Practical C Programming

A Manual for Professors

Practical C Programming A Manual for Professors

저자 김원선
인쇄일 2006년 12월 5일
발행일 2006년 12월 10일

발행처 이한출판사
발행인 한길만
편집 이재덕
디자인 인지영

주소 경기도 고양시 일산구 장항동 750 삼성스위트 501호
전화 031)924-1563, 팩스 031)924-0362

등록번호 제1-1520호
등록일자 1993. 5. 17

가격 비매품

홈페이지 http://www.ehan.co.kr
이메일 webmaster@ehan.co.kr

연습문제 해답

CHAPTER

01 C 언어 시작하기

문제 1-1

흐름도는 프로그램을 작성하고 실행시키는 과정에 대한 전체적인 구조를 보여주고 있다.

1. 어떠한 목적으로 프로그램을 작성하려고 하는지 문제 파악을 하는 것이 가장 먼저 해야 할 일이면서도 가장 중요한 일이다. 문제를 파악해야 문제해결을 할 수 있기 때문이다.
2. 문제 파악이 되면 문제 해결을 위한 프로그램 코드작성을 하게 된다. 프로그래밍 언어는 선택 사항일 뿐이며 여러 가지 언어로 표현이 가능하다.
3. 프로그램 코드 작성이 끝나면 원하는 결과가 나오는지 테스트를 수행하는데 만약 컴파일 과정에서 오류가 있다면 그것은 문법(Syntax) 오류이며, 링크가 수행된 후 프로그램을 실행했을 때 원하는 결과가 도출되지 않는다면 그것은 논리(Logic) 오류가 된다. 따라서 오류가 발생되면 원하는 결과가 도출될 때까지 위 과정을 반복해야 한다.
4. 원하는 결과가 도출 된다면 프로그램 작성은 끝나게 된다.

문제 1-2

프로그램의 실행결과는 다음과 같다.

결과

```
main() start.
Even : 2
Odd : 1
2 + 1 = 3
main() end.
```

#define문은 매크로 상수를 정의할 수 있으며Even 매크로상수는 “2”로 , Odd 매크로상수는 “1”로 전처리 단계에서 문자열 대치가 일어난다. 따라서 프로그램에서 매크로 상수를 사용 시 대체된 값을 사용한다. 또한 매크로 상수가 값을 갖는다면 연산에 참여할 수 있다.

문제 1-3

다음과 같은 결과를 구하기 위한 [ ] 안에 알맞은 코드는 다음과 같다.

"제 이름은" 을 출력할 함수의 내용이 func2()에 있으므로 func1()보다 func2()를 먼저 호출하고 "안재은 입니다"를 출력할 func1()을 나중에 호출 하여야 한다.

```
int main(void)
{
    printf("안녕하세요!!! \n");
    func2();
    func1();
return 0;
}
```

문제 1-4

②, ③, ④ 는 식별자로 사용할 수 없다.

② 7Count // 숫자가 첫 문자로 올 수 없다

③ func 1 // 식별자명에 공백을 포함할 수 없다

④ printf // 예약어는 식별자명으로 사용할 수 없다

CHAPTER

02 기본 자료형과 변수

문제 2-1

number_input( ) 함수에서 정수 7을 반환하여 main( ) 함수에서 그 수의 곱을 출력하는 프로그램은 다음과 같다.

```
1.  #include <stdio.h>
2.
3.  int number_input(void);
4.
5.  int main(void)
6.  {
7.      int num;
8.
9.      num = number_input();
10.     printf("%d * %d : %d \n", num, num, num*num);
11.
12.     return 0;
13. }
14.
15. int number_input(void)
16. {
17.    int number = 7;
18.
19.    return number ;
20. }
```

9행을 보면number_input( ) 함수를 호출하면 정수를 반환한다는 것을 알 수 있다. 15행의 number_input( ) 함수를 보면 number 변수에 7을 할당하고 19행에서 7을 반환한다. 이때 9행의 num 변수가 7을 대입 받게 될 것이다. 따라서 10행에서 num 변수에 저장된 값의 곱을 출력하고 있다.

문제 2-2

main( ) 함수의 age 변수는 10을 출력하려고 없다. 잘못된 곳이 무엇인가?

① 함수의 선언문이 없다
myfunc() 함수는 선언문이 없어도 정수를 반환하므로 문제가 발생하지 않는다.

② 13행에서 함수정의가 잘못되었다.
myfunc( ) 함수에서 인수전달이 없으면 ( ) 안에 void를 선언할 것을 권장하지만 생략할 수 있다.

③ myfunc( ) 함수에서 값을 반환하지 않는다.
7행을 보면 myfunc( ) 함수에서 정수값을 반환한다는 것을 알 수 있다. 따라서 myfunc( ) 함수에서 return 문과 함께 my_age 값을 반환해야 했다.

④ 7행은 문법적으로 바르지 않다. → 7행의 문법은 바르다.

정답 : ③

```
....
myfunc()
{
     int my_age;
    my_age=10;          ──→  값을 반환해야 한다.

return my_age ;
}
```

결과

```
age : 10
```

문제 2-3

다음을 출력하기 위한 완성된 프로그램이다.

결과

```
ch  is  B
count  is  200
pi   is  3.140000
```

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      char ch;
6.      int count;
7.      float pi;
8.
9.      ch = 'B'
10.     count = 200
11.     pi = 3.14
12.
13.     printf("ch  is  %c \n", ch );
14.     printf("count  is  %d \n", count );
15.     printf("pi   is  %f  \n", pi );
16.
17.     return 0 ;
18. }
```

변수는 선언만으로는 의미 있는 값을 보관하지 않는다. 따라서 변수에 값을 할당하고 사용하여야 한다. 또한 변수의 값을 출력하기 위해 prinf() 함수를 사용하는데 변수의 자료형을 출력할 수 있는 형식지정자를 일치시켜야 한다.

문제 2-4

프로그램의 실행 결과이다.

결과

```
가로: 5, 세로: 7
사각형의 넓이 :  35
사각형의 둘레 :  24
```

main( )함수에서 가로(width) 세로(height)를 대입 받아 각 함수에서 사각형의 넓이와 사각형의 둘레를 구한 값을 부모함수에게 반환하였다.

```
Int  func1( int w, int h )    사각형의 넓이를 구하는 함수
Int  func2( int w, int h )    사각형의 둘레를 구하는 함수
```

CHAPTER

03 자료형 수정자와 형 변환

문제 3-1

정수형 변수 num1=10, num2=3 에 저장된 몫과 나머지를 구하는 프로그램은 다음과 같다.

결과

```
10/3의 몫은 3 이고, 나머지는 0.333333 이다.
```

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      int num1=10, num2=3;
6.
7.      printf("%d/%d의 몫은 %d 이고, 나머지는 %f 이다. \n",
8.  num1, num2, num1/num2,
9.                          num1/(float)num2 - num1/num2 );
10. }
```

8행의 num1/num2 는 두 수의 몫을 구한다. 정수/정수이므로 정수의 몫만 구해진다.

9행은 나머지를 구하는 식이다. 먼저 두 변수 중 하나를 cast 연사자를 사용하여 실수형으로 변환해야 한다. 따라서 다음과 같이 num1/(float)num2 로 실수형 결과 "3.3333.."을 구하였다. 그 값에 num1/num2 의 결과인 몫을 빼면 나머지가 구해진다. 따라서 나머지를 구하기 위해 적절할 형 변환을 처리하고 있음을 보여준다.

문제 3-2

나이를 저자하기 위한 age 변수는 unsignedchar age; 의 선언문이 가장 적절하다.

① char age;

이 변수는 127을 넘으면 저장될 수 없다. 나이는 250을 넘지 않는다는 것을 127을 넘을 수 있다는 의미이므로 이 선언문은 잘못되었다.

② short int age;

이 변수는 32767 까지의 값을 저장하므로 관계없으나 메모리를 2바이트 할당한다. 따라서 1 바이트를 사용할 수 있는 방법을 생각해 보아야 한다.

③ int age;

이 변수는 약 -21억 ~ 21억 까지의 값을 저장하므로 관계없으나 메모리를 4바이트 할당한다. 따라서 1 바이트를 사용할 수 있는 방법을 생각해 보아야 한다.

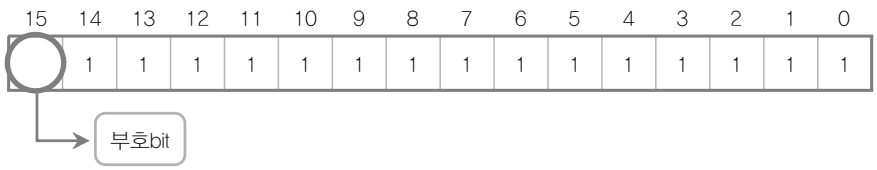
④ unsigned char age;

이 변수는 메모리에 1바이트를 할당하며 unsigned로 인하여 부호비트를 사용하지 않고 8비트 모두 데이터의 표현으로 사용하므로 0~265 까지의 값을 저장할 수 있게된다.

정답 : ④

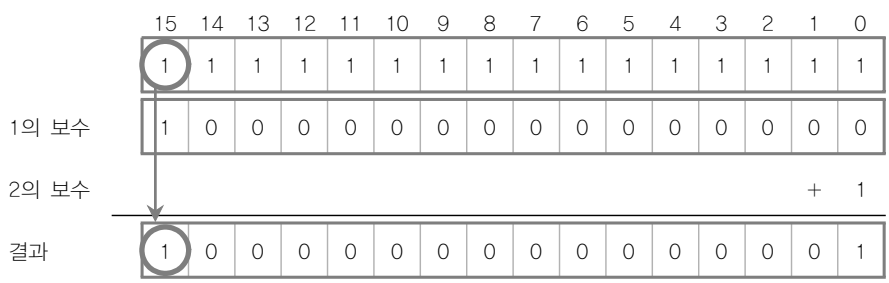
문제 3-3

short int형 변수에 32767을 저장하였다. 다음은 2진수로 표현이다.



■ 설명 1

위 표현에서 최상위 비트가 1이면, 이 수는 -1로 인식된다(2의 보수라 가정). 이유는 음수로 처리되므로 2의 보수를 위하면 -1이 된다.

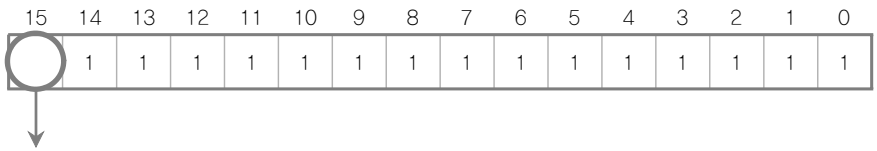


■ 설명 2

부호비트가 0 이면 32767가 된다. 이유는 부호가 0이면 양수이고, 나머지 비트의 값이 십진수 32767이기 때문이다.

■ 설명 3

이것이 unsigned 형으로 선언되었다면 즉, 최상위 비트가 1이 되면 65535가 된다.



unsigned 는 최상위 비트가 부호의 표현이 아닌 값으로 취급된다. 따라서 16비트에 저장된 비트의 십진수는 65535가 되기 때문이다.

문제 3-4

프로그램의 실행 결과이다.

결과

```
현재 시스템의 C 컴파일러의 데이터 유형의 크기 :
signed char :     1 바이트(8 비트)
        값의 범위 : -128 에서  127까지의 값

unsigned char:    1 바이트(8 비트)
        값의 범위 : 0 에서 255까지의 값

signed short int :        2 바이트
        값의 범위 : -32768 에서 32767까지의 값

unsigned short int :      2 바이트
        값의 범위 : 0 에서 65535까지의 값

signed int:       4 바이트
        값의 범위 : -2147483648 에서 2147483647까지의 값
unsigned int:     4 바이트
        값의 범위 : 0 에서 4294967295까지의 값

signed long int :         4 바이트
        값의 범위 : -2147483648 에서 2147483647까지의 값

unsigned long int :       4 바이트
        값의 범위 : 0 에서 4294967295까지의 값

float :  4 바이트
        값의 범위 : 1.175494E-38 에서  3.402823E+38)
        정확도(Precision) : 6 자리
```

```
double:  8 바이트
        값의 범위 :  2.225074E-308 에서 1.797693E+308)
        정확도(Precision) : 15 자리
```

2행의 헤더파일 선언문이다.

#include 〈limits.h〉
이 헤더파일은 C 언어에서 사용하고 있는 자료형 크기 및 자주 사용되는 여러 상수들을 정의해놓은 파일이다. 현재 프로그램은 헤더 파일의 정보 중 파일 자료형과 관련이 있는 상수들을 이용하여 자료형의 범위를 출력하고 있다.

CHAPTER

04 콘솔(Console) 입 · 출력 함수

문제 4-1

반지름을 입력 받아 원의 둘레와 넓이를 구하는 프로그램은 다음과 같다.

```
1.  #include <stdio.h>
2.  #define PI 3.14159
3.
4.  int main(void)
5.  {
6.       int radius;
7.       float cir_around, cir_area;
8.
9.       printf("반지름 ? ");
10.      scanf("%d", &radius);
11.
12.      cir_around =  radius * 2 * PI ;
13.      cir_area =  radius * radius * PI ;
14.
15.      printf("반지름이 %d인 원의 둘레 : %f \n",radius, cir_around );
16.      printf("반지름이 %d인 원의 넓이 : %f \n",radius, cir_area );
17.
18.      return 0 ;
19. }
```

결과

```
반지름 ? 7   // 첫번째 실행
반지름이 7인 원의 둘레 : 43.982262
반지름이 7인 원의 넓이 : 153.937912

반지름 ? 9  // 두 번째 실행
반지름이 9인 원의 둘레 : 56.548618
반지름이 9인 원의 넓이 : 254.468796
```

10행에서 radius 변수에 값을 입력받고 있다. 정수형이므로 scanf() 함수의 "%d" 지정자를 사용하였으며 프로그램을 실행할 때마다 새로운 반지름의 원의 둘레와 넓이를 출력하고 있다.

문제 4-2

다음과 같이 변수들이 선언되어 있다. 입력함수 바르게 표현된 것은 무엇인가?

```
    char ch;
int num;
float f_num;
double d_num;
```

① ch=getchar( );

ch는 문자변수로 선언되었다. getchar( ) 함수는 키도드로부터 한문자를 입력받을 수 있다.

② scanf("%hd", &num);

num 변수는 4바이트 정수형 변수이다. 형식지정자 "%hd"는 short int형 지정자이다. 따라서 문법오류는 아니지만 메모리 할당이 바르지 않으므로 의미없는 값을 저장하게 된다.

③ scanf("%f", f_num);

f_num은 "%f"로 실수를 입력받지만 변수명앞에 "&"인 주소연산자가 없어 값을 메모리에 저장할 수 없다. 이는 컴파일러에 따라 문법오류를 일으키기도 한다.

④ scanf("%Lf", &d_num);

d_num 변수는 8바이트 실수형 변수이다. 형식지정자 "%Lf"는 long double형 지정자이다. 따라서 문법오류는 아니지만 메모리 할당이 바르지 않으므로 의미없는 값을 저장하게 된다.

정답 : ①

문제 4-3

학생의 점수를 입력하여 총점과 평균을 구하여 출력하는 프로그램이다. 다음의 ☐ 안을 완성하라?

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.       int kor, eng, mat, sum ;
6.
7.       printf("점수를 입력하세요... \n");
8.       printf("국어, 영어, 수학 ? ");
9.       scanf("%d%d%d", &kor, &eng, &mat);
10.
11.      sum=kor + eng + mat ;
12.      printf("진달래: %d, %d, %d, 총점: %d, 평균: %f \n",
13. kor, eng, mat, sum, sum/(float)3);
14.
15.  return 0 ;
16. }9.
```

점수를 입력하기 위해 여러 개의 scanf()를 사용할 수도 있지만, 하나의 함수로 형식지정자를 여러 개 두어 여러 개의 값을 입력받을 수 있다.

문제 4-4

프로그램의 실행 결과이다.

결과

```
가로 ? 5
세로 ? 6
높이 ? 7
직육면체의 부피: 210.000000
```

CHAPTER

05 C 언어 연산자(Operators)

문제 5-1

연산자의 우선순위에 따라 괄호를 먼저 계산하고, 좌→우 방향으로 연산을 처리하게 된다.

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      int x, y, z;
6.
7.      printf("세 수를 입력하세요 ? ");
8.      scanf("%d%d%d", &x, &y, &z);
9.
10.     printf("(x+y) * (y-z) * (z/x) + (x%%z): %d \n",
11.                      (x+y) * (y-z) * (z/x) + (x%z));
12.
13.     return 0 ;
14. }
```

결과

```
세 수를 입력하세요 ? 1 2 3
(x+y) * (y-z) * (z/x) + (x%z): -8
```

문제 5-2

입력 받은 두 수는 정수형 자료이므로 몫과 나머지를 따로 구하기 위해서는 변수 중 하나를 실수형으로 형 변환하여 처리할 수 있다.

정답 : ④

결과

```
두 수를 입력하세요 ? 10 3
10 / 3 = 3,  0.333333
```

문제 5-3

간단한 조건을 묻기 위해 ?(3항) 연산자를 사용할 수 있다.

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      int x, y, z, max;
6.
7.      printf("세 수를 입력하세요 ? ");
8.      scanf("%d%d%d", &x, &y, &z);
9.
10.     max=(x > y ? x : y );
11.     max=(max > z ? max : z );
12.
13.     printf("가장 큰 값 : %d \n", max);
14.
15.     return 0 ;
16. }
```

결과

```
세 수를 입력하세요 ? 10 50 20
가장 큰 값 : 50
```

문제 5-4

프로그램의 실행 결과이다.

결과

```
1. x= 0
2. x= 1
3. x=2, y=2, z=2
```

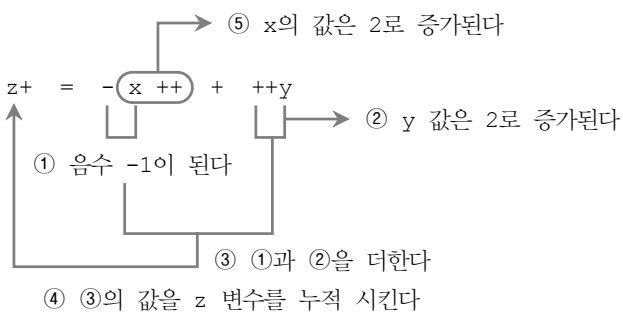
7행의 식이다. x=-3 * 4 % -6 / 5; 이 연산식은 우선순위에 의해 차례대로 좌 → 우로 계산되어 변수 x 에 저장된다.

x=2, y=1, z=0; 이다. 이 값을 12행에서 다음과 같이 처리한다. x=x && y || z ;

l 식은 x는 참, y도 참 이다. 따라서 두 논리값을 && 연산을 하면 참이다. 이 값과 z 을 || 연산을 하게 되는데 이미 앞의 피연산자가 참이므로 뒤의 피연산자의 값과 관계없이 참을 반환하게 된다.

x=y=z=1; 이다.

16행의 **z+= -x++ + ++y** 은 다음과 같이 처리된다.



CHAPTER

06 제어문

문제 6-1

키보드로부터 5개의 수를 입력 받아, 그 수 중 가장 큰 값을 찾아 출력하는 프로그램은 다음과 같다.

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.     int i, num, max=0;
6.
7.     for(i=1 i<6 i++)
8.     {
9.        printf("%d, 값 입력 ? ", i);
10.       scanf("%d", &num);
11.       if(max<num)
12.          max=num;
13.    }
14.    printf("max : %d \n", max);
15. }
```

변수 선언문이다.

```
5.     int i, num, max=0;
```

num은 값을 입력받기 위한 변수이고, max는 큰 값을 저장할 변수이다. max는 처음 입력되는 값부터 대소 비교를 하므로 가장 작은 값인 0을 초기화 하였다. 양수만 입력된다고 가정하였다.

7행에서 5번을 반복하고 있다. 하나의 문장이 아니므로 코드 블록을 주었으며 10행의 scanf( ) 가 값을 입력받아 num 변수에 저장한다.

```
10.    scanf("%d", &num);
11.    if(max<num)
12.        max=num;
```

max의 값이 입력된 num보다 작다면 max에 큰 값인 num을 대입하고, 그렇지 않으면 max가 크다는 의미이므로 max는 변경되지 않는다. 이렇게 5번을 비교하게 되면 max변수는 5개의 입력값 중 가장 큰 값을 저장하게 된다.

문제 6-2

어떤 수가 소수(prime number)인지를 결정하는 완성된 프로그램은 다음과 같다.

```
1.  # include <stdio.h>
2.
3.  int main(void)
4.  {
5.     int num,i, prime;
6.
7.     printf("Enter the number ? ");
8.     scanf("%d", &num);
9.
10.    prime=1;
11.    for(i=2; i<=num/2; i++)
12.       if((num%i)==0)
13.       {
14.           prime=0;
15.           break;
16.       }
17.
18.    if(prime==1)
19.       printf("The number is prime. \n");
20.    else
21.       printf("The number is not prime. \n");
22.
23.    return 0;
24. }
```

위 코드 내용 중 소수인지 체크하는 루틴이다.

```
11.    for(i=2; i<=num/2; i++)
12.       if((num%i)==0)
13.       {
14.           prime=0;
15.           break;
16.       }
```

소수란 1 과 자기 자신으로만 나누어 지는 수이다. 따라서 11행의 반복문은 2부터 시작하여 입력된 값을 2로 나눈 값까지 반복한다. 이때 입력한 값을 2로 나눈 이후의 값은 나누어 지는 수가 아니므로 반복에서 제외하였다.

12행의 if문은 입력한 값을 2로(2부터 입력된 값을 2로 나눈 값까지)나눈 나머지가 0인지 조건검사를 한다. 만약 참이라면 나누어 지는 수가 자기 자신 이외에도 있기 때문에 소수가 아니다. 따라서 prime 변수에 0을 할당하고 반복문을 빠져나간다. 이미 소수가 아니므로 나머지 반복을 계속할 필요가 없기 때문이다. 만약 12행의 조건이 거짓이라면 반복은 계속하게 된다. 즉 반복문은 둘 중 하나를 만족하면 종료하게 되는데 소수가 아니어서 break에 의해 빠져나오는 경우와 for문을 끝까지 반복하여 반복조건이 거짓이여서 빠져나오는 경우, 즉 소수인 경우이다.

```
18. if(prime==1)
19.        printf("The number is prime. \n");
20.    else
21.        printf("The number is not prime. \n");
```

반복문을 시작하기 전 prime 변수는 1이 대입된다. 그러나 소수가 아닌 경우 break를 만나기전 prime변수에 0을 대입하게 된다. 즉 반복문이 종료된 후에 prime변수가 여전히 1이라면 소수였다는 것이고, 1이 아니라면 소수가 아니라는 것을 알 수 있다.

문제 6-3

다음 중 바른 제어문의 표현은 어느것인가?

정답 : ③

① switch 문은 실수형 상수를 을 비교할 수 없다.

```
float f;

scanf("%f", &f);
switch(f)
{   case 10.5 : printf( "…");
    ……
}
```

② num은 정수형 변수이다. 식에 다른 자료형과 비교될 수 있지만 단, 형 변환이 가능한 자료형에 대해서만 가능하다. 문자열은 정수형과 비교될 수 없다.

```
num=10;
if(num=="C lang")
 printf("…");
else
 printf("…");
```

③ num 변수는 문자 'A'와 비교될 수 있다. char형 변수는 비교(연산)시 int형으로 자동 형 변환되기 때문이다.

```
num=10;
if(num=='A')
 printf("…");
else
 printf("…");
```

④ break 문의 사용은 오류를 일으킨다. break는 switch문이나 반복문 안에서만 사용 가능하다.

```
j=1;
if(j<=10)
{
   if(j%3==0)
      break;
   j=j+1;
}
```

문제 6-4

프로그램의 실행 결과이다.

이 프로그램은 다음은 2g, 3g, 5g 세 종류의 추가 각각 10개씩 있다. 이들 추를 하나 이상 이용하여 추의 전체 무게가 81g 인 추의 조합을 모두 찾아내는 프로그램이다.

결과

```
2g:  2, 3g:   9, 5g: 10
2g:  3, 3g:  10, 5g:  9
2g:  5, 3g:   7, 5g: 10
2g:  6, 3g:   8, 5g:  9
2g:  7, 3g:   9, 5g:  8
2g:  8, 3g:   5, 5g: 10
2g:  8, 3g:  10, 5g:  7
2g:  9, 3g:   6, 5g:  9
2g: 10, 3g:   7, 5g:  8
81g 인 경우의 수 : 9
```

프로그램의 반복문은 다음과 같다.

```
7.   for(i=1; i<=10;i++)
8.         for(j=1;j<=10;j++)
9.            for(k=1;k<=10;k++)
10.           {
11.               tot=(i*2)+(j*3)+(k*5);   // 각 추의 합 계산
12.               if(tot==81)    // 추의 합이 81 이냐
13.               {
14.                  count++;
15.                  printf("2g:%3d, 3g: %3d, 5g:%3d \n", i, j,
                                        k);    //각 g별 추의 수 출력
16.               }
17.           }
```

변수 i는 2g의 추, j는 3g의 추, k는 5g의 추 라 가정하고 각각 10번씩 반복문을 돌리고 있다. 예를 들어 i가 1 이고 j가 1일 때 k 는 10번 반복하면서 추의 무게가 81g인지를 비교하고 있다. 이처럼 각 반복에서 매번 추의 무게를 구하여 81g일 때 각 추의 수를 출력한다.

문제 6-5

프로그램의 실행 결과이다.

이 프로그램은 한 문자를 입력 받아 그 문자의 이진수를 출력하는 내용이다. 이를 해결하기 위해 5장에서 학습한 비트연산자 & 연산과 left 이동 연산자를 이용하였다.

```
13.  if(ch==27)
14.         break; //ch변수가 27이면 반복문 while 탈출로 25행을 빠져나간다.
```

13행은 ch변수에 저장된 값이 27이면 반복문을 탈출하여 프로그램을 종료하게 된다. 십진수 27은 ESC키를 입력했을 때의 값이다.

```
17.       for(i=128   i>0   i>>=1)
                              // ch변수에 저장된 2진수를 출력하는 반복문이다.
```

17행의 반복문 for문은 i 가 128부터 0보다 클 때까지 반복하게 되는데 증감식은 i〉〉=1 로 반복할 때마다 오른쪽 이동 시프트 연산을 1 bit씩 이동하고 있다. 이는 이전 값을 2로 나눈 값과 같다. 따라서 i 변수의 값은 128, 64, 32, 16, 8, 4, 2, 1 이 차례대로 저장되며 반복될 것이다.

```
19.          if(ch & i)
20.               printf("1 ");
21.            else
22.               printf("0 ");
```

19행의 조건식이다.

if(ch & I)

if문은 ( )안의 식이 참이면 다음 명령을, 거짓이면 else 이후의 명령을 실행시킨다. 이때 ch와 i 변수를 & 연산을 하게 되고 그 결과가 참이면 printf(“1 ”); , 거짓이면 printf(“0 ”); 을 호출하게 된다.

다음은 반복문의 전체적인 흐름을 다음과 같다

ch='A' → ‘A’ 문자가 저장되었다고 가정하자.

■ 반복이 첫 번째 실행된 변수의 구조

이 두 변수를 비트 연산 &를 하게 되면 결과값은 0 이다.

■ **반복이 두 번째 실행된 변수의 구조**

이 두 변수를 비트 연산 &를 하게 되면 결과값은 64이며, if조건이 참이 된다.

반복문에 의해 비교가 일어날 때마다 단 하나의 비트만 1로 되도록 i 의 값을 조정한다. i 변수의 최상위 비트가 1인 경우 128이므로, 이값을 시작으로 반복될 때마다 i 변수의 값은 반으로 나눈값을 보관하게 된다. 이것은 i 내에서 다음 비트의 위치를 1로 만들고 나머지는 모두 0이 되게 한다. 그러므로 반복될 때마다 ch에 있는 한 비트를 & 연산자로 검사하게 되는 것이다. 결국 반복문이 끝나게 되면 ch변수의 이진수 값을 출력하게 될 것이다.

문제 6-6

다음 프로그램은 커서 이동 키를 입력한 경우 어떤 이동 키를 선택했는지 확인하는 프로그램이다. 실행결과는 다음과 같다.

■ (windows o/s 기본)

결과

키보드로부터 입력 받는 기본 키 외에 확장키들은 운영체제에 따라 확장키 값이 달라진다. 이 예제에서는 windows o/s를 기준으로 하였다.

다음은 운영체제에 의해 약속된 이동 확장키 값이다.(16진수)

위 방향키 : 0x48

아래 방향키 : 0x450

왼쪽(left) 방향키 : 0x4b

오른쪽(right) 방향키 : 0x4d

다음은 중요 코드이다.

```
11. ch=getch();   // 한 문자 입력
12. if(ch==8)   // backspace 키를 누르면 종료한다
13.           exit(0);
14.
15.        switch(ch)
16.        {
17.           case 0x4b :  /* 좌측 움직임 */
18.                    printf("left... \n");
19.                    break;
20.           case 0x48 :  /*위로 움직임 */
21.                    printf("up... \n");
22.                    break;
23.           case 0x4d :  /* 우로 움직 */
24.                    printf("rihgt... \n");
25.                    break;
26.           case 0x50 :  /* 아래로 */
27.          printf("down... \n");
28.                    break;
29.          }
30.
```

CHAPTER

07 기억 클래스(Memory Class)

문제 7-1

어떤 수의 거듭제곱을 구하기 위한 power( ) 함수의 완성된 내용이다.

```
20. int power(int m, int e)
21. {
22.     int tmp;
23.
24.     tmp=1;
25.     for(;e>0;e--)
26.         tmp=tmp*m;
27.
28.     return tmp;
29. }
```

m과 e는 여전히 전역변수를 사용하지만 power( ) 함수에 인수를 전달하고 있다. 이때 power( )의 매개변수 m과 e는 지역변수로 전역변수에는 영향을 주지 않는다. 이러한 이유로 전역변수를 가지고 호출될 수도 있지만, 전역변수의 값이 아닌 다른 임의의 값을 가지고 power( )함수는 늘 어디서나 호출될 수 있다.

결과

```
3의5승 : 243
2의3승 : 8
4의3승 : 64
```

문제 7-2

선언문에 문제가 있는 것은 무엇인가?

정답 : ④

```
1.  #include <stdio.h>
2.  #define MAX 5
3.
4.  int main(void)
5.  {
6.      int a=7;
7.      int b=MAX;          ──────→ ①
8.      int c=a;            ──────→ ②
9.      {
10.         int d=10;       ──────→ ③
11.
12.         printf("%d, %d, %d %d \n", a, b, c, d);
13.         int e=20;       ──────→ ④
14.         printf("%d, %d, %d %d, %d \n", a, b, c, d, e);
15.     }
16.     return 0;
17. }
```

① 매크로 상수는 변수 선언에 값으로 사용될 수 있다.

② 지역변수는 선언 시점에서 값이 결정되는 어떤 식에 의해 초기화 될 수 있다.

③ 블록 안의 시작 부분에 지역변수를 선언할 수 있다.

④ 지역변수의 선언은 그 어떤 실행문 보다 먼저 선언되어야 한다. 즉 12행보다 먼저 선언되어야 한다.

문제 7-3

두 수를 입력 받아 두 수의 합을 구하여 출력하는 프로그램은 다음과 같다.

```
1.  #include <stdio.h>
2.
3.  void sum(int num1, int num2);
4.
5.  int main(void)
6.  {
7.     int num1, num2, cn ;
8.
9.     while(1)
10.    {
11.       printf("두 수를 입력하세요 ? ");
12.       cn=scanf("%d%d", &num1, &num2);
13.       if(cn==0 || cn==EOF)
14.           break;
15.       sum(num1, num2);
16.    }
17.    return 0;
18. }
19.
20. void sum(int num1, int num2)
21. {
22.    static int tot=0;
23.
24.    printf("%d + %d = %d \n", num1, num2, num1+num2);
25.    tot=tot+num1+num2;
26.    printf("지금까지의 합 : %d\n", tot);
27. }
```

결과

```
두 수를 입력하세요 ? 10 5
10 + 5 = 15
지금까지의 합 : 15
두 수를 입력하세요 ? 10 20
10 + 20 = 30
지금까지의 합 : 45
두 수를 입력하세요 ? 30 60
30 + 60 = 90
지금까지의 합 : 135
두 수를 입력하세요 ?
```

Ctrl+Z 또는 Ctrl+D를 입력한다.(입력종료)

sum( ) 함수에서 두 수의 합을 출력하고 있다. 이때 두 수의 합을 계속 누적한 값을 사용해야 하므로 누적변수는 함수 호출시 마다 값을 유지해야 한다. 따라서 tot는 정적변수로 선언되었다.

문제 7-4

피보나치 수열을 출력하는 프로그램이다.

피보나치(Fibonacci) 수열 알고리즘은 자연수를 입력받아 그 수보다 작은 수(혹은 같은수)까지의 피보나치 수열을 구한다.

피보나치 수열의 정의는 F(1) = 0, F(2) = 1, F(n) = F(n−1) + F(n−2) (n〉=3)이다. 따라서 제1항과 제2항은 예외이기 때문에 if문을 이용해서 정의를 해주었고, 3항부터는 일반항의 식을 따르도록 하였다.

```
void fibo_func(int num)
{
   int n1=0, n2=1, n3, i;

   if(num==1)
      printf("%4d,", n2);
   else
      printf("%4d,%4d,", n1, n2);

   for(i=2; i<=num;i++)
   {
      n3=n1+n2;
      printf("%4d,", n3);
      n1=n2;
      n2=n3;
   }
   printf("\n");
}
```

3항 이후의 값을 구하는 과정에 대해서 간단히 설명하면, 구하고자 하는 항보다 1작은 항과 2작은 항을 서로 더한다. 즉 3항은 0과 1의 합이 된다. 다음 동작은 구해진 값은 1작은 항으로 하고, 바로 전에 1작은 항이었던 값을 2작은 항으로 해서 위의 과정을 반복한다.

결과

```
출력할 피보나치 수 입력 ?(0보다 큰 값) 5
   0,   1,   1,   2,   3,   5,

출력할 피보나치 수 입력 ?(0보다 큰 값) 7
   0,   1,   1,   2,   3,   5,   8,  13,
```

CHAPTER

08 배열과 문자열

문제 8-1

str 배열의 문자열을 꺼꾸로 출력하기 위한 프로그램이다.

```
char str[20]="kingdom"
```

결과

```
m o d n g i n k
```

```
1.  #include <stdio.h>
2.  #include <string.h>
3.
4.  int main(void)
5.  {
6.     char str[20]="kingdom";
7.     int i;
8.
9.     i=strlen(str)-1;    // 문자수 반환
10.    for( ; str[i]; i--)    // 마지막 문자부터 조건이 참이냐 ?
11.       printf("%c ", str[i]);
12.    printf("\n");
13.
14.    return 0 ;
15. }
```

9행의 i 는 strlen( ) 함수를 통해 문자수에 – 1 의 값을 저장한다. 이유는 배열은 0부터 시작하며 만약 strlen( )이 5를 반환하였다면 배열은 0부터 4에 문자열이 저장되어 있기 때문이다. 따라서 반복문은 초기치를 생략하였으며 str[i]가 참이면 출력한다. 그리고 반복하기 위해 1씩 감소하고 있다. 이렇게 하면 문자열을 끝에서 부터 제어할 수 있게 된다.

문제 8-2

다음 단락 중 바른 것은 무엇인가?

정답 : ④

① 배열은 한번에 대입될 수 없고 요소마다 하나하나 대입하여야 한다.

```
int num1[5]={1,2,3,4,5};
int num2[5];

num2=num1;
```

② 배열범위를 넘어서 값을 저장하므로 잘못된 표현이다. 위 반복문은 10보다 작을 때 까지만 반복되어야 한다.

```
int count[10], i;

for(i=0; i<50;i++)
   count[i]=i;
```

③ str1은 1차원 문자 배열이다. 따라서 문자배열에 문자열을 입력하게 위해서는 문자배열 변수명을 첨자없이 호출하여야 하다. 배열 변수명은 그 배열의 선두주소이고 gets( )함수는 입력된 문자열을 인수의 주소에 저장하기 때문이다. str[0]는 문자열의 주소가 아닌 0열이며 하나의 원소인 값을 의미한다.

```
char str1[15];

printf("input string ?");
gets(str[0]);
```

④ 문자열을 추가하기 위해여 strcat( ) 함수를 호출할 수 있다.

```
char str1[15]="king", str2[15]="queen";

strcat(str1, str2);
```

문제 8-3

문자열을 입력받아 공백, 컴마, 마침표가 각각 몇 개인지 출력하는 프로그램은 다음과 같다.

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.     char str[40];
6.     int i, n1=0, n2=0, n3=0;
7.
8.     printf("input string ? ");
9.     gets(str);
10.    printf("str : %s \n", str);
11.
12.    for(i=0;str[i];i++) //저장된 문자가 참일 때(있을때) 까지 반복한다
13.       switch(str[i])
14.       {
15.       case 32 : n1++;    // 공백문자이냐 ?
16.                  break;
17.       case 44 : n2++;   // 컴마이냐 ?
18.                  break;
19.       case 46 : n3++;   // 마침표이냐 ?
20.                  break;
21.       }
22.
23.     printf("공백문자:%d, 컴마:%d, 마침표:%d \n", n1, n2, n3);
24.
25.     return 0 ;
26. }
```

12행에서 i 는0 부터 str[i] 번째가 참일 때 까지 1씩 증가하면서 반복하고 있다. 이는 문자열의 처음부터 끝까지 문자를 하나하나 제어하려는 의도이다. str[i] 번째에 저장된 문자가 공백인지, 컴마인지, 마침표인지 확인하기 위해 13행에서 switch문으로 다중 선택을 하고 있다. 부록 ASCII 코드표에 보면 공백문자는 32, 컴마는 44, 마침표는 46이다. 따라서 각 case 문에서 해당 상수를 만나면 누적하여 해당 문자수를 카운트 하게 될 것이다.

결과

```
input string ? a b,c.d공백,.ef,공백.p<Enter>
str : a b,c.d ,.ef, .p
공백문자:2, 컴마:3, 마침표:3
```

문제 8-4

전치행렬로 변환하는 프로그램이다.

전치행렬로 변환하는 코드를 살펴보자. 전치행렬은 행렬의 열 요소와 행 요소를 상호 교환하는 것이다. 따라서 원본의 행의 데이터를 사본의 열의 데이터 위치에 대입하면 이를 해결할 수 있다.

```
18. //a 배열을 전치행렬로 변환하여 b 배열에 저장
19.  for(i=0;i<3; i++)
20.       for(j=0;j<3;j++)
21.          b[j][i] = a[i][j];
```

결과

```
a 배열 리스트
   13    4   30
   34    2    5
    7   15   27
전치행열 b 배열 리스트
   13   34    7
    4    2   15
   30    5   27
```

문제 8-5

정수형 배열 num[7][7]을 1 부터 49 까지 대입한 후 다음과 같은 결과를 출력하는 프로그램이다.

결과

```
1
    8    9
   15   16   17
   22   23   24   25
   29   30   31   32   33
   36   37   38   39   40   41
   43   44   45   46   47   48   49
```

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.     int num[7][7], i, j, k=0;
6.
7.     for(i=0;i<7; i++)
8.        for(j=0;j<7;j++)
9.        {
10.          k++;    // 1싹 증가
11.          num[i][j]=k;    // 값 대입
12.       }
13.
14.    for(i=0;i<7; i++)
15.    {
16.       for(j=0;j<=i;j++)   // 안쪽 반복은 행의 변수 값 만큼 반복한다
17.          printf("%5d", num[i][j]);
18.       printf("\n");
19.    }
20.
21.    return 0 ;
22. }
```

이처럼 2차원 배열은 행과 열의 공통점을 잘 파악하여 반복을 처리할 때 정확하게 표현할 수 있어야 한다.

CHAPTER

09 포인터(pointer)

문제 9-1

키보드로부터 문자열을 입력받아 이를 반대로(끝에서부터) 출력하는 프로그램이다.

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.     char str[40], *ptr;
6.     int i
7.
8.     printf("input string ? ");
9.     gets(str);
10.
11.    ptr = str + strlen(str)-1;  //문자열의 끝 주소 얻어 낸다
12.    while(ptr>=str) //ptr 이 str의 선두주소보다 크거나 같으면 출력한다
13.       printf("%c ", *ptr--); //현재 문자를 출력하고 주소는 -1 이된다
14.    printf("\n");
15.
16.    return 0 ;
17. }a
```

중요 코드이다.

```
ptr = str + strlen(str)-1;  //문자열의 끝 주소 얻어 낸다
```

먼저 배열의 문자열의 길이를 얻어낸다. 이 값에 -1을 한 후str(배열의 선두주소) 주소와 이전에 얻어낸 문자열의 길이를 더하면 문자열의 끝 주소가 된다.

```
while(ptr>=str)
      printf("%c ", *ptr--);
```

ptr 이 str의 선두주소보다 크거나 같으면 아직 출력할 문자가 남아 있다는 것이므로 printf()에서 현재 문자를 출력하고 주소는 -1 이 된다. 즉 이전 주소로 이동하면서 거꾸로 출력하게 될 것이다.

결과

```
input string ? kingdom
m o d g n i k
```

문제 9-2

다음은 포인터를 잘못 사용하는 경우이다. 무엇이 문제인지 설명하라?

① 포인터 변수 p는 그 어떤 주소도 할당되어 있지 않다. 따라서 p를 이용하여 값을 치환할 수 없다.

```
int *p;

*p=200;
```

② 포인터 변수는 자신이 가르킬 객체의 자료형과 같아야 한다. 이유는 포인터 변수는 참조할 선두번지만 저장할 뿐 몇 바이트를 접근할 것인가에 대한 정보는 갖지 못한다. 따라서 포인터변수는 간접 접근시 자신의 자료형 만큼 접근한다. 이런 이유 때문에 포인터 변수와 객체의 자료형이 다르면 다른 메모리 영역을 가르키게 되며 메모리를 잘못 참조하는 오류의 원인이 되므로 주의해야 한다.

```
short int *p, q=100;

p=&q;
*p=300;
```

③ 포인터 연산은 정수형식만 가능하다. 실수형식은 사용될 수 없다. 이유는 메모리 주소는 정수이며, 항상 양의 정수로만 이루어져 있다.

```
int *p, q=100;

p=&q;
p=p+1.5;
```

④ gets( )함수는 인수로 문자열을 저장할 메모리 주소가 와야 한다. 그러나 포인터 변수 p는 주소를 할당받지 못하였으므로 잘못된 함수 호출이다.

```
char *p;

printf("input string ?");
gets(p);
```

문제 9-3

2차원 포인터 배열을 확인하는 프로그램의 결과는 다음과 같다.

결과

```
========================
     제품명       단 가
========================
     프린터,     5000
     키보드,     1000
     마우스,      500
       HDD,     7000
     모니터,     6000

검색할 제품명 ?(검색종료:end) 마우스
제품 : 마우스, 단가 : 500

검색할 제품명 ?(검색종료:end) 프린터
제품 : 프린터, 단가 : 5000

검색할 제품명 ?(검색종료:end) 광마우스
제품명을 다시 이력하세요.

검색할 제품명 ?(검색종료:end) end
```

product 포인터 배열의 메모리 구조는 다음과 같다.

```
char *product[][2]={{"프린터","5000"},
                    {"키보드","1000"},
                    {"마우스","500"},
                    {"HDD","7000"},
                    {"모니터","6000"},
                    {"",""}};
```

메모리 구조를 확인하여 보고 결과를 분석하여 보자.

product[6][2]

	[0]열	[1]열
[0] 행	40001058	40001060
[1] 행	40001068	40001070
[2] 행	40001078	40001080
[3] 행	40001088	40001090
[4] 행	40001098	400010a0
[5] 행	400010a8	400010b0

프린터	0x40001058
키보드	0x40001068
마우스	0x40001078
HDD	0x40001088
모니터	0x40001098
'₩0'	0x400010a8
....	
5000	0x40001060
1000	0x40001070
500	0x40001080
7000	0x40001090
6000	0x400010a0
'₩0'	0x400010b0

문제 9-4

다음 프로그램은 배열과 포인터를 이용하여 두 개의 문자열을 복사하거나, 문자열을 추가하는 완성된 프로그램 이다.

```
1.  #include <stdio.h>
2.
3.  void mystrcpy(char *to, char *from);
4.  void mystrcat(char *to, char *from);
5.
6.  int main(void)
7.  {
8.      char str1[40]="Hello ", str2[40]="안녕!!!";
9.
10.     printf("before -> str1:%s, str2:%s \n", str1, str2);
11.     mystrcpy(str1, str2);
12.     printf("after -> str1:%s, str2:%s \n", str1, str2);
13.
14.     printf("\nbefore -> str1:%s \n", str1);
15.     mystrcat(str1, "Hello world!!!");
16.     printf("after -> str1:%s \n", str1);
17.
```

```
18.      return 0 ;
19.  }
20.
21.  void mystrcpy(char *to, char *from)    // 문자열 복사
22.  {
23.     while(*from)
24.       *to++=*from++;
25.
26.     *to='\0';
27.  }
28.
29.  void mystrcat(char *to, char *from)    // 문자열 추가
30.  {
31.     while(*to)
32.        to++;
33.
34.      while(*from)
35.         *to++=*from++;
36.
37.      *to='\0';
38.  }
```

21행의 문자열 복사함수가 이해가 된다면 추가도 그래 어려운 작업이 아니다. 문자열 추가는 문자열을 추가할 to 포인터가 문자열의 끝에 먼저 이동된 후 위의 문자열 복사를 그대로 수행하면 된다.

다음은 to 포인터 변수가 null이 아닐 때 까지 주소만 이동시킨다.

```
while(*to)
      to++;
```

from 포인터 변수가 null이 아닐 때 까지 from을 to로 문자를 대입하고 주소는 다음 주소로 이동된다.

```
while(*from)
 *to++=*from++;
```

위 반복문이 종료되면 to에 문자열 대입이 종료되었다. 따라서 '₩0'를 주어 문자열의 종료임을 알린다.

```
*to='\0';
```

CHAPTER

10 C 언어 핵심! 함수

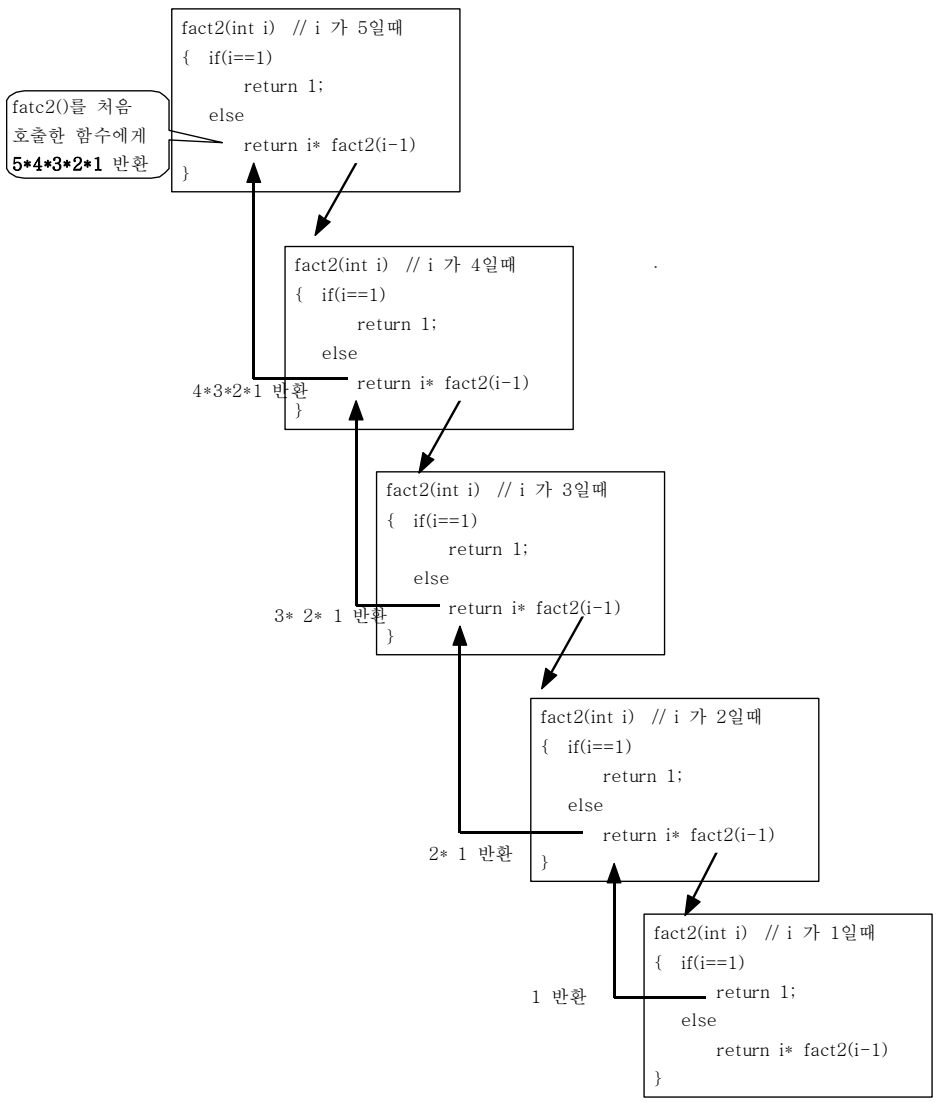
문제 10-1

어떤 수의 순열(factorial)을 구하는 프로그램이다. 하나의 수를 입력 받아 그 수의 순열을 구하는 fact2( ) 함수를 완성하였다.

```
#include <stdio.h>

int main(void)
{
   int num, result;

   printf("input number ? ");
   scanf("%d", &num);

   result=fact1(num);
   printf("fact1() -> %d! : %d \n", num , result);

   result=fact2(num);
   printf("fact2() -> %d! : %d \n", num , result);

   return 0;
}

int fact1(int num)
{
   int i, x=1;

   for(i=num;i>=1; i--)
      x=x*i;

   return x;
}

int fact2(int i)
{
   if(i==1)
      return 1;
   else
      return i * fact2(i-1);
}
```

30행에서 i 가 1이 아니면 i * 변수의 값을 반환하기 전에 자신의 함수를 -1을 주어 되 부르고 있다. 즉 i가 1이 될 때까지 함수 호출을 계속한다. i가 1이면 더 이상 함수는 호출하지 않고 반환을 시작하게 되는데, 함수의 최종 반환 값은 5*4*3*2*1이 된다.

다음 그림은 함수의 호출 단계를 보여준다.



결과

```
input number ? 5
fact1() -> 5! : 120
fact2() -> 5! : 120
```

문제 10-2

다음은 잘못된 표현을 사용하는 경우이다. 무엇이 문제인지 설명하라?

① 함수 func의 매개변수 중 첫번째 매개변수는 포인터 형으로 선언되어야 한다. 즉 함수의 형식이 일치하지 않는다.

```
void func(int x, int y, int z)

main()
{
    int num=10;

    func(&num, 20, 30);
    ….
}

void func(int x, int y, int z)
{
    ….
}
```

② argc 의 개수가1이라는 것은 실행파일명으로만 실행되는 경우이며, 다른 명령어 라인 인수는 입력되지 않은 경우이다. 명령어 라인 인수가 없더라도 파일명은 기본적으로 argv[0] 열에 파일명을 저장한 문자열 주소를 저장하고 있다. 따라서 문자열 출력인 "%s" 지정자를 주어야 한다.

```
void main(int argc, char *argv[])
{
    if(argc==1)
         printf("%d \n", argv[0]);
    …..
}
```

③ 이 함수는 프로그램이 작동하지 않을 때까지 자신의 함수를 반복적으로 호출한다. 이것은 순환 호출이 일어나는 것을 막는 조건이 없기 때문이다.

```
void func(void)
{
int i;

printf("func1(). \n");

for(i=0;i<10;i++)
        func();
}
```

문제 10-3

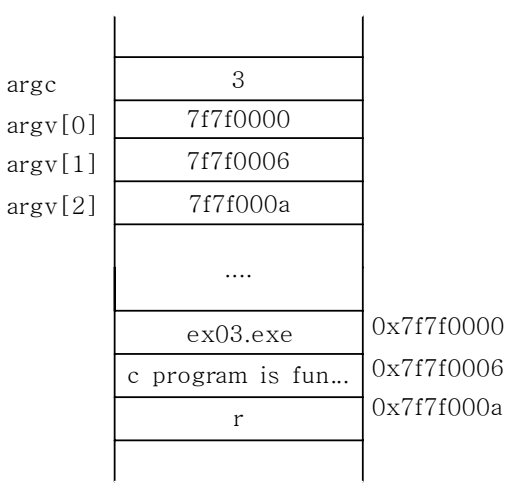
첫번째 인수의 문자열 중 두 번째 입력된 문자가 몇 번째 위치에 몇 개 있는지 출력하는 프로그램의 결과는 다음과 같다.

```
C:\> ex03.exe "c program is fun… " r <Enter>
```

결과

```
r 문자는 : 4, 7,  위치에 2 개 있다.
```

입력된 인수를 저장하는 메모리 구조이다. 구조를 먼저 이해하고 코드를 보자.



다음은 중요 소스 코드이다.

```
16. while(*argv[1])
        // 보관하고 있는 번지의 한 문자, 즉 처음이라면 문자 'c' 이다.
17. {
18.       position++;   // 몇 번째 인지를 가리키기 위한 count
19.       if(*argv[1]==*argv[2])   // 두 주소가 가리키는 문자가 같냐 ?
20.       {
21.          printf("%d, ", position);   // 위치 출력
22.          cn++;   // 개수 누적
23.       }
24.       argv[1]++;   // 다음 문자를 가리키기 위하여 주소 이동,
                       // 만약 argv[1] 보관된 처음 주소가 5000 이라면
                          이 명령으로 5001 로 변경된다.
25. }
26.
```

메모리 구조를 이해하고 소스코드를 보면 그리 어려운 내용이 아니다. 각자 분석하길 바란다.

또 다른 실행 결과를 보여준다.

```
C:\> ex03.exe "c program is fun…"  .  <Enter>
```

결과

```
. 문자는 : 17, 18, 19,  위치에 3 개 있다.
```

만약 첫번째 문자열 중 두번째 문자를 찾지 못할 경우의 출력이다.

```
C:\> ex03.exe "c program is fun…"  x <Enter>
```

결과

```
x 문자는 :  존재하지 않는다.
```

CHAPTER

11 구조체와 공용체

문제 11-1

학생의 정보를 구조체로 정의하야입출력 하는 프로그램은 다음과 같다.

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.     struct student {                  // 구조체 선언
6.             char st_no[20];
7.             char subject[20];
8.             char name[20];
9.             char entry;
10.            char telno[15];
11.     }st1 ;
12.
13.
14.    printf("학번 ? ");     // 구조체 데이터 입력
15.    gets(st1.st_no);
16.    printf("학과 ? ");
17.    gets(st1.subject);
18.    printf("성명 ? ");
19.    gets(st1.name);
20.    printf("등록여부(1:등록, 0:미등록) ? ");
21.    scanf("%d%*c", &st1.entry);         // 리턴값 제거
22.    printf("전화번호 ? ");
23.    gets(st1.telno);
24.
25.     // 구조체 데이터 출력
26.    printf("%s, %s, %s, %s, %s \n", st1.st_no, st1.subject,
27.        st1.name, st1.entry==1 ? "등록" : "미등록", st1.telno);
28.
29.    return 0;
30. }
```

프로그램 코드 자체는 크게 어려운 점이 없다. 따라서 구조체 선언문의 멤버 자료형에 따라 입력, 출력함수를 잘 정의 하였는지 생각해 보아야 할 것이다.

문제 11-2

다음은 잘못된 표현을 사용하는 경우이다. 무엇이 문제인지 설명하라?

①

```
struct A {
            int age;
            char name[20]
    } st;

.....
age=20;
```

구조체 멤버는 멤버명을 직접 사용할 수 없으며 구조체 변수와 함께 사용해야 한다. 만약 age 멤버에 접근 한다면 다음과 같다.

```
st.age = 20;
```

②

```
void main(void)
{
     struct A {
     int age;
     char name[20]
     } st, *ptr;

    ptr=st ;
….
}
```

구조체 변수 st는 변수이다. 따라서 구조체 포인터 변수 ptr에 변수의 주소를 대입하려면 다음과 같다.

```
ptr=&st;
```

③

```
void main(void)
{
     struct A {
     int age;
     char name[20]
     } st, *ptr;

    ptr=&st ;

    ptr.age=25;
.....
}
```

구조체 변수로 멤버에 접근하기 위해서는 구조체 포인터 연산자를 사용해야 한다.

```
ptr -> age =25;
```

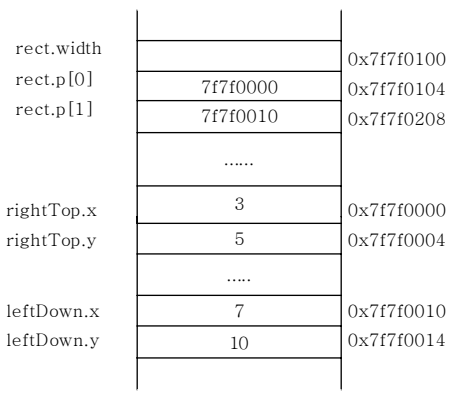
문제 11-3

다음 프로그램은 중첩된 구조체를 이용하여 사각형의 넓이를 구하는 프로그램이다. 잘못된 부분은 [ ] 의 내용이며 이와 같이 수정되여야 한다.

```
1.  #include <stdio.h>
2.
3.  typedef struct point {
4.                          int x;
5.                          int y;
6.                  }POINT;
7.
8.  struct rectangle {
9.                          int width;
10.                         POINT *p[2];
11.                 };
12.
13. int main(void)
14. {
15.     struct rectangle rect;      // rectangle은 구조체 테그명이다.
16.     POINT rightTop={3,5};    // POINT는 자료형 이름이다.
17.     POINT leftDown={7,10};
18.
```

```
19.     rect.p[0]=&rightTop;
20.     rect.p[1]=&leftDown;
21.
22.     rect.width=(rect.p[1]->x - rect.p[0]->x) ~*(rect.p[1]->y
                                                  - rect.p[0]->y);
23.     printf("가로:%d, 세로:%d, 사각형의 넓이 : %d 이다. \n",
24.  (rect.p[1]->x - rect.p[0]->x), (rect.p[1]->y -
                                    rect.p[0]->y),rect.width);
25.  // 구조체 멤버는 "."으로 접근
26.     return 0 ;
27.  }
```

15행부터 20행까지 rect, rightTop, leftDown의 구조체 변수가 메모리에 할당된 구조이다.



rect 의 width 멤버는 정수형이며, p 배열은 구조체포인터 배열이다. 따라서 p 배열의 원소에 저장된 주소를 참조하려면 구조체 포인터 연산자 "->"를 사용하여 접근하여야 한다.

결과

```
가로:4, 세로:5, 사각형의 넓이 : 20 이다.
```

문제 11-3

다음 프로그램은 구조체와 나열형을 활용하는 프로그램의 결과는 다음과 같다.

결과

```
1, 부서 ?(종료:end) 영업부
   성명 ? 홍길동
   직급 => 1:사장,2:부장,3:과장,4:대리,5:사원 ? 1
   월급 ? 2570000
2, 부서 ?(종료:end) 인사부
   성명 ? 성춘향
   직급 => 1:사장,2:부장,3:과장,4:대리,5:사원 ? 3
   월급 ? 1900000
3, 부서 ?(종료:end) 영업부
   성명 ? 이몽룡
   직급 => 1:사장,2:부장,3:과장,4:대리,5:사원 ? 5
   월급 ? 1700000
4, 부서 ?(종료:end) end

-----------------------------------
   부서   성명    직급    실수령액
-----------------------------------
영업부, 홍길동,  사장,    2620000
인사부, 성춘향,  과장,    1970000
영업부, 이몽룡,  사원,    1770000
```

이 프로그램에서 구조체 멤버로 나열형 상수를 사용하였다. 나열형 상수는 이전에 학습한 내용으로 나열형 변수는 enum에서 정의한 나열형 정수 값을 저장하여 사용하는 목적으로 이용된다.

다음 나열형 상수 정의를 보자.

```
enum B {manager1=1,manager2,manager3,employee1,employee2}
                                                    position;
```

position 나열형 변수는 1 부터 5까지의 정수(상수)를 저장할 것이다.

emps[i].position=manager2 // position 멤버는 정수 2를 저장한다.

입력된 데이터를 출력할 때 상수 값은 이해하기 어려우므로 직위 정보를 갖고있는 position 멤버를 삼항 연산자를 이용하여 문자열로출력하였다.

만약 position 멤버에 2가 저장되어 있다면 “부장”을 출력하게 된다.

```
printf("%s \n", emps[i].position==manager1?"사장":
         emps[i].position==manager2?"부장":
         emps[i].position==manager3?"과장":
         emps[i].position==employee1?"대리":"사원" );
```

CHAPTER

12 파일 입 · 출력

문제 12-1

알파벳 대문자 A~Z, 소문자 a~z 까지 파일에 출력하는 완성된 프로그램이다. 추가한 부분은 굵은문자로 표시하였다.

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.
4.  int main(void)
5.  {
6.     FILE *fp;
7.     int i;
8.     char ch='A';
9.
10.    fp=fopen("ex01.dat", "wt");   // "wt"로 스트림과 연결한다
11.    if(fp==NULL)
12.    {
13.       printf("file open error. \n");
14.       exit(1);
15.    }
16.
17.    for(i=0;i<26;i++)
18.    {
19. fputc(ch, fp);      // 대문자 저장
20. fputc(ch+32, fp);  // 소문자 저장
21.       fputc(32, fp);      // 공백문자 저장
22. ch++;
23.    }
24.    printf("파일 저장. \n")
25.
26.    fclose(fp);   //스트림을 닫는다
27.
28.    return 0;
29. }
```

"ex01.dat" 파일을 열어보면 내용은 다음과 같다.

```
Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo Pp Qq Rr Ss Tt Uu
Vv Ww Xx Zz
```

문제 12-2

학생정보를 입력 받아 "st1.dat"파일에 저장한 후 그 파일을 temp 구조체 변수에 읽어 들여 출력하는 프로그램은 다음과 같다.

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.
4.  int main(void)
5.  {
6.     struct student {
7.            char st_no[20];
8.            char subject[20];
9.            char name[20];
10.           char entry;
11.           char telno[15];
12.     }st1 , temp;
13.
14.    FILE *fp;
15.
16. printf("학번 ? ");
17.    gets(st1.st_no);
18. printf("학과 ? ");
19. gets(st1.subject);
20. printf("성명 ? ");
21. gets(st1.name);
22. printf("등록여부(1:등록, 0:미등록) ? ");
23. scanf("%d%*c", &st1.entry);
24.    printf("전화번호 ? ");
25. gets(st1.telno);
26.
27.    fp=fopen("st1.dat", "wb");
28.    if(fp==NULL)
29.    {
30.       printf("file open error. \n");
31.       exit(1);
32.    }
33.    fwrite(&st1, sizeof(st1), 1, fp);
34.
35.    fp=freopen("st1.dat", "rb", fp);
36.    fread(&temp, sizeof(temp), 1, fp);
37.
38. printf("temp 구조제 출력.\n");
39. printf("%s, %s, %s, %s, %s \n", temp.st_no, temp.subject,
40. temp.name, temp.entry==1 ? "등록" : "미등록", temp.telno);
41.
42.    return 0;
43. }
```

문제 12-3

다음은 잘못된 표현을 사용하는 경우이다. 무엇이 문제인지 설명하라?

①

```
#include <stdio.h>
....
FILE  fp;
fp=fopen("st.dat", "wt");
...
```

FILE은 구조체 자료형이다. 파일 포인터 fp는 *fp 로 선언되어야 한다.

②

```
#include <stdio.h>
void main(void)
{
     FILE  *fp;
     struct A {
         int age;
         char name[20]
     } st,;
  fp=fopen("st.dat", "wt");
  fwrite(&st, sizeof(st), 1, fp);
  ….
```

fwrite(), fread() 함수는 이진 입출력 함수이다. 따라서 fopen() 함수의 모드는 다음과 같아야 한다.

```
fp=fopen("st.dat", "wb");
```

③

```
#include <stdio.h>
void main(void)
{
      FILE  *fp;
      struct A {
```

```
        int age;
        char name[20]
    } st,;
fp=fopen("st.dat", "wb");
fwrite(st, sizeof(strcuct A), 1, fp);
```

구조체 선언에서 st 는 구조체 변수이다. 따라서 fwrite( ) 함수의 첫번째 인수는 저장할 메모리의 선두주소인 포인터이여야 한다. 즉 fwrite( ) 는 다음과 같아야 한다.

```
fwrite(&st, sizeof(strcuct A), 1, fp);
```

문제 12-4

다음은 작성된 텍스트 파일을 문자열 단위로 읽어 표준 출력에 출력시키는 프로그램의 실행 결과는 다음과 같다.

결과

```
진달래 서울시 성동구 옥수동 100번지 011-111-1111
이몽룡 서울시 강남구 역삼동 120번지 016-222-2222
까꿍이 속초시 대포동 바다 2번지      017-333-3333
```

데이터 파일은 처음부터 끝까지 행(80 바이트) 단위로 차례대로 읽어들이다. 다음 코드를 보자.

```
16. while(1)
17. {
18. if(!fgets(line, 80, fp))
            break;
19. printf("%s", line);
20. }
```

스트림의 현재 위치에서 80바이트를 읽어 line 배열에 저장한다. 이때 한 행이 80바이트가 안되더라도 현재 줄의 끝에 줄 빠꿈 기호가 저장되어 있다. 따라서 한 행의 문자열이 80문자가 안된다면 한 줄씩 읽을 수 밖에 없다.

CHAPTER

13 동적 메모리 할당

문제 13-1

다음과 같은 실수형 데이터를 힙 영역에 저장한 뒤 값을 출력하는 프로그램은 다음과 같다.

■ double 자료형

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.
4.  int main(void)
5.  {
6.       double *ptr;
7.       int i;
8.
9.       ptr=(double *)malloc(sizeof(double)*10);//동적 메모리 할당
10.      if(ptr==NULL)
11.      {
12.           printf("Melory Allocation Error. \n");
13.              exit(1);
14.      }
15.
16.      for(i=0;i<10;i++)     // 할당된 영역에 값 저장
17.      {
18.              ptr[i]=i+1;
19.
20.      }
21.
22.      for(i=0;i<10;i++)    // 저장된 데이터 출력
23.              printf("%.2f ", ptr[i]);
24.
25.      free(ptr);    // 메모리 해제
26.
27.       return 0;
28. }
```

프로그램에서 보는 바와 같이동적 메모리 할당은 자료형에 구애받지 않고 저장되며, 단지 자료형이 다를 때 접근 방법의 차이가 있다는 점만 유의하면 된다

문제 13-2

구조체 데이터를 원하는 만큼 힙 영역에 저장한 뒤 데이터를 검색하는 완성된프로그램이다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
   struct student {
          char st_no[20];
          char subject[20];
          char name[20];
          char entry;
          char telno[15];
          struct student *next;
    }*temp, *head, *tail;

    char s_st_no[20];
    int found;

    head=tail=NULL;

   while(1)    //데이터 입력
   {
      temp=(struct student *)malloc(sizeof(struct student));
      if(temp==NULL)
      {
         printf("mamory allocation fail... \n");
         exit(1);
      }

printf("학번 ? (입력종료:Enter) ");
gets(temp->st_no);
      if(temp->st_no[0]==NULL)
         break;
printf("학과 ? ");
gets(temp->subject);
printf("성명 ? ");
gets(temp->name);
printf("등록여부(1:등록, 0:미등록) ? ");
scanf("%d%*c", &temp->entry);
printf("전화번호 ? ");
      gets(temp->telno);

      if(head==NULL)
         head=tail=temp;
```

```
45.       else
46.        {
47.           tail->next=temp;
48.           tail=temp;
49.        }
50.     }//while end
51.     free(temp);
52.
53.     //데이터 검색
54.  while(1)
55.     {
56.  printf("\n검색할 학번 ? (검색종료:Enter) ");
57.  gets(s_st_no);
58.        if(!s_st_no[0])
59.           break;
60.
61.        temp=head;     // 첫 노드의 주소 대입
62.  found=1;
63.        while(temp)
64.        {
65.           if(!strcmp(s_st_no,temp->st_no))     // 학번이 같으냐 ?
66.  {
67.              found=0;
68.              printf("%s, %s, %s, %s, %s \n", temp->st_no,
69.                  temp->subject, temp->name, temp->entry==1 ?
70.  "등록" : "미등록", temp->telno);
71.  break;
72.  }
73.           temp=temp->next;
74.        }
75.        if(found)
76.  printf("데이터는 검색되지 않습니다. \n");
77.
78.     }
79.     return 0;
80.  }
```

문제 13-3

다음은 잘못된 표현을 사용하는 경우이다. 무엇이 문제인지 설명하라?

①

```
char *ptr;
*ptr=malloc(20);
gets(ptr);
…
```

malloc( ) 함수는 힙영역에 필요한 공간을 할당하고 선두번지를 반환한다. 따라서 ptr 포인터 변수는 반환된 주소를 대입하여 사용하게 된다. 따라서 다음과 같이 수정되어야 한다.

```
ptr=malloc(20);
```

②

```
int *ptr, i;
ptr=malloc(10);

for(i=0;i<100;i++)
    ptr[i]=i;
…  …
```

문법적으로는 유효하나 잘못된 것이다. 할당된 기억장소의 경계를 넘고있다. 반복문은 10보다 작을 때 까지만 반복해야 한다.

CHAPTER

14 선행처리와 고급문제

문제 14-1

다음은 잘못된 표현을 사용한 매크로이다. 무엇이 문제인지 설명하라?

①

```
#define COUNT 100;

#define COUNT 100    // 변경
```

전처리지시자는 세미콜론(;)으로 끝나지 않는다.

②

```
#define MSG 입력방식이 바르지 않습니다.

#define MSG "입력방식이 바르지 않습니다."  // 변경
```

문자열은 인용부호("")로 묶어 전달한다.

③

```
#define MSG "입력방식이 바르지 않습니다. 다시 한번"
            "확인하여 주세요!!!"

#define MSG "입력방식이 바르지 않습니다. 다시 한번" \
            "확인하여 주세요!!!"                          // 변경
```

매크로 상수는 한 행에 기술해야 하나 한 행을 넘어가는 경우는 '₩'를 사용하여 다음 줄과 연결된다는 것을 알려 주어야 한다.

문제 14-2

두 수를 입력받아 큰 값을 구하는 매크로 함수를 구현하는 프로그램은 다음과 같다.

```
1.  #include <stdio.h>
2.
3.  #define MAX(x,y)  (x>y ? x : y )
4.
5.  int main(void)
6.  {
7.     int num1, num2, max;
8.
9.  printf("두 수를 입력하세요 ? ");
10.    scanf("%d%d", &num1, &num2);
11.
12.    max=MAX(num1, num2);
13.
14.    printf("max : %d \n", max);
15.
16.    return 0;
17. }
```

3행의 매크로 함수 선언을 보면 3항 연산자를 사용하였다. x가 y보다 크면 x를 반환하고 작으면 y를 반환하게 된다.

```
#define MAX(x,y)  ( x>y ? x : y )
                    조건  참  거짓
```

결과

```
두 수를 입력하세요 ?  7  50
max : 50
```