# Chapter 4. 제어의 흐름

연습문제

4-1. 다음 수식을 부정을 사용하지 않는 동일한 논리 수식으로 다시 써라.

!(a>b)

!(a <= b && c <=d)

!(a + 1 == b + 1)

!(a<1 || b< 2 && c <3)

Sol)

!(a>b) == (a<=b)

!(a <=b && c<=d) == (a>b || c>d)

!(a + 1 == b + 1) == (a+ 1 != b+ 1)

!(a<1 || b<2 && c<3) == (a>=1 && b>=2 || c>=3)

4-2. 다음 표를 완성하여라.

| 선언 및 초기화 | | |
|---|---|---|
| int a = 1, b = 2, c = 3;<br>double x = 1.0; | | |
| 수식 | 동일한 수식 | 값 |
| a > b && c< b | a > b > c | 1 |
| a < ! b \|\| !! a | (a< ( ! b) ) \|\| (!(!a)) | 1 |
| a + c < ! c + c | (a + c) < ( (!c) + c ) | 0 |
| a– x \|\| b *c && b /a | (a – x) \|\| (b * c) && (b/a) | 1 |

4-3. EOF를 만날때까지 표준 입력 파일로부터 문자를 읽는 프로그램을 작성하여라.
숫자와 다른 문자의 개수를 세기 위해 digit_cnt와 other_cnt 변수를 사용하여라.

C code

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

```
int main(void)
{
int digit_cnt=0,other_cnt=0, c=0;

while((c = getchar()) != EOF)
    if (c == ' '|| c == '₩n')
       {}
    else if (c >='0'&& c<='9')
       { + + digit_cnt; }
    else {+ + other_cnt; }
printf ("digit_cnt = %d , other cnt = %d ₩n",digit_cnt,other_cnt);

return 0;
}
```

4-4. 한 파일에서 알파벳의 첫 세 문자(a,A,b,B,c,C) 의 발생 횟수를 세는 프로그램을 작성하여라. 소문자와 대문자는 구별 하지 않는다.

Sol)

```
#include <stdio.h>

int main(void)
{
int abc_cnt=0,c=0;

while((c = getchar()) != EOF)
    if (c == ' '|| c == '₩n')
       {}
    else if (c >='a'&& c <='c' || c >='A' && c <='Z' )
       { + + abc_cnt; }
printf ("abc_cnt = %d₩n ",abc_cnt);

return 0;
}
```

4-5. 다음 루프를 포함하는 프로그램을 작성하여라.

```
while (scanf(%lf",&salary) == 1) {
 ....
}
```

루프의 몸체 부분에서 17%의 기본 원천 세금과 3%의 지방 세금을 계산한 후, 봉급과 함께 출력하여라. 그리고 출력 되는 모든 봉급과 세금을 합여여, 프로그램에서 while루프를 빠져 나올 때 그 합들을 출력하여라.

Sol)

C code

```
#include <stdio.h>
int main(void)
{ double salary = 0,sum_salary = 0,tax_3 = 0, sum_tax_3 = 0,tax_17 = 0 ,
sum_tax_17 = 0;
while (scanf("%lf",&salary) == 1) {
tax_3 = salary*0.03;   tax_17 = salary*0.17;
printf("%lf,%lf,%lf₩n",salary,tax_3,tax_17);
sum_salary + = salary; sum_tax_3 + = tax_3; sum_tax_17 + = tax_17;   }
printf("%lf,%lf,%lf₩n",sum_salary,sum_tax_3,sum_tax_17);
return 0; }
```

Result (결과)

```
[romance@161s ch4_code]$ ./a.out
10000
10000.000000,300.000000,1700.000000
2000
2000.000000,60.000000,340.000000
20000
20000.000000,600.000000,3400.000000
30000
30000.000000,900.000000,5100.000000
4210
4210.000000,126.300000,715.700000
66210.000000,1986.300000,11255.700000
[romance@161s ch4_code]$
```

다음 코드는 어떤 값을 출력하겠는가?

```
char c = 'A';
int i= 5,j = 10;
printf("%d %d %d₩n",!c,!!c,!!!c);
printf("%d %d %d₩n",-!i,!-i,!-i-!j);
printf("%d %d %d₩n",-!i,!-i,!-i-!j);
printf("%d %d %d₩n",!(6*j+ i-c),!i-5,!j - 10);
```

Sol)

```
[romance@161s ch4_code]$ ./a.out
0 1 0
0 0 0
0 0 0
1 -5 -10
[romance@161s ch4_code]$
```

4-6. 다음 코드의 실행 결과를 설명하여라.

```
int i;
……
while ( i= 2) { printf(“some even numbers : %d %d %d ₩n”,i,i+ 2,i+ 4);
i= 0;
}
```

Sol) while 루프 안으로 들어가는 순간 I = 2가 되고 2 4 6 을 무한히 출력한다.

4-7. 다음 코드는 어떤 값을 출력하겠는가?

```
char   c = ‘A’;
double x = 1e+ 33, y = 0.001;
printf(“%d %d %d₩n”, c == ‘a’, c == ‘b’,c != ‘c’);
printf(“%d₩n”,c ==’A’ && c < = ‘b’ || ‘c’);
printf(“%d₩n”,x +  y > x – y   );
```

Sol) 0 0 1

1

1

0

4-8. 다음 코드를 실행시킬 경우, 어떤 값이 출력되는가? 설명해 보아라.

```
int i = 7, j = 7;
if ( i == 1) if (j == 2)
printf ("%d₩n",i=i+ j);
else ("%d₩n", i = i -j);
printf("%d₩n",i);
```

Sol) I가 1이 아니니깐 if 루틴 안으로 들어오지 않는다. 따라서
I를 출력하고 끝난다.
따라서 7을 출력한다.

4-9. 다음을 포함하는 프로그램을 작성하여, 컴파일러가 이 구문 오류에 대해 어떤 메시지를 출력하는지 확인해 보아라. 그리고 그 이유를 설명하여라.

```
while(+ + i < LIMIT ) do {
j = 2 * i + 3;
printf("j = %d₩n",j);
}
```

```
[romance@161s ch4_code]$ cc 4_10.c
4_10.c: In function `main':
4_10.c:11: parse error before `return'
[romance@161s ch4_code]$
```

4-10. 다음 코드는 무한 루프에 빠질 수 있는가? 설명하여라.

```
printf("Input two integers : ");
scanf ("%d%d",&i,&j);
while( i * j < 0 && + + i != 7 && j+ +  != 9)
{ }
```

Sol) 없다.

4-11. 4.8절 “while 문”에서 n이 음수이면, 다음 whlie 문은 무한 루프가 된다고 하였다.

```
While(--n )
  .....
```

그러나 실제적으로 이것은 컴퓨터에 종속적이다. 그 이유를 설명해 보아라.

Sol) 만약 n 이 한계에 다달아 Infinity가 되면 거짓이 되어 빠져 나올 것이다.

4-12. n으로 정수 값을 하나 읽어 들여서, n이 양수이면 n에서 2 * n까지의 정수의 합을 구하고, n이 음수이면 2 * n 에서 n 까지의 정수의 합을 구하려고 한다. 이것을 수행하는 2개의 프로그램을 작성하여라. 단, 한 프로그램은 for루프를 사용하고, 다른 하나는 whlie 루프를 사용하도록 하여라.

Sol)

```
#include <stdio.h>
int main(void)
{ int i,x,sum = 0;
scanf("%d",&x);
printf("for loop ");
if (x > 0){   for (i = x; i <= 2*x ; + + i) sum + =i; }
else { for (i = 2*x; i <= x ; + + i) sum + =i; }
printf ("%d₩n",sum);
printf("while loop ");
sum = 0;
if ( x> 0 )
{ i = x;   while ( i <=   2*x ) { sum + =i; + + i; } }
else { i = 2*x; while( i <= x ) {sum + =i; + + i; } }
printf ("%d₩n",sum);

return 0;
}
```

4-13. putchar() 함수는 출력한 문자의 값을 int형으로 리턴한다. 다음 코드는 어떤 값을 출력하겠는가?

```
for( putchar('1'); putchar('2'); putchar('3'))
putchar('4');
```

Sol)

1234123412341234 .... 무한히 출력한다.

4-14. C언어에 대한 다음 문장 중 잘못 된 것은 어느 것인가?

1) 모든 문장은 세미콜론으로 끝난다.
2) 모든 문장은 적어도 하나의 세미콜론을 포함한다.
3) 모든 문장은 많아야 하나의 세미콜론을 포함한다.
4) 정확히 33개의 세미콜론을 갖는 문장이 존재한다.
5) 33개의 세미콜론을 포함하는 35개의 문자로 된 문장이 존재한다.

Sol) 1),2),4),5)

한 문장에는 하나혹은 0개의 세미콜론을 포함한다.

4-15. 4.10절 "예제:부울 변수"에서 부울 함수 값을 표로 출력하는 프로그램을 제시하였다. 이 프로그램을 실행하고 출력을 조사해 보아라. 32개의 다른 입력에 대해서, 정확히 반(16개)의 우위 값은 1이다. 7개의 부울 변수에 대한 우위 함수의 값을 표로 출력하는 프로그램을 작성하여라. 128개의 다른 입력에 대해서 우위 값이 1인 개수는 몇 개 이겠는가?

C code

```
#include <stdio.h>

int main(void)
{
int b1=0,b2=0,b3=0,b4=0,b5=0,b6=0,b7=0,cnt = 0;
int bb = 0 , bb_cnt = 0;
printf("₩n%5s %5s %5s %5s %5s %5s %5s %5s %11s₩n₩n","Cnt","b1","b2","b3"
,"b4","b5","b6","b7","majority");
for (b1 = 0; b1<=1;++b1)
```

```
for (b2 = 0; b2<=1;++b2)
for (b3 = 0; b3<=1;++b3)
for (b4 = 0; b4<=1;++b4)
for (b5 = 0; b5<=1;++b5)
for (b6 = 0; b6<=1;++b6)
for (b7 = 0; b7<=1;++b7)
{if (b1+b2+b3+b4+b5+b6+b7 >= 4) { bb = 1; ++bb_cnt; }
 printf("₩n%5d %5d %5d %5d %5d %5d %5d %5d %5d",++cnt,b1,b2,b3,b4,b5,b6,b7,
bb );
}
printf("₩n%5d",bb_cnt);
putchar('₩n');


return 0;
}
```

Result (결과)

```
[romance@161s ch4_code]$ ./a.out

  Cnt     b1     b2     b3     b4     b5     b6     b7 majority

    1      0      0      0      0      0      0      0      0
    2      0      0      0      0      0      0      1      0
    3      0      0      0      0      0      1      0      0
          .......(중략)
  127      1      1      1      1      1      1      0      1
  128      1      1      1      1      1      1      1      1
   64                (우위 값 출력)
```

4-16. 첫 n개의 짝수의 합과 첫 n개의 홀수의 합을 계산하는 3개의 프로그램을 작성하여라. n값은 사용자가 입력하도록 하여라. 첫 번째 프로그램에서는 다음 코드를 사용하여라.

a) for문과 comma를 이용

```
#include <stdio.h>


int main(void)
{
int cnt,i,j,odd_sum = 0,even_sum = 0;
int n;
scanf("%d",&n);
for(cnt = 0,i=1,j=2;cnt <n; + + cnt,i+ =2,j+ =2)
 odd_sum + = i, even_sum  + = j;
printf ("%d %d₩n",odd_sum,even_sum);

return 0;
}
```

b) for문 만을 이용

```
#include <stdio.h>
int main(void)
{ int cnt,i,j,odd_sum = 0,even_sum = 0;
int n;
scanf("%d",&n);
 for(i=0;i <= 2*n;i+ =2)
 even_sum + = i;
 for(j=1;j <= 2*n-1;j+ =2)
 odd_sum + = j;
printf ("%d %d₩n",odd_sum,even_sum);
return 0;
}
```

c) while 루프를 이용

```
int main(void)
{
int cnt,i,j,odd_sum = 0,even_sum = 0;
int n;
scanf("%d",&n);

cnt=0;
i=1;
j=2;
while(cnt < n)
{
 odd_sum + = i, even_sum   + = j;
  + + cnt,i+ =2,j+ =2;
}
printf ("%d %d₩n",odd_sum,even_sum);
return 0;
}
```

4-17. 연습 문제 16번에서 작성한 프로그램 중 하나에 다음 코드를 삽입하여 다시 작성하여라.

```
do {
    printf("Input a positive integer : " );
    scanf("%d",&n);
    if (error = (n<=0))
        printf("₩nERROR : Do it again !₩n₩n");
    }while(error);
```

그리고 이 do문을 사용하는 대신 while문을 사용하여 같은 결과가 나오도록 수정하여라.

Sol)

```
error = 1;
while( error ) {
    printf("Input a positive integer : " );
    scanf("%d",&n);
    if (error = (n<=0))
        printf("₩nERROR : Do it again !₩n₩n");
    }
```

4-18. 다음 코드는 인터럽트가 걸릴 때 까지 화면에 TRUE FOREVER를 반복적으로 출력하는 것이다.

```
while(1)
   printf(" TRUE FOREVER ");
```

while 문 대신에 for 문을 사용하여 같은 작업을 수행하는 간단한 프로그램을 작성하여라.

Sol)

```
for(;;) printf(" TRUE FOREVER ");
```

4-19. 다음 코드는 "단축 평가"를 연습할 수 있는 코드이다.

```
int a = 0,b = 0,x;
x = 0 && (a = b = 777);
printf("%d %d %d₩n",a,b,x);
x = 777 || (a   = + + b);
printf("%d %d %d₩n",a,b,x);
```

Sol)

0 0 0

0 0 1

4-20. 논리 수식의 의미는 계산할 때 평가 순서가 매우 중요하다는 것을 암시한다. 다음 2개의 수식 중 올바른 것은 어느 것인가? 설명하여라.

(a) if ((x != 0.0) && ((z-x) / x*x <2.0))

(b) if (((z – x)/x *x <2.0) && (x != 0.0))

Sol)

(a)가 올바른 순서이다. X가 0이 되면 어떤 정수를 0으로 나눌수 없기 때문에 에러가 난다. 이를 방지 하기 위해서 (a)와 같이 해 두면 x=0이 되면 그 다음 수식에 들어가지 않기 때문에 적절한 표현이라고 할 수 있다.

4-21. st1, st2, st3의 세 문장이 있다고 가정하자. if-else 문을 사용하여, int 변수 I의 값을 검사하고, 그에 따라 각기 다른 조합의 문장을 실행하도록 하려고 한다. 다음 표는 세가지 조합을 보여주고 있다.

| i | 수행 | i | 수행 | i | 수행 |
|---|---|---|---|---|---|
| 1 | st1 | 1 | st2 | 1 | st1,st2 |
| 2 | st2 | 2 | s1,st3 | 2 | st1,st2 |
| 3 | st3 | 3 | st1 | 3 | st2,st3 |

Sol)

첫번째 수행은

```
if ( I == 1) st1;
else if ( I ==2 ) st2;
else st3;
```

두번째 수행은

```
if (I == 1) st2;
else { if ( I == 2) st3;
       st1;
     }
```

세번째 수행은

```
if (I = 3) st3;
else st1;
st2;
```

2-23. 다음과 같은 x에 대한 2차 다항식을 보자.

$ax^2 + bx + c$ 이식의 판별식은 다음과 같다. $b^2 - 4ac$

a,b,c 의 값을 읽어, 판별식의 제곱근을 출력하는 프로그램을 작성하여라.

C code

```
#include <stdio.h>
#include <math.h>
int main(void)
{
float a,b,c,d;
scanf("%f",&a);
scanf("%f",&b);
scanf("%f",&c);
d = b*b-4*a*c;
if (d == 0.0) {
        printf("%5f ₩n", ((-b +  sqrt(d)) /(2*a) ) );
        printf("%5f ₩n", ((-b - sqrt(d)) /(2*a) ) );
                }
else { printf( "%5f + i*( %5f )₩n" , -b/(2*a) ,sqrt(-d)/(2*a) ); }
return 0;
}
```

2-24. 23번 문제에서 a,b,c 값에 대하여 근을 계산하고, 그근을 다음 메시지 중 하나와 함께 출력하는 프로그램을 작성하여라.

degenerate  two real roots

Extremely degenerate  two complex roots

Multiple real roots

Sol)

```
#include <stdio.h>
#include <math.h>
int main(void)
{
float a,b,c,d;
scanf("%f",&a);
scanf("%f",&b);
scanf("%f",&c);
d = b*b-4*a*c;
if ( a == 0 && b == 0) { printf ("extremely degenerate₩n"); }

else if (a == 0 && b != 0) { printf("degenerate₩n"); }

else if (d > 0.0) { printf("two real roots : ");
                    printf("%5f ", ((-b - sqrt(d)) /(2*a) ) );
                    printf("%5f₩n", ((-b +  sqrt(d)) /(2*a) ) );
                  }
else if (d == 0.0) { printf("multiple real roots : ");
                     printf("%5f₩n", ((-b +  sqrt(d)) /(2*a) ) );
                   }

else {      printf( " two complex roots : ");
            printf( "%5f - i*( %5f )" , -b/(2*a) ,sqrt(-d)/(2*a) );
            printf( "%5f +  i*( %5f )₩n" , -b/(2*a) ,sqrt(-d)/(2*a) );
     }
return 0;
}
```

4-25. 부울 함수의 진리표는 변수들의 모든 가능한 값과 각 경우에 대한 부울 함수의 값으로 구성된 표이다. 4.10절 “예제 : 부울 변수”에서 우위 함수와 두 개의 다른 함수에 대한 진리표를 만들었다. 그 표에서는 참과 거짓을 각각 1,0으로 나타내었다. 다음 부울 함수 에 대한 진리표를 작성하여라.

(a) b1||b2 || b3 || b4

(b) b1 && b2 && b3 && b4

(c) !(!b1 || b2 ) && (! b3||b4)

```
#include <stdio.h>

int main(void)
{
int b1=0,b2=0,b3=0,b4=0,b5=0,b6=0,b7=0,cnt = 0;
int bb = 0 , bb_cnt = 0;
printf("₩n%5s %5s %5s %5s %5s %5s %5s %5s %5s₩n₩n","Cnt","b1","b2","b3"
,"b4","(a)","(b)","(c)","majority");
for (b1 = 0; b1<=1;++b1)
for (b2 = 0; b2<=1;++b2)
for (b3 = 0; b3<=1;++b3)
for (b4 = 0; b4<=1;++b4)
{if (b1+b2+b3+b4 >= 2) { bb = 1; ++bb_cnt; }
 printf("₩n%5d %5d %5d %5d %5d %5d %5d %5d %5d",++cnt,b1,b2,b3,b4,
b1||b2||b3||b4,b1&&b2&&b3&&b4,!(!b1||b2)&&(!b3||b4), bb );
}
printf("₩n%5d",bb_cnt);
putchar('₩n');


return 0;
}
```

4-26. 중괄호 쌍이 올바로 사용되었는 지 조사하는 프로그램을 작성하여라. 프로그램에서 다음 두개의 변수를 사용해야 한다:왼쪽 중괄호를 조사하는 left_cnt 변수와 오른쪽 중괄호를 조사하는 right_cnt 변수. 각각의 변수는 0으로 초기화되어야 한다. 프로그램은 입력 파일에서 각 문자를 읽고 출력할 수 있어야 한다. 중괄호를 만나면 해당변수를 증가 시킨다. 입출력을 위해 getchar()와 putchar() 매크로를 사용하여라.

Sol)

```
#include <stdio.h>

int main(void)
{
char a;
int i;
int right_cnt = 0,left_cnt = 0;

while((a = getchar()) != EOF)
{
   if (a == '{') ++left_cnt;
   if (a == '}') ++right_cnt;
   if (right_cnt > left_cnt) { putchar('?');
                                        putchar('?');}
   else {putchar(a); }

  }
printf("₩n");
if (right_cnt > left_cnt)
  { printf("ERROR : Missing right braces :");
     for(i=0; i < right_cnt - left_cnt ; ++i)
     printf("}");
     printf("₩n");
  }
else if (right_cnt < left_cnt)
  { printf("ERROR : Missing right braces :");
     for(i=0; i < left_cnt - right_cnt ; ++i)
     printf("{");
     printf("₩n");
  }
else { printf ("complete!₩n"); }



return 0;
}
```

4-27. 앞의 연습문제에서 작성한 프로그램을 수정하여 중괄호와 괄호를 동시에 처리할 수 있도록 하여라.

Sol)

```
#include <stdio.h>

int main(void)
{
char a;
int i;
int right_cnt = 0,left_cnt = 0;
int right_cnt2 = 0,left_cnt2 = 0;

while((a = getchar()) != EOF)
{
   if (a == '{') ++left_cnt;
   if (a == '(') ++left_cnt2;
   if (a == '}') ++right_cnt;
   if (a == ')') ++right_cnt2;



   if (right_cnt2 > left_cnt2) { putchar('?'); }
   else if (right_cnt > left_cnt) { putchar('?');
                                          putchar('?');}
   else {putchar(a); }

 }
printf("₩n");
if ( right_cnt != left_cnt || right_cnt2 != left_cnt2)
printf("ERROR : Missing right braces :");

if (right_cnt > left_cnt || right_cnt2 > left_cnt2)
 {
    for(i=0; i < right_cnt - left_cnt ; ++i)
    printf("}");
```

```
    for(i=0; i < right_cnt2 - left_cnt2 ; ++i )
    printf(")");
    printf("₩n");
 }

else if (right_cnt < left_cnt || right_cnt2 < left_cnt2)
 {
    for(i=0; i < left_cnt - right_cnt ; ++i)
    printf("{");
    for(i=0; i < left_cnt2 - right_cnt2 ; ++i)
    printf("(");
    printf("₩n");
 }
else { printf ("complete!₩n"); }



return 0;
}
```

4-28. 4.8절 “while”문에서 공백,숫자,문자의 수를 세는 프로그램을 작성하였다. 이 프로그램을 수정하여 소문자와 대문자를 구별하도록 세라.

Sol)

C code

```
#include <stdio.h>

int main(void)
{
int big=0,small=0;
int c;

while((c = getchar()) != EOF)
 if (c >='a' || c<='z' ) { ++small; }
 else if (c>='A' || c <= 'Z' ) {++big; }
```

```
printf("%7s%7s₩n","BIG","SMALL");
printf("%7d%7d₩n",big,small);
return 0;
}
```

4-29. 다음 코드는 break나 continue 문을 사용하지 않는 코드로 수정하여라.

```
int c,i,n,cnt = 0,total=0,digit_cnt = 0;
while( c= getchar()) {
  if ( c == 'E' ) break;
  + + cnt;
  if (c >= '0' && c <= '9')
  + + digit_cnt;
}
i = -5;
n = 50;
while(i<n) {+ + i; if (i ==0) continue;
total + =i;
printf(" i = %d and total = %d₩n",i,total);
}
```

Sol)

```
c = getchar();
while( c != 'E') {
  + + cnt;
  if (c >= '0' && c <= '9')
  + + digit_cnt;
  c = getchar();              }
i = -5;   n = 50;
while(i<n)
{ + + i;    if (i == 0) goto L1;
total + =i;   printf(" i = %d and total = %d₩n",i,total);
 L1 :  }
```

4-30. while 문을 goto와 if문을 사용하여 구현 할 수 있음을 보여라. 그리고 어떤 구문이 더 좋은지 설명하여라.

Sol) while ( exp1 ) {st1; }
st2;

Loop : If ( !exp1 ) goto OutLoop
st1;
goto Loop;
OutLoop : st2;

goto문이 많이 들어가면 프로그램의 맥락이 복잡해지기 때문에 최근에는 프로그래머들은 goto문 사용을 금기시 하고 있다.

4-31. 다음 코드를 goto가 없는 코드로 다시 작성하여라.

```
d = b * b – 4.0 * a *c ;
if ( d == 0.0) goto L1;
else if (d >0.0) {
  if (a != 0.0) {
        st1;      goto L4;
         }
   else goto L3;
                    } goto L2;
 L1: if (exp1)   st2; else gotoL3;
      Goto L4;
  L2 : if (exp2)   { st3; goto L4;}
  L3 : ……st4;
  L4 : ……st5;
```

Sol)

```
….
if ( a == 0.0) { st4; }
else if ( d > 0.0 ) {st1; }
```

```
else if (d == 0) {st2}
else {st3}
st5;
```

4-32. 다음은 for 루프의 몸체에서 continue 문이 어떻게 동작하는 지를 볼 수 있는 예제이다. 어떤 값이 출력 되는가?

```
for (putchar('1');   putchar('2');putchar('3')) { putchar('4');
continue; putchar('5'); }
```

Sol) 1234가 무한히 출력된다.

4-33. 수학연산 min(x,y)는 다음과 같은 조건부 수식으로 표현할 수 있다.
(x < y)? x : y
유사한 방법으로 조건부 수식만을 사용하여 다음 수학 연산을 기술해 보아라.
min(x,y,z)
max(x,y,z,w)

Sol) min(x,y,z) == ((x<y) && (x<z))? x : ((y<z)? y :z)
max(x,y,z,w) == ((x>y) && (x>z) && (x>w))? x : ((y>z) && (y > w)? y
: ((z>w)? z : w )))

4-34. 다음 두 문장 중 옳은 것은어느 것인가? Switch문의 구문을 살펴보고 먼저 답을 작성한 다음, 프로그램을 작성하여 검사해 보아라.

```
switch (1);                    /* version 1*/
switch(1) switch(1);           /* version 2*/
```

Sol) version 1이 옳다.

4-35. ANSI C에서 레이블은 자신의 이름 영역을 가지고 있지만, 전통적인 C에서는 그렇지 못하다. 비록 좋은 프로그래밍 방법은 아니지만, ANSI C에서는 하나의 식별자를 변수와 레이블로 동시에 사용할 수 있다. 다음 코드를 포함하는 프로그램을 작성하고, ANSI C 컴파일러를 실행해 보아라.

```
int L = -3;
if (L < 0)
  goto L;
printf("L = %d₩n",L);
L :printf("Exiting label test!₩n");
```

Sol)

```
[romance@161s ch4_code]$ cc 4_35.c
[romance@161s ch4_code]$ ./a.out
Exiting label test!
[romance@161s ch4_code]$
```

4-36. a를 양의 실수라고 하고, 실수 $x_i$의 수열을 다음과 같이 정의하자.

$$x_0 = 1, x_{i+1} = \frac{1}{2}(x_i + \frac{a}{x_i}) \quad \text{for I=0,1,2 ...}$$

이 식은 수학적으로 I가 무한대로 접근할 때 다음과 같음을 의미한다.

$$x_i -> \sqrt{a}$$

이를 구현하시오.

Sol)

```
#include <stdio.h>

int main(void)
{
double x0,x1 = 1.0;
double a;
```

```
scanf("%f",&a);

do
{
 x0 = x1;
 x1 = 0.5 *( x1 +  (a / x1));
 printf("x0 = %.5f ,   x1 =   %.5f   , a-x1*x1 %.5f ₩n",x0,x1,a-x1*x1);
} while(x0 != x1);

return 0;
}
```

4-37. 연습문제 36번에서 작성한 프로그램을 수정하여, 1부터 입력 받은 n값까지의 모든 정수에 대한 제곱근을 구할 수 있도록 하여라.

```
#include <stdio.h>

int main(void)
{
float x0,x1 = 1.0;
float a,c,i;

scanf("%f",&a);

for (i =1.0; i <= a; i+ =1.0)
{
printf("SQRT (%f)₩n",i);

do
{
 x0 = x1;
 x1 = 0.5 *( x1 +  (i / x1));
 printf("x0 = %.5f ,   x1 =   %.5f   , i-x1*x1 %.5f ₩n",x0,x1,i-x1*x1);
 if (x0 > x1) { c = x0 - x1;} else {c = x1 - x0; }
} while( c > 0.000001 );
printf("₩n₩n");
```

```
}
return 0;
}
```

4-38. 자연로그의 밑수인 상수e를 구현하시오.

```
#include <stdio.h>

int main(void)
{
float x1 = 1.0;
float a,i;
for (a= 1.0; a < 50; a+ =1.0)
{
   for (i = 1.0; i < a ; i+ =1.0)
          x1 =x1*(1+ 1/a) ;
 printf("x0 = %.5f ₩n",x1);
 x1 = 1.0;
}

return 0;
}
```

4-39. 연습문제 38번을 다른 알고리즘으로 구현하시오.

```
#include <stdio.h>

int main(void)
{
float x1 = 1.0;
float a,i,e = 0;
for (a= 1; a < 50; a+ =1.0)
{
   for (i = 1.0; i < a ; i+ =1.0)
          x1 =x1*1/i;
   e=e+ x1;
```

```
   x1=1.0;

 printf("e = %.5f ₩n",e);

}

return 0;
}
```

Chapter 4 End Point.