
  6) 단, printf 함수 호출문 (즉, printf ("...", ...))내에 어떠한 숫자도 보이지 않아야 한다.

```
23을 3으로 나눈 몫은 7이고 나머지는 2입니다.
현재의 b의 값은 7이고 c의 값은 1입니다.
최종적으로 a는 23, b는 8, c는 1입니다.
```

29

프로그램 연습 (Lab 2)

- scanf와 printf 함수를 이용한 프로그램
  1) 키보드로부터 문자 1개를 입력 받아 변수 d에 저장
  2) d에 저장된 문자를 출력
  3) 정수형 변수 a, b, c와 문자형 변수 d를 선언
  4) 키보드로부터 정수 값 3개를 입력 받아 각각 a, b, c에 저장
  5) 이들 a, b, c의 합을 구해서 출력
  6) 단, printf 문장 내에는 숫자 및 문자는 보이지 않도록 한다.

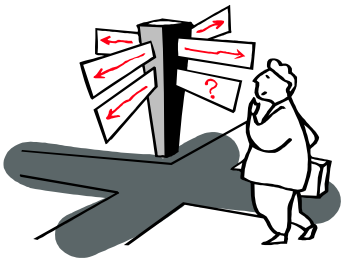
```
문자 한 1개를 입력하세요 : A
입력한 문자는 A입니다.
세 개의 정수를 입력하세요 : 10 20 30
세 개의 정수 10, 20, 30의 총 합은 60입니다.
```

30

실습시간 (2019년 3월 20일)

- 교재 예제: (11개)
  - VarDeclAndInit.c,
  - SimpleAddTwo.c, SimpleAddThree.c, SimpleAddFour.c
  - OperatorOne.c, OperatorTwo.c, OperatorThree.c, OperatorFour.c, OperatorFive.c, OperatorSix.c, OperatorSeven.c,
- 실습문제: (2개)
  - Lab1, Lab2

31



Chapter 03이 끝났습니다. 질문 있으신지요?