

Basic Computer Programming

Lecture 6

Electrical & Electronics Engineering
Chung-Ang University

Contents

- Understand and use the while iterative structure.
- Understand and use the do-while iterative structure.
- Understand and use the for iterative structure.
- Understand and use the overlapping iterative structure.
- Understand how to use break and continue in iterations.

The program we are going to make in this chapter.

- Let's create a program that automatically presents arithmetic questions for elementary school students. If you get hit once, the program will be over.



```
Microsoft Visual Studio 디버그 콘솔
산수 문제를 자동으로 출제합니다.
41 + 67 = 109
틀렸습니다.
34 + 0 = 34
맞았습니다.
```

- Let's calculate which is more advantageous, when you receive 100 million won, or when you receive 1 won at first, and double every day for a month.



```
Microsoft Visual Studio 디버그 콘솔
26일날 현재 금액=134217728.000000
27일날 현재 금액=268435456.000000
28일날 현재 금액=536870912.000000
29일날 현재 금액=1073741824.000000
30일날 현재 금액=2147483648.000000
```

The program we are going to make in this chapter.

- Let's write a number matching game.



```
Microsoft Visual Studio 디버그 콘솔
정답을 추측하여 보시오: 50
제시한 정수가 높습니다.
정답을 추측하여 보시오: 25
제시한 정수가 높습니다.
정답을 추측하여 보시오: 12
제시한 정수가 높습니다.
정답을 추측하여 보시오: 6
제시한 정수가 낮습니다.
정답을 추측하여 보시오: 9
제시한 정수가 낮습니다.
정답을 추측하여 보시오: 10
제시한 정수가 낮습니다.
정답을 추측하여 보시오: 11
축하합니다. 시도횟수=7
```

Iterations

- Iterations causes a step to be repeated, and using a iteration structure makes the program simple and fast. For example, to repeat the same task, using a repeating structure makes the program simpler than copying and pasting the same sentence.

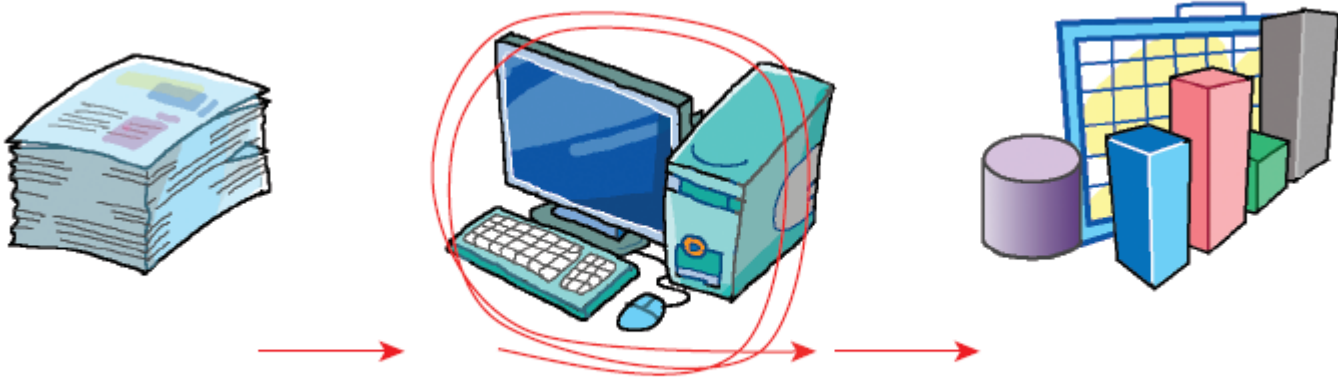


그림 6.1 반복은 같은 처리 과정을 반복하는 것이다.

A program that converts miles into meters

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

```
int main(void)
```

```
{
```

```
    int meter;
```

```
    meter = 0 * 1609;  
    printf("0 마일은 %d미터입니다\n", meter);
```

```
    meter = 1 * 1609;  
    printf("1 마일은 %d미터입니다\n", meter);
```

```
    meter = 2 * 1609;  
    printf("2 마일은 %d미터입니다\n", meter);
```

```
    return 0;
```

```
}
```

Same process
#1

Same process
#2

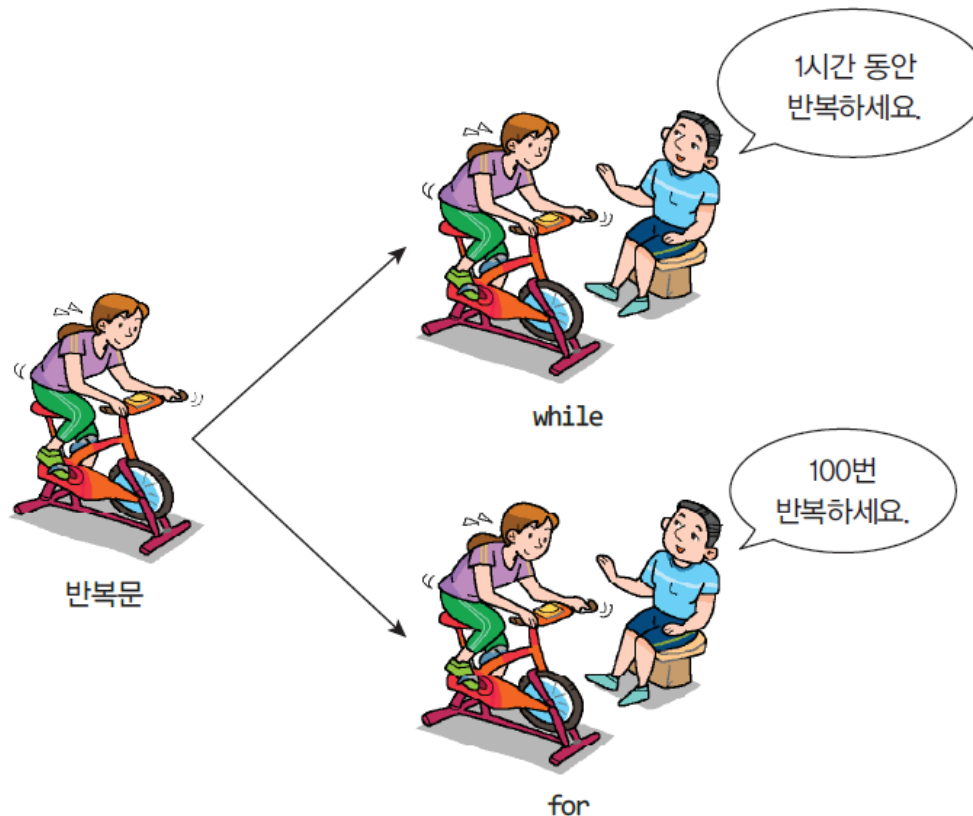
Same process
#3

Microsoft Visual Studio 디버그 콘솔

```
0 마일은 0미터입니다  
1 마일은 1609미터입니다  
2 마일은 3218미터입니다
```

Types of iterations

- Condition control iteration (while statement): If a specific condition is satisfied, it is repeated continuously.
- Repeated number of times (for statement): Repeated a predetermined number of times.



Interim check

1. Why does the program need a repetitive structure?
2. Repeating statements include _____, _____.



while statement

- The while statement executes the sentences repeatedly if the conditional expression is true. The grammar of the while statement is as follows.

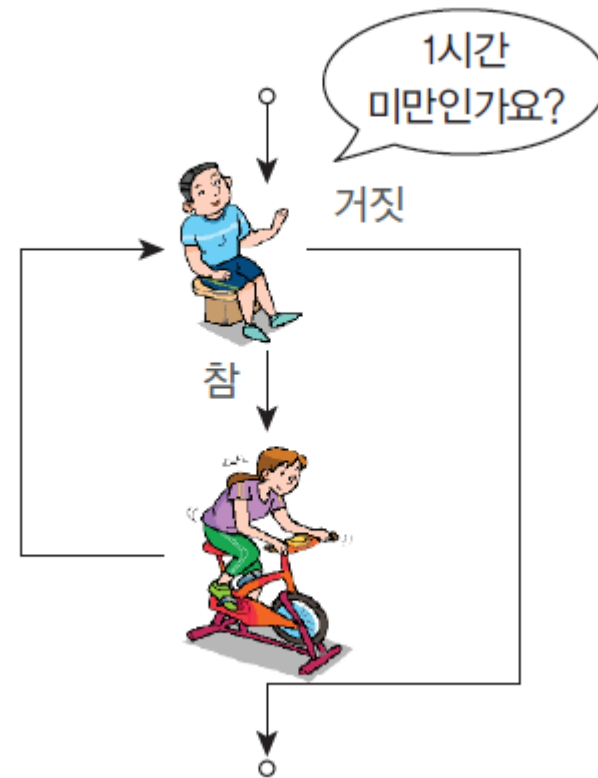
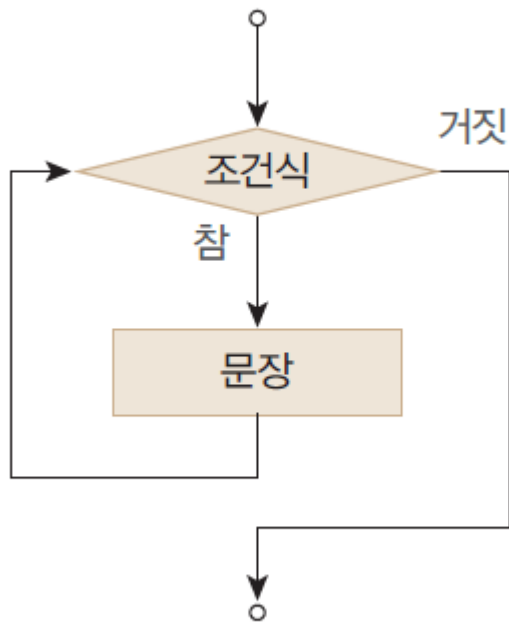
Syntax 6.1 while 문

형식 `while(조건식) {
 문장(들);
}`

예 `while(i < 10) {
 printf("Hello World!\n");
 i++;
}`

설명 조건식이 참이면 문장을 반복 실행한다.
반복되는 문장이 하나이면 중괄호는 생략할 수 있다.

Flowchart of while statement



Write "I won't make any noise" 10 times on the blackboard.

```
#include <stdio.h>
int main(void)
{
    int i = 0;
    while (i < 10)
    {
        printf("수업 시간에 떠들지 않습니다.\n");
        i++;
    }
    return 0;
}
```



Microsoft Visual Studio 디버그 콘솔

다다다다다다다다다다
스스스스스스스스스스
게게게게게게게게게게
아아아아아아아아아아
지지지지지지지지지지
떠떠떠떠떠떠떠떠떠떠
에에에에에에에에에에
간간간간간간간간간간
시시시시시시시시시시
진진진진진진진진진진

Output of repeated control variables

```
#include <stdio.h>

int main(void)
{
    int i = 0;
    while (i < 10)
    {
        printf("i=%d ", i);
        i++;
    }
    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

i=0 i=1 i=2 i=3 i=4 i=5 i=6 i=7 i=8 i=9

Convert miles into meters

```
#include <stdio.h>

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

    while (i < 3)
    {
        meter = i * 1609;
        printf("%d 마일은 %d미터입니다\n", i, meter);
        i++;
    }
    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
0 마일은 0미터입니다
1 마일은 1609미터입니다
2 마일은 3218미터입니다
```

Finding a Factual

- *Factorial*

- It refers to the multiplication of all integers from 1 to a certain positive integer n , expressed as $n!$
- $0! = 1$.
- (예) $3! = 3 \times 2 \times 1$

- $n! = n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$



Finding a Factual

```
#include <stdio.h>

int main(void)
{
    int i = 5;
    int factorial = 1;

    while (i >= 1)    // i를 감소시키면서 i가 1이상이면 반복한다.
    {
        factorial *= i;
        i--;
    }
    printf("%d \n", factorial);

    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

120

Multiplication output

```
// while 문을 이용한 구구단 출력 프로그램
#include <stdio.h>

int main(void)
{
    int n;
    int i = 1;

    printf("구구단 중에서 출력하고 싶은 단을 입력하시오: ");
    scanf("%d", &n);

    while (i <= 9)
    {
        printf("%d*%d = %d \n", n, i, n*i);
        i++;
    }

    return 0;
}
```



```
구구단 중에서 출력하고 싶은 단을 입력하시오: 9
9*1 = 9
9*2 = 18
9*3 = 27
....
9*9 = 81
```

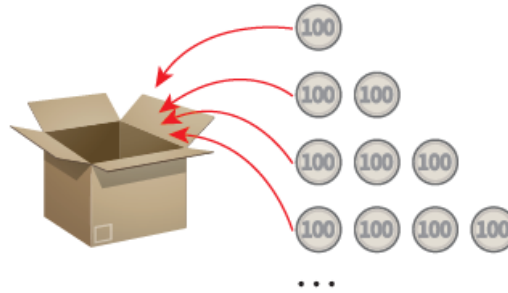

Sum of 1 to 1000

- A program for calculating the sum of 1 to n

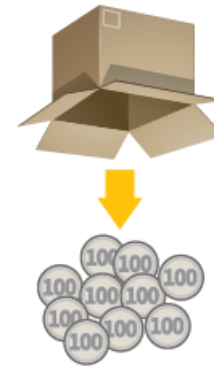
① 빈통을 준비한다.



② 통에 1부터 n까지를 넣는다.



③ 통에 들어있는 동전의 개수를 출력한다.



Sum of 1 to 1000

```
#include <stdio.h>

int main(void)
{
    int i, sum;

    i = 1;
    sum = 0;
    while (i <= 1000)
    {
        sum += i;
        i++;
    }
    printf("합은 %d입니다.\n", sum);
    return 0;
}
```

선택 Microsoft Visual Studio 디버그 콘솔

합은 500500입니다.

Interim check



중간점검

1. `if` 문과 `while` 문을 비교하여 보라. 똑같은 조건이라면 어떻게 동작하는가?
2. `while` 루프를 이용하여 무한 루프를 만들어 보라.
3. 다음 코드의 출력을 쓰시오.

```
int n = 10;
while (n > 0) {
    printf("%d\n", n);
    n = n - 3;
}
```

Lab: origami

- In this exercise, we will fold A4 paper and calculate how many times you have to fold it to reach the height of Mt. Everest (8,800 meters). Let's say the thickness of the paper is 1mm. The thickness of the paper doubles each time it folds. And suppose that the paper can be folded without restrictions on collection. The log function is not used.



Sol: origami

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

int main(void) {
    double pheight = 0.001; // 단위 미터
    const double everest = 8800.0; // 단위 미터
    int count = 0;

    while (pheight < everest) {
        pheight *= 2.0;
        count++;
    }
    printf("종이 접는 횟수=%d \n\n", count);
    return 0;
}
```



Microsoft Visual Studio 디버그 콘솔

종이 접는 횟수=24

Lab: a digital clock.

- There is always a clock on the computer. Let's make one, too. All we need is a function that tells us that a second has passed. You can use `sleep()` on UNIX, and `Sleep()` on Windows. `Sleep(ms)` stops a program that runs for a given time.



Sol: a digital clock.

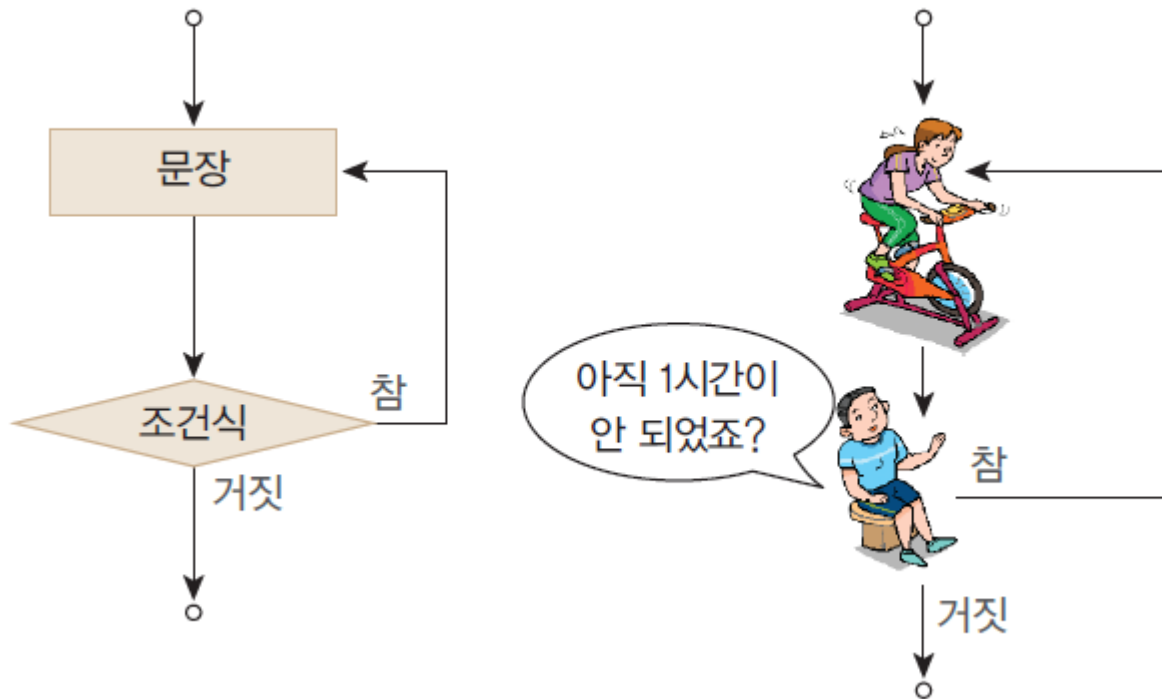
```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

int main(void)
{
    int hour, min, sec;
    hour = min = sec = 0;

    while (1) {
        system("cls");    // 화면을 지운다.
        printf("%02d: %02d: %02d", hour, min, sec);
        sec++;
        if (sec == 60) { min++;      sec = 0; }
        if (min == 60) { hour++;    min = 0; }
        if (hour == 24) { hour = min = sec = 0; }
        Sleep(1000);      // 1초 동안 프로그램을 재운다.
    }
    return 0;
}
```

do...while statement

- The do-while statement is similar to the while statement, but the difference is that the repeating condition is examined at the end of the loop, not at the beginning of the loop.



do-while statement

- It execute commands at least once.

Syntax 6.2 do-while 반복문

형식 `do {`
 문장(들);

 `} while(조건식);`

예 `do {`
 `sum = sum + i;`
 `i++;`
 `} while(i < 3);`

설명 일단 문장을 실행한 후에 조건을 검사하여 반복 여부를 결정한다.

Example #1

- The do..while statement is often used in the processing of inputs. For example, it is used in a part that receives a user's input after outputting a menu as follows.



```
Microsoft Visual Studio 디버그 콘솔
1---파일열기
2---파일저장하기
3---종료
하나를 선택하시오: 1
선택된 메뉴=1
```

Example #1

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

int main(void)
{
    int i = 0;
    do
    {
        printf("1---파일열기\n");
        printf("2---파일저장하기\n");
        printf("3---종료\n");
        printf("하나를 선택하시오: ");
        scanf("%d", &i);
    } while (i < 1 || i > 3);

    printf("선택된 메뉴=%d\n", i);
    return 0;
}
```

Interim check



중간점검

1. 다음 코드의 출력을 쓰시오.

```
int n = 0;
while (n > 0) {
    printf("%d\n", n);
    n = n - 3;
}
```

2. 1번 문제의 반복 구조를 **do-while**로 변경하면 출력이 어떻게 변화되는가?



Lab: number guessing game

- A game in which users guess an integer
- When the user presents an answer, the program only informs whether the presented integer is higher or lower than the restored integer.



Algorithm

do

사용자로부터 숫자를 guess로 입력받는다.

시도횟수를 증가한다.

if(guess < answer)

숫자가 낮다고 출력한다.

if(guess > answer)

숫자가 높다고 출력한다.

while(guess != answer);

"축하합니다"와 시도횟수를 출력한다.

```
#define _CRT_SECURE_NO_WARNINGS
```

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

```
int main(void)
```

```
{
```

```
    int answer;    // 정답
```

```
    int guess;
```

```
    int tries = 0;
```

```
    srand(time(NULL));
```

```
    answer = rand() % 100;
```

```
    // 반복 구조
```

```
    do {
```

```
        printf("정답을 추측하여 보시오: ");
```

```
        scanf("%d", &guess);
```

```
        tries++;
```

```
        if (guess > answer) // 사용자가 입력한 정수가 정답보다 높으면  
            printf("제시한 정수가 높습니다.\n");
```

```
        if (guess < answer) // 사용자가 입력한 정수가 정답보다 낮으면  
            printf("제시한 정수가 낮습니다.\n");
```

```
    } while (guess != answer);
```

```
    printf("축하합니다. 시도횟수=%d\n", tries);
```

```
    return 0;
```

```
}
```

Lab: Calculating integer digits

- Write a program that receives numbers from users and calculates the total number of digits of a given integer using the while loop.



```
Microsoft Visual Studio 디버그 콘솔
정수를 입력하시오: 111222233333
총 자리수: 12
```


Sol:

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

int main(void)
{
    long long num; // 64비트 정수형
    int count = 0;

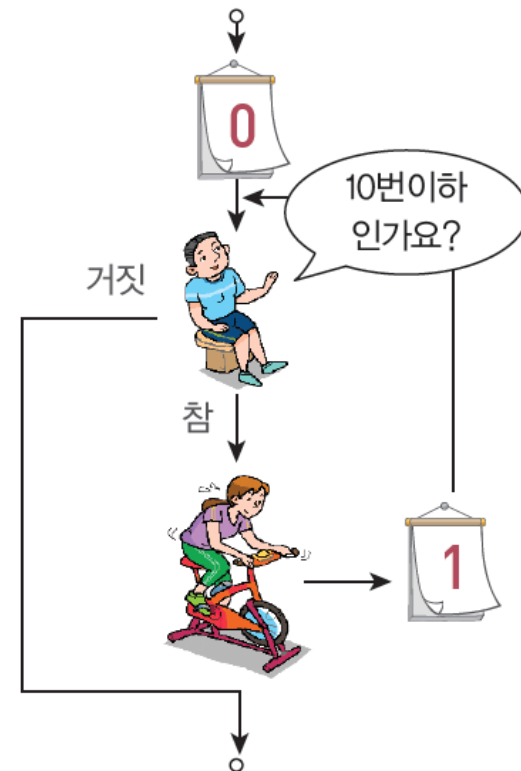
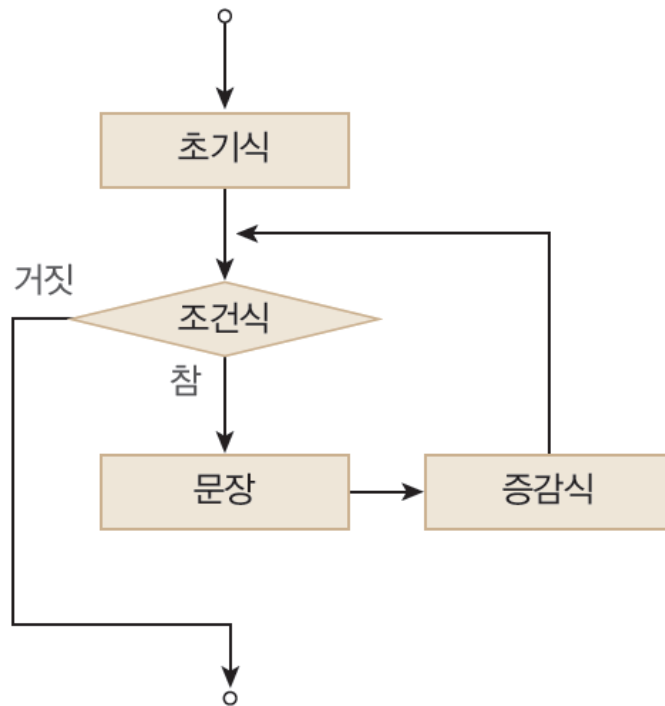
    printf("정수를 입력하시오: ");
    scanf("%lld", &num); // long long 타입은 %lld를 사용해야 한다.

    do
    {
        count++;
        num /= 10; // 정수 나눗셈
    } while (num != 0);

    printf("총 자리수: %d", count);
    return 0;
}
```

for loop

- Structure that repeats a fixed number of times



for loop

Syntax 6.3 for 반복문

형식 `for` (초기식; 조건식; 증감식){
 문장(들);
}

예 `for(i=0; i<10; i++) {`
 `sum += i;`
}

설명 초기식을 실행한 후에 조건식의 값이 참인 동안, 문장을 반복한다.
한번 반복이 끝날 때마다 증감식이 실행된다.

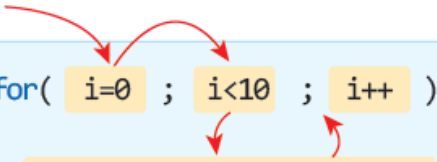
Print 10 times

- Print "Hello World!" 10 times on the screen

[illegible]

for loop

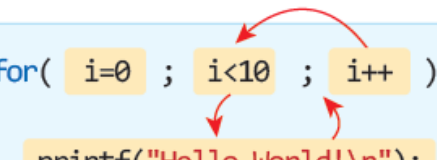
```
for( i=0 ; i<10 ; i++ )  
    printf("Hello World!\n");
```



1번째 루프 i값은

0

```
for( i=0 ; i<10 ; i++ )  
    printf("Hello World!\n");
```



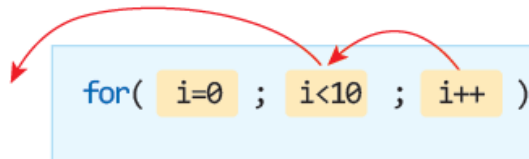
2-10번째 루프 i값은

1 2 3 4 5 6 7 8 9

...

...

```
for( i=0 ; i<10 ; i++ )  
    printf("Hello World!\n");
```



11번째 루프 i값은

10

Example #1

- As a simple example, let's make a program that adds integers from 1 to 10 to find the sum.



The image shows a screenshot of a Microsoft Visual Studio debug console window. The title bar is orange and contains the text "Microsoft Visual Studio 디버그 콘솔" (Microsoft Visual Studio Debug Console) and standard window control buttons (minimize, maximize, close). The main area of the console is white and displays the text "1부터 10까지의 정수의 합= 55" (The sum of integers from 1 to 10 = 55). A vertical scrollbar is visible on the right side of the console area.

Sol:

```
#include <stdio.h>

int main(void)
{
    int i, sum;

    sum = 0;
    for (i = 1; i <= 10; i++)    // i는 1부터 10까지 증가
        sum += i;              // sum = sum + i;와 같음

    printf("1부터 10까지의 정수의 합= %d\n", sum);
    return 0;
}
```

Example #2: square drawing

- Let's draw the following square using * letters on the screen. If the for loop is not used, printf() should be called for each line.



```
Microsoft Visual Studio 디버그 콘솔

*****
*           *
*           *
*           *
*           *
*           *
*           *
*****
```


Example #2: square drawing

```
// 반복을 이용한 네모 그리기
#include <stdio.h>

int main(void)
{
    int i;
    printf("*****");

    for(i = 0; i < 5; i++)
        printf("*");

    printf("*****");

    return 0;
}
```

Example #3: Finding a Factual

- *Factorial*

- $n! = n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$



A screenshot of a Microsoft Visual Studio debug console window. The title bar is orange and contains the text "Microsoft Visual Studio 디버그 콘솔" and standard window control buttons. The console area is white and contains two lines of Korean text: "정수를 입력하세요: 10" and "10!은 3628800입니다.". A vertical scrollbar is visible on the right side of the console area.

Example #3: Finding a Factual

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

int main(void)
{
    int fact = 1;
    int i, n;

    printf("정수를 입력하세요: ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++)
        fact = fact * i;

    printf("%d!은 %d입니다.\n", n, fact);

    return 0;
}
```

If $n=5$

	i의 값	$i \leq 5$	반복여부	fact의 값
1번째 반복	1	$1 \leq 5$ (참)	반복	$1*1$
2번째 반복	2	$2 \leq 5$ (참)	반복	$1*1*2$
3번째 반복	3	$3 \leq 5$ (참)	반복	$1*1*2*3$
4번째 반복	4	$4 \leq 5$ (참)	반복	$1*1*2*3*4$
5번째 반복	5	$5 \leq 5$ (참)	반복	$1*1*2*3*4*5$
6번째 반복	6	$6 \leq 5$ (거짓)	중단	

Points to note in the for loop

- The conditional expression can be any formula that can determine true or false. Various conditions may be combined using a logical operator as follows.

```
for (i = 0; i < 100 && sum < 2000; i++ )  
    sum += i;
```

- The for statement consists of three parts, but some of the three parts may be empty. If all three parts are empty, the loop repeats indefinitely.

```
for(;;)  
    printf("Hello World!\n");
```

Points to note in the for loop

- Increasing equations are mainly used, but decreasing equations are also used.

```
for (i = 10; i > 0; i-- )  
    printf("Hello World!\n");
```

- If the repeated sentence is more than two lines, combine it using brackets.

Notice



참고: for 루프에서 변수 선언

최신의 C 규격에서는 C++ 언어처럼 다음과 같이 **for** 루프 안에서 변수를 선언하여 사용할 수 있다.

```
for(int i=0; i<10; i++){ ... }
```

하지만 임베디드 시스템의 경우, 여러 가지 이유로 예전 버전의 C 컴파일러를 사용하는 경우도 무척 흔하다. 따라서 이 책에서는 가급적 **for** 루프 안에서는 변수를 선언하지 않았다.



Q 3가지의 반복문 **for**, **while**, **do-while** 중에서 어떤 것을 사용해야 하는가?

A 부분적으로는 개인적인 취향의 문제이다. 일반적인 선택 기준은 루프의 반복 횟수를 아는 경우에는 **for** 루프가 **while** 루프에 비하여 약간 더 편리하다고 할 수 있다. 즉 루프 제어 변수를 증가하는 것을 잊어버린다거나 하는 일이 **while** 루프에 비하여 덜 발생한다. 만약 조건만 존재하고 정확한 반복 횟수는 모르는 경우에는 **while** 구조가 좋다. 만약 반드시 한번은 수행되어야 하는 문장들이 있다면 **do-while** 구조가 제격이다.

또한 **while**과 **for**는 반복하기 전에 조건을 검사하는 구조이고 **do-while**은 먼저 실행한 후에 반복 조건을 검사한다. 특별한 경우가 아닌 일반적인 경우에는 반복을 하기 전에 조건 검사를 하는 것이 좋다. 뭐든지 실행하기 전에 면밀하게 사전 조사를 하는 것이 좋은 것과 마찬가지이다.

Interim check



중간점검

1. 다음 코드의 출력을 쓰시오.

```
for(i = 1; i < 5; i++)  
    printf("%d ", 2 * i);
```

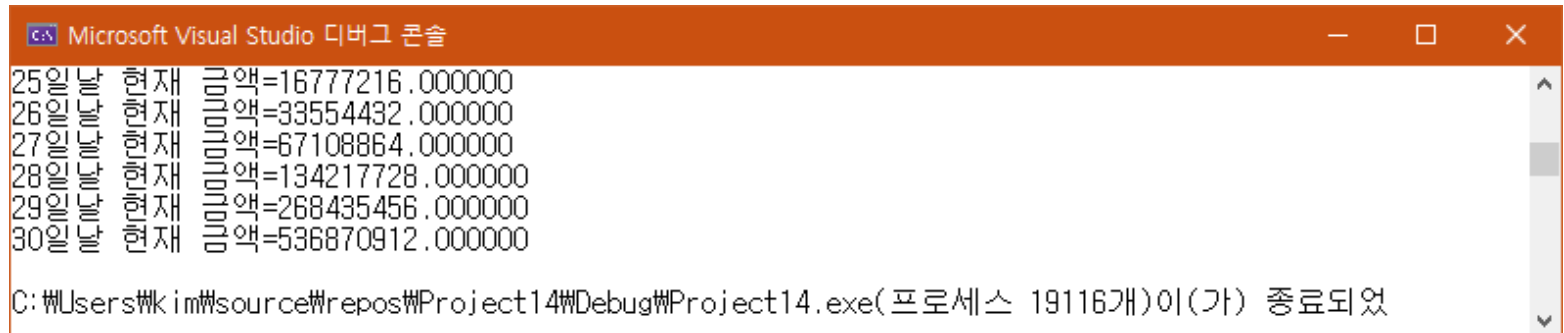
2. 다음 코드의 출력을 쓰시오.

```
for(i = 10; i > 0; i = i - 2)  
    printf("Student%d\n", i);
```



Lab: compound interest

- Will you receive 100 million won? Or will you choose to receive one won on the first day, but double the previous day for the next 30 days?



The screenshot shows the 'Microsoft Visual Studio 디버그 콘솔' (Debug Console) window. It displays the output of a program that calculates compound interest over 30 days. The output is as follows:

일 (Day)	현재 (Current)	액 (Amount)
25일	현재	액=16777216.000000
26일	현재	액=33554432.000000
27일	현재	액=67108864.000000
28일	현재	액=134217728.000000
29일	현재	액=268435456.000000
30일	현재	액=536870912.000000

At the bottom of the console, a message indicates the program has finished: 'C:\Users\k\source\repos\Project14\Debug\Project14.exe(프로세스 19116개)이(가) 종료되었'.

Sol:

```
#include <stdio.h>

int main(void) {
    double money = 1.0;

    for (int i = 2; i <= 30; i++) {                // 교과서 오타
        money *= 2.0;
        printf("%d일날 현재 금액=%lf\n", i, money);
    }
    return 0;
}
```

Lab: divisors

- Use the iterative statement to find all divisors of the given integers. Given an integer num, the loop control variable i increases from 1 to num one by one, and if $\text{num} \% i == 0$, i becomes a divisor of num.



```
Microsoft Visual Studio 디버그 콘솔
정수를 입력하시오:36
1 2 3 4 6 9 12 18 36
```

The screenshot shows a Windows-style window titled "Microsoft Visual Studio 디버그 콘솔" (Microsoft Visual Studio Debug Console). The window has a red title bar with standard minimize, maximize, and close buttons. The console area is white and contains two lines of text: "정수를 입력하시오:36" (Enter an integer: 36) and "1 2 3 4 6 9 12 18 36". A vertical scrollbar is visible on the right side of the console area.

Sol:

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

int main(void) {
    int num, i = 1;
    printf("정수를 입력하시오:");
    scanf("%d", &num);
    while (i <= num) {
        if (num % i == 0)
            printf("%d ", i);
        i++;
    }
    printf("\n\n");
    return 0;
}
```

Lab: harmonic sequence calculation

- Let's write a program that receives n from the user and displays the sum of the harmonic sequences.
- Each term gradually decreases to become infinitely close to zero, but the sum becomes infinite. Is that true? Let's calculate up to 10 million on the computer.

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots = \infty$$



```
Microsoft Visual Studio 디버그 콘솔
항의 개수: 10000000
수열의 합=16.695311
```

The screenshot shows the 'Microsoft Visual Studio 디버그 콘솔' (Debug Console) window. It contains two lines of text: '항의 개수: 10000000' (Number of terms: 10,000,000) and '수열의 합=16.695311' (Sum of the sequence = 16.695311). The window has a standard Windows title bar with minimize, maximize, and close buttons.

Sol:

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

int main(void) {
    int i, n;
    double sum = 0.0;

    printf("항의 개수: ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++)
        sum += 1.0 / (double)i;

    printf("수열의 합=%lf \n\n", sum);
    return 0;
}
```

Nested loop

- Nested loop: another iteration is located in the iteration statement

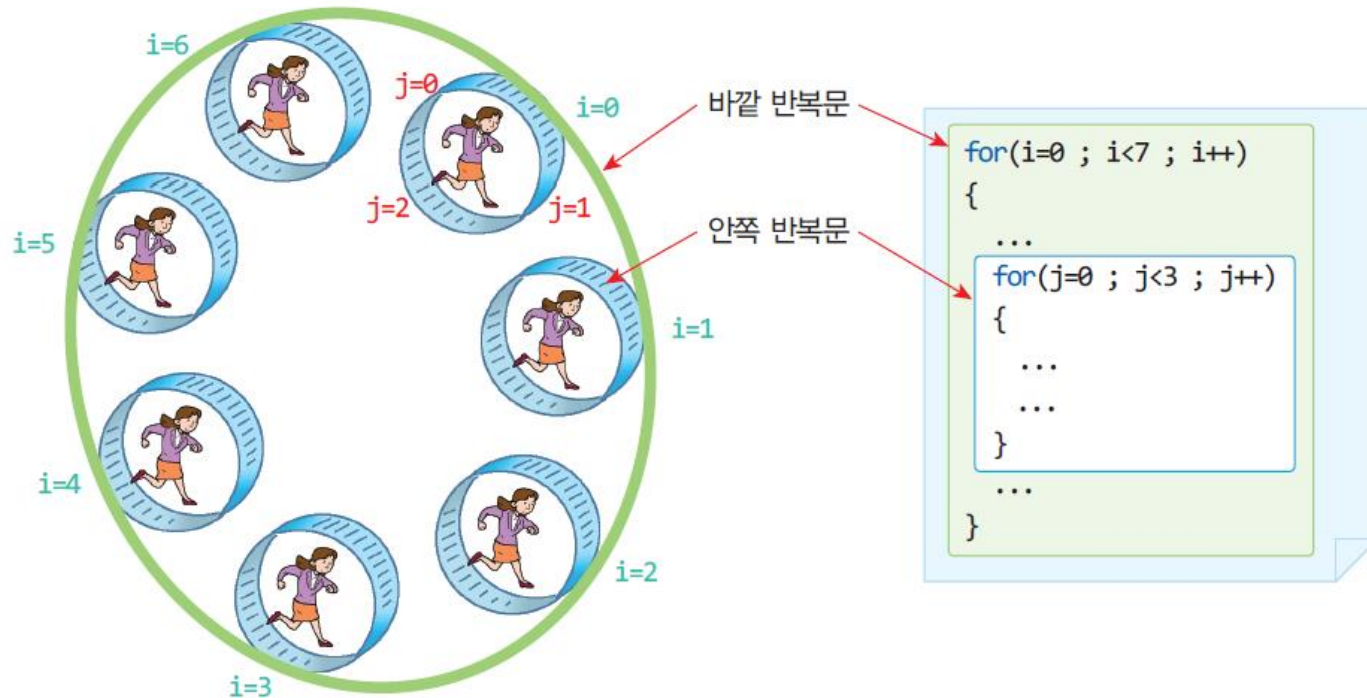


그림 6.9 중첩 반복문은 반복 루프 안에 또 다른 반복 루프가 있는 것이다.

square drawing

- Let's create a program that outputs symbols in a square shape.



The image shows a screenshot of a Microsoft Visual Studio debug console window. The title bar is orange and contains the text "선택 Microsoft Visual Studio 디버그 콘솔" (Select Microsoft Visual Studio Debug Console) along with standard window control buttons (minimize, maximize, close). The console area is white and displays a square shape composed of asterisks. The square is 5 rows high and 10 columns wide. A vertical scrollbar is visible on the right side of the console.

```
*****  
*****  
*****  
*****  
*****
```


square drawing

```
// 중첩 for 문을 이용하여 *기호를 사각형 모양으로 출력하는 프로그램
#include <stdio.h>

int main(void)
{
    int x, y;

    for (y = 0; y < 5; y++)    // 바깥 반복문
    {
        for (x = 0; x < 10; x++) // 안쪽 반복문
            printf("*");

        printf("\n");    // 안쪽 반복문이 종료될 때마다 실행
    }

    return 0;
}
```

Interim check



중간점검

1. 다음 코드의 출력을 쓰시오.

```
for(i = 1; i < 3; i++)  
    for(j = 3; j >= 1; j--)  
        printf("%d 곱하기 %d은 %d\n", i, j, i*j);
```

Lab: multiplication output

- Let's print out the entire multiplication loop using a nested loop.



A screenshot of the Microsoft Visual Studio 디버그 콘솔 (Debug Console) window. The window has a title bar with the text "Microsoft Visual Studio 디버그 콘솔" and standard window controls (minimize, maximize, close). The console area displays the following output:

```
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
2 X 1 = 2
```

The output shows a nested loop structure where the first loop (labeled '1') iterates from 1 to 9, and the second loop (labeled '2') iterates from 1 to 1. The console has a vertical scrollbar on the right side.

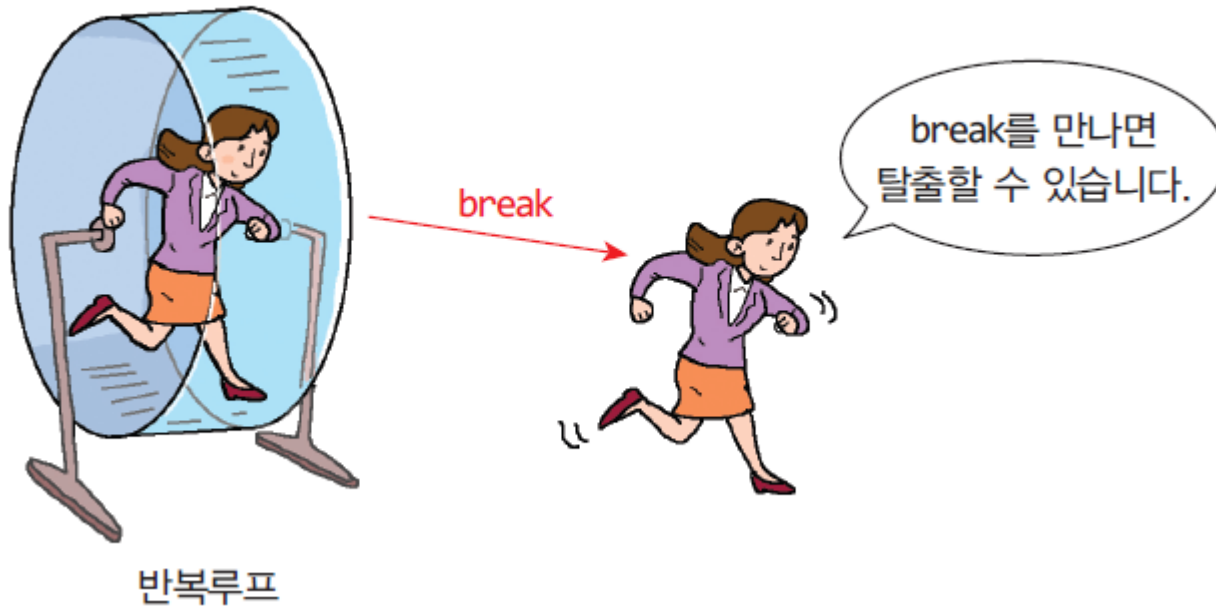
Sol: multiplication output

```
#include <stdio.h>

int main(void)
{
    int i, k;
    for (i = 1; i <= 9; i++) {
        for (k = 1; k <= 9; k++) {
            printf("%d X %d = %d \n", i, k, i * k);
        }
    }
    return 0;
}
```

break statement

- The break statement is used to exit the loop.



Example

- The next program receives 10 real numbers from the user and outputs a total. If the user inputs a negative number in the middle, the repetitive loop is terminated early.



The screenshot shows the 'Microsoft Visual Studio 디버그 콘솔' (Debug Console) window. It contains the following text:

```
1번째 실수를 입력하시오: 10.0  
2번째 실수를 입력하시오: 20.0  
3번째 실수를 입력하시오: 30.0  
4번째 실수를 입력하시오: -1  
합계 = 60.000000
```

The text is in Korean, indicating the program prompts for 10 real numbers. The user has entered 10.0, 20.0, 30.0, and -1. The program has calculated a sum of 60.000000, suggesting it terminated early upon receiving the negative number -1.

Example

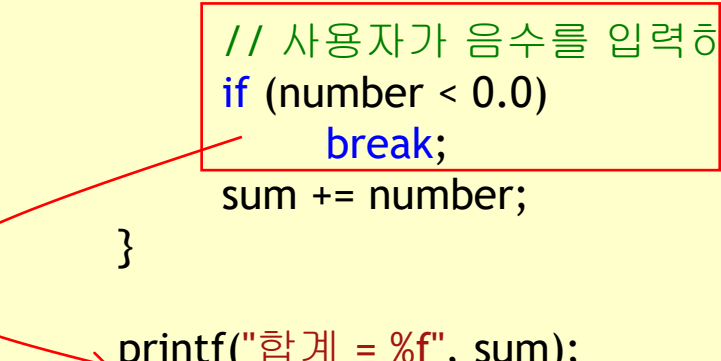
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void)
{
    int i;
    double number, sum = 0.0;

    for (i = 1; i <= 10; i++)
    {
        printf("%d번째 실수를 입력하시오: ", i);
        scanf("%lf", &number); // double형 실수는 %lf를 사용한다.

        // 사용자가 음수를 입력하면 반복 루프가 종료된다.
        if (number < 0.0)
            break;
        sum += number;
    }

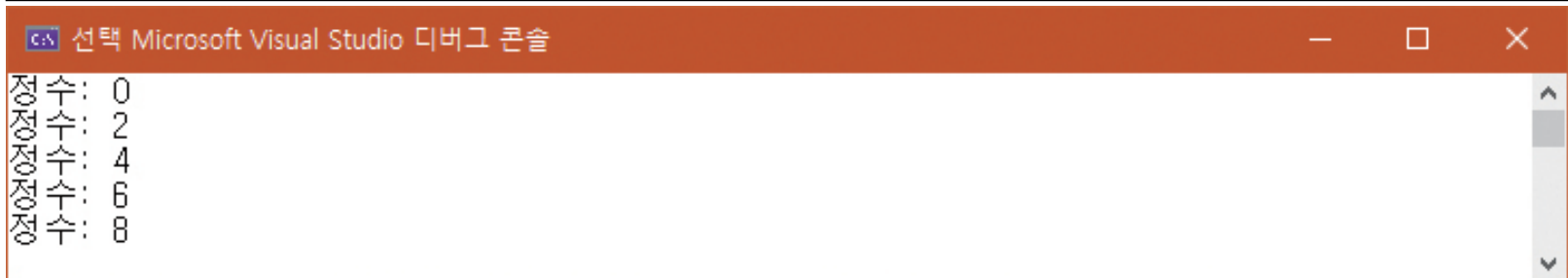
    printf("합계 = %f", sum);
    return 0;
}
```



continue statement

- Stop the current iteration and start the next iteration.

```
#include <stdio.h>
int main(void)
{
    int i;
    for (i = 0; i < 10; i++) {
        if (i % 2 == 1)
            continue;
        printf("정수: %d \n", i);
    }
    return 0;
}
```



The screenshot shows a debug console window titled "선택 Microsoft Visual Studio 디버그 콘솔". The output displays the program's execution results, showing only even integers from 0 to 8, as the odd integers were skipped due to the 'continue' statement.

```
정수: 0
정수: 2
정수: 4
정수: 6
정수: 8
```


Interim check



중간점검

1. _____ 문이 반복문에서 실행되면 현재의 반복을 중단하고 다음번 반복 처리가 시작된다.
2. _____ 문이 반복문에서 실행되면 반복문을 빠져 나온다.
3. 다음 코드의 출력을 쓰시오.

```
int i;  
for(i = 1; i < 10; i++) {  
    if( i % 3 == 0 ) break;  
    printf("%d\n", i);  
}
```

4. 3번 문제에서 **break**를 **continue**로 변경하면 어떻게 되는가?



Lab: automatic arithmetic questions

- Let's create a program that automatically presents arithmetic questions for elementary school students. Addition problems should be generated automatically. Let the operand be a random number between 0 and 99. If you get hit once, the program terminates.



```
Microsoft Visual Studio 디버그 콘솔
산수 문제를 자동으로 출제합니다.
41 + 67 = 109
틀렸습니다.
34 + 0 = 34
맞았습니다.
```

Sol:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int i, ans;
    printf("산수 문제를 자동으로 출제합니다. \n");

    while (1) {
        int x = rand() % 100;
        int y = rand() % 100;
        printf("%d + %d = ", x, y);
        scanf("%d", &ans);
        if (x + y == ans) {
            printf("맞았습니다.\n");
            break;
        }
        else
            printf("틀렸습니다.\n");
    }
    return 0;
}
```

Lab: Let's find all the primes from 1 to n.

- Write a program that outputs all primes between 1 and n using the iterative structure.

```
Microsoft Visual Studio 디버그 콘솔
어디까지 찾을까요? : 50
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
```

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

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

```
int main()
{
    int n, is_prime;

    printf("어디까지 찾을까요? : ");
    scanf("%d", &n);

    for (int i = 2; i <= n; i++) {
        is_prime = 1;
        for (int k = 2; k < i; k++) {
            if (i % k == 0) {
                is_prime = 0;
                break;
            }
        }
        if (is_prime == 1)
            printf("%d, ", i);
    }
    printf("\n\n");
    return 0;
}
```

Q & A

