

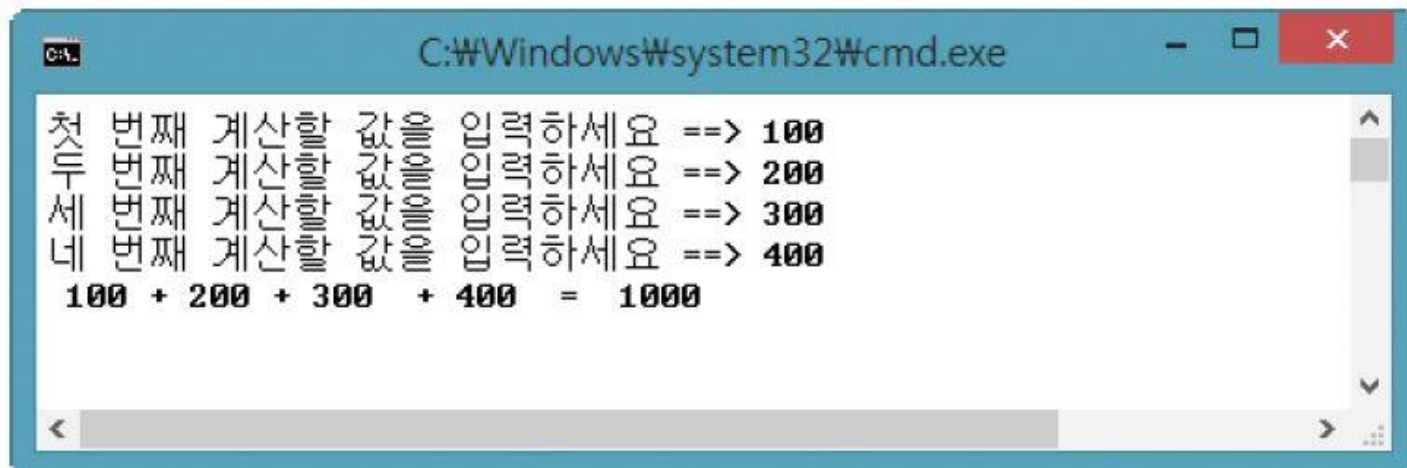
C언어

Chapter 02

[예제모음 01] 숫자 4개를 더하는 프로그램

예제 설명 숫자 4개를 입력받아 그 합을 구하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

첫 번째 계산할 값을 입력하세요 ==> 100
두 번째 계산할 값을 입력하세요 ==> 200
세 번째 계산할 값을 입력하세요 ==> 300
네 번째 계산할 값을 입력하세요 ==> 400
100 + 200 + 300 + 400 = 1000
```

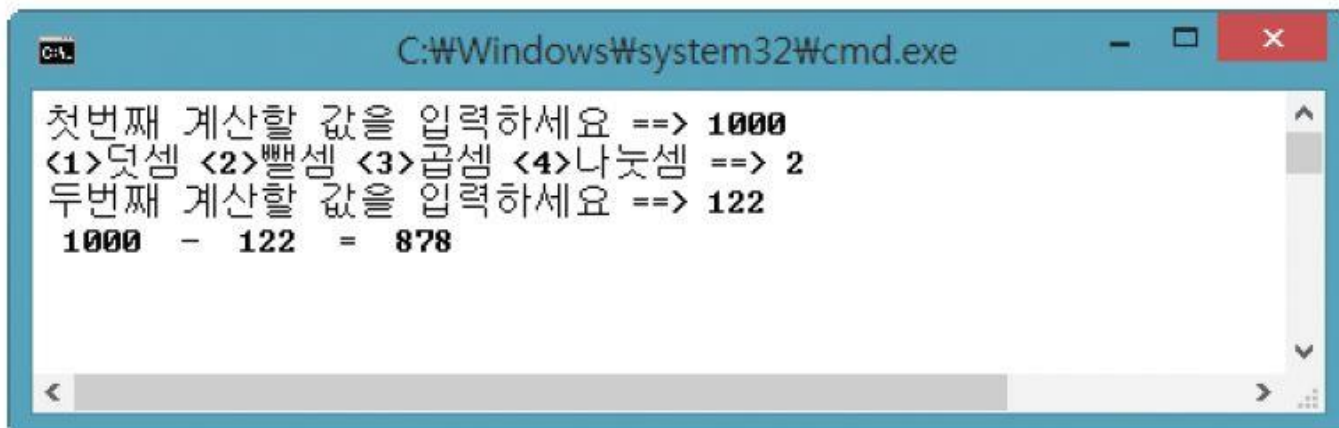
[예제모음 01] 숫자 4개를 더하는 프로그램

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b, c, d;          ----입력받을 변수 4개를 선언한다.
06     int result;
07
08     printf("첫 번째 계산할 값을 입력하세요 == > ");
09     scanf_s("%d", &a);      ----변수 a에 들어갈 값을 키보드로 직접 입력한다.
10     printf("두 번째 계산할 값을 입력하세요 == > ");
11     scanf_s("%d", &b);      ----변수 b에 들어갈 값을 키보드로 직접 입력한다.
12     printf("세 번째 계산할 값을 입력하세요 == > ");
13     scanf_s("%d", &c);      ----변수 c에 들어갈 값을 키보드로 직접 입력한다.
14     printf("네 번째 계산할 값을 입력하세요 == > ");
15     scanf_s("%d", &d);      ----변수 d에 들어갈 값을 키보드로 직접 입력한다.
16
17     result = a + b + c + d;  ----변수 a, b, c, d의 값을 모두 더해 변수 result에 입력한다.
18
19     printf(" %d + %d + %d + %d = %d \n", a, b, c, d, result); ----변수 a, b, c, d와 result 값을 모니터에 출력한다.
20 }
```

[예제모음 02] if문을 활용한 계산기

예제 설명 if문으로 덧셈, 뺄셈, 곱셈, 나눗셈 중 하나를 선택하여 계산하는 프로그램이다. 5장에서 배울 if문이 미리 나와서 좀 어렵게 느껴지겠지만 직접 코딩하고 실행해보자.

실행 결과



```
C:\Windows\system32\cmd.exe

첫번째 계산할 값을 입력하세요 ==> 1000
<1>덧셈 <2>뺄셈 <3>곱셈 <4>나눗셈 ==> 2
두번째 계산할 값을 입력하세요 ==> 122
1000 - 122 = 878
```

[예제모음 02] if문을 활용한 계산기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b;
06     int result;
07     int k;
08
09     printf("첫번째 계산할 값을 입력하세요 == > ");
10     scanf("%d", &a);
11     printf("<1>덧셈 <2>뺄셈 <3>곱셈 <4>나눗셈 == > ");
12     scanf("%d", &k);
13     printf("두번째 계산할 값을 입력하세요 == > ");
14     scanf("%d", &b);
15
16     if (k == 1) {
17         result = a + b;
18         printf(" %d + %d = %d \n", a, b, result);
19     }
20
```

---계산 방식을 선택할 변수를 선언한다

---계산할 수를 입력한다

---연산자를 선택한다(1:덧셈, 2:뺄셈, 3:곱셈, 4:나눗셈).

---계산할 수를 입력한다.

---입력한 k가 1이면 덧셈을 수행한다.

[예제모음 02] if문을 활용한 계산기

```
21  if (k == 2) {  
22      result = a - b;  
23      printf(" %d - %d = %d \n", a, b, result);  
24  }
```

----입력한 k가 2이면 뺄셈을 수행한다.

25

```
26  if (k == 3) {  
27      result = a * b;  
28      printf(" %d * %d = %d \n", a, b, result);  
29  }
```

----입력한 k가 3이면 곱셈을 수행한다.

30

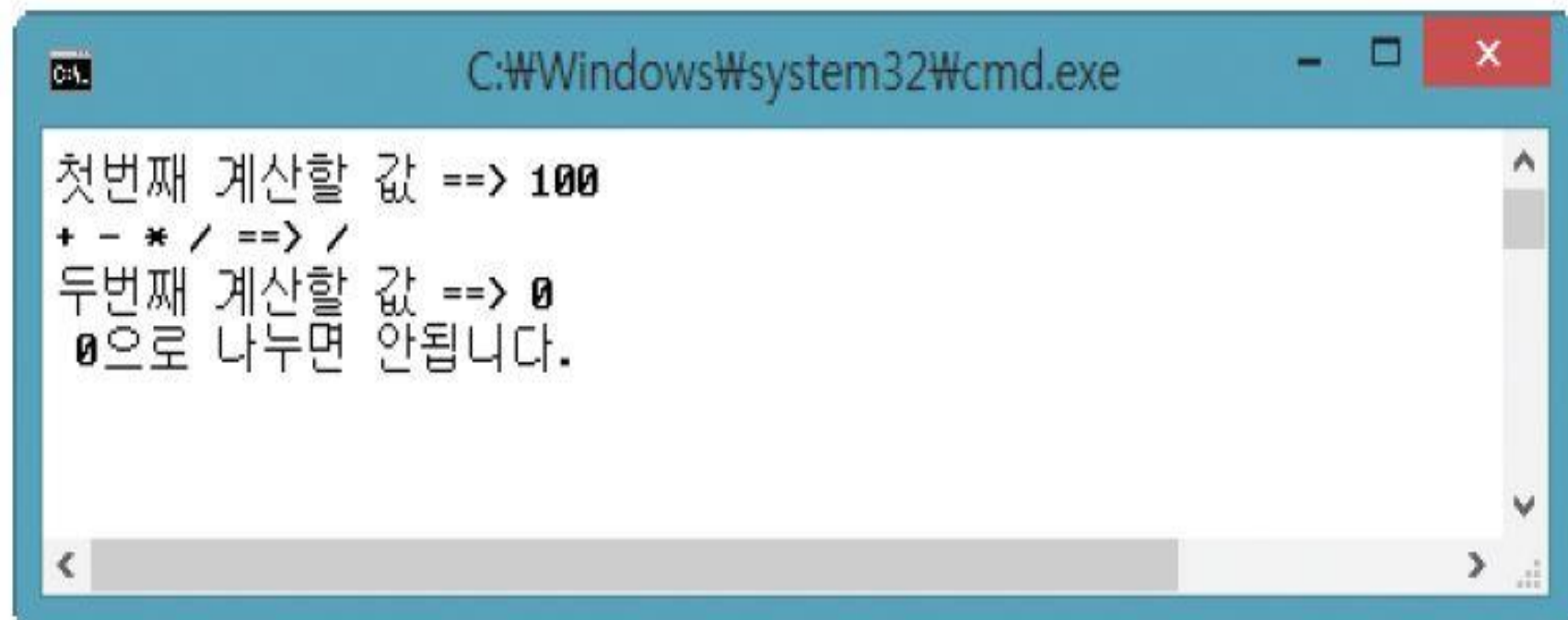
```
31  if (k == 4) {  
32      result = a / b;  
33      printf(" %d / %d = %d \n", a, b, result);  
34  }
```

----입력한 k가 4이면 나눗셈을 수행한다.

```
35 }
```

설명 [예제모음 02]는 0으로 나누면 오류가 발생하는데, 오류 없는 계산기 프로그램을 작성하는 기호(+, -, *, /)를 사용해서 직접 입력하고 나머지 값 연산자인 %를 추가한다.

결과



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window contains the following text:

```
첫번째 계산할 값 ==> 100  
+ - * / ==> /  
두번째 계산할 값 ==> 0  
0으로 나누면 안됩니다.
```

[예제모음 03] 오류 없는 계산기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b;
06     int result;
07     char k;
08
09     printf("첫번째 계산할 값 == > ");
10     scanf("%d", &a);
11     printf("+ - * / % == > ");
12     scanf(" %c", &ch);
13     printf("두번째 계산할 값 == > ");
14     scanf("%d", &b);
15
16     if (k == '+') {
17         result = a + b;
18         printf(" %d + %d = %d \n", a, b, result);
19     }
20
```

----연산자를 입력받을 변수를 문자형으로 선언한다

----%c의 앞에 공백이 있어야 한다.

[예제모음 03] 오류 없는 계산기

```
21  if (k == '-') {
22      result = a - b;
23      printf(" %d - %d = %d \n", a, b, result);
24  }
25
26  if (k == '*') {
27      result = a * b;
28      printf(" %d * %d = %d \n", a, b, result);
29  }
30
31  if (k == '/') {
32      if (b != 0) {
33          result = a / b;
34          printf(" %d / %d = %d \n", a, b, result);
35      } else
36          printf(" 0으로 나누면 안됩니다. \n");
37  }
38
39  if (k == '%') {
```

---0으로 나누거나 나머지 값을 구하면 처리하지 않고
오류 메시지를 보여준다.

[예제모음 03] 오류 없는 계산기

```
40     if (b != 0) {  
41         result = a % b;  
42         printf(" %d %% %d = %d \n", a, b, result);  
43     } else  
44         printf(" 0으로 나누면 나머지 값이 안됩니다. \n");  
45 }  
46 }
```

---0으로 나누거나 나머지 값을 구하면 처리하지 않고
오류 메시지를 보여준다.

[2장 요약]

1 C 프로그램 작성 순서

프로젝트 만들기

프로그램 코딩

빌드(컴파일/링크)

실행

2 변수의 개념

변수는 값을 저장하는 그릇과 비슷한 개념이다.

변수에 한 번 들어간 값은 다른 값이 들어오기 전까지 그대로 유지된다.

3 scanf() 함수

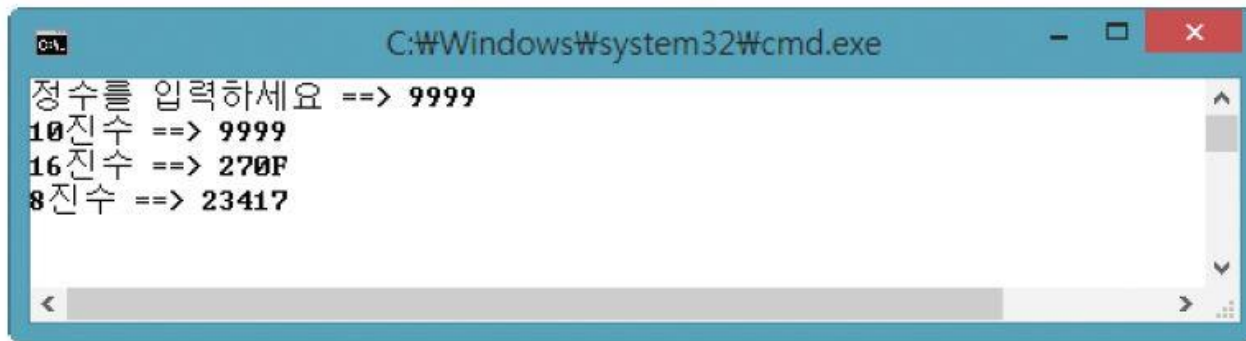
키보드로 값을 입력할 때 사용하는 함수이다.

변수에 값을 입력받으려면 반드시 변수 앞에 & 기호를 붙여야 한다.

[예제모음 04] 정수형을 출력하는 프로그램

예제 설명 정수를 하나 입력받아 10진수, 16진수, 8진수로 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
정수를 입력하세요 ==> 9999
10진수 ==> 9999
16진수 ==> 270F
8진수 ==> 23417
```

[예제모음 04] 정수형을 출력하는 프로그램

```
01 #include <stdio.h>
```

```
02
```

```
03 int main()
```

```
04 {
```

```
05     int data;      ----정수형 변수를 선언한다.
```

```
06
```

```
07     printf("정수를 입력하세요 == > ");
```

```
08     scanf("%d", &data);  ----키보드로 정수를 입력받는다.
```

```
09
```

```
10     printf("10진수 == > %d \n", data);  ----10진수(%d), 16진수(%X), 8진수(%o)를 출력한다.
```

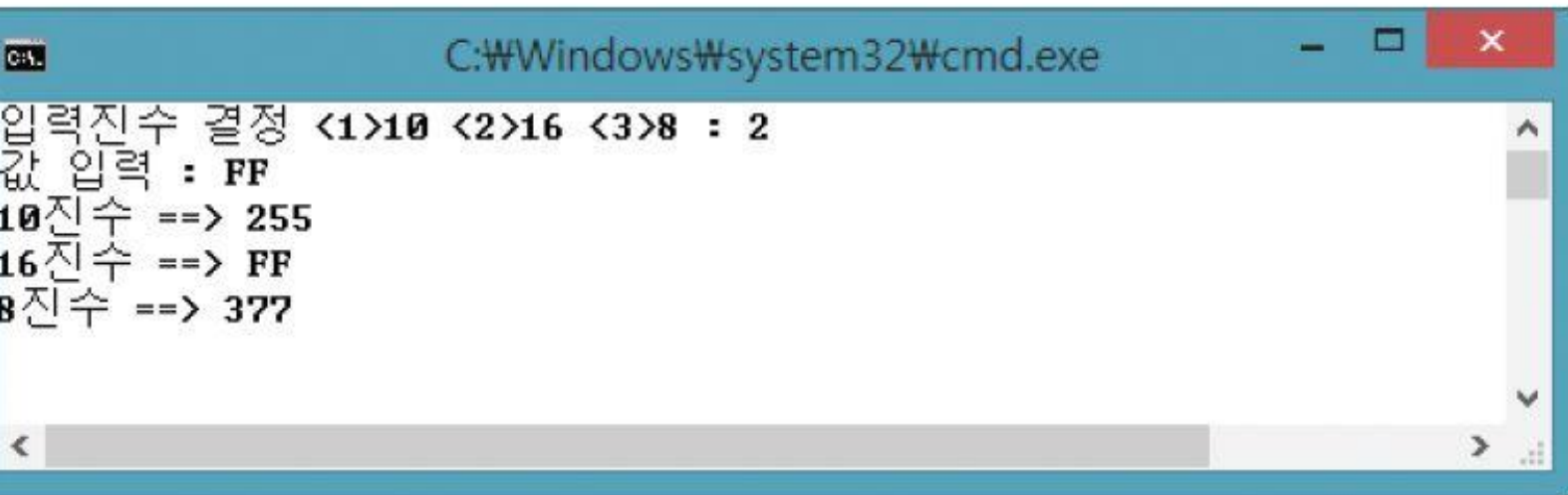
```
11     printf("16진수 == > %X \n", data);
```

```
12     printf("8진수 == > %o \n", data);
```

```
13 }
```

[예제모음 05] 입력하는 정수의 진수 결정

10진수, 16진수, 8진수 중 어떤 진수의 값을 입력받을지 결정하고, 입력받은 수를 10진수, 16진수, 8진수로 출력하는 프로그램이다.



```
C:\Windows\system32\cmd.exe

입력진수 결정 <1>10 <2>16 <3>8 : 2
값 입력 : FF
10진수 ==> 255
16진수 ==> FF
8진수 ==> 377
```

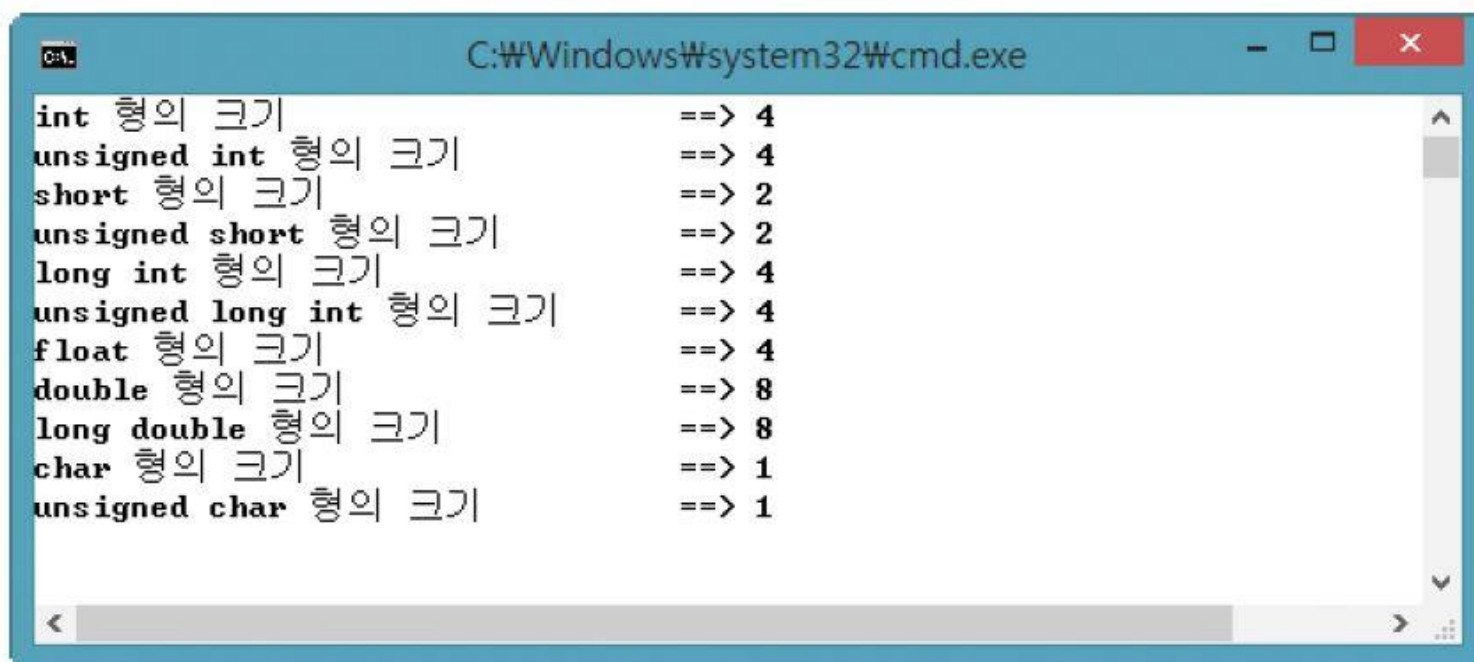
[예제모음 05] 입력하는 정수의 진수 결정

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int type, data;
06
07     printf("입력진수 결정 <1>10 <2>16 <3>8 : ");
08     scanf("%d", &type); ----키보드로 1~3 중 하나를 입력받는다.
09
10     printf("값 입력 : ");
11
12     if(type == 1) ----입력값이 1이면 10진수를 입력받는다.
13     { scanf("%d", &data); }
14
15     if(type == 2) ----입력값이 2이면 16진수를 입력받는다.
16     { scanf("%x", &data); }
17
18     if(type == 3) ----입력값이 3이면 8진수를 입력받는다.
19     { scanf("%o", &data); }
20
21     printf("10진수 == > %d \n", data); ----입력받은 data 값을 10진수, 16진수, 8진수로 변환하여 출력한다.
22     printf("16진수 == > %X \n", data);
23     printf("8진수 == > %o \n", data);
24 }
```

[예제모음 06] 데이터형의 크기 확인

예제 설명 sizeof() 함수를 사용해서 각 데이터형의 크기를 확인하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

int 형의 크기                ==> 4
unsigned int 형의 크기        ==> 4
short 형의 크기               ==> 2
unsigned short 형의 크기      ==> 2
long int 형의 크기            ==> 4
unsigned long int 형의 크기    ==> 4
float 형의 크기               ==> 4
double 형의 크기              ==> 8
long double 형의 크기         ==> 8
char 형의 크기                ==> 1
unsigned char 형의 크기       ==> 1
```


[예제모음 06] 데이터형의 크기 확인

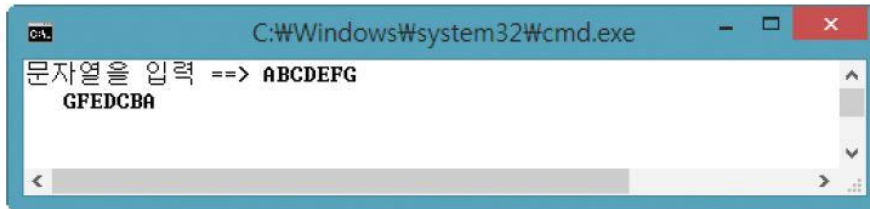
```
01 #include <stdio.h>
02
03 int main()
04 {
05     printf("int 형의 크기\t\t\t == > %d\n", sizeof(int));
06     printf("unsigned int 형의 크기\t\t == > %d\n", sizeof(unsigned int));
07     printf("short 형의 크기\t\t\t == > %d\n", sizeof(short));
08     printf("unsigned short 형의 크기\t == > %d\n", sizeof(unsigned short));
09     printf("long int 형의 크기\t\t\t == > %d\n", sizeof(long int));
10     printf("unsigned long int 형의 크기\t == > %d\n", sizeof(unsigned long int));
11     printf("float 형의 크기\t\t\t\t == > %d\n", sizeof(float));
12     printf("double 형의 크기\t\t\t == > %d\n", sizeof(double));
13     printf("long double 형의 크기\t\t == > %d\n", sizeof(long double));
14     printf("char 형의 크기\t\t\t\t == > %d\n", sizeof(char));
15     printf("unsigned char 형의 크기\t\t == > %d\n", sizeof(unsigned char));
16 }
```

--sizeof() 함수로 각 데이터형의 크기(바이트 수)를 출력한다. 이때 컴파일러에 따라서 long double 형은 16바이트 크기일 수도 있다.

[예제모음 07] 입력된 문자열을 반대 순서로 출력

예제 설명 열 글자 미만의 문자열을 입력받고, 입력받은 문자열을 반대 순서로 출력하는 프로그램이다(아직 배우지 않은 내용이 나오지만 나중에 위해 미리 살펴보자).

실행 결과



```
C:\Windows\system32\cmd.exe
문자열을 입력 ==> ABCDEFG
GFEDCBA
```

The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt is "문자열을 입력 ==>". The user has entered "ABCDEFG", and the program has output "GFEDCBA". The window has a blue title bar and standard Windows window controls (minimize, maximize, close).

[예제모음 07] 입력된 문자열을 반대 순서로 출력

```
01 #include <stdio.h>
02
03 int main()
04 {
05     char str[10]= " "; ----문자열을 입력받을 str 배열을 준비한다.
06     int i; ----첨자를 준비한다.
07
08     printf("문자열을 입력 == > ");
09     scanf("%s", str); ----문자열을 입력받는다.
10
11     for(i = sizeof(str) - 1; i >= 0; i- -) ----str 배열에 들어 있는 문자열을 맨 뒤의 str[9]부터 str[0]까지
12     { 출력한다. 즉 입력한 순서의 반대로 출력되는 것이다.
13         printf ("%c", str[i]);
14     }
15     printf("\n");
16 }
```

[3장. 요약]

1 printf() 함수

- ❶ 모니터에 무언가를 출력하는 역할.
- ❷ 형식: printf("서식", 인자 ...)
- ❸ 정수는 %d, 실수는 %f, 문자는 %c, 문자열은 %s 서식 사용.

2 printf() 함수의 서식

- ❶ %5d는 다섯 자리로 정수 출력.
- ❷ %7.3f는 전체 일곱 자리에 소수점 아래 세 자리의 실수 출력.
- ❸ \n은 새로운 줄로 이동, \t는 다음 탭으로 이동, \\는 \를 출력하는 등의 다양한 서식 문자 존재.

3 변수에 값을 대입하는 방법

- '10=100'처럼 왼쪽에 상수가 오면 안 되고 'a=100'과 같이 왼쪽에 변수가 와야 한다.
- 오른쪽에는 상수, 변수, 계산값 등 무엇이든지 올 수 있다.

[3장. 요약]

4 데이터 형식

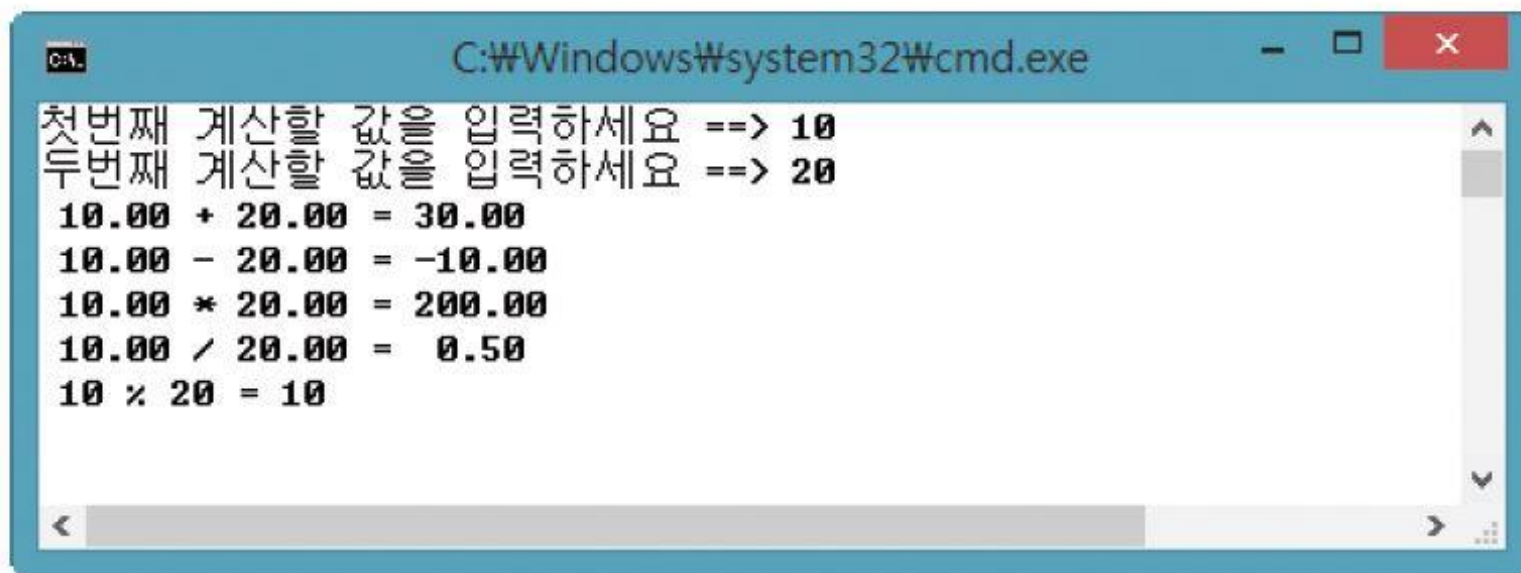
- ❶ 컴퓨터에서 내부적으로 사용되는 것은 2진수, 2진수 네 자리를 묶은 것이 16진수.
- ❷ 정수 데이터 형식에는 short, int, long 등이 있으며 부호 없이 사용하려면 unsigned를 붙인다.
- ❸ 실수 데이터 형식에는 float, double 등이 있다.
- ❹ 한 글자를 저장하는 문자 데이터 형식에는 char, 여러 글자를 저장하기 위한 문자열 데이터 형식은 문자 데이터 형식의 배열 사용. 단, 문자열의 끝을 표시하는 널 문자를 저장하기 위해 '실제 문자열 길이+1'로 크기 선언.

[예제모음 08] 입력된 두 실수의 산술 연산

설명 실수를 입력받아 두 수의 다양한 연산을 출력하는 프로그램이다.

힌트_ 나머지를 구할 때는 강제 형 변환을 사용한다.

결과



```
C:\Windows\system32\cmd.exe

첫번째 계산할 값을 입력하세요 ==> 10
두번째 계산할 값을 입력하세요 ==> 20
10.00 + 20.00 = 30.00
10.00 - 20.00 = -10.00
10.00 * 20.00 = 200.00
10.00 / 20.00 = 0.50
10 % 20 = 10
```

[예제모음 08] 입력된 두 실수의 산술 연산

```
01 #include <stdio.h>
02
03 int main()
04 {
05     float a, b;
06     float result;
07
08     printf("첫번째 계산할 값을 입력하세요 == > ");
09     scanf("%f", &a);
10     printf("두번째 계산할 값을 입력하세요 == > ");
11     scanf("%f", &b);
12
13     result = a + b;
14     printf(" %5.2f + %5.2f = %5.2f \n", a, b, result);
15     result = a - b;
16     printf(" %5.2f - %5.2f = %5.2f \n", a, b, result);
17     result = a * b;
18     printf(" %5.2f * %5.2f = %5.2f \n", a, b, result);
19     result = a / b;
20     printf(" %5.2f / %5.2f = %5.2f \n", a, b, result);
21     result = (int)a % (int)b;
22     printf(" %d %% %d = %d \n", (int)a, (int)b, (int)result);
23 }
```

---실수형 변수를 선언한다.

---실수를 입력받는다.

---실수를 입력받는다.

---실수의 덧셈이다.

---실수의 뺄셈이다.

---실수의 곱셈이다.

---실수의 나눗셈이다.

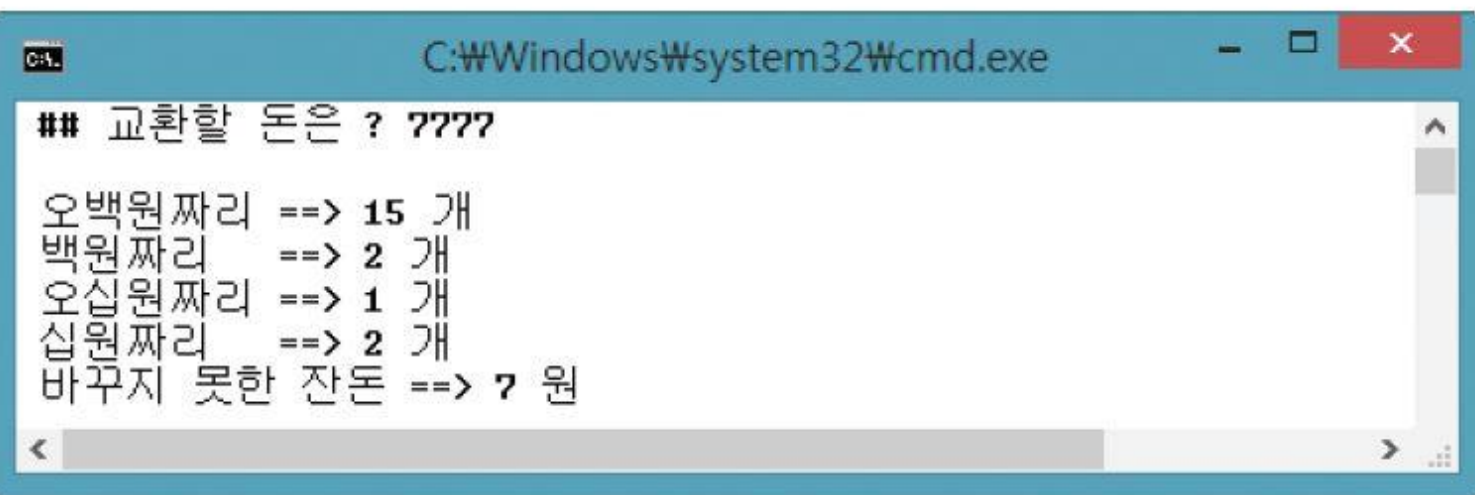
---나머지 연산을 위해 실수를 정수로 강제 형 변환한다.

[예제모음 09] 동전 교환 프로그램

입력된 액수만큼 500원, 100원, 50원, 10원짜리 동전으로 교환해주는 프로그램이다.

❶ 동전의 총수를 최소화한다.

❷ 고액의 동전을 먼저 바꿔준다.



```
C:\Windows\system32\cmd.exe

## 교환할 돈은 ? 7777

오백원짜리 ==> 15 개
백원짜리   ==> 2 개
오십원짜리 ==> 1 개
십원짜리   ==> 2 개
바꾸지 못한 잔돈 ==> 7 원
```


[예제모음 09] 동전 교환 프로그램

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int money, c500, c100, c50, c10;    ---입력한 돈과 각 동전의 개수를 저장할 변수이다.
06
07     printf(" ## 교환할 돈은 ? ");
08     scanf("%d", &money);    ---교환할 액수를 입력한다
09
10     c500 = money / 500;    ---500원짜리 동전의 개수를 계산한다.
11     money = money % 500;    ---500원짜리로 바꾼 후 나머지 금액이다.
12
13     c100 = money / 100;    ---100원짜리 동전의 개수를 계산한다.
14     money = money % 100;    ---100원짜리로 바꾼 후 나머지 금액이다.
15
16     c50 = money / 50;    ---50원짜리 동전의 개수를 계산한다.
17     money = money % 50;    ---50원짜리로 바꾼 후 나머지 금액이다.
18
19     c10 = money / 10;    ---10원짜리 동전의 개수를 계산한다.
20     money = money % 10;    ---10원짜리로 바꾼 후 나머지 금액이다.
21
```

[예제모음 09] 동전 교환 프로그램

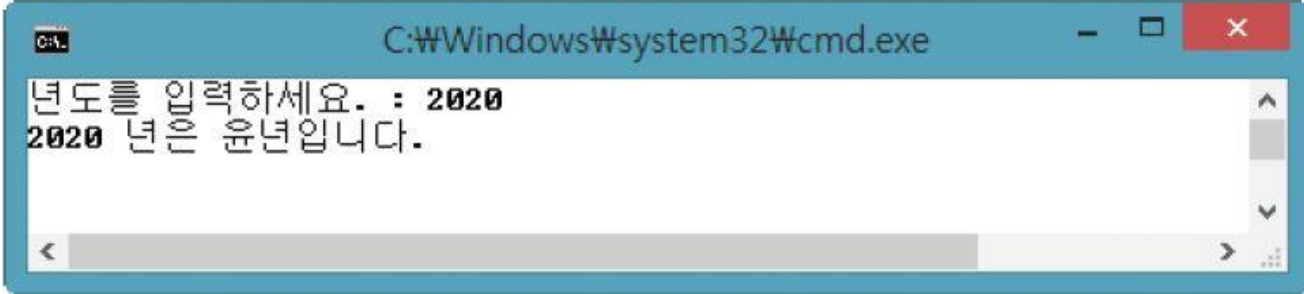
```
22  printf("\n 오백원짜리 == > %d 개 \n", c500);
23  printf(" 백원짜리 == > %d 개 \n", c100);
24  printf(" 오십원짜리 == > %d 개 \n", c50);
25  printf(" 십원짜리 == > %d 개 \n", c10);
26  printf(" 바꾸지 못한 잔돈 == > %d 원 \n", money); ---바꾸지 못한 나머지 돈은 money에 들어 있다.
27 }
```

[예제모음 10] 윤년 계산 프로그램

예제 설명 입력된 연도가 윤년인지 계산하는 프로그램이다.

- ❶ 4로 나누어 떨어지고 100으로 나누어 떨어지지 않으면 윤년이다.
- ❷ 400으로 나누어 떨어지는 해도 윤년에 포함된다.

실행 결과



```
C:\Windows\system32\cmd.exe

년도를 입력하세요. : 2020
2020 년은 윤년입니다.
```

The screenshot shows a Windows command prompt window with the title bar 'C:\Windows\system32\cmd.exe'. The prompt is '년도를 입력하세요. : 2020' and the output is '2020 년은 윤년입니다.'.

[예제모음 10] 윤년 계산 프로그램

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int year;
06
07     printf("년도를 입력하세요. : ");
08     scanf("%d", &year);
09
10     if ( ((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0) )
11         printf ("%d 년은 윤년입니다. \n", year);
12     else
13         printf ("%d 년은 윤년이 아닙니다. \n", year);
14 }
```

---계산할 연도를 입력한다.

---윤년은 입력한 연도가 4로 나누어 떨어지고 100으로는 나누어 떨어지지 않아야 한다. 또는 400으로 나누어 떨어져도 된다.

[4장 요약]

1 산술 연산자

- ❶ 더하기, 빼기, 곱하기, 나누기 등의 기호로 C에서 처리해야 할 가장 기본적인 연산자
- ❷ 연산자 우선순위는 *, /가 +, -보다 우선이다. 또한()는 가장 먼저, =은 가장 나중에 처리된다.
- ❸ 정수를 실수로 강제 형 변환하려면 정수 앞에 (float)를 붙인다.

표 4-1 산술 연산자

연산자	명칭	사용 예	설명
=	대입 연산자	$a = 3$	정수 3을 a에 대입한다.
+	더하기	$a = 5 + 3$	정수 5와 3을 더한 값을 a에 대입한다.
-	빼기	$a = 5 - 3$	정수 5에서 3을 뺀 값을 a에 대입한다.
*	곱하기	$a = 5 * 3$	정수 5와 3을 곱한 값을 a에 대입한다.
/	나누기	$a = 5 / 3$	정수 5를 3으로 나눈 값을 a에 대입한다.
%	나머지값	$a = 5 \% 3$	정수 5를 3으로 나눈 뒤 나머지 값을 a에 대입한다.

[4장 요약]

2 대입 연산자와 증감 연산자

표 4-2 대입 연산자와 증감 연산자

연산자	명칭	사용 예	설명
<code>+=</code>	대입 연산자	<code>a += 3</code>	<code>a = a + 3</code> 과 동일하다.
<code>-=</code>	대입 연산자	<code>a -= 3</code>	<code>a = a - 3</code> 과 동일하다.
<code>*=</code>	대입 연산자	<code>a *= 3</code>	<code>a = a * 3</code> 과 동일하다.
<code>/=</code>	대입 연산자	<code>a /= 3</code>	<code>a = a / 3</code> 과 동일하다.
<code>%=</code>	대입 연산자	<code>a %= 3</code>	<code>a = a % 3</code> 과 동일하다.
<code>++</code>	증가 연산자	<code>a++</code> 또는 <code>++a</code>	<code>a += 1</code> 또는 <code>a = a + 1</code> 과 동일하다.
<code>--</code>	감소 연산자	<code>a--</code> 또는 <code>--a</code>	<code>a -= 1</code> 또는 <code>a = a - 1</code> 과 동일하다.

3 관계 연산자

두 값을 비교하는 관계 연산자의 결과는 항상 참이나 거짓으로 표현된다

표 4-3 관계 연산자

연산자	의미	설명
<code>==</code>	같다.	두 값이 동일하면 참이다.
<code>!=</code>	같지 않다.	두 값이 다르면 참이다.
<code>></code>	크다.	왼쪽이 크면 참이다.
<code><</code>	작다.	왼쪽이 작으면 참이다.
<code>>=</code>	크거나 같다.	왼쪽이 크거나 같으면 참이다.
<code><=</code>	작거나 같다.	왼쪽이 작거나 같으면 참이다.

[4장 요약]

4 논리 연산자

두 가지 이상의 조건을 표현하는 경우에 사용하며 복잡한 조건을 표현할 수 있다.

표 4-4 논리 연산자

연산자	의미		사용 예	설명
&&	~ 이고	그리고(AND)	(a>100) && (a<200)	둘 다 참이어야 참이다.
	~ 이거나	또는(OR)	(a>100) (a<200)	둘 중 하나만 참이어도 참이다.
!	~ 아니다	부정(NOT)	!(a==100)	참이면 거짓, 거짓이면 참이다.

5 비트 연산자

정수나 문자 등을 2진수로 변환한 후 각 자리의 비트끼리 연산을 수행한다.

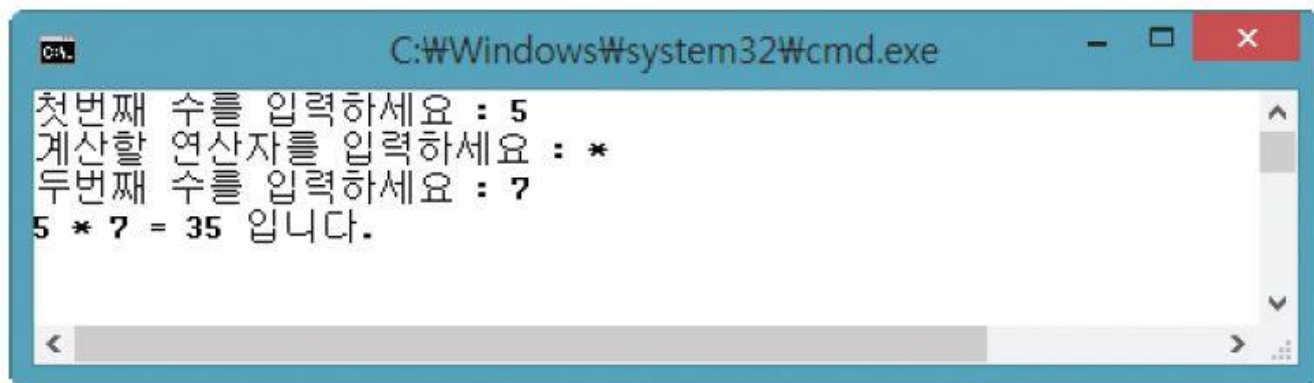
표 4-5 비트 연산자

연산자	명칭	설명
&	비트 논리곱(AND)	둘 다 1이면 1이다.
	비트 논리합(OR)	둘 중 하나만 1이면 1이다.
^	비트 배타적 논리합(XOR)	둘이 같으면 0, 둘이 다르면 1이다.
~	비트 부정	1은 0으로, 0은 1로 변경한다.
<<	비트 왼쪽 시프트(이동)	비트를 왼쪽으로 시프트(이동)한다.
>>	비트 오른쪽 시프트(이동)	비트를 오른쪽으로 시프트(이동)한다.

[예제모음 11] 단순 if문을 활용한 간단한 계산기

예제 설명 단순 if문을 활용하여 두 수의 $+$, $-$, $*$, $/$, $\%$ 연산을 수행하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
첫번째 수를 입력하세요 : 5
계산할 연산자를 입력하세요 : *
두번째 수를 입력하세요 : 7
5 * 7 = 35 입니다.
```


[예제모음 11] 단순 if문을 활용한 간단한 계산기

```
01 #include <stdio.h>
```

```
02
```

```
03 int main()
```

```
04 {
```

```
05     int a, b;
```

```
06     char ch;
```

```
07
```

```
08     printf("첫번째 수를 입력하세요 : ");
```

```
09     scanf("%d", &a);
```

```
10     printf("계산할 연산자를 입력하세요 : ");
```

```
11     scanf(" %c", &ch);
```

```
12     printf("두번째 수를 입력하세요 : ");
```

```
13     scanf("%d", &b);
```

```
14
```

```
15     if (ch == '+')
```

```
16         printf("%d + %d = %d 입니다. \n", a, b, a+b);
```

```
17
```

```
18     if (ch == '-')
```

```
19         printf("%d - %d = %d 입니다. \n", a, b, a-b);
```

```
20
```

----입력받을 정수 2개와 연산자 문자 1개를 선언한다.

----계산할 첫 번째 숫자를 입력한다.

----연산자를 입력한다.

----계산할 두 번째 숫자를 입력한다.

----기본 if문을 사용한 연산을 수행한다.

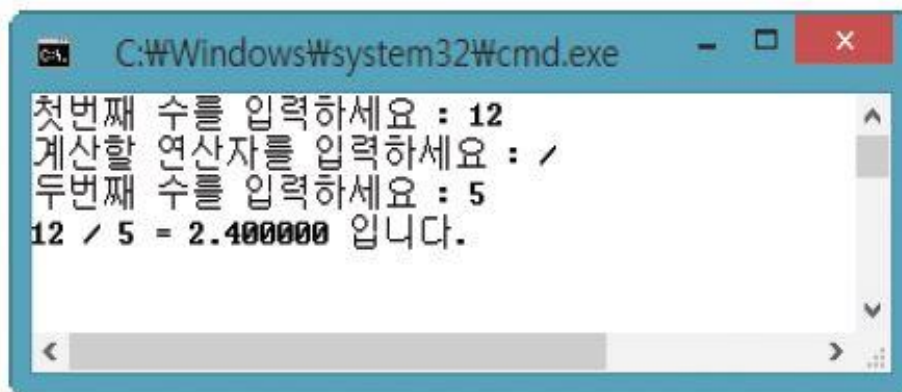
[예제모음 11] 단순 if문을 활용한 간단한 계산기

```
21  if (ch == '*')
22      printf("%d * %d = %d 입니다. \n", a, b, a*b);
23
24  if (ch == '/')
25      printf("%d / %d = %f 입니다. \n", a, b, a/(float)b);
26
27  if (ch == '%')
28      printf("%d %% %d = %d 입니다. \n", a, b, a%b);
29 }
```

[예제모음 12] 중복 if문을 활용한 간단한 계산기

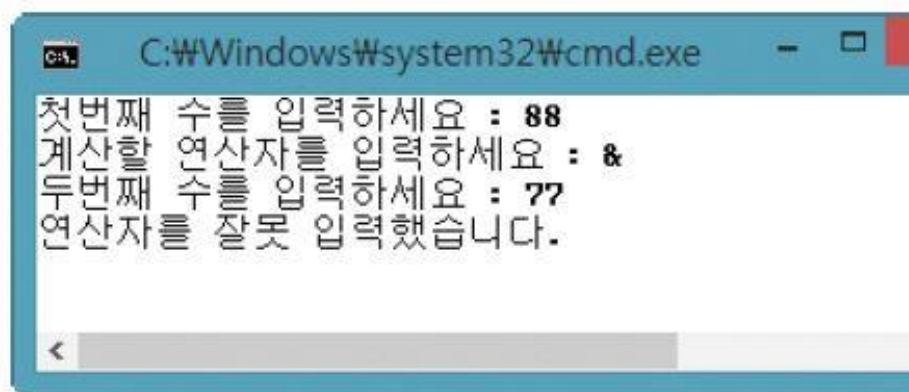
설명 중복 if문을 활용하여 두 수의 $+$, $-$, $*$, $/$, $\%$ 연산을 수행하는 프로그램이다.

결과



```
C:\Windows\system32\cmd.exe

첫번째 수를 입력하세요 : 12
계산할 연산자를 입력하세요 : /
두번째 수를 입력하세요 : 5
12 / 5 = 2.400000 입니다.
```



```
C:\Windows\system32\cmd.exe

첫번째 수를 입력하세요 : 88
계산할 연산자를 입력하세요 : &
두번째 수를 입력하세요 : 77
연산자를 잘못 입력했습니다.
```

[예제모음 12] 중복 if문을 활용한 간단한 계산기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b;
06     char ch;
07
08     printf("첫번째 수를 입력하세요 : ");
09     scanf("%d", &a);
10     printf("계산할 연산자를 입력하세요 : ");
11     scanf(" %c", &ch);
12     printf("두번째 수를 입력하세요 : ");
13     scanf("%d", &b);
14
15     if (ch == '+')
16         printf("%d + %d = %d 입니다. \n", a, b, a+b);
17     else if (ch == '-')
18         printf("%d - %d = %d 입니다. \n", a, b, a-b);
19     else if (ch == '*')
20         printf("%d * %d = %d 입니다. \n", a, b, a*b);
```

---중복 if문을 사용한 연산을 수행한다.

[예제모음 12] 중복 if문을 활용한 간단한 계산기

```
21  else if (ch == '/')
22      printf("%d / %d = %f 입니다. \n", a, b, a/(float)b);
23  else if (ch == '%')
24      printf("%d %% %d = %d 입니다. \n", a, b, a%b);
25  else
26      printf("연산자를 잘못 입력했습니다. \n");
27 }
```

---- +, -, *, /, % 외의 문자를 입력하면
오류 메시지를 보여준다.

[예제모음 13] switch~case문을 활용한 간단한 계산기

Scanf_s 사용시에는 인수 숫자 1을 삽입

예제 설명 수식을 띄어쓰기로 한 줄에 입력받고 switch~case문을 활용하여 두 수의 +, -, *, /, % 연산을 수행하는 프로그램이다.

실행 결과

```
C:\Windows\system32\cmd.exe

수식을 한줄로 띄어쓰기로 입력하세요 : 99 - 22
99 - 22 = 77 입니다.
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe

수식을 한줄로 띄어쓰기로 입력하세요 : 55 # 99
연산자를 잘못 입력했습니다.
계속하려면 아무 키나 누르십시오 . . .
```

[예제모음 13] switch~case문을 활용한 간단한 계산기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b;
06     char ch;
07
08     printf("수식을 한줄로 띄어쓰기로 입력하세요 : ");
09     scanf_s("%d %c %d", &a, &ch, 1, &b);
10
11     switch (ch)
12     {
13     case '+':
14         printf("%d + %d = %d 입니다. \n", a, b, a+b);
15         break;
16     case '-':
17         printf("%d - %d = %d 입니다. \n", a, b, a-b);
18         break;
19     case '*':
20         printf("%d * %d = %d 입니다. \n", a, b, a*b);
21         break;
```

----일반 수식처럼 띄어쓰기를 해서
한 줄에 변수 3개를 입력한다.
//scanf_s 사용시 %c의 데이터크기 숫자 1을 포함한다

----switch~case문을 사용한 연산을 수행한다.

[예제모음 13] switch~case문을 활용한 간단한 계산기

```
22  case '/':  
23      printf("%d / %d = %d 입니다. \n", a, b, a/b);  
24      break;  
25  case '%':  
26      printf("%d %% %d = %d 입니다. \n", a, b, a%b);  
27      break;  
28  default:  
29      printf("연산자를 잘못 입력했습니다. \n");  
30  }  
31 }
```


[5장 요약]

1 if문

❶ if문은 조건식이 참일 때와 거짓일 때
때 사용하며

각각 다른 일을 수행하는 제어문이다.

```
if(조건식)
    { 참일 때 실행할 문장들 }
else
    { 거짓일 때 실행할 문장들 }
```

❸ 중첩 if문은 처리할 조건이
세 가지 이상일 때 사용한다.

2 switch~case문

❶ 다양한 경우의 수가 있을

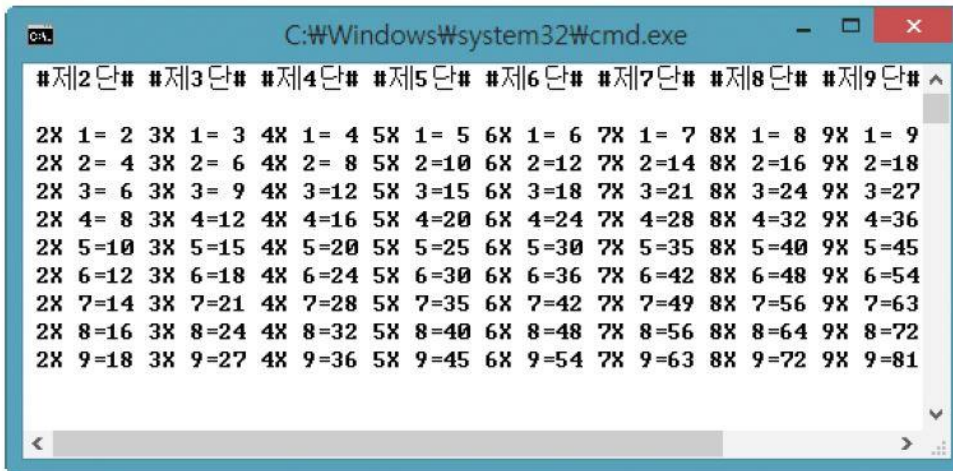
중첩 if문보다 구문을 깔끔

```
switch(정숫값){
    case 정숫값 1 :
        실행할 문장 1;
        break;
    case 정숫값 2 :
        실행할 문장 2;
        break;
    default :
        실행할 문장 3;
        break;
}
```

[예제모음 14] 구구단 출력 프로그램

예제 설명 중첩 for문을 사용하여 제목과 구구단을 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
#제2 단# #제3 단# #제4 단# #제5 단# #제6 단# #제7 단# #제8 단# #제9 단# ^
2X 1= 2 3X 1= 3 4X 1= 4 5X 1= 5 6X 1= 6 7X 1= 7 8X 1= 8 9X 1= 9
2X 2= 4 3X 2= 6 4X 2= 8 5X 2=10 6X 2=12 7X 2=14 8X 2=16 9X 2=18
2X 3= 6 3X 3= 9 4X 3=12 5X 3=15 6X 3=18 7X 3=21 8X 3=24 9X 3=27
2X 4= 8 3X 4=12 4X 4=16 5X 4=20 6X 4=24 7X 4=28 8X 4=32 9X 4=36
2X 5=10 3X 5=15 4X 5=20 5X 5=25 6X 5=30 7X 5=35 8X 5=40 9X 5=45
2X 6=12 3X 6=18 4X 6=24 5X 6=30 6X 6=36 7X 6=42 8X 6=48 9X 6=54
2X 7=14 3X 7=21 4X 7=28 5X 7=35 6X 7=42 7X 7=49 8X 7=56 9X 7=63
2X 8=16 3X 8=24 4X 8=32 5X 8=40 6X 8=48 7X 8=56 8X 8=64 9X 8=72
2X 9=18 3X 9=27 4X 9=36 5X 9=45 6X 9=54 7X 9=63 8X 9=72 9X 9=81
```

[예제모음 14] 구구단 출력 프로그램

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int i, k;
06
07     for( i = 2; i <= 9; i++ )
08         printf(" #제%d단#", i);      ----맨 위에 단의 제목을 출력한다
09
10     printf("\n\n");                  ----두 줄을 띄운다.
11
12     for ( i = 1; i <= 9; i++ )
13     {
14         for ( k = 2; k <= 9; k++ )
15         {
16             printf("%2dX%2d=%2d", k, i, k*i);
17         }
18         printf("\n");
19     }
20 }
```

----중첩 for문으로 구구단을 출력한다.

[예제모음 15] 아스키코드표 출력 프로그램

예제 설명 for문과 if문을 사용하여 아스키코드의 0~127을 10진수, 16진수, 문자로 출력하는 프로그램이다.

실행 결과

```
C:\Windows\system32\cmd.exe

-----
10진수  16진수  문자
-----
0       0
1       1
2       2
3       3
4       4
5       5
6       6
7       7
8       8
9       9
A       A
B       B
C       C
D       D
E       E
F       F
-----
10진수  16진수  문자
-----
16      10
17      11
```

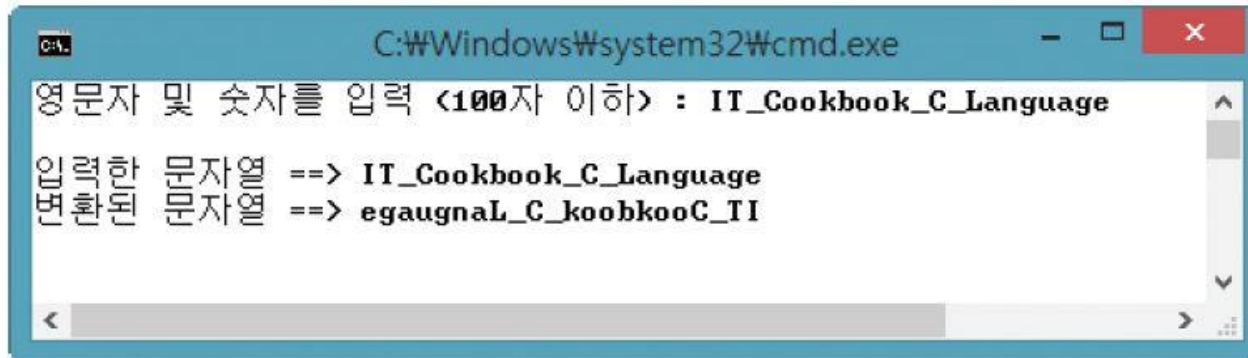
[예제모음 15] 아스키 코드표 출력 프로그램

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int i;
06
07     for ( i= 0; i < 128; i++ )    ---0~127을 처리한다.
08     {
09         if ( i%16 == 0 )    ---16행마다 제목 줄을 출력한다.
10         {
11             printf("-----\n");
12             printf("10진수 16진수 문자 \n");
13             printf("-----\n");
14         }
15         printf ("%5d %5x %5c\n", i, i, i);    ---i 값을 10진수, 16진수, 문자로 출력한다.
16     }
17 }
```

[예제모음 16] 입력한 문자를 반대 순서로 출력

예제 설명 입력된 영문자나 숫자를 for문을 사용하여 반대 순서로 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
영문자 및 숫자를 입력 <100자 이하> : IT_Cookbook_C_Language
입력한 문자열 ==> IT_Cookbook_C_Language
변환된 문자열 ==> egaugnaL_C_koobkooC_TI
```

[예제모음 16] 입력한 문자를 거꾸로 출력

```
01 #include <string.h>
02
03 int main()
04 {
05     char str[100];          ----입력받을 문자 배열이다.
06     int str_cnt;            ----입력한 문자의 개수를 저장할 변수이다
07     int i;
08
09     printf("영문자 및 숫자를 입력 (100자 이하) : ");
10     scanf("%s", str);      ----최대 99자까지 문자를 입력한다.
11
12     printf("\n");
13     printf("입력한 문자열 == > %s\n", str);  ----입력한 문자열을 출력한다.
14     printf("변환된 문자열 == >");
15
16     str_cnt = strlen(str);  ----입력한 문자의 개수를 계산한다.
17
18     for ( i= str_cnt; i >=0; i-- )          ----입력된 개수만큼 반대 순서로 출력한다.
19     {
20         printf ("%c", str[i]);
21     }
22
23     printf("\n");
24 }
```

[6장 요약]

1 for문

❶ for문은 반복할 문장을 원하는 만큼 반복함.

❷ for문의 형식

```
for ( 초기값 ; 조건식 ; 증감식 )  
{  
    반복할 문장들;  
}
```

2 중첩 for문

❶ for문 안에 또 다른 for문이 있는 형태.

```
for ( i = 0 ; i < 반복 횟수 ; i ++ )  
{  
    for ( k = 0 ; k < 반복 횟수 ; k ++ )  
    {  
        반복할 문장들;  
    }  
}
```

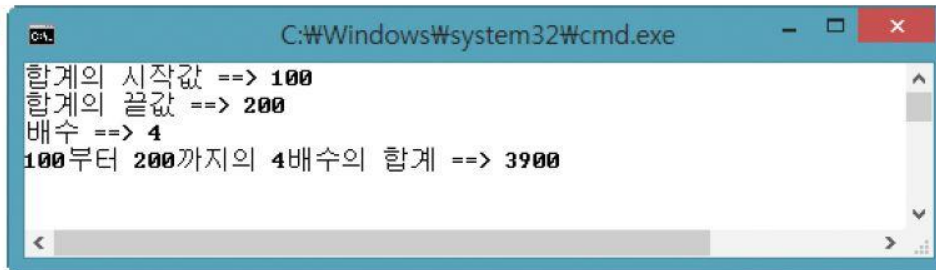
3 for문의 다른 형태

❶ for문의 초기값, 조건식, 증감식을 하나 이상 생략 가능함

[예제모음 17] 배수의 합계를 구하는 계산기

예제 설명 입력한 두 수 사이의 합계를 구하되 원하는 배수를 선택하는 프로그램이다. 예를 들어 100~200 중에서 4배수의 합계를 구할 수 있다.

실행 결과



```
C:\Windows\system32\cmd.exe
합계의 시작값 ==> 100
합계의 끝값 ==> 200
배수 ==> 4
100부터 200까지의 4배수의 합계 ==> 3900
```

[예제모음 17] 배수의 합계를 구하는 계산기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int start, end;          ----변수 선언과 함께 초기화한다.
06     int basu, i;
07     int hap = 0;
08
09     printf("합계의 시작값 == > ");
10     scanf("%d", &start);    ----시작값을 입력한다.
11     printf("합계의 끝값 == > ");
12     scanf("%d", &end);      ----끝값을 입력한다
13     printf("배수 == > ");
14     scanf("%d", &basu);     ----배숫값을 입력한다
15
16     i = start;               ----i 값을 시작값으로 초기화한다.
17     while (i <= end)         ----i 값이 끝값보다 작은 동안 반복한다.
18     {
19         if (i % basu == 0)   ----i 값이 입력한 배수라면 합계에 누적된다.
20             hap = hap + i;
21
22         i++;
23     }
```

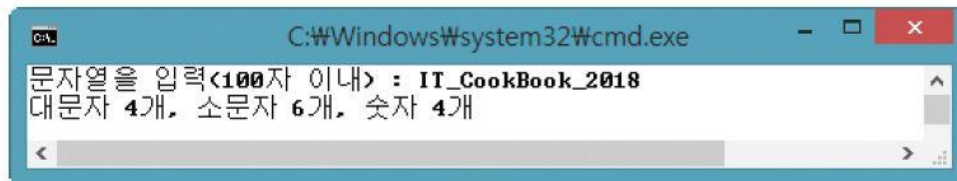
[예제모음 17] 배수의 합계를 구하는 계산기

```
24  
25     printf("%d부터 %d까지의 %d배수의 합계 == > %d\n", start, end, basu, hap);  
26 }
```

[예제모음 18] 입력한 문자열의 종류 구분

예제 설명 입력한 문자열에 대문자와 소문자, 숫자가 각각 몇 개 입력되었는지 세는 프로그램이다. 그 외는 무시한다.

실행 결과



```
C:\Windows\system32\cmd.exe
문자열을 입력<100자 이내> : IT_CookBook_2018
대문자 4개, 소문자 6개, 숫자 4개
```

[예제모음 18] 입력한 문자열의 종류 구분

```
01 #include <stdio.h>
02
03 int main()
04 {
05     char str[100];
06     char ch;
07
08     int upper_cnt = 0, lower_cnt=0, digit_cnt=0;
09     int i;
10
11     printf("문자열을 입력(100자 이내) : ");
12     scanf("%s", str);
13
14     i = 0;
15     do {
16         ch = str[i];
17
18         if(ch >= 'A' && ch <= 'Z')
19             upper_cnt ++;
20         if(ch >= 'a' && ch <= 'z')
21             lower_cnt ++;
```

----문자열 배열과 문자형 변수를 선언한다

----대문자, 소문자, 숫자의 개수를 초기화한다.

----문자열을 입력받는다.

----문자열의 위치를 나타낼 변수 i이다.

----입력한 문자열의 끝(\0)까지 반복한다.

-----문자열에서 한 글자를 추출한다

-----추출한 글자 하나가 A~Z이면 대문자의 개수가 하나 증가한다.

-----추출한 글자 하나가 a~z이면 소문자의 개수가 하나 증가한다.

[예제모음 18] 입력한 문자열의 종류 구분

```
22     if(ch >= '0' && ch <= '9')
```

```
23         digit_cnt++;
```

```
24
```

```
25     i++;
```

```
26 } while (ch != '\0');
```

```
27
```

```
28 printf("대문자 %d개, 소문자 %d개, 숫자 %d개\n", upper_cnt, lower_cnt, digit_cnt);
```

```
29 }
```

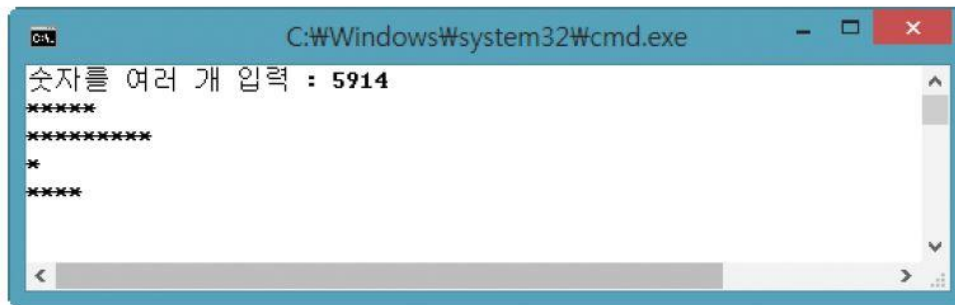
-----추출한 글자 하나가 0~9이면 숫자의 개수가
하나 증가한다.

-----다음 글자를 추출하기 위해 i 값을 증가시킨다.

[예제모음 19] 입력된 숫자만큼 별표 출력

예제 설명 입력한 숫자만큼의 별 모양을 출력하는 프로그램이다. 예를 들어 5914를 입력하면 각 줄에 별을 5개, 9개, 1개, 4개 출력한다.

실행 결과



```
C:\Windows\system32\cmd.exe
숫자를 여러 개 입력 : 5914
*****
*****
*
****
```

[예제모음 19] 입력된 숫자만큼 별표 출력

```
01 #include <stdio.h>
```

```
02
```

```
03 int main()
```

```
04 {
```

```
05     char str[100];
```

----문자열 배열과 문자형 변수를 선언한다.

```
06     char ch;
```

```
07
```

```
08     int i, k;
```

----정수형 변수를 선언한다. i, k는 반복문에서 사용한다.
star는 별의 개수를 추출한다.

```
09     int star;
```

```
10
```

```
11     printf("숫자를 여러 개 입력 : ");
```

```
12     scanf("%s", str);
```

----문자열(숫자만)을 입력받는다.

```
13
```

```
14     i = 0;
```

----문자열의 위치를 나타낼 변수 i이다.

```
15     ch = str[i];
```

----문자열에서 한 글자(숫자)를 추출한다

```
16     while (ch != '\0') {
```

----문자가 있는 동안 반복한다(4회 반복).

```
17         star = (int)ch - 48;
```

----아스키코드 값으로 계산해서 문자를
숫자로 변환한다.

```
18
```

```
19         for(k=0; k<star; k++)
```

----별의 개수만큼 *를 화면에 출력한다

```
20             printf("*");
```

```
21
```


[예제모음 19] 입력된 숫자만큼 별표 출력

```
22     printf("\n");
```

-----한 줄을 띄운다.

```
23     i = i + 1;
```

-----다음 문자를 추출하기 위해 i 값을 증가시킨다.

```
24     ch = str[i];
```

```
25 }
```

```
26 }
```

[7장 요약]

1 while문

- ❶ while문은 for문과 같이 특정 동작을 반복하기 위해 사용함.
- ❷ 무한 루프를 돌리려면 while(1) 형식을 사용함
- ❸ while문의 기본 형식.

```
while (조건식)
{
    반복할 문장들;
}
```

2 do~while문

- ❶ do~while문은 while문과 거의 동일하지만 조건이 참이든 거짓이든 무조건 반복할 문장을 한 번은 수행함.
- ❷ do~while문의 기본 형식.

```
do
{
    반복할 문장들;
} while (조건식);
```

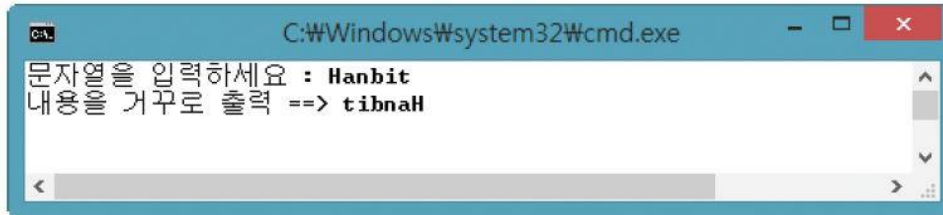
3 기타 제어문

- ❶ break문을 만나면 현재의 반복문을 무조건 탈출함.
- ❷ continue문을 만나면 무조건 블록의 끝으로 이동한 후 다시 반복문의 처음으로 돌아감

[예제모음 20] 입력된 문자열을 반대 순서로 출력

예제 설명 문자열 배열을 이용해서 입력받은 문자열을 반대 순서로 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
문자열을 입력하세요 : Hanbit
내용을 거꾸로 출력 ==> tibnaH
```

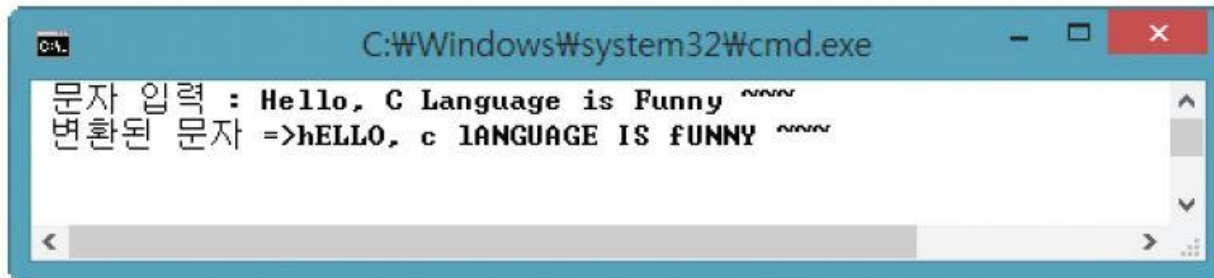
[예제모음 20] 입력된 문자열을 반대 순서로 출력

```
01 #include <stdio.h>
02 #include <string.h>
03
04 int main()
05 {
06     char ss[100];      ----문자형 배열 ss를 선언한다.
07     char tt[100];      ----문자형 배열 tt를 선언한다.
08     int count, i;
09
10     printf("문자열을 입력하세요 : ");
11     scanf("%s", ss);   ----문자열을 입력받는다.
12
13     count = strlen(ss); ----입력받은 문자열의 개수를 구한다.
14
15     for(i=0; i<count; i++) ----문자열의 개수만큼 반복해서 tt 배열에
16     {                      문자열을 반대 순서로 저장한다.
17         tt[i] = ss[count-(i+1)];
18     }
19     tt[count] = '\0';   ----tt 배열의 마지막에 널 문자를 입력한다
20
21     printf("내용을 거꾸로 출력 == > %s \n", tt);
22 }
```

[예제모음 21] 대문자와 소문자의 변환

예제 설명 입력된 문자열이 대문자이면 소문자로, 소문자이면 대문자로 변환하고 그 외의 문자는 그대로 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
문자 입력 : Hello, C Language is Funny ~~~~
변환된 문자 =>hELLO, c LANGUAGE IS FUNNY ~~~~
```

[예제모음 21] 대문자와 소문자의 변환

```
01 #include <stdio.h>
02 #include <string.h>
03
04 int main()
05 {
06     char in[50], out[50];    ---입력 문자형 배열 in과 출력 문자형 배열 out이다.
07     int i, len;
08     int diff = 'a' - 'A';    ---대문자와 소문자의 값 차이를 diff에 저장한다.
09
10     printf(" 문자 입력 : ");
11     gets(in);                ---문자를 입력받는다. 실제 최대 입력 문자는
                                ' 배열 크기-1'이다.
12
13     len = strlen(in);        ---입력한 문자열의 길이를 구한다.
14
15     for(i=0; i<len; i++)
16     {
17         if ( ('A' <= in[i]) && (in[i] <= 'Z') )    ---문자가 대문자이면 대소문자 차이값을 더한다.
18             out[i] = in[i] + diff;
19         else if ( ('a' <= in[i]) && (in[i] <= 'z') )    ---문자가 소문자이면 대소문자 차이값을 뺀다.
20             out[i] = in[i] - diff;
21         else    ---영문자가 아닌 기호, 숫자 등은 그대로 둔다.
22             out[i] = in[i];
```

[예제모음 21] 대문자와 소문자의 변환

```
23 }
```

```
24 out[i] = '\0';
```

----마지막에 널 문자를 입력한다.

```
25
```

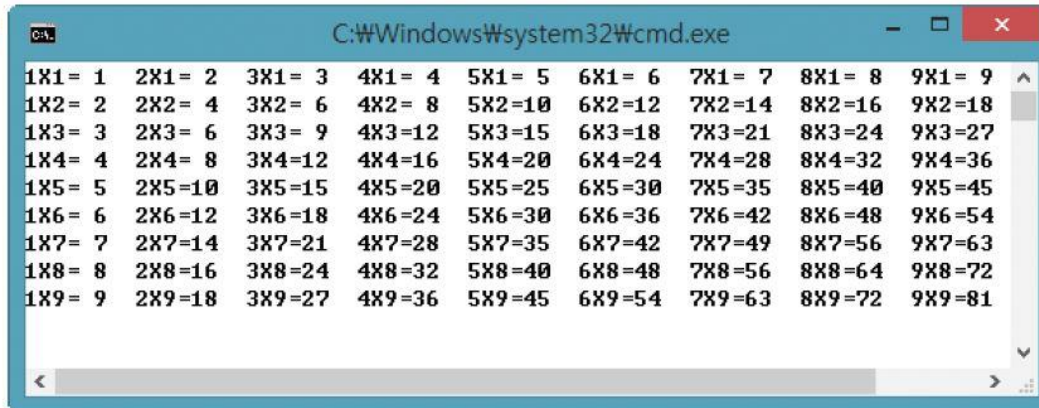
```
26 printf(" 변환된 문자 = >%s \n", out);
```

```
27 }
```

[예제모음 22] 구구단의 결과를 2차원 배열에 저장

예제 설명 구구단의 결과를 2차원 배열에 저장한 후 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
1x1= 1  2x1= 2  3x1= 3  4x1= 4  5x1= 5  6x1= 6  7x1= 7  8x1= 8  9x1= 9
1x2= 2  2x2= 4  3x2= 6  4x2= 8  5x2=10  6x2=12  7x2=14  8x2=16  9x2=18
1x3= 3  2x3= 6  3x3= 9  4x3=12  5x3=15  6x3=18  7x3=21  8x3=24  9x3=27
1x4= 4  2x4= 8  3x4=12  4x4=16  5x4=20  6x4=24  7x4=28  8x4=32  9x4=36
1x5= 5  2x5=10  3x5=15  4x5=20  5x5=25  6x5=30  7x5=35  8x5=40  9x5=45
1x6= 6  2x6=12  3x6=18  4x6=24  5x6=30  6x6=36  7x6=42  8x6=48  9x6=54
1x7= 7  2x7=14  3x7=21  4x7=28  5x7=35  6x7=42  7x7=49  8x7=56  9x7=63
1x8= 8  2x8=16  3x8=24  4x8=32  5x8=40  6x8=48  7x8=56  8x8=64  9x8=72
1x9= 9  2x9=18  3x9=27  4x9=36  5x9=45  6x9=54  7x9=63  8x9=72  9x9=81
```


[예제모음 22] 구구단의 결과를 2차원 배열에 저장

```
01 #include <stdio.h>
```

```
02
```

```
03 int main()
```

```
04 {
```

```
05     int gugu[9][9];
```

---문자형 2차원 배열 gugu와 첨자 변수 i, k를 선언한다.

```
06     int i, k;
```

```
07
```

```
08     for(i=0; i<9; i++)
```

---구구단을 곱한 결과를 2차원 배열에 저장한다.
i, k가 0부터 시작되므로 1을 더해서 곱한다.

```
09         for(k=0; k<9; k++)
```

```
10             gugu[i][k] = (i+1) * (k+1);
```

```
11
```

```
12     for(i=0; i<9; i++)
```

---구구단 결과를 출력한다.

```
13     {
```

```
14         for(k=0; k<9; k++)
```

```
15         {
```

```
16             printf("%dX%d= %2d ", k+1, i+1, gugu[i][k]);
```

```
17         }
```

```
18         printf("\n");
```

-----한 행을 출력한 후 줄을 넘긴다.

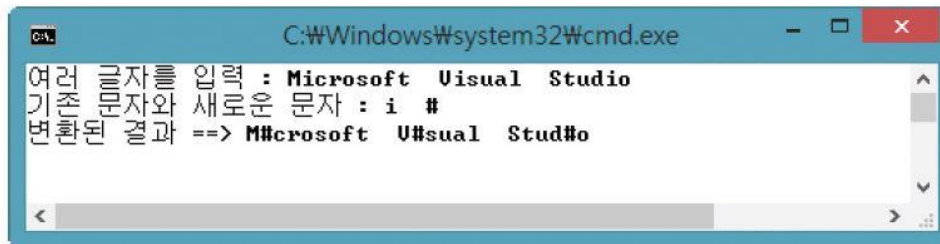
```
19     }
```

```
20 }
```

[예제모음 23] 문자열 내 특정 문자의 변환

예제 설명 문자열을 입력받고 그 문자열에서 변환할 기존 문자와 새로운 문자를 각각 입력받은 뒤 변환된 문자열을 반환하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
여러 글자를 입력 : Microsoft Visual Studio
기존 문자와 새로운 문자 : i #
변환된 결과 ==> M#icrosoft U#sual Stud#o
```

[예제모음 23] 문자열 내 특정 문자의 변환

```
01 #include <stdio.h>
02 #include <string.h>
03
04 int main()
05 {
06     char str[100];      ----문자형 배열 str을 선언한다.
07     char ch1, ch2;      ----기존 문자와 새 문자를 위한 문자형 변수이다.
08     int i;
09
10     printf("여러 글자를 입력 : ");
11     gets(str);          ----최대 99자를 입력받는다.
12
13     printf("기존 문자와 새로운 문자 : ");
14     scanf("%c %c", &ch1, &ch2); ----기존 문자(ch1)와 새 문자(ch2)를 한 글자씩
15                                     입력받는다(띄어쓰기로 구분).
16
17     for(i=0; i<strlen(str); i++)
18     {
19         if(str[i] == ch1)
20             str[i] = ch2;
21     }
22     printf("변환된 결과 = > %s \n", str);
23 }
```

[8장 요약]

1 배열의 기본

- ❶ 배열은 변수 여러 개를 나란히 나열해놓은 개념이다.
- ❷ 변수 여러 개를 개별 선언하지 않고 공통된 변수 이름에 첨자만 변경해서 사용할 수 있다.
- ❸ 배열의 첨자는 대개 0부터 시작한다.
- ❹ for문 등의 반복문과 함께 사용하는 경우가 많다.
- ❺ 배열의 개수를 알아내려면 sizeof() 함수를 사용한다.

2 배열과 문자열

- ❶ 문자 여러 개를 나열한 문자열은 배열 형태로 표현할 수 있다.
- ❷ 배열에 문자열 저장하려면 문자열 끝을 나타내는 '\0'을 고려해 '문자열 길이 +1' 크기 필요하다.

3

문자열 함수	기능
strlen()	문자열의 길이를 알려준다.
strcpy()	문자열을 복사한다.
strcat()	두 문자열을 이어준다.
strcmp()	두 문자열을 비교한다.
gets(), puts()	문자열을 키보드로 입력하거나 화면에 출력한다.

[8장 요약]

4 2차원 배열

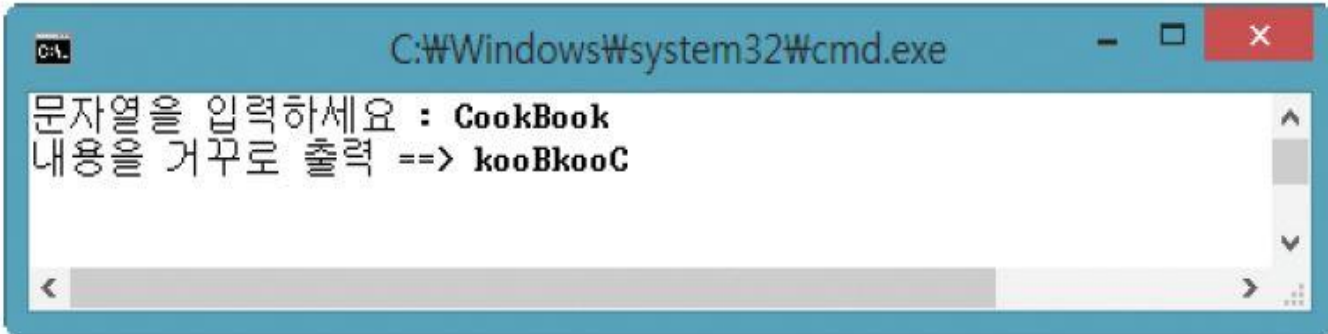
- ❶ 행과 열로 만든 배열, 2차원 배열의 개수는 '행 수×열 수'로 계산한다.
- ❷ 선언과 동시에 2차원 배열을 초기화하는 형식

```
int aa[3][4] = {  
    { 1, 2, 3, 4 },  
    { 5, 6, 7, 8 },  
    { 9, 10, 11, 12 }  
};
```

[예제모음 24] 포인터를 이용하여 문자열을 반대 순서로 출력

예제 설명 8장의 [예제모음 20]에서처럼 입력한 문자열을 반대 순서로 출력해보자. 이번에는 포인터를 활용하여 작성한 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

문자열을 입력하세요 : CookBook
내용을 거꾸로 출력 ==> kooBkooC
```

The screenshot shows a Windows command prompt window with the title bar "C:\Windows\system32\cmd.exe". The prompt is "C>". The user has entered "문자열을 입력하세요 : CookBook" and the program has responded with "내용을 거꾸로 출력 ==> kooBkooC". The output is the reverse of the input string "CookBook".

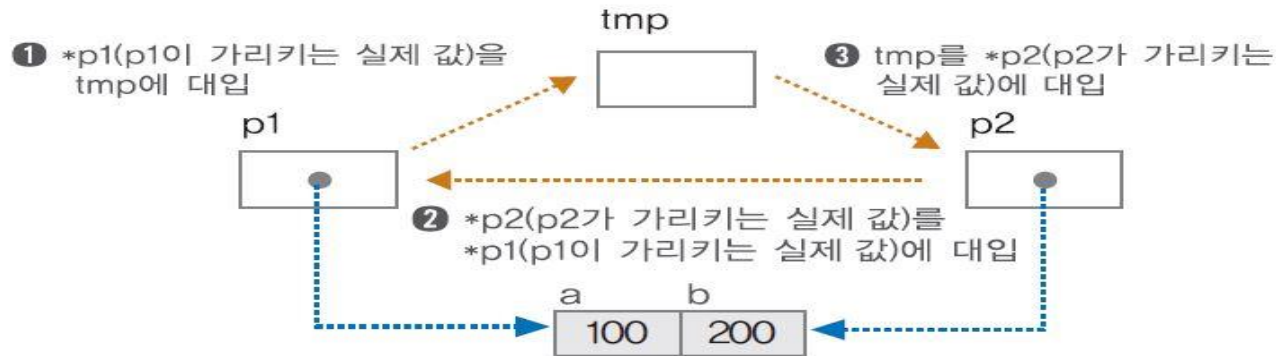
[예제모음 24] 포인터를 이용하여 문자열을 반대 순서로 출력

```
01 #include <stdio.h>
02 #include <string.h>
03
04 int main()
05 {
06     char ss[100];      ----입력받을 문자 배열을 선언한다.
07     int count, i;
08     char *p;           ----문자형 포인터를 선언한다.
09
10     printf("문자열을 입력하세요 : ");
11     scanf("%s", ss);   ----문자열을 입력한다.
12
13     count = strlen(ss); ----입력한 문자열의 개수이다.
14
15     p = ss;            ----배열 ss의 주소를 포인터 변수 p에 대입한다.
16
17     printf("내용을 거꾸로 출력 == > ");
18     for(i=0; i<count; i++)
19     {
20         printf("%c", *(p+count-(i+1)));
21     }
22     printf("\n");
23 }
```

----포인터 p에 있는 실제 값을 문자열의 맨 뒤부터 출력한다.

[예제모음 25] 포인터를 이용한 두 값의 교환

예제 설명 입력한 두 값을 포인터를 활용하여 교환하는 프로그램이다. 값을 바꾸는 개념은 다음과 같다.



실행 결과

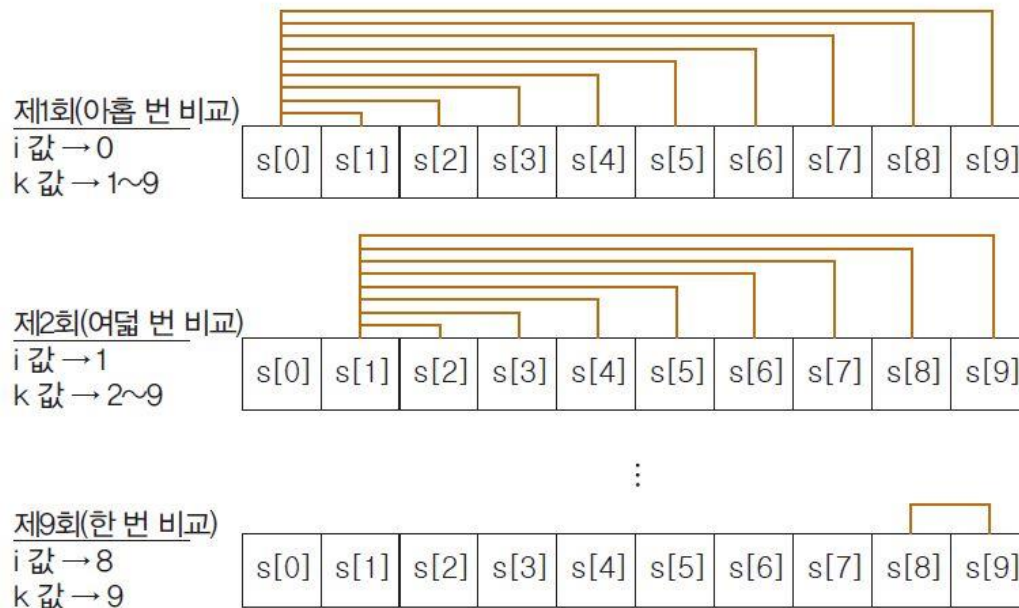
```
C:\Windows\system32\cmd.exe
a 값 입력 : 100
b 값 입력 : 200
바뀐 값 a는 200, b는 100
```


[예제모음 25] 포인터를 이용한 두 값의 교환

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b, tmp;           --정수형 변수 3개와 포인터 변수 2개를 선언한다.
06     int *p1, *p2;
07
08     printf("a 값 입력 : ");   --a와 b에 값을 입력한다.
09     scanf("%d", &a);
10     printf("b 값 입력 : ");
11     scanf("%d", &b);
12
13     p1 = &a;                 --변수 a의 주솟값을 p1에 대입한다.
14     p2 = &b;                 --변수 b의 주솟값을 p2에 대입한다.
15
16     tmp = *p1;               --① p1이 가리키는 곳의 실제 값을 tmp에 넣는다.
17     *p1 = *p2;               --② p2가 가리키는 곳의 실제 값을 p1이 가리키는 곳에 넣는다.
18     *p2 = tmp;               --③ tmp에 저장된 값을 p2가 가리키는 곳에 넣는다.
19
20     printf("바뀐 값 a는 %d, b는 %d \n", a, b);
21 }
```

[예제모음 26] 포인터를 이용한 배열의 정렬

예제 설명 포인터를 이용하여 배열에 들어 있는 값 10개를 정렬하는 프로그램이다. 다음 그림을 참고하여 두 값을 비교하고 작은 것을 앞으로 옮기는 선택 정렬을 사용한다.



실행 결과

```
C:\Windows\system32\cmd.exe

정렬 전 배열 s ==> 1 0 3 2 5 4 7 6 9 8
정렬 후 배열 s ==> 0 1 2 3 4 5 6 7 8 9
```

[예제모음 26] 포인터를 이용한 배열의 정렬

```
01 #include <stdio.h>
```

```
02
```

```
03 int main()
```

```
04 {
```

```
05     int s[10] = {1, 0, 3, 2, 5, 4, 7, 6, 9, 8};
```

----배열 s에 정숫값 10개를 초기화하고
관련 변수를 선언한다.

```
06     int tmp;
```

```
07     int i, k;
```

```
08
```

```
09     int *p;
```

----포인터 변수를 선언한다.

```
10
```

```
11     p = s;
```

----배열 s의 주소를 포인터 변수 p에 대입한다.

```
12
```

```
13     printf("정렬 전 배열 s = = > ");
```

```
14     for(i=0; i<10; i++)
```

----정렬 전의 상태로 데이터 10개를 출력한다.

```
15     {
```

```
16         printf("%d ", *(p+i));
```

```
17     }
```

```
18     printf("\n");
```

```
19
```

[예제모음 26] 포인터를 이용한 배열의 정렬

```
20  for (i=0 ; i<9 ; i++)
21  {
22      for(k=i+1 ; k<10 ; k++)
23      {
24          if ( *(p+i) > *(p+k) )
25          {
26              tmp = *(p+i);
27              *p(+i) = *(p+k);
28              *p(+k) = tmp;
29          }
30      }
31  }
32  printf("정렬 후 배열 s == > ");
33  for(i=0 ; i<10 ; i++)
34  {
35      printf("%d ", *(p+i));
36  }
37  printf("\n");
38 }
```

----9회 반복한다. 내부 for문으로 비교할 두 값 중 첫 번째 값의 자리는 s[0]~s[8]이다.

-----내부 for문으로 9회, 8회, 7회, ..., 1회 반복한다. 비교할 두 값 중 두 번째 값의 자리는 s[1]~s[9]이다.

-----p+i의 실제 값이 p+k의 실제 값보다 크면 두 값을 바꾼다.

----정렬 후의 상태로 데이터 10개를 출력한다.

[9장 요약]

1 스택

- ❶ 한쪽 끝이 막혀 있는 데이터 구조(자료 구조)를 말하며 LIFO(Last In First Out) 구조이다.
- ❷ 스택을 구현하려면 배열이라는 데이터 구조를 사용해야 한다.

2 메모리와 주소

- ❶ 프로그램에서 사용된 변수, 배열 등은 모두 메모리(램)에 존재하며 이 메모리의 각 자리에는
주소가 할당되어 있다.
- ❷ 변수의 주소를 표시하려면 변수 이름 앞에 &를 붙인다.
- ❸ 배열의 이름 자체는 배열의 시작 주소이다.
- ❹ 정수형 변수는 4바이트를, 문자형 변수는 1바이트를 차지한다.

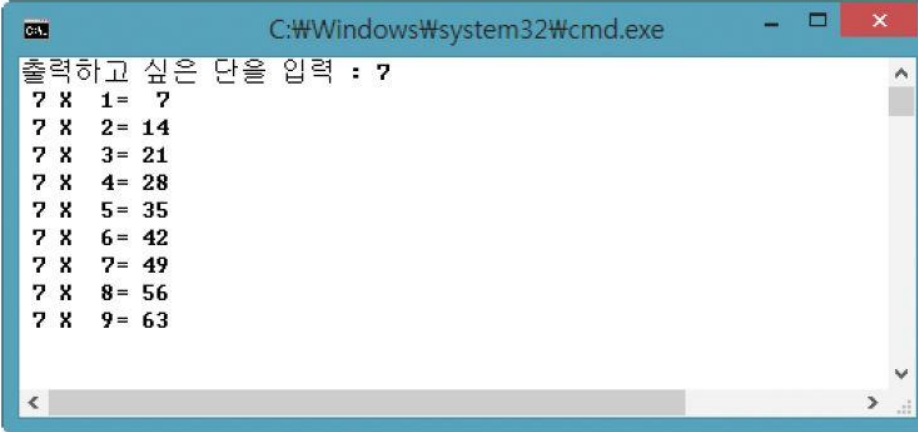
3 포인터

- ❶ 포인터 변수란 주소를 저장하는 변수로, 포인터 변수를 선언할 때는 *를 붙인다.

[예제모음 27] 함수를 이용한 구구단 프로그램

예제 설명 함수를 사용하여 구구단을 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
출력하고 싶은 단을 입력 : 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
```

[예제모음 27] 함수를 이용한 구구단 프로그램

```
01 #include <stdio.h>
```

```
02
```

```
03 void gugu(int dan)
```

```
04 {
```

```
05     int i;
```

```
06
```

```
07     for (i=1; i<=9; i++)
```

```
08     {
```

```
09         printf("%2d X %2d= %2d \n", dan, i, dan*i);
```

```
10     }
```

```
11 }
```

```
12
```

```
13 int main()
```

```
14 {
```

```
15     int input;
```

```
16
```

```
17     printf("출력하고 싶은 단을 입력 : ");
```

```
18     scanf("%d", &input);
```

```
19
```

```
20     gugu(input);
```

```
21 }
```

----gugu() 함수를 정의한다(매개변수는 정수형 dan이다).

-----1~9를 반복하며 매개변수로 받은 dan의 단을 출력한다.

----출력할 단을 입력한다.

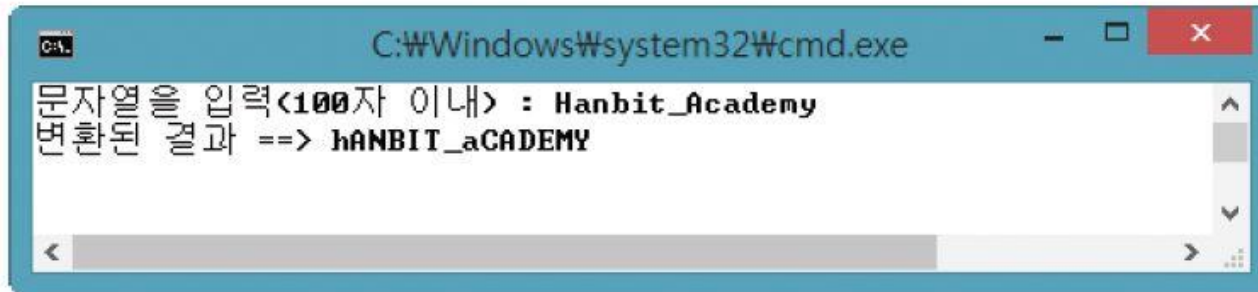
----구구단을 계산하고 출력할 함수를 출력한다.

[예제모음 28] 함수를 이용한 대소문자 변환 프로그램

예제 설명 대문자는 소문자로, 소문자는 대문자로 변환하는 프로그램이다.

- ① 대문자 변환 방법: 소문자에서 대·소문자 차이를 뺀다.
- ② 소문자 변환 함수: 대문자에서 대·소문자 차이를 더한다.

실행 결과



```
C:\Windows\system32\cmd.exe
문자열을 입력<100자 이내> : Hanbit_Academy
변환된 결과 ==> hANBIT_aCADEMY
```


[예제모음 28] 함수를 이용한 대소문자 변환 프로그램

```
01 #include <stdio.h>
```

```
02
```

```
03 char upper(char ch)
```

---대문자로 변환하는 함수이다.

```
04 { return ch - ('a' - 'A'); }
```

```
05
```

```
06 char lower(char ch)
```

---소문자로 변환하는 함수이다.

```
07 { return ch + ('a' - 'A'); }
```

```
08
```

```
09 int main()
```

```
10 {
```

```
11     char in[100], out[100];
```

```
12     char ch;
```

```
13     int i = 0;
```

```
14
```

```
15     printf("문자열을 입력(100자 이내) : ");
```

```
16     scanf("%s", in);
```

---문자열을 입력받는다.

```
17
```

[예제모음 28] 함수를 이용한 대소문자 변환 프로그램

```
18  do {
19      ch = in[i];
20      if(ch >= 'A' && ch <= 'Z')
21          out[i] = lower(ch);
22      else if(ch >= 'a' && ch <= 'z')
23          out[i] = upper(ch);
24      else
25          out[i] = ch;
26      i++;
27  } while (ch != '\0');
28
29  out[i] = '\0';
30  printf("변환된 결과 = > %s\n",out);
31 }
```

-----문자열이 널이 아닌 동안 반복한다.
즉 문자열의 끝까지 반복한다.

-----문자형 배열에서 한 문자만 추출한다.

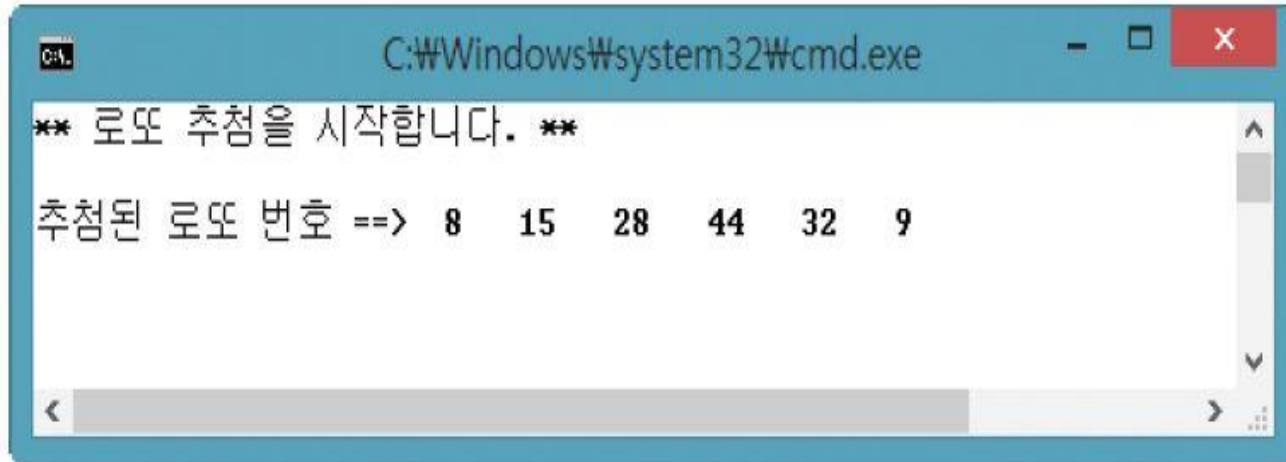
-----문자가 대문자이면 lower()함수를,
소문자이면 upper()함수를 호출한다.
숫자나 기호 등은 그대로 사용한다.

----출력 문자열의 맨 뒤에 널 문자를 추가한다.

[예제모음 29] 숫자 자동 추첨 프로그램

예제 설명 1~45 중에서 숫자 6개를 자동으로 뽑는 프로그램이다.

실행 결과



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The window contains two lines of text: the first line is '*** 로또 추첨을 시작합니다. ***' and the second line is '추첨된 로또 번호 ==> 8 15 28 44 32 9'. The window has a blue title bar and a white background with a scrollbar on the right.

```
C:\Windows\system32\cmd.exe
*** 로또 추첨을 시작합니다. ***
추첨된 로또 번호 ==> 8 15 28 44 32 9
```

[예제모음 29] 숫자 자동 추첨 프로그램

```
01 #include <stdio.h>
```

```
02 #include <stdlib.h>
```

----관련 함수를 사용하기 위해 포함한 헤더 파일이다.

```
03 #include <time.h>
```

```
04
```

```
05 int getNumber() {
```

----1~45 중에서 숫자 하나를 추출하는 함수이다.

```
06     return rand() % 45 + 1;
```

-----rand() 함수는 0~32767 중 하나를 임의로 반환한다.

```
07 }
```

```
08
```

```
09 int main()
```

```
10 {
```

```
11     short int lotto[6] = {0,};
```

----추첨된 숫자를 담을 배열이다.

```
12     int i, k, num;
```

----반복 변수 i, k와 뽑힌 숫자를 담을 변수 num이다.

```
13     char dup= 'N';
```

----이미 뽑힌 숫자인지 체크하기 위한 변수이다.

```
14
```

```
15     printf("** 로또 추첨을 시작합니다. ** \n\n");
```

```
16     srand((unsigned)time(NULL));
```

----rand() 함수를 초기화하는 함수이다.

```
17
```

이 행이 없으면 늘 같은 숫자가 뽑힌다.

[예제모음 29] 숫자 자동 추첨 프로그램

```
18  for (i=0;i<6;) {
19      num = getNumber();
20
21      for(k=0; k<6; k++)
22          if (lotto[k] == num)
23              dup = 'Y';
24
25      if(dup == 'N')
26          lotto[i++] = num;
27      else
28          dup = 'N';
29  }
30
31  printf("추첨된 로또 번호 == > ");
32  for (i= 0 ; i < 6 ; i++) {
33      printf("%d ", lotto[i]);
34  }
35
36  printf("\n\n");
37 }
```

-----다른 숫자 6개가 뽑힐 때까지 반복한다(18~28행).
다른 숫자가 뽑히면 18행에서 i를 1 증가시킨다.

-----로또 숫자를 1개 뽑는다.

-----뽑은 숫자가 이미 뽑은 숫자와 동일한지 체크하고,
동일하면 중복 확인 변수에 'Y'를 대입한다.

-----뽑은 숫자가 처음 뽑혔다면 로또 배열에 넣고
i(뽑힌 개수)를 1 증가시킨다(25~28행).
아니면 다시 중복 확인 변수에 'N'을 대입한다.

-----뽑힌 로또 숫자 6개를 출력한다.

[10장 요약]

1 함수의 이해

- ❶ 어떤 값이 들어가면 그것을 처리한 후 하나의 결과값을 돌려준다.
- ❷ 간단히 '함수 이름()' 형식으로 사용한다.
- ❸ 함수는 반복적인 것을 처리할 때 유용하다

2 출력하는 예

```
int plus (int v1, int v2)
{
    int result;
    result = v1 + v2;
    return result;
}
hap = plus (100, 200);
```

3 지역변수와 전역변수

지역변수는 선언된 함수 안에서만 유효한 변수이고, 전역변수는 모든 범위에서 유효한 변수이다

[10장 요약]

4 함수의 반환값

- ❶ 함수에서 값을 돌려주기 위해서는 return문을 사용한다.
- ❷ 함수가 돌려줄 값에 따라 함수 이름 앞에 데이터 형식이 붙는다.
즉 정숫값을 반환하려면 'int 함수 이름()'처럼 사용한다.
- ❸ 돌려줄 값이 없다면 함수를 void 형으로 선언한다.

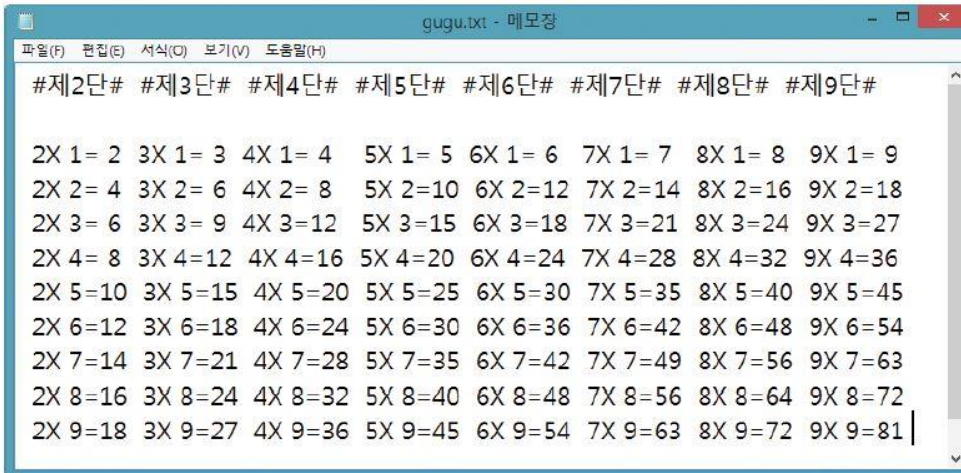
5 매개변수 전달 방법

- ❶ 값으로 전달 : 값을 복사해서 해당 함수에서 사용하는 것이므로 기존의 변수에 들어가는 값은 변하지 않는다.
- ❷ 주소로 전달 : 값이 들어있는 주소를 넘겨주기 때문에 연산 결과에 따라 기존의 값이 변한다.

[예제모음 30] 구구단을 파일에 출력

예제 설명 [예제모음 14]의 내용을 모니터가 아닌 'gugu.txt' 파일에 쓰는 프로그램이다.

실행 결과



```
#제2단# #제3단# #제4단# #제5단# #제6단# #제7단# #제8단# #제9단#  
  
2X 1= 2 3X 1= 3 4X 1= 4 5X 1= 5 6X 1= 6 7X 1= 7 8X 1= 8 9X 1= 9  
2X 2= 4 3X 2= 6 4X 2= 8 5X 2=10 6X 2=12 7X 2=14 8X 2=16 9X 2=18  
2X 3= 6 3X 3= 9 4X 3=12 5X 3=15 6X 3=18 7X 3=21 8X 3=24 9X 3=27  
2X 4= 8 3X 4=12 4X 4=16 5X 4=20 6X 4=24 7X 4=28 8X 4=32 9X 4=36  
2X 5=10 3X 5=15 4X 5=20 5X 5=25 6X 5=30 7X 5=35 8X 5=40 9X 5=45  
2X 6=12 3X 6=18 4X 6=24 5X 6=30 6X 6=36 7X 6=42 8X 6=48 9X 6=54  
2X 7=14 3X 7=21 4X 7=28 5X 7=35 6X 7=42 7X 7=49 8X 7=56 9X 7=63  
2X 8=16 3X 8=24 4X 8=32 5X 8=40 6X 8=48 7X 8=56 8X 8=64 9X 8=72  
2X 9=18 3X 9=27 4X 9=36 5X 9=45 6X 9=54 7X 9=63 8X 9=72 9X 9=81
```


[예제모음 30] 구구단을 파일에 출력

```
01 #include <stdio.h>
02
03 int main()
04 {
05     FILE *wfp;          ----파일 포인터와 변수를 선언한다.
06     int i, k;
07
08     wfp=fopen("c:\\temp\\gugu.txt", "w");  ----쓰기 모드로 파일을 연다.
09
10     for(i = 2; i <= 9; i++)              ----첫 줄에 단 제목을 출력한다.
11         fprintf(wfp, " #제%d단# ", i);
12
13     fprintf(wfp, "\n\n");                ----줄 넘김을 출력한다.
14
15     for (i = 1; i <= 9; i++)
16     {
17         for (k = 2; k <= 9; k++)
18         {
19             fprintf(wfp, "%2dX%2d= %2d ", k, i, k*i);
20         }
```

----반복문을 돌면서 출력되는 구구단을
'gugu.txt' 파일에 저장한다.

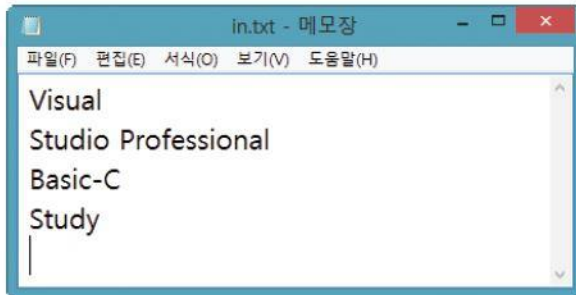
[예제모음 30] 구구단을 파일에 출력

```
21     fprintf(wfp, "\n");
22 }
23
24 fclose (wfp);
25 }
```

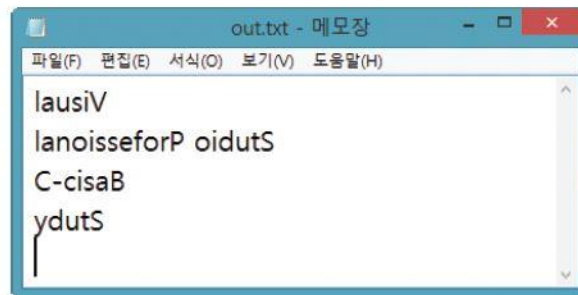
[예제 모음 31] 파일에서 읽어온 문자열을 파일에 반대 순서로 출력

예제 설명 미리 만들어둔 'in.txt' 파일의 내용을 읽어와 'out.txt' 파일에 쓰는데 각 행의 문자를 반대 순서로 저장하는 프로그램이다(반드시 'in.txt' 파일의 마지막 행에서 **Enter**를 누르고 저장한다).

실행 결과



```
in.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
Visual
Studio Professional
Basic-C
Study
|
```



```
out.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
lausiv
lanoisseforP oidutS
C-cisaB
ydutS
|
```

[예제모음 31] 파일에서 읽어온 문자열을 파일에 반대 순서로 출력

```
01 #include <stdio.h>
02 #include <string.h>
03
04 int main()
05 {
06     FILE *rfp, *wfp;          ----파일 포인터를 선언한다.
07     char str1[200], str2[200]; ----입력 문자열, 출력 문자열, 변수를 선언한다.
08     int size, i;
09
10     rfp=fopen("c:\\temp\\in.txt", "r"); ----입력 파일과 출력 파일을 연다.
11     wfp=fopen("c:\\temp\\out.txt", "w");
12
13     while(1)                  -----무한 루프이다.
14     {
15         fgets(str1, 200, rfp); ----입력 파일의 문자열을 읽는다.
16
17         if (feof(rfp))        -----입력 파일의 끝이면 종료한다.
18             break;
19
20         size = strlen(str1);
21         for(i=size-1; i>=0; i- -) ----'문자열 길이 -1'만큼 반복하면 입력 문자열과
22             str2[size-1-i] = str1[i-1]; 출력 문자열의 위치를 바꾼다.
```

[예제모음 31] 파일에서 읽어온 문자열을 파일에 반대 순서로 출력

23

24 str2[size-1] = '\\0';

----출력 문자열의 맨 끝에 널 문자를 추가한다

25 fputs(str2, wfp);

-----출력 문자열을 출력 파일에 쓰고 줄 바꿈을 한다.

26 fputs("\\n", wfp);

27 }

28

29 fclose(rfp);

30 fclose(wfp);

31 }

[11장 요약]

1 표준 입출력 함수

❶ 키보드로 입력하는 것을 '표준 입력'이라 하며 표준 입력 함수에는 scanf(), gets(),

getchar() 등이 있다.

❷ 모니터로 출력하는 것을 '표준 출력'이라 하며 표준 출력 함수에는 printf(), puts(), putchar() 등이 있다.

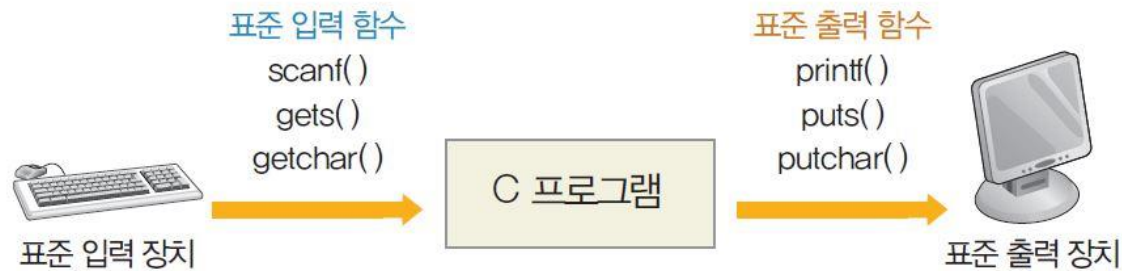


그림 11-1 표준 입출력의 개념

[11장 요약]

2 파일 입출력 함수

❶ 키보드 대신 파일을 통해 입력받는 함수를 '파일 입력 함수'라고 하며 `fscanf()`, `fgets()`, `fgetc()` 등이 있다.

❷ 실행 결과를 모니터 대신 파일에 출력하는 함수를 '파일 출력 함수'라고 하

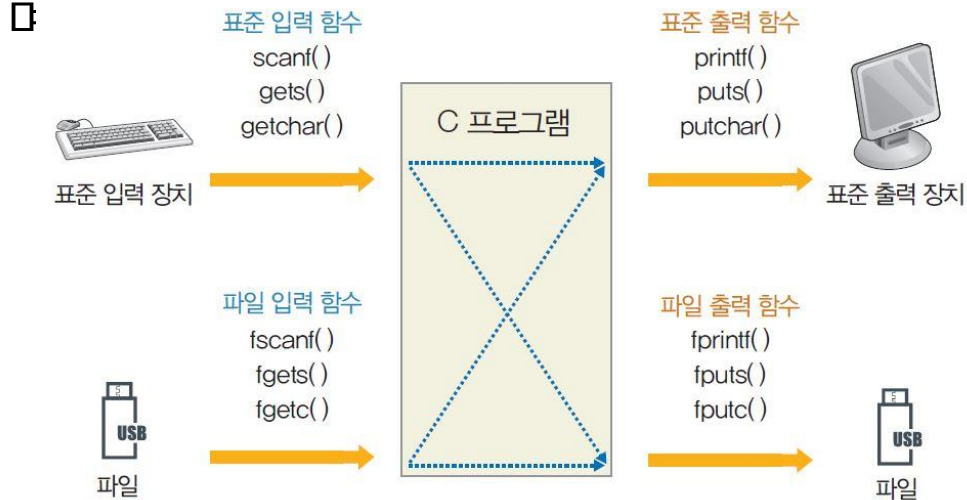


그림 11-4 표준 입출력 함수와 파일 입출력 함수

3

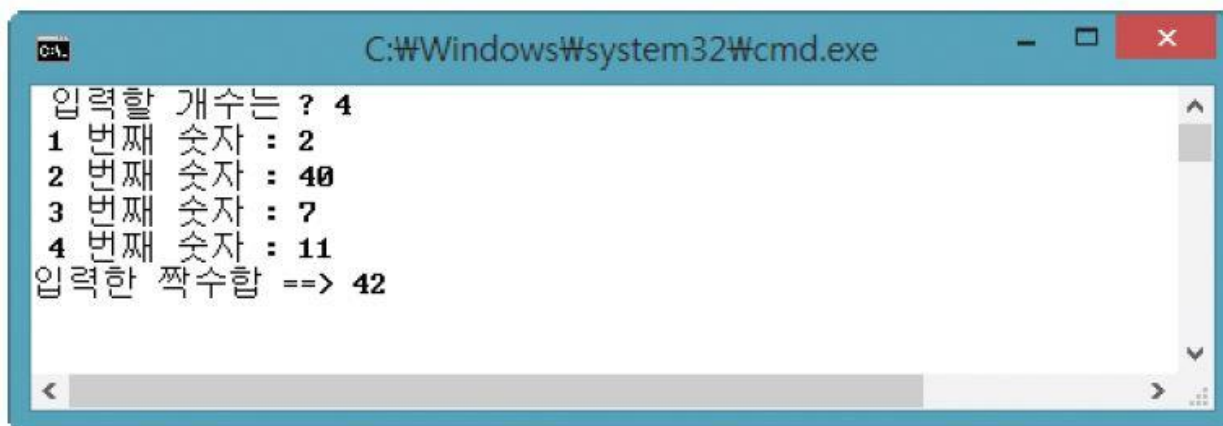


그림 11-5 파일 입출력의 기본 과정

[예제모음 32] 여러 숫자 중 짝수만 더하기

예제 설명 사용자가 입력한 여러 숫자 중에서 짝수의 합계를 출력하는 프로그램이다([응용 12-3] 활용).

실행 결과



```
C:\Windows\system32\cmd.exe
입력할 개수는 ? 4
1 번째 숫자 : 2
2 번째 숫자 : 40
3 번째 숫자 : 7
4 번째 숫자 : 11
입력한 짝수합 ==> 53
```


[예제모음 32] 여러 숫자 중 짝수만 더하기

```
01 #include <stdio.h>
02 #include <malloc.h>
03
04 int main()
05 {
06     int* p;          ----정수형 포인터 변수를 선언한다.
07     int i, hap=0;
08     int cnt;
09
10     printf(" 입력할 개수는 ? ");
11     scanf("%d", &cnt); ----사용자가 입력할 숫자의 개수를 입력한다.
12     p = (int*) malloc(sizeof(int) * cnt); ----입력한 개수(cnt)에 따라 메모리를 확보한다.
13
14     for(i=0; i<cnt; i++)
15     {
16         printf(" %d 번째 숫자 : ", i+1);
17         scanf("%d", p+i);
18     }
19
```

----cnt만큼 배열에 숫자를 입력한다.

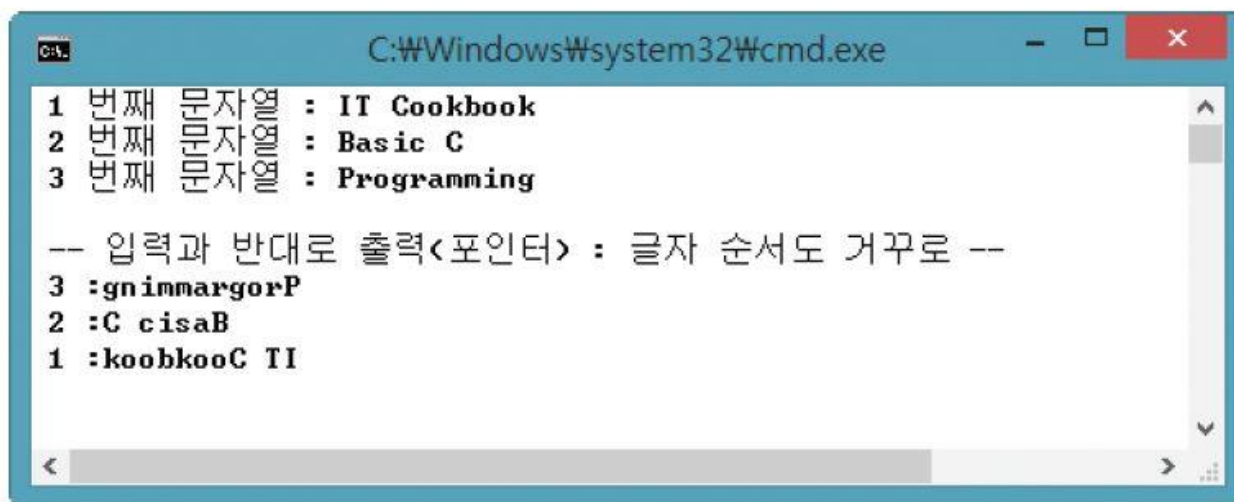
[예제모음 32] 여러 숫자 중 짝수만 더하기

```
20  for(i=0; i<cnt; i++)
21  {
22      if (p[i] % 2 == 0)      ----짝수일 때만 값을 누적한다.
23          hap = hap + p[i];
24  }
25
26  printf("입력한 짝수합 = > %d\n", hap);    ----합계를 출력한다.
27
28  free(p);    ----메모리를 해제한다.
29 }
```

[예제모음 33] 입력된 문자열을 반대 순서로 출력

예제 설명 [응용 12-7]과 유사한데, 입력한 순서의 반대로 그리고 각 행의 문자를 반대 순서로 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

1 번째 문자열 : IT Cookbook
2 번째 문자열 : Basic C
3 번째 문자열 : Programming

-- 입력과 반대로 출력<포인터> : 글자 순서도 거꾸로 --
3 :gnimmargorP
2 :C cisaB
1 :koobkooC TI
```

[예제모음 33] 입력된 문자열을 반대 순서로 출력

```
01 #include <stdio.h>
02 #include <malloc.h>
03 #include <string.h>
04
05 int main()
06 {
07     char* p[3];          ---포인터 배열을 세 칸 선언한다.
08     char imsi[100];      ---입력값을 저장할 임시 공간을 마련한다.
09     int i, k, size;
10
11     for (i=0; i<3; i++)  ---20행까지 세 번 반복한다.
12     {
13         printf(" %d 번째 문자열 : ", i+1);
14         gets(imsi);      ---임시 공간에 문자열을 입력한다.
15
16         size = strlen(imsi); ---입력한 문자열의 길이를 계산한다.
17         p[i] = (char*) malloc( (sizeof(char) * size) + 1 ); ---'입력한 길이 +1' 크기의 메모리를 확보한다.
18
19         strcpy(p[i], imsi); ---확보된 메모리에 입력한 문자열을 복사한다.
20     }
```

[예제모음 33] 입력된 문자열을 반대 순서로 출력

```
21
22  printf("\n -- 입력과 반대로 출력(포인터) : 글자 순서도 거꾸로 --\n");
23  for(i=2; i>=0; i--) ----31행까지 세 번 반복한다.
24  {
25      size = strlen(p[i]); ----문자열의 길이를 체크한다.
26      imsi[size] = '\0'; ----문자열의 끝부분에 널 문자를 입력한다.
27      for(k=size-1; k>=0; k--) ----'문자열 길이 -1'만큼 반복하며 p[i]와 imsi의
28          imsi[size-1-k] = p[i][k]; ----문자열 위치를 바꾼다.
29
30      printf(" %d :%s\n", i+1, imsi); ----imsi 배열에 저장된 문자열을 출력한다.
31  }
32
33  for(i=0; i<3; i++) ----할당했던 메모리 3개를 운영체제에 반납한다.
34      free(p[i]);
35 }
```

[12장 요약]

1 메모리 할당 함수

- 필요에 따라 할당되는 메모리의 크기가 다르면 `malloc()` 함수를 사용하여 메모리 확보

포인터 변수 = (포인터 변수의 데이터형*) `malloc`(포인터 변수의 데이터형 크기 × 필요한 크기)

- 동적으로 할당한 메모리의 사용이 모두 끝나면 `free()` 함수로 사용한 공간 해제해야 함.
- 포인터 변수 = (포인터 변수의 데이터형*) `realloc`(기본 포인터, 포인터 변수의 데이터형 크기 × 필요한 크기);
- 이미 할당한 메모리의 크기를 변경할 때는 `realloc()` 함수 사용함.

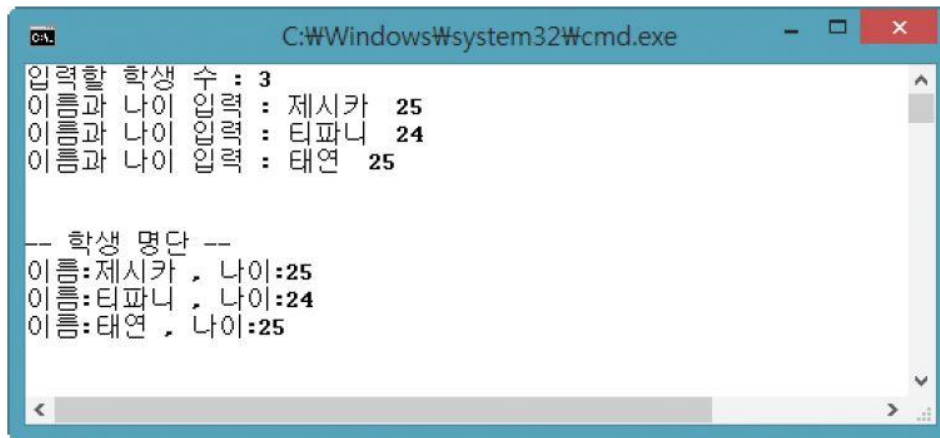
2 포인터 배열

- 포인터 변수를 배열로 선언한 것
- 메모리 낭비 없이 여러 행의 문자열을 처리하는 데 유용함.

[예제모음 34] 구조체 포인터를 활용한 학생 관리

예제 설명 구조체 포인터와 메모리 할당 함수를 이용하여 학생의 이름과 나이를 입력받아 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

입력할 학생 수 : 3
이름과 나이 입력 : 제시카 25
이름과 나이 입력 : 티파니 24
이름과 나이 입력 : 태연 25

-- 학생 명단 --
이름:제시카 , 나이:25
이름:티파니 , 나이:24
이름:태연 , 나이:25
```

[예제모음 34] 구조체 포인터를 활용한 학생 관리

```
01 #include <stdio.h>
02 #include <malloc.h>
03
04 int main()
05 {
06     struct student{
07         char name[10];
08         int age;
09     };
10
11     struct student *s;
12
13     int cnt, i;
14
15     printf("입력할 학생 수 : ");
16     scanf("%d", &cnt);
17
18     s = (struct student *) malloc((sizeof(struct student)) * cnt);
19
```

----구조체형을 선언한다.

----구조체 포인터 변수를 선언한다.

----관리할 학생 수를 입력한다.

----학생 수만큼 메모리를 할당한다.

[예제모음 34] 구조체 포인터를 활용한 학생 관리

```
20  for (i=0; i<cnt; i++)
21  {
22      printf("이름과 나이 입력 : ");
23      scanf("%s %d", s[i].name, &s[i].age);
24  }
```

----학생 수만큼 반복하며 이름과 나이를 입력한다.

```
25
26  printf("\n\n-- 학생 명단 --\n");
```

```
27  for (i=0; i<cnt; i++)
28      printf("이름:%s , 나이:%d \n", s[i].name, s[i].age);
```

----학생 수만큼 반복하며 이름과 나이를 출력한다.

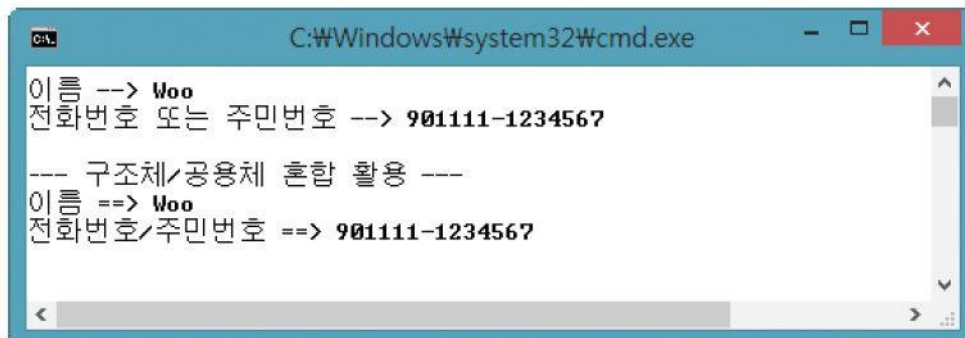
```
29
30  free(s);
31 }
```

----메모리를 해제한다.

[예제모음 35] 구조체와 공용체의 혼합

예제 설명 사용자 이름을 입력하고 전화번호 또는 주민번호 중 한 가지만 입력하는 프로그램으로, 전화번호와 주민번호는 공용체로 동일한 메모리를 차지함으로써 공간을 절약한다. 구조체와 공용체를 혼합해서 사용해보자.

실행 결과



```
C:\Windows\system32\cmd.exe

이름 --> Woo
전화번호 또는 주민번호 --> 901111-1234567

--- 구조체/공용체 혼합 활용 ---
이름 ==> Woo
전화번호/주민번호 ==> 901111-1234567
```

[예제모음 35] 구조체와 공용체의 혼합

```
01 #include <stdio.h>
02
03 int main()
04 {
05     typedef struct _person {
06         char name[10];
07         union _id {
08             char phone[15];
09             char jumin[15];
10         } id;
11     } person;
12
13     person p1;
14
15     printf("이름 --> ");
16     scanf("%s", p1.name);
17     printf("전화번호 또는 주민번호 --> ");
18     scanf("%s", p1.id.jumin);
19
20     printf("\n--- 구조체/공용체 혼합 활용 ---\n");
21     printf("이름 == > %s\n", p1.name);
22     printf("전화번호/주민번호 == > %s\n", p1.id.phone);
23 }
```

----구조체형을 정의할 때 그 내부에 공용체를 사용한다.

-----공용체 변수 id를 선언한다.

----p1의 크기는 총 25바이트이다.

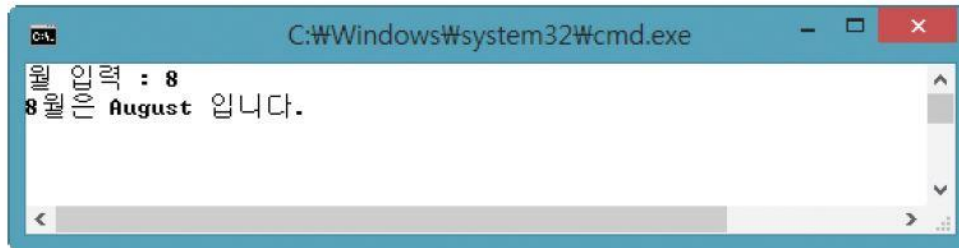
----p1 안 공용체 id의 멤버 변수 jumin에 입력받는다.
이 부분은 phone으로 해도 동일하다.

----p1 안 공용체 id의 멤버 변수 phone을 출력한다.
이 부분은 jumin으로 해도 동일하다.

[예제모음 36] 열거형을 활용한 월 이름 출력

예제 설명 열거형을 활용해서 입력된 월의 이름을 출력하는 프로그램이다.

실행 결과



[예제모음 36] 열거형을 활용한 월 이름 출력

```
01 #include <stdio.h>
```

```
02
```

```
03 int main()
```

```
04 {
```

```
05     enum month {
```

```
06         January=1, February, March, April,
```

```
07         May, June, July, August,
```

```
08         September, October, November, December
```

```
09     };
```

```
10
```

```
11     enum month mm;
```

```
12
```

```
13     printf("월 입력 : ");
```

```
14     scanf("%d", &mm);
```

```
15
```

```
16     switch (mm)
```

```
17     {
```

```
18         case January : printf("%d월은 January 입니다.", mm); break;
```

```
19         case February : printf("%d월은 February 입니다.", mm); break;
```

```
20         case March : printf("%d월은 March 입니다.", mm); break;
```

```
21         case April : printf("%d월은 April 입니다.", mm); break;
```

----열거형을 선언한다(1부터 시작).

----열거형 변수 mm을 선언한다.

----값을 입력한다.

----입력값에 따라 해당 월을 출력한다.

[예제모음 36] 열거형을 활용한 월 이름 출력

```
22     case May : printf("%d월은 May 입니다.", mm); break;
23     case June : printf("%d월은 June 입니다.", mm); break;
24     case July : printf("%d월은 July 입니다.", mm); break;
25     case August : printf("%d월은 August 입니다.", mm); break;
26     case September : printf("%d월은 September 입니다.", mm); break;
27     case October : printf("%d월은 October 입니다.", mm); break;
28     case November : printf("%d월은 November 입니다.", mm); break;
29     case December : printf("%d월은 December 입니다.", mm); break;
30     default : printf("잘못 입력했군요.");
31 }
32 printf("\n\n");
33 }
```

[13장 요약]

1 구조체

- 서로 다른 데이터 형식의 변수를 하나로 묶어 놓은 것임.

▼ 구조체의 기본 형식

```
struct 구조체형_이름 {  
    데이터_형식  멤버_변수_1;  
    데이터_형식  멤버_변수_2;  
    :  
};  
  
struct 구조체형_이름 구조체_변수;
```

▼ 구조체의 예

```
struct bibim {  
    int a;  
    float b;  
    char c;  
};  
  
struct bibim b1;  
b1.a = 10;
```

- 구조체 변수 초기화 방법의 예

```
struct student {  
    char name[10];  
    int kor;  
    int eng;  
    float avg;  
};  
  
struct student s = {"LeeSan", 90, 80};
```

- 구조체 변수도 일반 변수처럼 배열과 포인터로 사용할 수 있다.

[13장 요약]

2 공용체

- 하나의 공간을 서로 다른 두 변수가 같이 사용하는 것임.

▼ 공용체의 기본 형식

```
union 공용체형_이름 {  
    데이터_형식 멤버_변수_1;  
    데이터_형식 멤버_변수_2;  
    :  
};  
  
union 공용체형_이름 공용체_변수;
```

▼ 공용체의 예

```
union student {  
    int tot;  
    char grade;  
};  
  
union student u;  
u.tot = 300;
```

3 열거형

- 변수가 가질 수 있는 범위가 정해져 있을 때 숫자보다 쉽게 파악할 수 있는 문자나 단어로 표현하는 자료형

▼ 열거형의 기본 형식

```
enum 열거형_이름 {  
    기호_1;  
    기호_2;  
    :  
};  
  
enum 열거형_이름 열거형_변수;
```

▼ 열거형의 예

```
enum week {  
    sun, mon, tue, wed, thu, fri, sat  
};  
  
enum week ww;  
ww = sat;
```