

```
#include <stdio.h>
int main(void)
{
    int count;
    for(count=1; count<=500; count++)
        printf("I will not throw paper airplanes in class.");
    return 0;
}
```

AMEND 10-3



Copyright 2004, FoxTrot by Bill Amend
www.ucomics.com/foxtrot/2003/10/03

제1장

변수와 흐름제어

변수와 치환

Variables and Assignment

hello.c

프로그램의 모든 부분을 내가 직접 작성하는 것은 아니다. 미리 작성되어 있는 프로그램(라이브러리)을 내 프로그램에 포함시켜서 사용한다. 표준입출력 라이브러리를 이렇게 `include`한다.

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

```
int main(void)
```

```
{
```

```
    printf("Hello world!\n");
```

```
    return 0;
```

```
}
```

main 함수는 프로그램 실행이 시작되는 곳이다.

화면에 문자열을 출력한다. 출력하고 싶은 문자열을 껌따 옴표("")로 묶는다. '\n'은 줄바꿈 문자이다.

code01.c

1에서 100까지의 합을 구하는 코드이다.

```
#include <stdio.h>

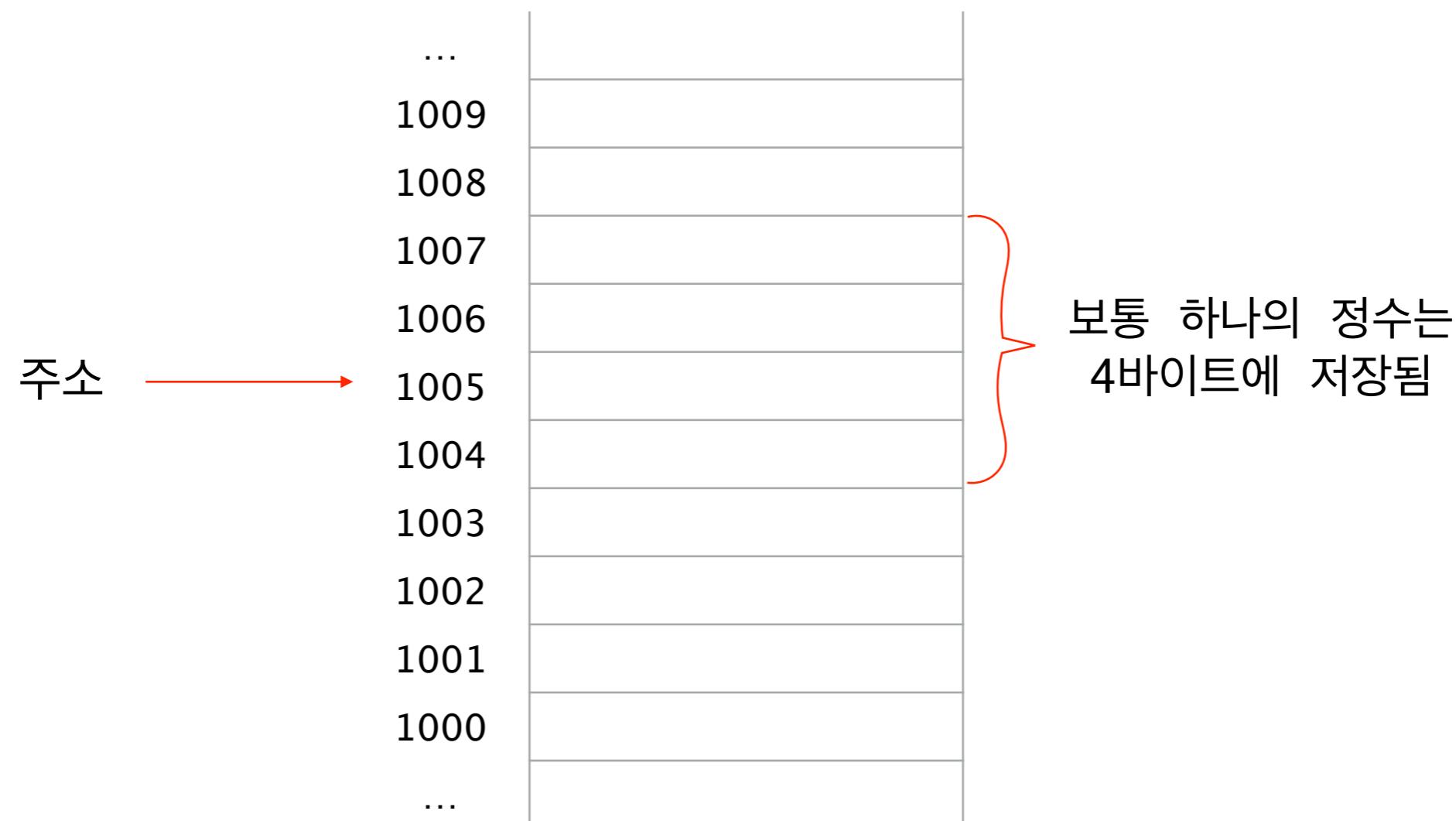
int main(void)
{
    int sum = 0;           sum과 i를 변수라고 부른다.
    int i;

    for (i=1; i<=100; i++)
        sum = sum + i;
    printf("The sum from 1 to 100 is %d.\n", sum);

    return 0;
}
```

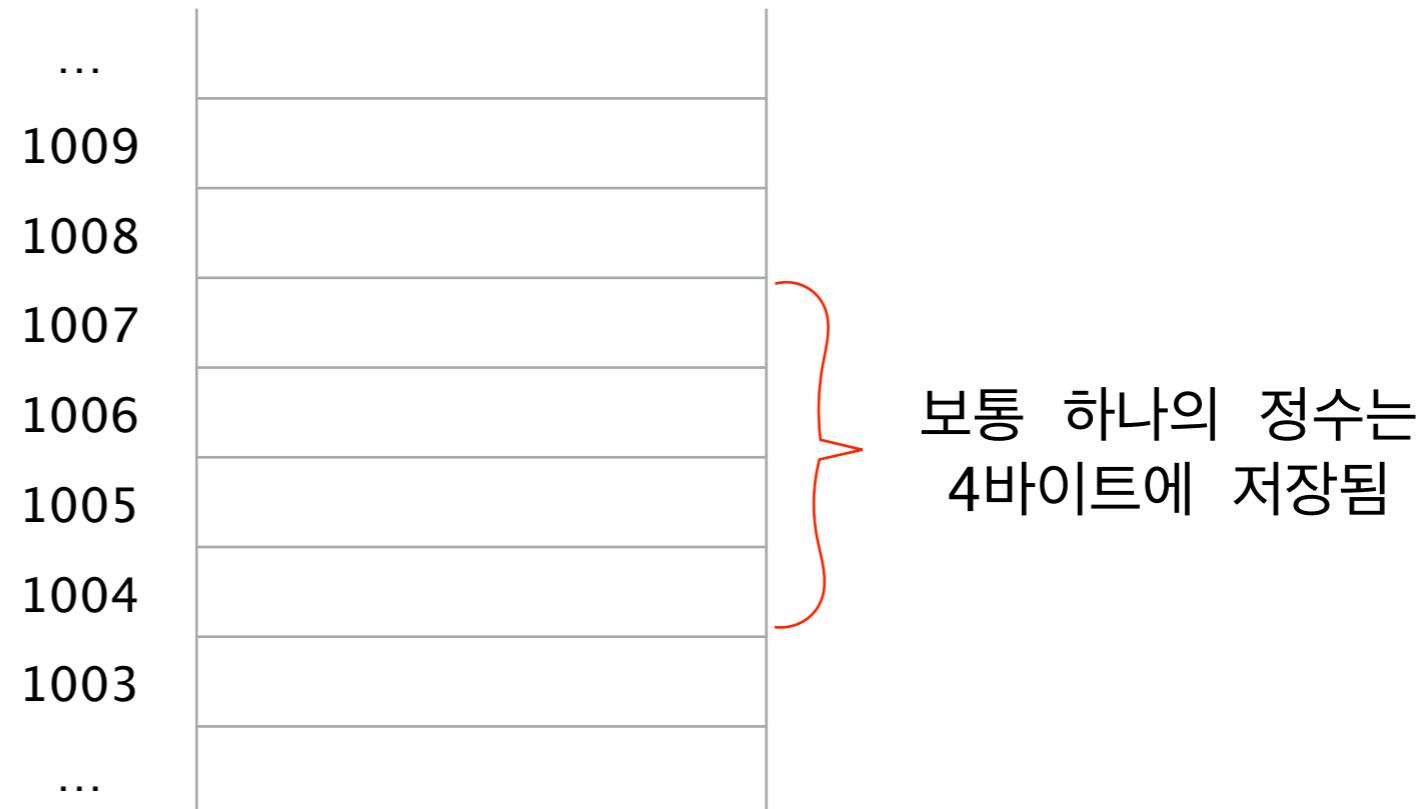
메모리 (RAM)

- 컴퓨터의 메모리는 데이터를 보관하는 장소
- 바이트(8 bits) 단위로 주소가 지정됨



메모리 (RAM)

- 기계어나 어셈블리 언어 프로그램에서는 직접 메모리 주소를 사용

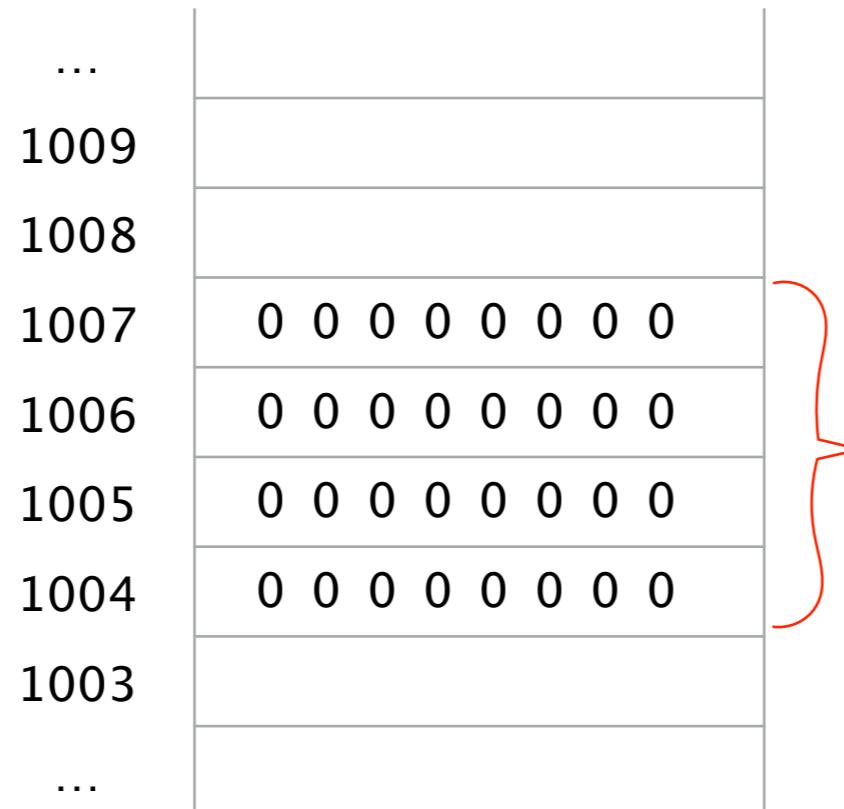


LOAD R0 1004

메모리의 1004번지에서부터 연속된 4바이트(32비트)의 데이터를 CPU의 R0레지스터로 읽어오라.

메모리 (RAM)

- C와 같은 고급언어 프로그램에서는 메모리 주소 대신 **변수**를 사용



```
int sum = 0;
```

프로그램에서 변수의 이름(sum)과 타입(int)를 정의해 준다. 그러면 시스템에 의해서 이 변수에게 4바이트의 메모리가 할당된다. 프로그래머는 이 메모리의 주소를 알 필요가 없으며 변수의 이름을 사용해 데이터를 읽고 쓴다.

변수

- 변수는 데이터를 보관하는 장소(memory)
- 변수는 사용하기 전에 선언해야 한다. 변수의 선언이란 “이름”과 “타입”을 정해주는 것
- 타입을 정해주는 이유는 그 변수에게 몇 바이트의 메모리를 할당할지 결정하기 위해서이다.

타입 이름	설명
int	정수
float	소수
double	소수
char	문자
char *	문자열
그밖의 타입들	나중에 배울 것임

} 기본 타입

실제로는 훨씬 더 많은 타입들이 있으나 우리는 당분간 int, double, char, char *타입만을 사용한다.

모든 변수는 다음의 6가지를 가진다 !

	<code>int sum = 0;</code>	<code>int i;</code>	<code>double degree = 10.0;</code>
이름 (name)			
타입 (type)			
값 (value)			
주소 (address)			
사용 범위 (scope)			
수명 (life time)			

code01.c

```
#include <stdio.h>

int main(void)
{
    int sum = 0;
    int i;
    for (i=1; i<=100; i++)
        sum = sum + i;
    printf("The sum from 1 to 100 is %d.\n", sum);
    return 0;
}
```

두 개의 정수형 변수 sum과 i를 선언하였다. sum은 선언과 동시에 0으로 초기화해주었다.

i처럼 초기화되지 않은 변수는 예측할 수 없는 값을 가진다. 컴파일러에 따라서는 초기화하지 않은 변수의 값을 엑세스하는 것을 오류로 처리하기도 한다.

치환문과 치환 연산자

치환 연산자

sum = 0;

- 치환 연산자(=)은 수학에서의 =과 다른 뜻이다.
- 수학에서 =은 양쪽이 “동일하다”라는 사실을 표현하는 기호이다.
- 프로그램에서 =은 오른쪽의 값을 왼쪽의 변수(장소)에 저장하라는 명령이다.

치환문과 치환 연산자

statements	correct ?
sum = 10;	
sum = sum + 1;	
sum = sum + sum;	
sum = 2*sum + i/j + 100;	
sum = i/2 + 100;	
sum + 1 = sum;	
sum * 2 = i;	
sum + i = sum + i;	
sum = sum;	
10 = sum;	

흐름제어

for, if-else, while

code01.c

```
#include <stdio.h>

int main(void)
{
    int sum = 0;
    int i;
    for (i=1; i<=100; i++)
        sum = sum + i;
    printf("The sum from 1 to 100 is %d.\n", sum);
    return 0;
}
```

for문은 대표적인 반복문이다.

for 문

- 2) 이 조건을 검사하여 만족되면
statements를 한 번 실행하고,
 아니면 for문을 벗어난다.

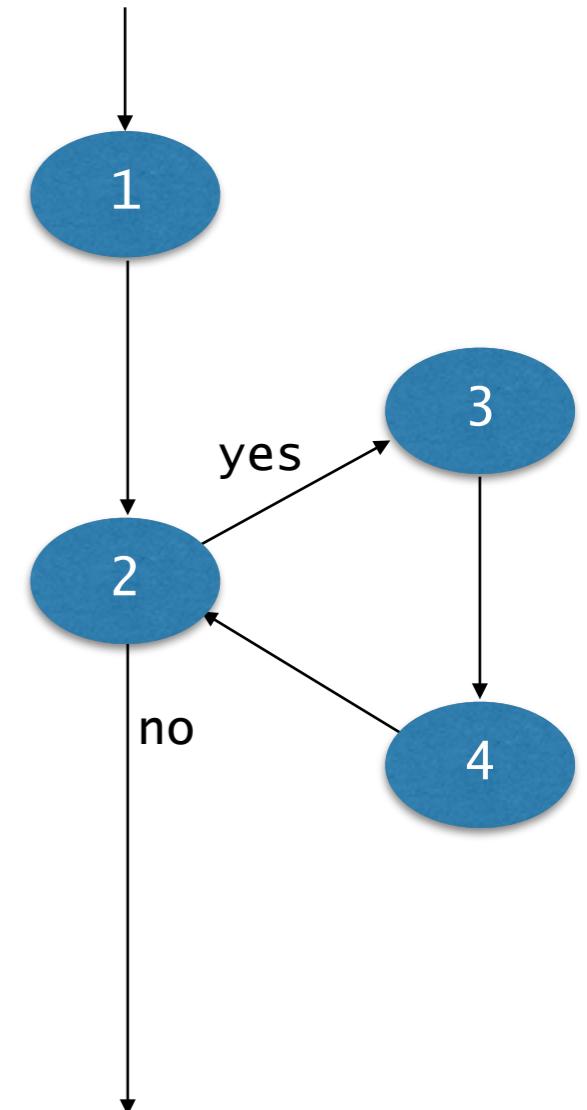
- 1) 처음에 한 번 실행
 하고 잊어버린다.

```
for ( statement1; condition; statement2 )
```

```
{  
    statements;  
}
```

- 3) 중괄호로 둘러싸인 문장들이다.
condition이 만족되면 실행
 된다. 이 부분이 단지 하나의
 문장이라면 중괄호를 생략할 수
 있다.

- 4) **statements**가 실행되고
 나면 자동으로 한 번 실행
 된다. 그런 후 다시
condition을 검사하려
 간다.



code01.c

```
#include <stdio.h>

int main(void)
{
    int sum = 0;
    int i;

    for (i=1; i<=100; i++)
        sum = sum + i;

    printf("The sum from 1 to 100 is %d.\n", sum);

    return 0;
}
```

i=1에서 시작한다. i가 100을 초과할 때 까지 sum에 i를 더하고, i는 1씩 증가시킨다. 결과적으로 sum의 값은?

code01.c

```
#include <stdio.h>

int main(void)
{
    int sum = 0;
    int i;

    for (i=1; i<=100; i++)
        sum = sum + i;

    printf("The sum from 1 to 100 is %d.\n", sum);

    return 0;
}
```

이 자리에 하나의 정수를 끼워 넣겠다.

이것이 끼워 넣을 정수이다.

1에서 100까지 더하기

i	sum
1	1
2	2
3	6
4	10
5	15
6	21
7	...
8	
...	
100	5050

code02.c

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

#define MIN 0
#define MAX 360
#define STEP 10

void main() {
    double radian, pi, value;
    pi = 4.0*atan(1.0);
    printf ("Angle Sine \n");
    for (int degree=MIN; degree<=MAX; degree+=STEP ){
        radian = pi*(degree/180);
        value = sin(radian);
        printf (" %3d %f \n ", degree, value);
    }
}
```

0도에서 360도까지 10도 단위로 sine함수값을
계산하여 출력한다.

code02.c

```
#include <stdio.h>
#include <math.h>
#define MIN 0
#define MAX 360
#define STEP 10

void main() {
    double radian, pi, value;
    pi = 4.0*atan(1.0);
    printf ("Angle sine \n");
    for (int degree=MIN; degree<=MAX; degree+=STEP ){
        radian = pi*(degree/180);
        value = sin(radian);
        printf (" %3d %f \n ", degree, value);
    }
}
```

sin함수나 atan함수를 계산하기 위해서 math 라 이브러리를 include한다.

code02.c

```
#include <stdio.h>
#include <math.h>
#define MIN 0
#define MAX 360
#define STEP 10

void main() {
    double radian, pi, value;
    pi = 4.0*atan(1.0);
    printf ("Angle sine \n");
    for (int degree=MIN; degree<=MAX; degree+=STEP ){
        radian = pi*(degree/180);
        value = sin(radian);
        printf (" %3d %f \n ", degree, value);
    }
}
```

define문을 이용하여 상수들을 정의하였다.

변수 degree를 이렇게 for문 내에서 선언하는 것
과 바깥에서 선언하는 것의 차이는?

code02.c

```
#include <stdio.h>
#include <math.h>
#define MIN 0
#define MAX 360
#define STEP 10

void main() {
    double radian, pi, value;
    pi = 4.0*atan(1.0);
    printf ("Angle sine \n");
    for (int degree=MIN; degree<=MAX; degree+=STEP ){
        radian = pi*(degree/180);
        value = sin(radian);
        printf (" %3d %f \n ", degree, value);
    }
}
```

실수는 float이나 double형 변수에 저장한다.

실행 결과는? 이유는?

code02.c

```
#include <stdio.h>
#include <math.h>
#define MIN 0
#define MAX 360
#define STEP 10

void main() {
    double radian, pi, value;
    pi = 4.0*atan(1.0);
    printf ("Angle sine \n");
    for (int degree=MIN; degree<=MAX; degree+=STEP ){
        radian = pi*degree/180.0;
        value = sin(radian);
        printf (" %3d %f \n ", degree, value);
    }
}
```

/ 연산에서 양쪽 피연산자가 모두 정수인 경우
결과도 정수이다. 즉 엄밀히 말하면 나눗셈이 아
니라 몫을 구한다.

연습 1-2

1. 입력으로 2차방정식 $ax^2+bx+c=0$ 의 계수 a, b, c 를 받아서 근을 구해 출력하는 프로그램을 작성하라. 여기서 a, b, c 는 정수이다. (`math.h`가 제공하는 `sqrt(double)` 함수를 사용하라. 실수근이 존재하지 않으면 NO ROOT를 출력한다.)
2. 섭씨 0도에서 100도까지를 5도 단위로 화씨 온도로 변환하여 출력하는 프로그램을 작성하라. 섭씨 온도를 화씨로 변환하는 규칙은 다음과 같다.

$${}^{\circ}\text{F} = {}^{\circ}\text{C} \times 1.8 + 32$$

code03.c

```
#include <stdio.h>

int main(void)
{
    int sum = 0;
    for (int i=1; i<=100; i++) {
        if (i%2==0 || i%3==0)
            sum = sum + i;
    }
    printf("The sum is %d.\n", sum);
    return 0;
}
```

if 문은 조건분기문이다. 괄호 안의 조건이 만족될 때만 sum=sum+i를 실행한다.

==은 양쪽이 동일하다는 조건 연산자이고, &&는 AND, ||는 OR 논리 연산자이다.

%는 나머지를 구하는 연산자이다.

code04.c

```
#include <stdio.h>
int main(void)
{
    int count1 = 0, count2 = 0;
    for (int i=1; i<=100; i++)  {
        if (i%2==0 || i%3==0)
            count1 = count1 + 1;
        else
            count2++;
    }
    printf("The results are %d %d.\n", count1, count2);
    return 0;
}
```

if-else 문

- 1) condition1이 만족되면 statements1을 실행한다.



```
if ( condition1 ) {  
    statements1;  
}  
else if ( condition2 ) {  
    statements2;  
}  
else if ( condition3 ) {  
    statements3;  
}  
else {  
    statements4;  
}
```

- 2) condition1이 만족되지 않으면서 condition2가 만족되면 statements2를 실행한다.



- 3) condition1, 2, 3가 모두 만족되지 않으면 statements4를 실행한다.



소득세율

소득 (income)	세율 (rate)
\$0 - \$40,000	22%
\$40,000 - \$100,000	25%
\$100,000 - \$170,000	28%
\$170,000 - \$300,000	33%
-\$300,000	35%

```
double rate;
if (income < 40000)    rate = 0.22;
if (income < 100000)   rate = 0.25;
if (income < 170000)   rate = 0.28;
if (income < 300000)   rate = 0.33;
if (income >= 300000)  rate = 0.35;
```

correct ?

소득세율

소득 (income)	세율 (rate)
\$0 - \$40,000	22%
\$40,000 - \$100,000	25%
\$100,000 - \$170,000	28%
\$170,000 - \$300,000	33%
-\$300,000	35%

```

if (income < 40000) rate = 0.22;
else {
    if (income < 100000) rate = 0.25;
    else {
        if (income < 170000) rate = 0.28;
        else {
            if (income < 300000) rate = 0.33;
            else rate = 0.35;
        }
    }
}

```

correct ?

소득세율

소득 (income)	세율 (rate)
\$0 - \$40,000	22%
\$40,000 - \$100,000	25%
\$100,000 - \$170,000	28%
\$170,000 - \$300,000	33%
-\$300,000	35%

```
double rate;
if      (income < 40000) rate = 0.22;
else if (income < 100000) rate = 0.25;
else if (income < 170000) rate = 0.28;
else if (income < 300000) rate = 0.33;
else                      rate = 0.35;
```

조건 분기문의 예

절대값	<pre>if (x < 0) x = -x;</pre>
x와 y를 크기순으로 정렬	<pre>if (x > y) { int t = x; x = y; y = t; }</pre>
x와 y중 큰 값	<pre>int max; if (x > y) max = x; else max = y;</pre>

연습 3

3. 입력으로 세 정수 a, b, c를 받은 후 오름차순으로 정렬하여 출력하는 프로그램을 작성하라.

```
#include <stdio.h>
int main(void)
{
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);

    printf("%d %d %d\n", a, b, c);
    return 0;
}
```

이렇게 출력했을 때 정렬되어 있어야 한다.

연습 4

4. 삼각형은 어떤 두 변의 길이의 합도 다른 한 변의 길이보다 커야 한다. 이것을 triangular inequality라고 부른다. 입력으로 세 개의 양의 정수를 받아서 triangular inequality를 만족하는지 검사하여 YES 혹은 NO를 출력하는 프로그램을 작성하라.

```
#include <stdio.h>
int main(void)
{
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);

    return 0;
}
```

연습 5

5. 입력으로 5개의 서로 다른 정수를 받아서 중간값(median)을 찾아 출력하는 프로그램을 작성하라. 중간값이란 크기순으로 순서를 매길 때 3번째인 값을 말한다. 즉 자신 보다 작은 값이 2개이고 큰 값이 2개인 값이다. ■ 비교횟수 6번 이하로 해결해 보라.

```
#include <stdio.h>
int main(void)
{
    int a, b, c, d, e;
    scanf("%d %d %d %d %d", &a, &b, &c, &d, &e);

    return 0;
}
```

연습 6-7

어떤 정수의 끝 두자리를 구하려면?

6. 10~10,000 사이의 정수들 중 끝 두 자리가 2 혹은 3의 배수인 정수들의 개수를 세는 프로그램을 작성하라. 예를 들어 1115의 끝 두자는 15이고 3의 배수이다.
7. 10,000~99,999사이의 정수들 중 처음 두 자리가 3 혹은 7의 배수인 것들의 개수를 세는 프로그램을 작성하라. 예를 들어 12,345의 처음 두자는 12이고 3의 배수이다.

5자리 정수의 처음 두자리를 구하려면?

연습 8

8. 입력으로 두 개의 정수 a와 b를 받은 후 a^b 을 계산하는 프로그램을 작성하라. b는 음이 아닌 정수이다.

```
#include <stdio.h>
int main(void)
{
    int a, b;
    scanf("%d %d", &a, &b);

    return 0;
}
```

while문

- ➊ 키보드로부터 여러 개의 정수를 순차적으로 입력 받는다. 짹수가 입력되면 무시하고 홀수가 입력되면 더해 나간다. 더해진 홀수의 개수가 10개가 되면 합을 출력하고 종료한다.

code05.c

```
#include <stdio.h>
void main() {
    int count = 0, sum = 0;
    int tmp;
    while(count<10) {
        scanf("%d", &tmp);
        if (tmp%2==1) {
            sum += tmp;
            count++;
        }
    }
    printf("The sum is %d.\n", sum);
}
```

while문은 for문과 함께 가장 자주 사용되는 반복문이다.

반복의 횟수가 미리 정해진 경우에는 보통 for문을 사용하고 그렇지 않은 경우에는 while문을 사용한다.

모든 for문은 while문으로 바꿀 수 있고 반대의 경우도 마찬가지이다.

code05_2.c

```
#include <stdio.h>
void main() {
    int count = 0, sum = 0;
    int tmp;
    while(1) {
        scanf("%d", &tmp);
        if (tmp%2==0)
            continue;
        sum += tmp;
        count++;
        if (count>=10)
            break;
    }
    printf("The sum is %d.\n", sum);
}
```

엄밀히 말해서 while문은 ()안의 값이 0이면 종료한다.
관습적으로 0은 false를 0이 아닌 값은 true를 의미한다.
따라서 while(1)은 무한루프이다.

continue문은 자신을 둘러싼 가장 안쪽 루프 내에서 자신의 다음에 나오는 모든 문장들을 실행하지 않고 건너뛰게 한다.

break문은 자신을 둘러싼 가장 안쪽 루프를 빠져 나간다.

code06.c

- 키보드로부터 여러 개의 정수들을 입력받아 그 합을 구한다. 사용자가 -1을 입력하면 합을 출력하고 종료한다.

```
#include <stdio.h>
void main() {
    int sum = 0;
    int tmp;
    while(1) {
        scanf("%d", &tmp);
        if (tmp== -1) {
            printf("The sum is %d.\n", sum);
            break;
        }
        sum += tmp;
    }
}
```

입력된 숫자가 -1이면 합을 출력하고 while문을 빠져 나간다.

연습 9

9. 키보드로 부터 양의 정수들을 연속해서 입력받는다. 홀수번째 수는 더하고 짝수번째 수는 뺀다. 합이 0이되면 입력된 정수의 개수를 출력하고 종료 한다. 예를 들어 입력이 1 2 2 3 2의 순서로 들어오면 $1-2+2-3+2=0$ 이 되어 5를 출력하고 종료한다.

Code07.c

- 입력으로 하나의 양의 정수 N을 받은 후 N보다 작거나 같으면서 가장 큰 2의 거듭제곱수를 출력하는 프로그램을 작성하라. 예를 들어 N=23이면 16을 출력한다.

```
#include <stdio.h>
int main(void)
{
    int N;
    scanf("%d", &N);
    int p=1;
    while (p*2<=N)
        p *= 2;
    printf("The answer is %d.\n", p);
    return 0;
}
```

N=115인 경우 p는
 1
 2
 4
 8
 16
 32
 64
 순으로 증가한다.

Code08.c: 이진수

```
#include <stdio.h>
int main(void) {
    int N;
    scanf("%d", &N);
    int v = 1;
    while (v*2 <= N)
        v = 2*v;
    while (v > 0) {
        if (N < v)
            printf("0");
        else {
            printf("1");
            N -= v;
        }
        v = v/2;
    }
    return 0;
}
```

입력으로 하나의 양의 정수 N을 받은 후 이진수로 변환하여 출력하는 프로그램이다.

v는 N보다 작거나 같으면서 가장 큰 2의 거듭제곱수이다.

N=115인 경우:

N=115	v=64	출력=1
N=51	v=32	출력=1
N=19	v=16	출력=1
N=3	v=8	출력=0
N=3	v=4	출력=0
N=3	v=2	출력=1
N=1	v=1	출력=1

GCD

- 입력으로 두 양의 정수를 받은 후 두 정수의 최대공약수(GCD)를 구해서 출력하는 프로그램을 작성하라. GCD를 구하기 위해서 Euclid 알고리즘을 사용하라. Euclid 알고리즘은 다음의 성질을 이용한다: 두 정수 x, y 중에 크거나 같은 쪽을 x 라고 하자. 만약 x 가 y 로 나누어 떨어지면 GCD는 y 이다. 그렇지 않다면 x 와 y 의 GCD는 $x \% y$ 와 y 의 GCD와 같다.

code09.c: GCD

```
#include <stdio.h>

int main(void)
{
    int m, n;
    scanf("%d %d", &m, &n);
    if (m<n) { int tmp = m; m=n; n=tmp; }
    while (m%n!=0) {
        m = m%n;
        int tmp = m;
        m=n;
        n=tmp;
    }
    printf("The GCD is %d", n);
    return 0;
}
```

Euclid 알고리즘

code10.c

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

```
int main(void)
{
    int N;
    scanf("%d", &N);
    int sum = 0;
    while (N>0) {
        sum += (N%10);
        N /= 10;
    }
    printf("The sum is %d.\n", sum);
    return 0;
}
```

입력으로 하나의 양의 정수 N을 받은 후 각 자리수의 합을 구하여 출력하는 프로그램이다.

연습 10: π

10. 파이(π)의 값은 다음과 같이 근사적으로 계산할 수 있다.

$$\frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \dots$$

항의 개수가 많을 수로 실제 π 의 값에 가까워진다. 입력으로 양의 정수 n 을 받아서 이 무한 시리즈의 n 번째 항까지 계산하여 출력하는 프로그램을 작성하라. 실제 π 의 값과 얼마나 다른지 비교해보라.

연습 11: checksum

11. 책에 부여되는 ISBN 번호는 10자리로 구성된다. 이 중 가장 오른쪽 자리는 다음과 같은 규칙으로 다른 9자리의 값에 의해서 자동으로 결정되는 checksum 값이다.

$$d_1 + 2d_2 + 3d_3 + \dots + 10d_{10} = 11의 배수$$

여기서 d_i 는 i -번째 자리수를 나타내고 0과 9사이의 값이다 (원래는 0에서 10 사이의 값이다. 10은 x로 표시된다). 가령 처음 9자리가 020131452이면 마지막 자리는 5이다. 왜냐하면 5는

$10*0+9*2+8*0+7*1+6*3+5*1+4*4+3*5+2*2+1*x$ 를 11의 배수로 만드는 유일한 0과 9사이의 값이기 때문이다.

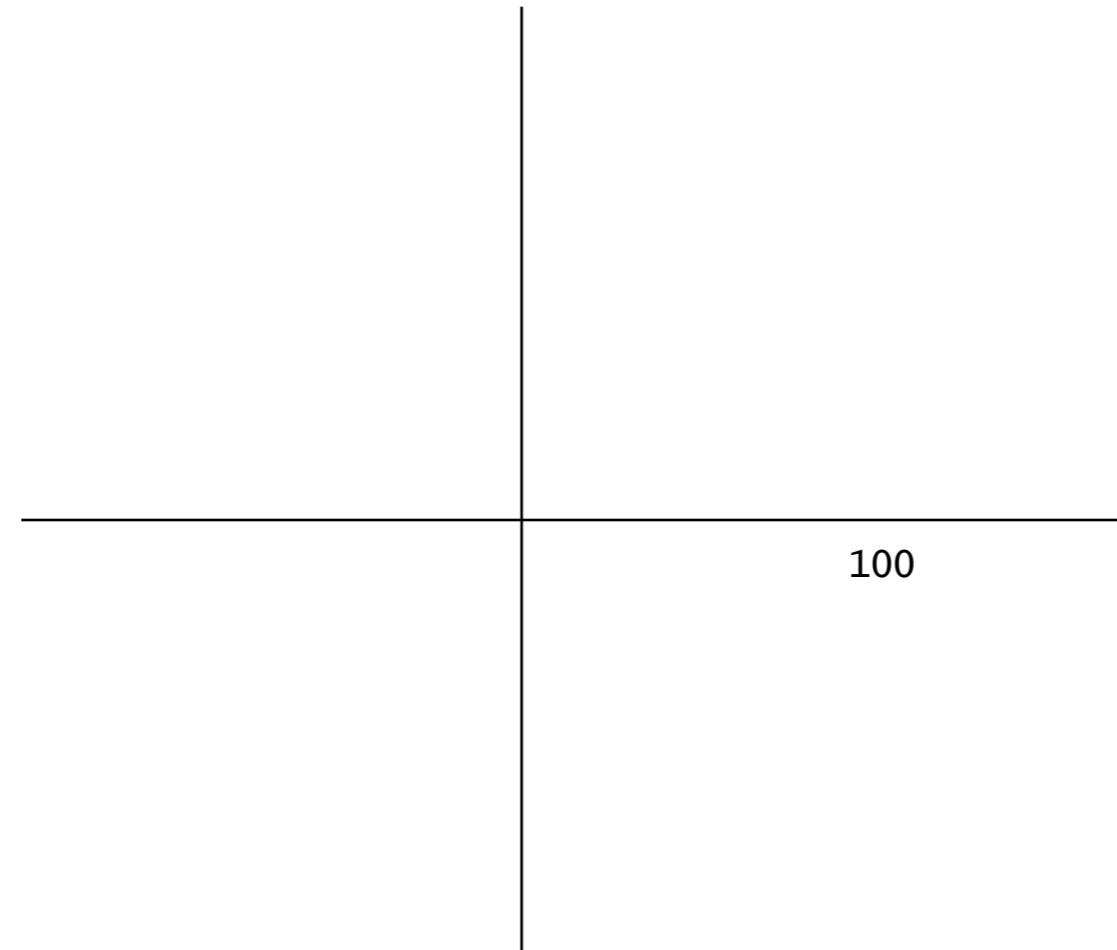
ISBN 번호의 처음 9자리수를 받아서 마지막 자리수를 계산하는 프로그램을 작성하라. 9자리수는 각각 별개의 정수로 입력된다. (답이 없을 경우 x를 출력하라.)

중첩된 루프

Nested Loops

code11.c

- 2차원 평면의 1사분면에서 원점으로부터 거리가 100 이하인 정수 좌표점의 개수는? x-좌표나 y-좌표가 0인 경우도 포함한다.



code11.c

```
#include <stdio.h>

int main(void)
{
    int count = 0;
    for ( int x=0; x<=100; x++ )    {
        for ( int y=0; y<=100; y++) {
            if (x*x + y*y <= 10000)
                count++;
    }
    printf("The number of points is %d.\n", count);
    return 0;
}
```

두 개의 중첩된 for문을 이용하여 점들을 다음과 같은 순서로 검사한다.

(0,0), (0,1), (0,2), ..., (0,100)

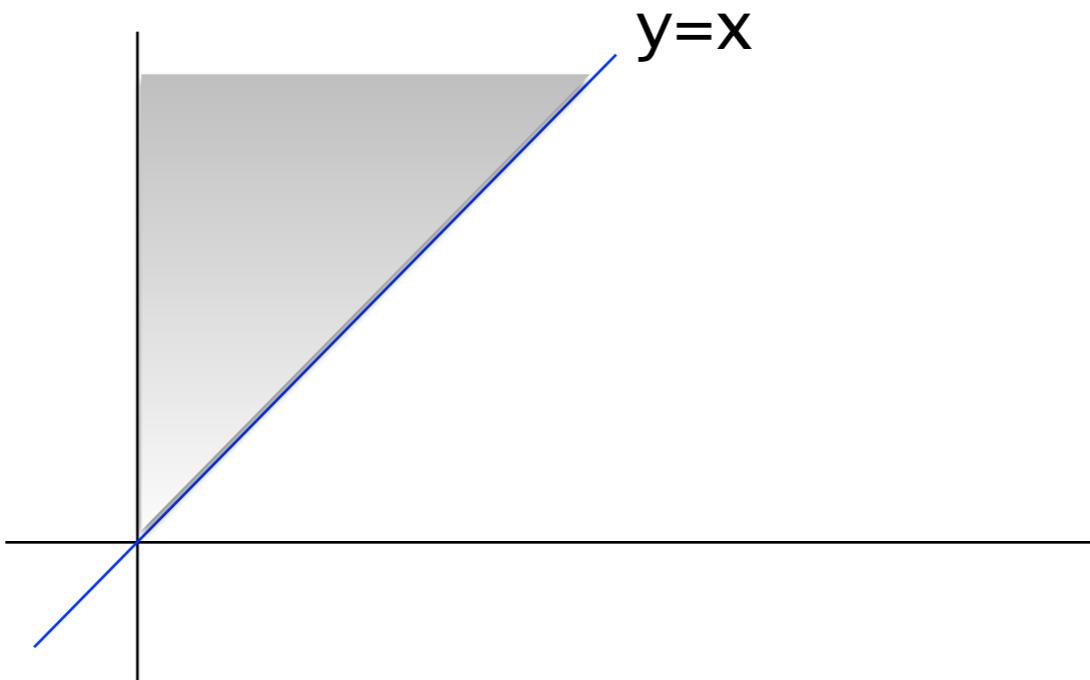
(1,0), (1,1), (1,2), ..., (1,100)

...

(100,0), (100,1), ..., (100,100)

code12.c

- 2차원 평면의 1사분면에서 그래프 $y=x$ 의 위쪽에 있으면서 원점으로부터 거리가 100 이하인 정수 좌표 점의 개수는? x-좌표나 y-좌표가 0이거나 그래프 $y=x$ 상에 있는 점도 포함한다.



code12.c

```
#include <stdio.h>

int main(void)
{
    int count = 0;
    for ( int x=0; x<=100; x++ )    {
        for ( int y=x; y<=100; y++) {
            if (x*x + y*y <= 10000)
                count++;
    }
    printf("The number of points is %d.\n", count);
    return 0;
}
```

두 개의 중첩된 for문을 이용하여 점들을 다음과 같은 순서로 검사한다.

(0,0), (0,1), (0,2), ..., (0,100)

(1,1), (1,2), ..., (1,100)

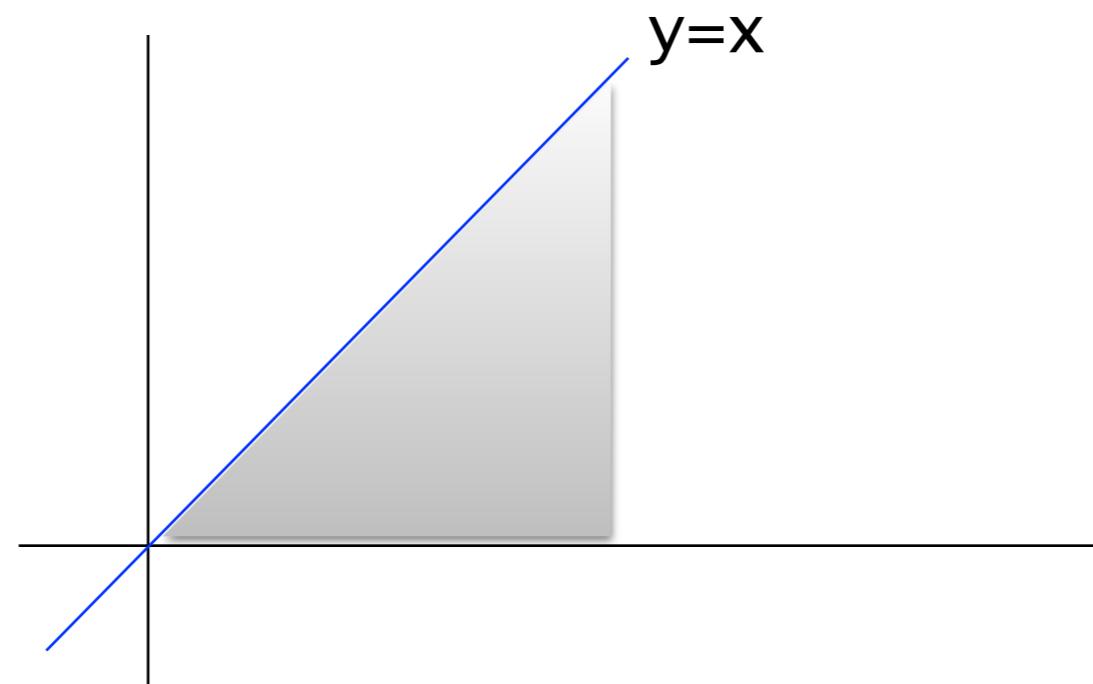
(2,2), ..., (2,100)

...

(100,100)

연습 12

12. 2차원 평면의 1사분면에서 그래프 $y=x$ 의 아래쪽에 있으면서 원점으로부터 거리가 100 이하인 정수 좌표 점의 개수는? x -좌표나 y -좌표가 0이거나 그래프 $y=x$ 상에 있는 점도 포함한다.



code13.c: Divisor Pattern

- 1~N 사이의 모든 정수들 중에서 서로 약수-배수 관계인 정수 쌍들을 모두 찾아 아래 그림과 같이 출력한다.

*	*	*	*	*	*	*	*	1
*	*	*	*	*	*	*	2	
*	*	*	*				3	
*	*	*			*		4	
*			*				5	
*	*	*		*			6	
*				*			7	
*	*	*			*		8	

N=8인 경우

code13.c: Divisor Pattern

```
#include <stdio.h>

int main(void)
{
    int N;
    scanf("%d", &N);
    for (int i = 1; i <= N; i++) {
        for (int j = 1; j <= N; j++) {
            if ((i%j == 0) || (j%i == 0))
                printf("* ");
            else
                printf("  ");
        }
        printf("%d\n", i);
    }
    return 0;
}
```

code14.c

```
#include <stdio.h>

int main(void)
{
    int N;
    scanf("%d", &N);
    int count = 0;
    for ( int i=1; i<=N; i++ )    {
        for ( int j=i+1; j<=N; j++) {
            if (j%i==0)
                count++;
        }
    }
    printf("The number of pairs is %d.\n", count);
    return 0;
}
```

1~N 사이의 모든 정수들 중에서 (약수, 배수)관계인 서로 다른 정수 쌍의 개수는? (a,b)와 (b,a)는 같은 쌍으로 간주하고 (a,a)는 카운트하지 않는다.

연습 13

13. 1~N 사이의 정수들 중 각 자리의 합이 7의 배수인 것들의 개수를 세는 프로그램을 작성하라. 예를 들어 12,345의 각 자리의 합은 $1+2+3+4+5=15$ 이며 7의 배수가 아니다.

code15.c: Prime Numbers

- 1에서 100,000사이의 모든 소수들을 찾아서 출력하는 프로그램을 작성하라.

code15.c: Prime Numbers

```
#include <stdio.h>

int main(void)
{
    for ( int i=2; i<=100000; i++ )      {
        int isPrime = 1;
        for (int j=2; j<=i/2 && isPrime==1; j++)
        {
            if (i%j==0)
                isPrime = 0;
        }
        if (isPrime==1)
            printf("%d\n", i);
    }
    return 0;
}
```

1~100000 사이의 모든 소수들을 찾아서 출력하는 프로그램이다.

C언어에서는 보통 `true`를 정수 1로 `false`를 정수 0으로 표현하곤 한다.

각각의 정수 `i`에 대해서 이 `for` 문을 돌면서 2보다 크거나 같은 약수가 있는지 검사한다. 하나라도 약수가 있다면 이미 소수가 아니므로 더이상 검사할 필요가 없다. 변수 `isPrime`이 어떤 역할을 하는지 잘 생각해보라.

이 프로그램의 오류는?

code15_2.c: Prime Numbers

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

```
int main(void)
{
    printf("2\n");
    for ( int i=3; i<=100000; i++ ) {
        int j=2;
        while (j*j<=i && i%j !=0)
            j++;
        if (j*j>i)
            printf("%d\n", i);
    }
    return 0;
}
```

1~100000 사이의 모든 소수들을 찾아서
출력하는 프로그램이다.

code16.c: Factoring

```
#include <stdio.h>

int main(void)
{
    int n;
    scanf("%d", &n);
    for(int i=2; i*i<=n; i++) {
        while (n % i == 0) {
            n /= i;
            printf("%d ", i);
        }
    }
    if (n > 1)
        printf("%d", n);
    return 0;
}
```

입력으로 하나의 양의 정수 n을 받은 후 소인수분해하는 프로그램을 작성하라.

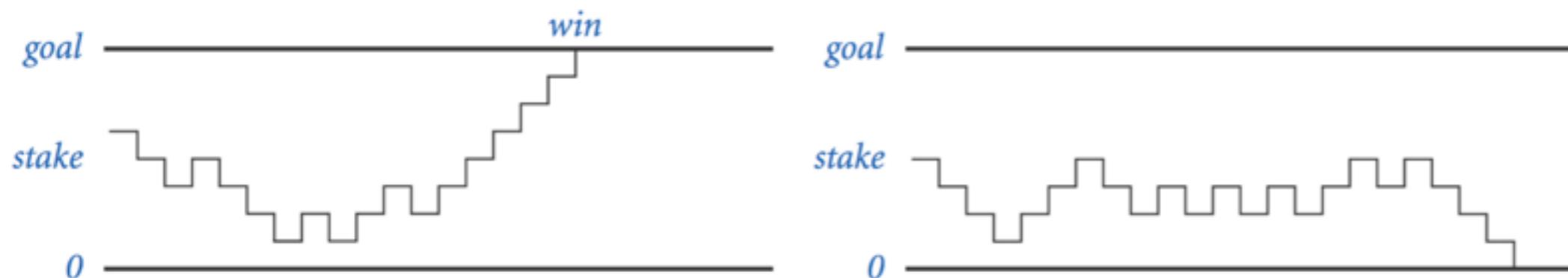
연습 14: 서로소

14. 입력으로 하나의 양의 정수 N 을 받아서 $N \times N$ 크기의 테이블을 출력하는 프로그램을 작성하라. 테이블의 (i -행, j -열)에는 i 와 j 가 서로소이면 *을 출력하고 아니면 빈칸으로 남겨둔다. 서로소란 GCD가 1인 경우를 말한다.

*	*	*	*	*	*	*	*	*	1
*		*		*		*		*	2
*	*		*	*		*	*		3
*		*		*		*		*	4
*	*	*	*		*	*	*		5
*				*		*		*	6
*	*	*	*	*	*	*	*	*	7
*		*		*		*		*	8

Gambler's Ruin Simulation

- 동전을 던져서 앞면이 나오면 \$1을 따고 뒷면이 나오면 \$1을 잃는다.
- 초기자본(stake)과 목표액(goal)이 주어진다.
- 목표액에 도달하면 이기고(win), 돈을 모두 잃으면 진다.
- 게임에 이길 확률을 계산하기 위해서 시뮬레이션을 수행한다.
즉 게임을 T번 반복하여 승률을 출력한다.
- T와 초기자본, 목표액은 입력으로 주어진다.



Gambler's Ruin Simulation

```

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

int main(void) {
    int stake, goal, T;
    scanf("%d %d %d", &stake, &goal, &T);
    int wins = 0;
    for (int t = 0; t < T; t++) {
        int cash = stake;
        while (cash > 0 && cash < goal) {
            if (rand()%2 == 0 ) cash++;
            else cash--;
        }
        if (cash == goal)
            wins++;
    }
    printf("%d%% wins\n", 100*wins/T);
    return 0;
}

```

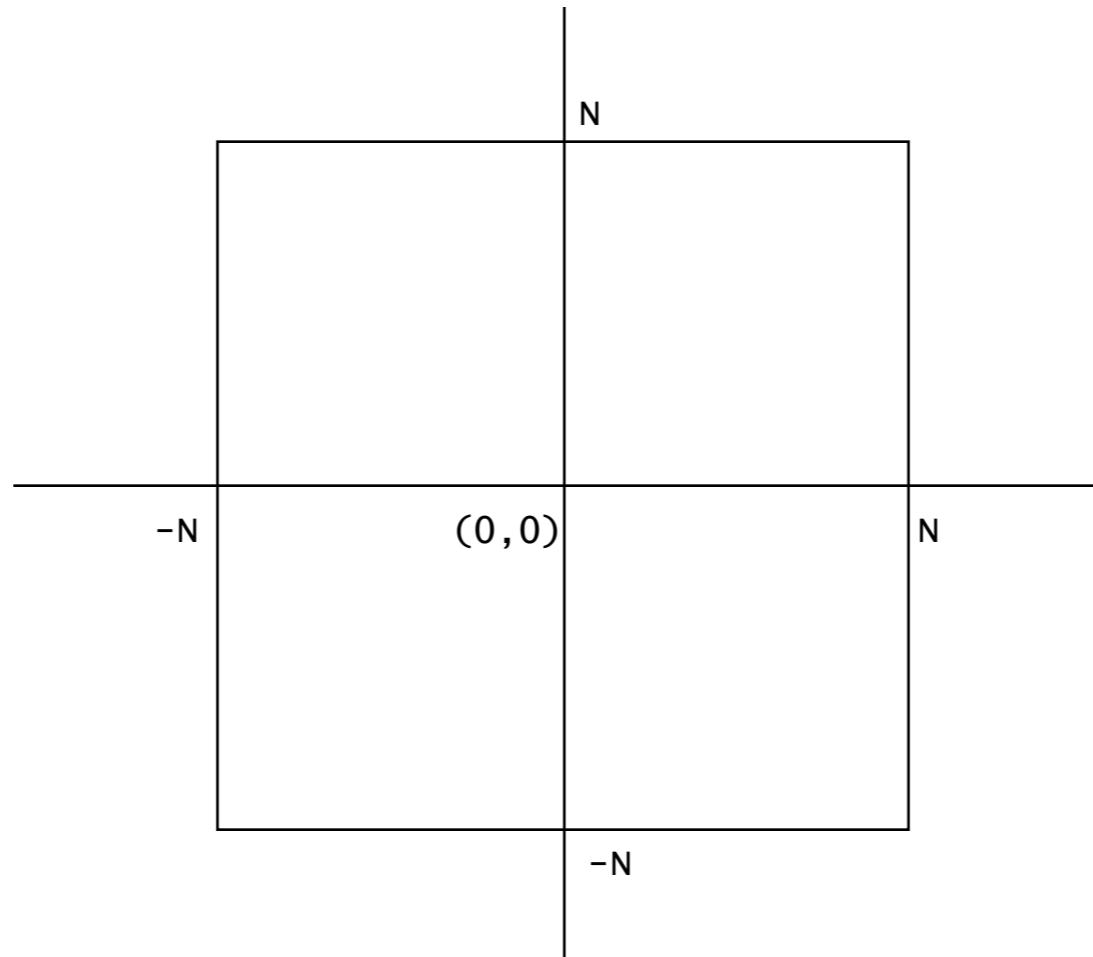
stdlib가 제공하는 rand()함수는 0에서 RAND_MAX 사이의 정수를 랜덤하게 생성해준다.

연습 15: Sum of Two Cubes

15. 입력으로 하나의 양의 정수 N 을 받는다. N 이하의 양의 정수들 중에 적어도 두 가지 서로 다른 “세제곱의 합”으로 표현될 수 있는 정수들을 찾아서 출력하는 프로그램을 작성하라. 즉 $K=a^3+b^3=c^3+d^3$ 이 되는 서로 다른 두 정수쌍 (a, b) 와 (c, d) 가 존재하는 N 이하의 정수 K 를 모두 찾아서 출력한다. 여기서 a, b, c, d 는 모두 양의 정수이다. 예를 들어 1729는 이런 조건을 만족하는 가장 작은 양의 정수이다.

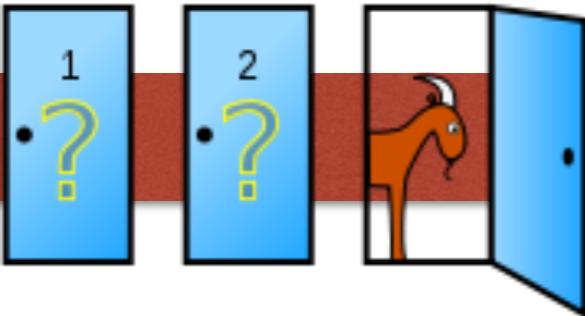
연습 16: 2D Random Walk

- 원점 $(0, 0)$ 에서 출발하여 매 스텝마다 동서남북 4방향 중 한 방향을 각각 $1/4$ 의 확률로 랜덤하게 선택하여 1만큼 움직인다. x -좌표나 y -좌표가 N 이나 $-N$ 에 도달하면 종료한다. 얼마 만큼의 스텝이 필요한지 평가하는 프로그램을 작성하라. 즉 동일한 실험을 T 번 반복하여 가장자리에 도달하는데 걸린 스텝수의 평균을 구하라. N 과 T 는 입력으로 주어진다.



연습 17: Pepys problem

16. 1693년에 Samuel Pepys는 뉴튼에게 다음 중 어느 쪽이 더 일어날 확률이 높은지 질문하였다: “주사위를 6번 던져서 적어도 한 번 1이 나오는 것과 주사위를 12번 던져서 적어도 두 번 1이 나오는 것”. 이 질문에 대한 답을 시뮬레이션을 통해 알아내는 프로그램을 작성하라.



연습 18: Monty Hall Problem

17. Monty Hall 문제는 위키피디아에도 한자리를 차지하고 있고 아직도 인터넷 상에서 논쟁이 벌어지곤 하는 유명한 문제이다. 1970년대 “Let’s make a deal”이라는 TV쇼에서 유래한 문제이다. 세 개의 문이 있고 그 중 하나의 문 뒤에는 스포츠카가 있고 다른 두 개의 문 뒤에는 염소가 있었다고 한다. 진행자는 어떤 문 뒤에 스포츠카가 있는지 알고 있다. 게임의 참가자는 먼저 3개의 문 중 하나를 선택한다. 그러면 진행자는 나머지 두 개의 문들 중에서 염소가 있는 문을 하나 개방한다. 그런 다음 참가자에서 자신의 선택을 바꿀 기회를 한 번 준다. 즉 자신이 선택한 문과 아직 열리지 않은 남아있는 문 중에서 선택을 바꿀 기회를 주는 것이다. 이때 참가자가 선택을 바꾸는 것이 이길 확률에 영향을 미칠까? 이 문제에 대한 답을 알기 위한 시뮬레이션 프로그램을 작성하라. 즉 T번 실험을 반복하여 하여 선택을 바꾸는 경우와 바꾸지 않는 두 경우의 승률을 계산하는 프로그램을 작성하라.