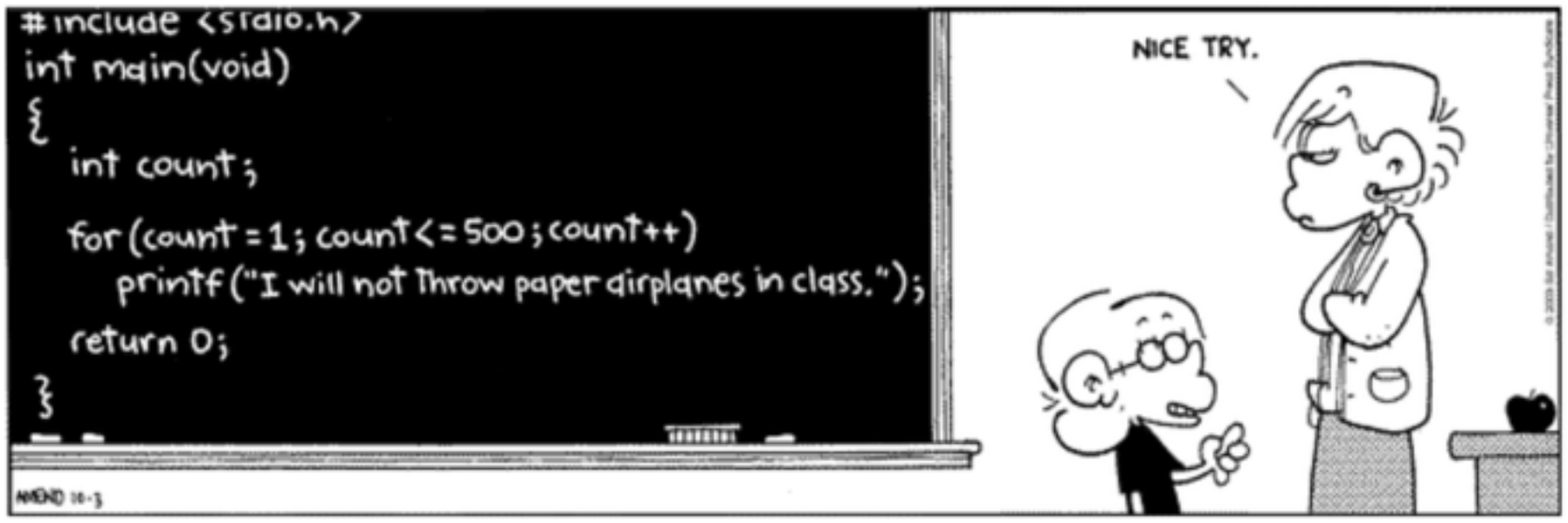


제1장

변수와 흐름제어

Programming Language

문법과 그 의미에 대한 이해 및 숙련

Problem Solving Skills

논리적 사고. 자료구조와 알고리즘에 대한 이해

변수와 치환

Variables and Assignment

문제 01

- **1에서 100까지 정수들의 합을 구해 출력하는 프로그램을 작성하라.**

1에서 100까지 더하기

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

code01.c

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

```
#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;
}
```

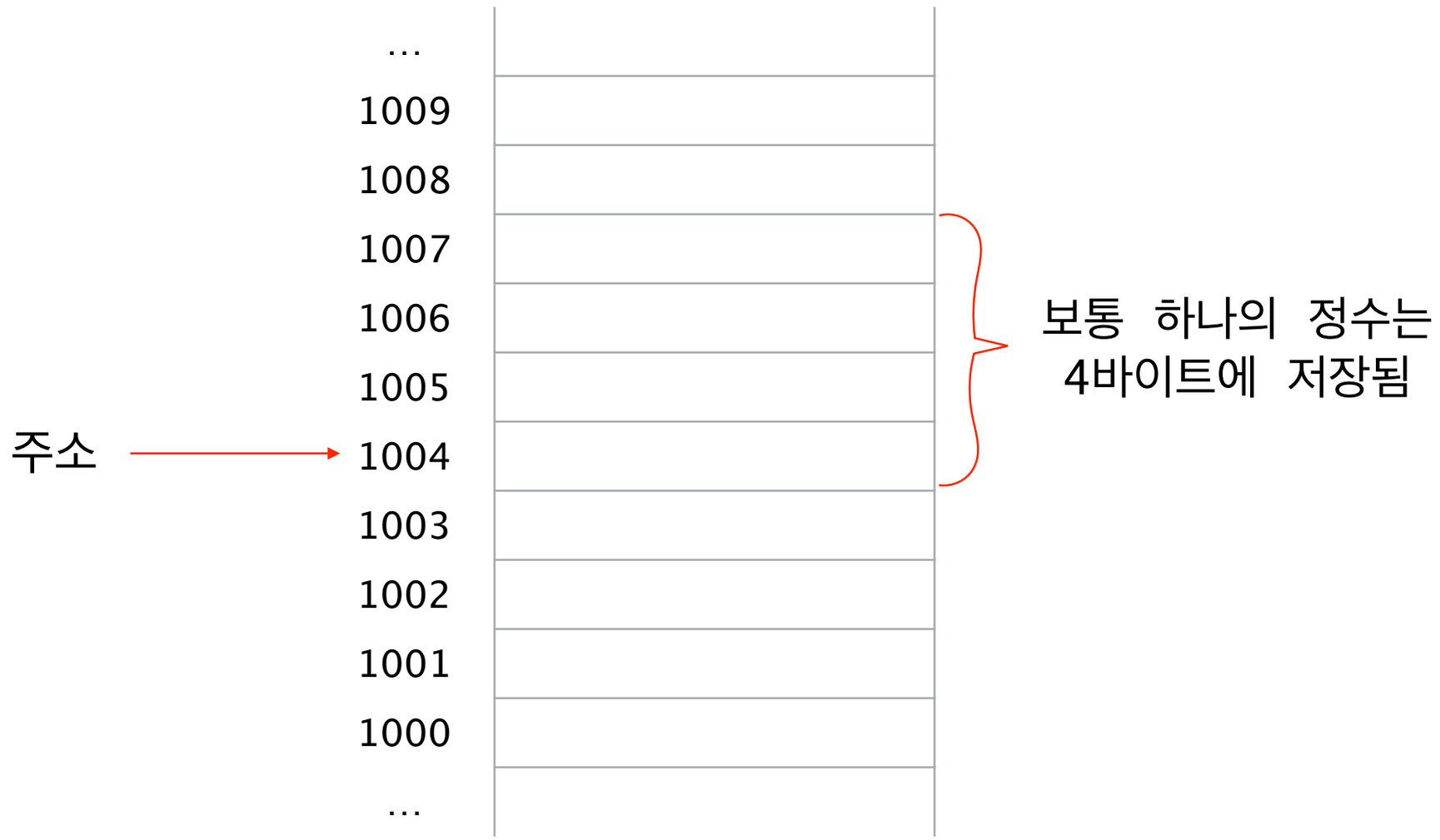
main 함수는 프로그램 실행이 시작되는 곳이다.

sum과 i를 변수라고 부른다.

화면에 문자열을 출력한다. 출력하고 싶은 문자열을 겹따옴표(“”)로 묶는다. ‘\n’은 줄바꿈 문자이다.

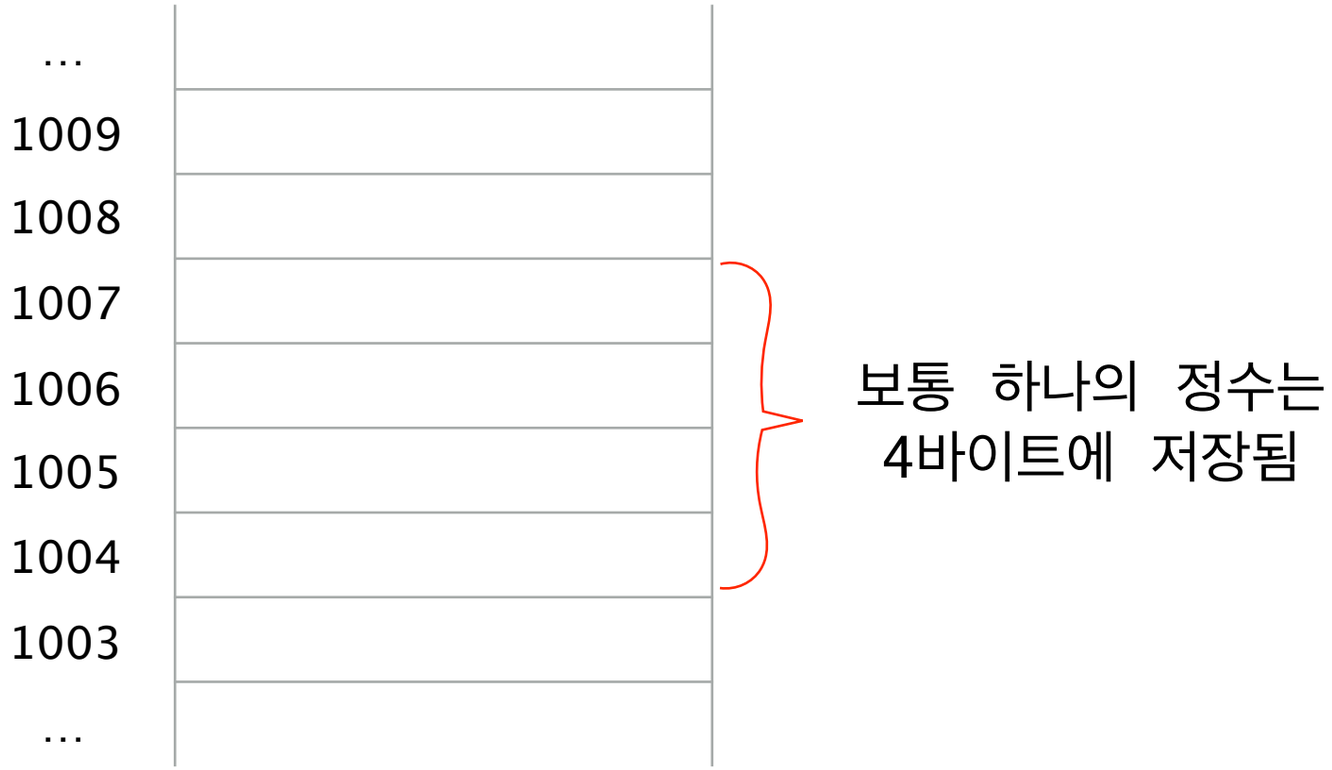
메모리 (RAM)

- 컴퓨터의 메모리는 데이터를 보관하는 장소
- 바이트(8 bits) 단위로 주소가 지정된 거대한 테이블



메모리 (RAM)

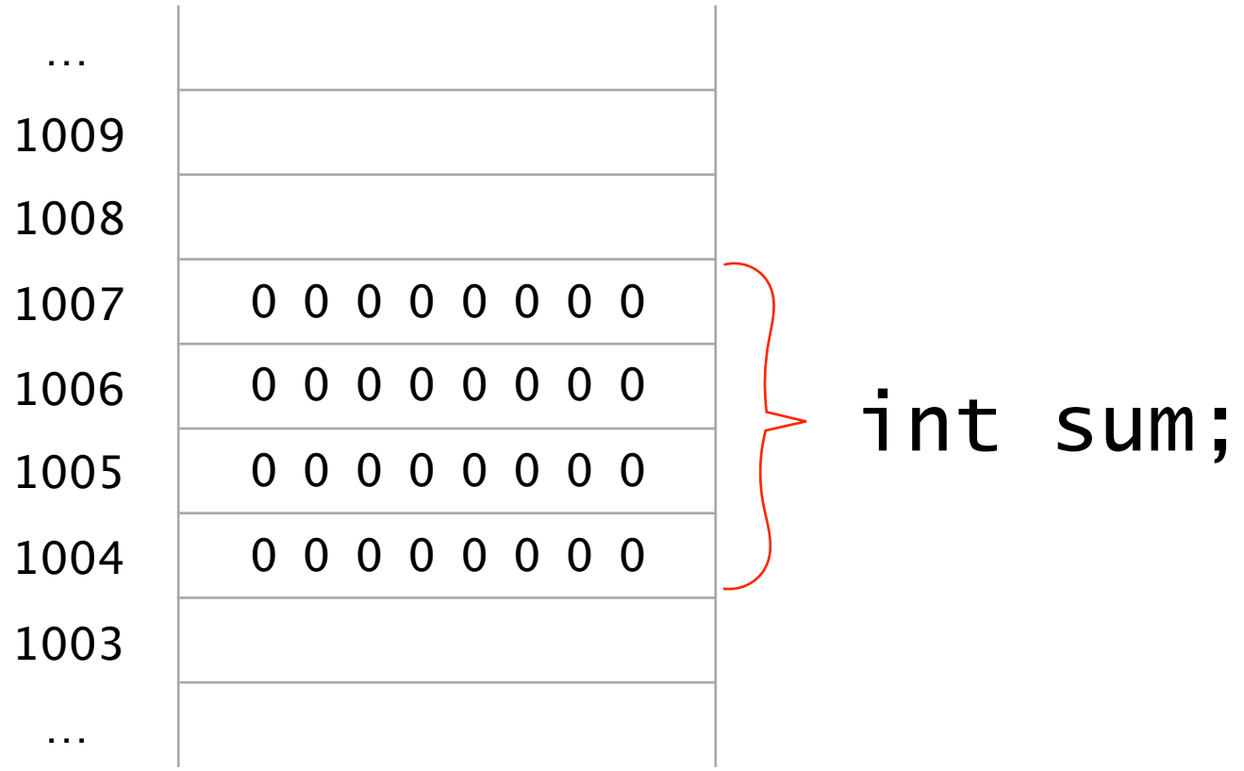
- 기계어나 어셈블리 언어 프로그램에서는 직접 메모리 주소를 사용



LOAD R0 1004

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

- C와 같은 고급언어 프로그램에서는 메모리 주소 대신 변수를 사용



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 위해서

타입 이름	설명
int	정수 (integer)
float	실수 (floating-point number)
double	실수 (double precision number)
char	문자 (character)
char *	문자열 (string)
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 !

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

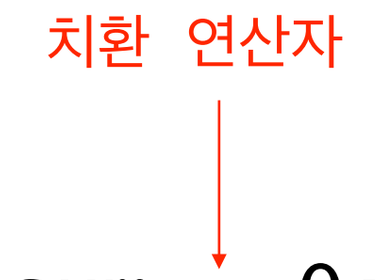
code01.c

```
#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.

치환문



- 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.

치환문

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;	

치환문

```
int x, y;
y = 2*x + 1;
for ( x = 0; x < 10; x++ )
    printf("x = %d, y = %d\n", x, y);
```

흐름제어

for, if-else, while

code01.c

```
#include <stdio.h>

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

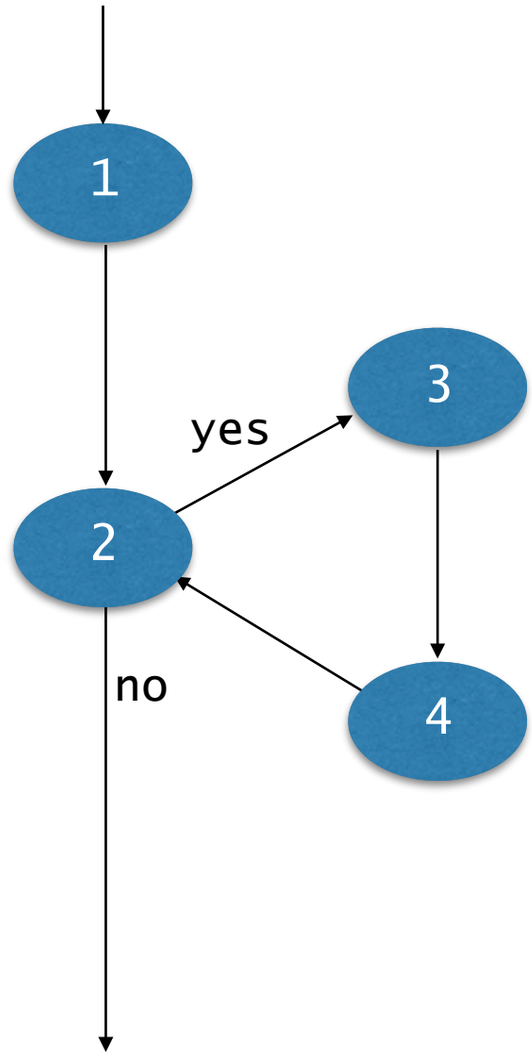
2) 이 조건을 검사하여 만족되면 statements를 한 번 실행하고, 아니면 for문을 벗어난다.

1) 처음에 한 번 실행하고 잊어버린다.

```
for ( statement1; condition; statement2 )
{
    statements;
}
```

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

4) statements가 실행되고 나면 자동으로 한 번 실행된다. 그런 후 다시 condition을 검사하러 간다.



code01.c

```
#include <stdio.h>

int main(void)
{
    int sum = 0;

    for (int 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;

    for (int 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.

문제 02

- 각도 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);
    }
}
```

실수 값을 저장하기 위해서 double형 변수들을 선언하였다.

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 라이브러리를 include한다.

code02.c

```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.

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 아니라 몫을 구한다.

실습1: for 반복문, 실수 계산

Enough Talk
Let's Code

- 실수 계산
- for 반복문 연습
- 실습 문제는 별개의 파일로 주어짐

문제 03

- 두 정수 s와 t를 입력받아 (단 s≤t) s에서 t 사이의 정수들 중에서 2 혹은 3의 배수인 것들의 합을 계산하여 출력한다.
- 2 혹은 3의 배수인 것들과 그렇지 않은 것들의 개수를 각각 구하여 출력한다.

code03.c

```
#include <stdio.h>

int main(void)
{
    int s, t, sum = 0;
    scanf(“%d”, &s);
    scanf(“%d”, &t);

    for (int i=s; i<=t; 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.

code03_2.c

```
#include <stdio.h>

int main(void)
{
    int s, t, sum = 0;
    scanf(“%d”, &s);
    scanf(“%d”, &t);
    int count1 = 0, count2 = 0;

    for (int i=s; i<=t; i++)  {
        if (i%2==0 || i%3==0)
            count1 = count1 + 1;
        else
            count2++;;
    }

    printf("The results are %d %d.\n”, count1, count2);
}
```

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	`if (x < 0)` `  x = -x;`

x와 y를
크기순으로 정렬
	`if (x > y) {`
	`  int t = x;`
	`  x = y;`
	`  y = t;`
	`}`

x와 y중 큰 값	`int max;`
	`if (x > y)`
	`  max = x;`
	`else`
	`  max = y;`

실습2: 조건문

Enough Talk
Let's Code

- if - else 조건문 연습

문제 04

- 키보드로부터 여러 개의 정수를 순차적으로 입력 받는다. 짝수가 입력되면 무시하고 홀수가 입력되면 더해 나간다. 더해진 홀수의 개수가 10개가 되면 합을 출력하고 종료한다.
- 키보드로 부터 연속해서 정수를 입력받아 합을 구한다. 사용자가 -1을 입력하면 합을 출력하고 종료한다.
- 키보드로 부터 연속해서 정수(음수일 수도 있음)를 입력받아 합을 구한다. 합이 0이되면 입력된 정수의 개수를 출력하고 종료한다.

code04.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.

code04_2.c

```
#include <stdio.h>
void main() {
    int count = 0, sum = 0;
    int tmp;
    while(1) {
        scanf(“%d”, &tmp);
        if (tmp%2==1)  {
            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.

break문은 자신을 둘러싼 가장 안쪽 루프를 빠져 나간다.

code04_3.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);
}
```

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

code04_4.c

- 키보드로 부터 연속해서 정수를 입력받아 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;
    }
}
```

문제 05

- 입력으로 하나의 양의 정수 **N**을 받은 후 각 자리수의 합을 구하여 출력하는 프로그램을 작성하라.
- 입력으로 하나의 양의 정수 **N**을 받은 후 **2**진수로 변환했을 때 **1**의 개수를 구하여 출력하는 프로그램을 작성하라. 예를 들어 **N=13**이면 $13=1101_2$이므로 **1**은 **3**개이다.

code5.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;
}
```

code5_2.c

```
#include <stdio.h>

int main(void)
{
    int N;
    scanf(“%d”, &N);
    int count = 0;
    while (N>0) {
        count += (N%2);
        N /= 2;
    }
    printf(“The number of 1 is %d.\n”, count);
    return 0;
}
```

문제 06

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

code06.c

```
#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.

문제 07: 2진수로 변환하기

- 입력으로 양의 정수 N을 받아서 2진수로 변환하여 출력하는 프로그램을 작성하라
- 예를 들어서 115는 다음과 같이 2진수로 변환할 수 있다.

	v	N		Binary
2^6	64	115	115 ≥ 64	1
2^5	32	115-64 = 51	51 ≥ 32	1
2^4	16	51-32 = 19	19 ≥ 16	1
2^3	8	19-16 = 3	3 < 8	0
2^2	4	3	3 < 4	0
2^1	2	3	3 ≥ 2	1
2^0	1	3-2 = 1	1 ≥ 1	1

문제 07: 2진수로 변환하기

```
#include <stdio.h>

int main(void) {
    int N;
    scanf("%d", &N);

    /* N보다 작거나 같으면서 가장 큰 2의 거듭제곱수 v를 구한다. */
    while (v > 0) {

        /* N ≥ v이면 N = N-v가되고   */
        /* 1을 출력한다. 그렇지 않으면 */
        /* 0을 출력한다. 두 경우 모두  */
        /* v는 1/2한다.               */

    }

}
```

	v	N		Binary
2^6	64	115	115 ≥	1
2^5	32	115-64 = 51	51 ≥ 32	1
2^4	16	51-32 = 19	19 ≥ 16	1
2^3	8	19-16 = 3	3 < 8	0
2^2	4	3	3 < 4	0
2^1	2	3	3 ≥ 2	1
2^0	1	3-2 = 1	1 ≥ 1	1

문제 08: 최대공약수(GCD)

- 입력으로 두 양의 정수를 받은 후 두 정수의 최대공약수(GCD)를 구해서 출력하는 프로그램을 작성하라. GCD를 구하기 위해서 Euclid 알고리즘을 사용하라. Eulid 알고리즘은 다음의 성질을 이용한다: 두 정수 x,y중에 크거나 같은 쪽을 x라고 하자. 만약 x가 y로 나누어 떨어지면 GCD는 y이다. 그렇지 않다면 x와 y의 GCD는 x%y와 y의 GCD와 같다.

실습3: while 반복문

Enough Talk
Let's Code

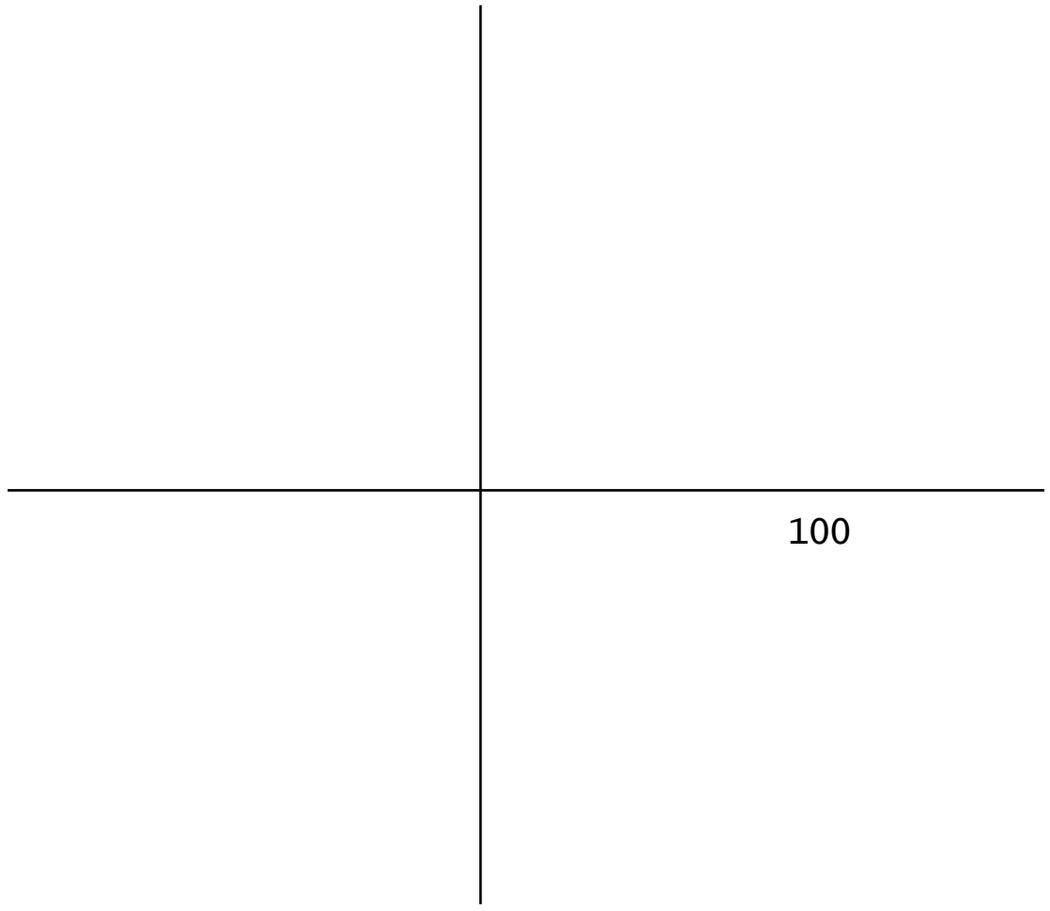
- while 반복문 연습
- 반복문의 다양한 형태

중첩된 루프

Nested Loops

문제 9: 2차원 평면에서

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



code9.c

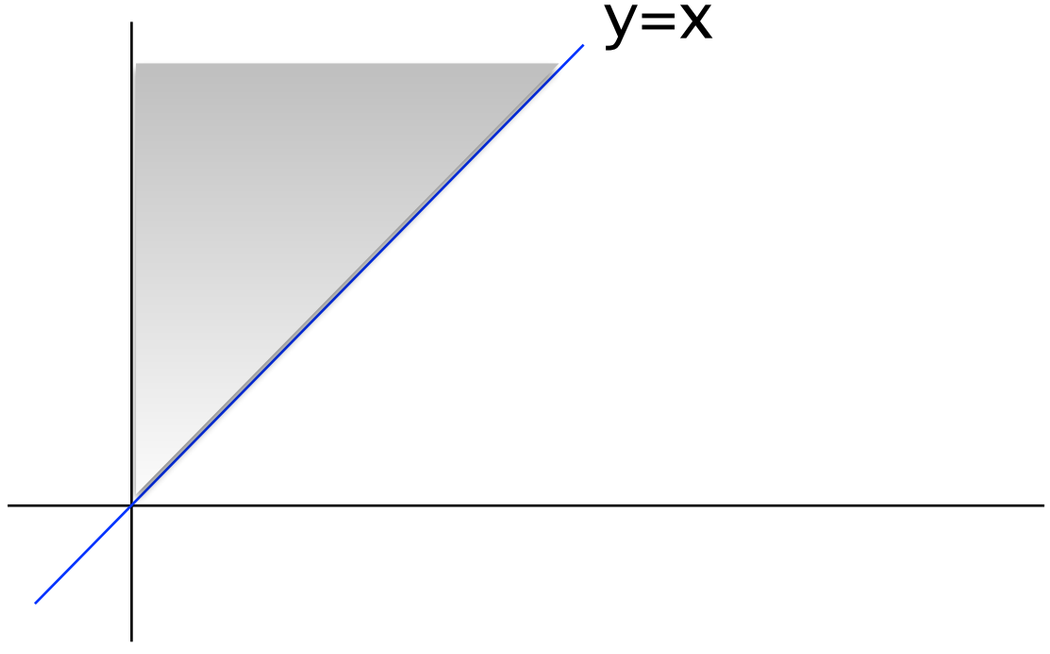
두 개의 중첩된 for문을 이용하여 점들을 다음과 같은 순서로 검사한다.

(0,0), (0,1), (0,2),…,(0,100)
(1,0), (1,1), (1,2),…,(1,100)
…
(100,0), (100,1),…, (100,100)

```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;
}
```

문제 10

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



code10.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)

문제 11: Divisors

- **1~N 사이의 정수들 중에서 서로 약수-배수 관계인 정수 쌍의 개수를 계산해 출력하라. (a,b)와 (b,a)는 같은 쌍으로 간주하고 (a,a)는 카운트하지 않는다.**

code11.c: Divisors

```
#include <stdio.h>

int main(void)
{
    int N;
    scanf(“%d”, &N);
    int count = 0;
    for (int i = 2; i <= N; i++)  {
        for (int j = 1; j < i/2; j++)  {
            if (i%j == 0)
                count++;
        }
    }
    printf(“%d\n”, count);
    return 0;
}
```

문제 12: Sum of Two Cubes

- **10,000,000**이하의 양의 정수들 중에 적어도 두 가지 서로 다른 "세제곱의 합"으로 표현될 수 있는 모든 정수를 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 모든 정수 K를 찾아서 출력한다. 여기서 a,b,c,d는 모두 양의 정수이다. 중복 출력되어도 상관없다.

문제 12: Sum of Two Cubes

```
#include <stdio.h>

int main()
{
    for (int a=1; a*a*a<10000000; a++) {
        for (int c=a+1; c*c*c<10000000; c++) {
            for (int d=c; d*d*d<10000000; d++) {
                for (int b=d+1; b*b*b<10000000; b++) {
                    if (a*a*a + b*b*b == c*c*c + d*d*d) {
                        printf("%d: %d %d %d %d\n", a*a*a+b*b*b, a, b, c ,d);
                    }
                }
            }
        }
    }
}
```

문제 13: 소수 (prime number)

- **1~100000** 사이의 모든 소수들을 찾아서 출력하라.

code13.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;
}
```

C언어에서는 보통 true를 정수 1로 false를 정수 0으로 표현하곤 한다.

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

code13_2.c: Prime Numbers

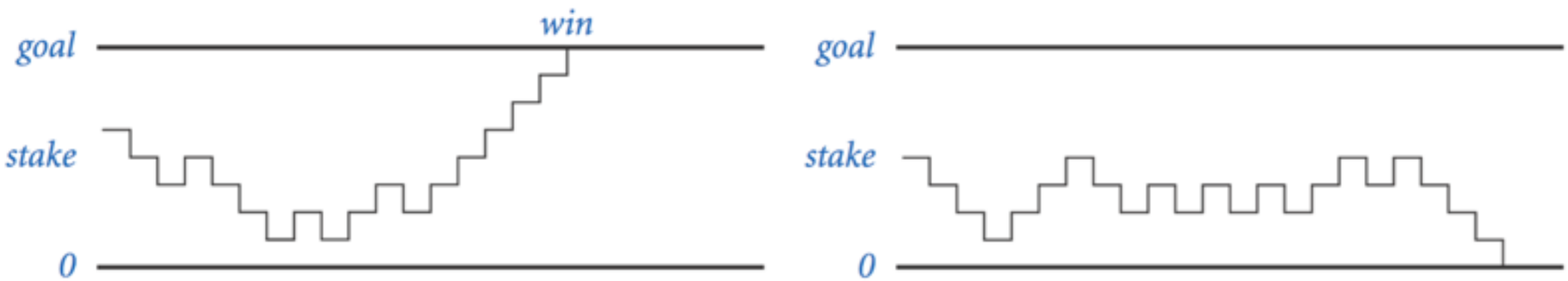
1~100000 사이의 모든 소수들을 찾아서 출력하는 프로그램이다.

```
#include <stdio.h>

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

문제 14: Gambler's Ruin Simulation

- 동전을 던져서 앞면이 나오면 **$1**을 따고 뒷면이 나오면 **$1**을 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 **T**번 반복하여 승률을 출력한다.

- **T**와 초기자본, 목표액은 입력으로 주어진다.



code14.c

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

int main(void)  {
    int stake, goal, T;
    srand((unsigned int) time(NULL));
    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.

실습4: 중첩된 반복문

Enough Talk
Let's Code

- 중첩된 반복문 연습
- random number generation
- 고급 예제들