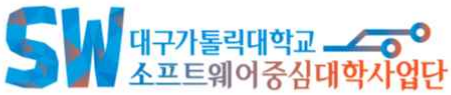
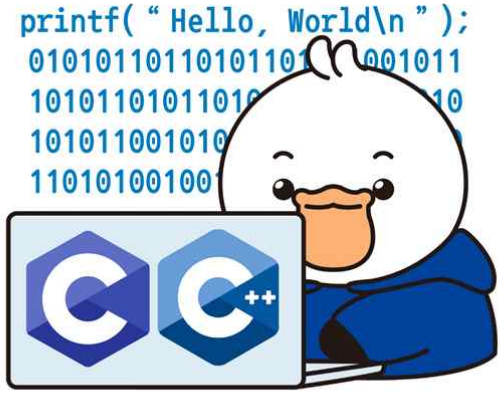
학번 : _____ 성명 : _____.

- 본 강의자료는 과학기술정보통신부 및 정보통신기획평가원에서 지원하는『소프트웨어중심대학』사업의 결과물입니다.
- 본 강의자료는 내용은 전재할 수 없으며, 인용할 때에는 반드시 과학기술정보통신부와 정보통신기획평가원의 '소프트웨어중심대학'의 결과물이라는 출처를 밝혀야 합니다.



Part 5. 제어문 - 반복문

목차



5.1 반복문 개요

- 반복문(Iteration Statement)이란?
  - ✓ 제어문 중 하나, 코드 내 특정 부분을 반복적 수행 하고자 할 때 사용
    - ➢ 반복(repeat) 또는 순환(loop), 루틴(routine)으로 부름.
  - ✓ C언어의 경우 3가지 명령어 제공
    - ➢ 반복 횟수를 명확히 알 수 있는 경우
      - for
    - ➢ 반복 횟수를 특정할 수 없을 때
      - while, do ~ while

```
Hello World!!!
Hello World!!!
Hello World!!!
```

<단순 반복 프로그램 출력 예시>

5.1 반복문 개요

- for
  - ✓ 특정 작업을 n회 실행 하는 경우
- while
  - ✓ 반복 횟수를 특정할 수 없을 때
  - ✓ 반복 여부를 사전에 검사
    - ➢ 단, 한번도 실행되지 않을 가능성 존재

5.1 반복문 개요

- do ~ while
  - ✓ 반복 횟수를 특정할 수 없을 때
  - ✓ 반복 수행해야 하는 문장들을 수행한 이후 반복 조건을 검사
    - ➢ 적어도 한 번은 반복 문장들을 실행
  - ✓ 사용자 입력 값 검사시에 많이 사용
    - ➢ 센티널 검사(Sentinel Test) 또는 센티널 루프(Sentinel Loop)

5.1 반복문 개요

소스 5-1

```c
#include <stdio.h>
int main(void)
{
    int i = 1;
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    printf("%d월\n", i++);
    return 0;
}
```

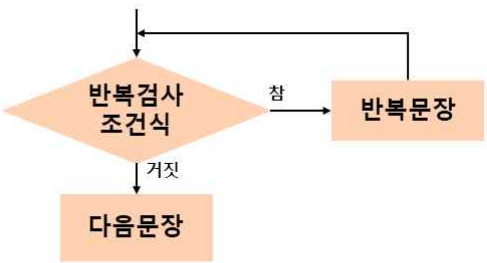
출력 예

```
1월
2월
3월
4월
5월
6월
7월
8월
9월
10월
11월
12월
```


5.2 반복문 while

- while문이란?
  - ✓ 조건이 참인 경우 반복을 수행하고 다시 조건 비교 방식 동작
    - ➢ 조건식이 참인 경우에만 반복 수행
  - ✓ if문에 반복적인 검사과정이 추가된 형태

```
while(조건식) {
        반복 문장;
}
다음 문장;
```



<while문 형식과 제어 흐름>

5.2 반복문 while

소스 5-2

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int i = 1;
5     while (i < 4) {
6        printf("Hello World!!!\n");
7        i++;
8     }
9     return 0;
10 }
```

출력 예

```
Hello World!!!
Hello World!!!
Hello World!!!
```


5.2 반복문 while

소스 5-3

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int i = 1;
5     while (i < 13) {
6        printf("%d월\n",i);
7        i++;
8     }
9     return 0;
10 }
```

출력 예

```
1월
2월
3월
4월
5월
6월
7월
8월
9월
10월
11월
12월
```

5.2 반복문 while

소스 5-4

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4    int n = 1;
 5    int sum = 0;
 6    int count = 0;
 7    double avg;
 8    while (n != 0) {
 9       scanf("%d", &n);
10       if (n % 2 == 0) {
11          sum += n;
12          count++;
13       }
14    }
15    printf("%d\n", sum);
16    avg = (double)sum / count;
17    printf("%.2f\n", avg);
18    return 0;
19 }
```

입력 예	10 3 4 5 2 3 0
출력 예	16 4.00

5.2 반복문 while

소스 5-5

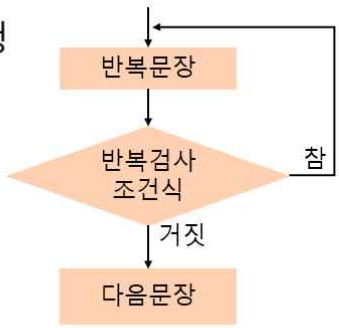
```
 1 /* 입력한 정숫값 이하의 3의 거듭제곱을 오름차순으로 출력 */
 2 #include <stdio.h>
 3 int main(void)
 4 {
 5    int threePow = 3, limit;
 6    printf("양의 정수를 입력하세요: ");
 7    scanf("%d", &limit);
 8    while (threePow <= limit) {
 9       printf("%d ", threePow);  // 3 거듭제곱 값을 출력
10       threePow *= 3;     // 3 거듭제곱 값 계산
11    }
12    printf("\n");         // 줄 바꿈
13    return 0;
14 }
```

입력 예	40
출력 예	양의 정수를 입력하세요: 40 3 9 27

5.3 반복문 do ~ while

- do ~ while문이란?
  - ✓ while 문과 매우 유사, 반복 조건 검사 순서의 차이
    - ➢ 반복 수행 후 조건식 판단
      - 적어도 한 번은 반복 문장을 수행

```
do {
      반복문장;
} while(조건식);
다음 문장;
```



<do ~ while문 형식과 제어 흐름>

5.3 반복문 do ~ while

소스 5-6

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int i = 1;
5     do {
6        printf("[inloop]%d\n", i++);
7     } while (i<=10);
8     printf("[outloop] i = %d\n", i);
9     return 0;
10 }
```

출력 예

```
[inloop]1
[inloop]2
[inloop]3
[inloop]4
[inloop]5
[inloop]6
[inloop]7
[inloop]8
[inloop]9
[inloop]10
[outloop] i = 11
```

5.3 반복문 do ~ while

소스 5-7

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int from, to, i, temp;
5      int sum = 0;
6      printf("From : ");
7      scanf("%d", &from);
8      printf("To : ");
9      scanf("%d", &to);
10     if (from > to) {
11         temp = from;
12         from = to;
13         to = temp;
14     }
15     i = from;
16     do {
17         sum = sum + i;
18         i = i +1;
19     } while (i <= to);
20     printf("Total from %d to %d is %d.\n", from, to, sum);
21     return 0;
22 }
```

입력 예	10 40
출력 예	From : 10 To : 40 Total from 10 to 40 is 775.

5.3 반복문 do ~ while

소스 5-8

```
1  #include <stdio.h>
2  int main(void){
3     int n;
4     do {
5        printf("Input a positive number or 0(Quit) : ");
6        scanf("%d", &n);
7     } while (n!=0);
8     printf("Teminated...");
9     return 0;
10 }
```

입력 예	10 20 0
출력 예	Input a positive number or 0(Quit) : 10 Input a positive number or 0(Quit) : 20 Input a positive number or 0(Quit) : 0 Teminated...

5.4 반복문 for

- for문이란?
  - ✓ 반복 횟수를 아는 경우 주로 사용
    - ➢ 주어진 횟수만큼 반복 문장을 반복하는 경우 주로 사용
      - while문이나 do ~ while문에 비해 조금은 복잡한 구조
  - ✓ 구성
    - ➢ 초기식 - 반복 제어 변수 초기화
      - 반복 안에서 사용할 반복을 제어하기 위한 변수 초기화
    - ➢ 조건식 - 반복 횟수 제어
      - 반복문 내부 반복 문장 실행 전에 검사를 통해 참이면 반복, 수행 거짓이면 반복 종료
    - ➢ 증감식 - 1회 반복할 때 마다 반복 제어 변수 값 변경
      - 반복문 내부 반복 문장 수행 후 실행하는 문장, 반복 제어 변수 값 증감 용도

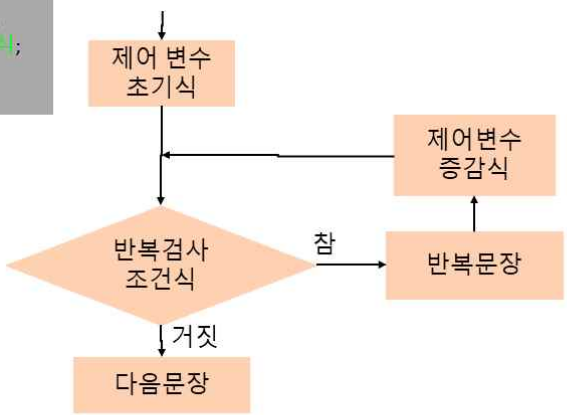
5.4 반복문 for

```
이전문장
for(초기식; 조건식; 증감식)
{
    문장;
}
다음문장;
```

변환 가능

```
이전문장;
초기화식;
while(조건식)
{
    문장;
    증감식;
}
다음문장;
```

```
for(초기식; 조건식; 증감식) {
    반복문장;
}
다음 문장;
```



<for문 형식과 제어 흐름>

5.4 반복문 for

소스 5-9

```
1 #include <stdio.h>
2 int main(void) {
3    int i;
4    for(i = 1; i <= 3; i++) {
5       printf("Hello World!!!\n");
6    }
7    return 0;
8 }
```

출력 예

```
Hello World!!!
Hello World!!!
Hello World!!!
```


5.4 반복문 for

<for문과 while문의 비교표>

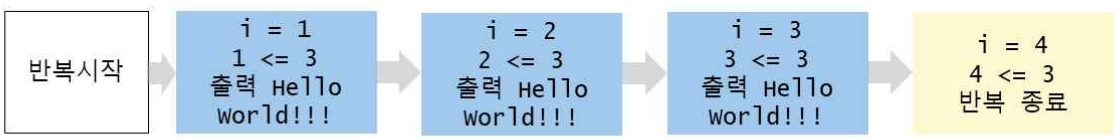
for 문	while 문

`for(i = 1; i <= 3; i++)`
`{`
`  printf("Hello World\n");`
`}`
	`int i = 1;`
	`while(i <= 3)`
	`{`
	`  printf("Hello World!!!\n");`
	`  i++;`
	`}`

이전문장
for(초기식; 조건식; 증감식)
{
  문장;
}
다음문장;

이전문장;
초기화식;
while(조건식)
{
  문장;
  증감식;
}
다음문장;

변환 가능



5.4 반복문 for

소스 5-10

```
1 #include <stdio.h>
2 int main(void) {
3    int i;
4    for (i = 1; i <= 12; i++) {
5       printf("%d월\n", i);
6    }
7    return 0;
8 }
```

출력 예

```
1월
2월
3월
4월
5월
6월
7월
8월
9월
10월
11월
12월
```


5.4 반복문 for

소스 5-11

```
1  #include <stdio.h>
2  int main(void) {
3     int dan, i;
4     printf("출력할 구구단 입력 : ");
5     scanf("%d", &dan);
6     for (i=1; i<=9; i++) {
7        printf("%d * %d = %d\n", dan, i, dan*i);
8     }
9     return 0;
10 }
```

입력 예

```
7
```

출력 예

```
출력할 구구단
입력 : 7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
```

5.4 반복문 for

소스 5-12

```
1  #include <stdio.h>
2  int main(void) {
3     int dan, i;
4     do {
5        printf("출력할 구구단(2~9) 입력 : ");
6        scanf("%d", &dan);
7     } while (dan<2 || dan>9);
8     for (i=1; i<=9; i++) {
9        printf("%d * %d = %d\n", dan, i, dan*i);
10    }
11    return 0;
12 }
```

입력 예	10 7
출력 예	출력할 구구단(2~9) 입력 : 10 출력할 구구단(2~9) 입력 : 7 7 * 1 = 7 7 * 2 = 14 7 * 3 = 21 7 * 4 = 28 7 * 5 = 35 7 * 6 = 42 7 * 7 = 49 7 * 8 = 56 7 * 9 = 63

5.4 반복문 for

소스 5-13

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int i, n;
5     int sum=0;
6     printf("Input n : ");
7     scanf("%d", &n);
8     for (i=1; i<=n; i++)
9        sum += i;
10    printf("Sum from 1 to %d is %d.\n", input_i, sum);
11    return 0;
12 }
```

=
동일하게
동작

```
for(i=1, sum=0; i<=n; sum+=i++)
```

입력 예	10
출력 예	Input n : 10 Sum from 1 to 10 is 55.

5.4 반복문 for

소스 5-14

```
1  #include <stdio.h>
2  int main(void) {
3     int i, n;
4     int sum;
5     printf("Input n : ");
6     scanf("%d", &n);
7     for (i=1, sum =0; i<=n; sum+=i++) ;
8     printf("Sum from 1 to %d is %d.\n", n, sum);
9     return 0;
10 }
```

입력 예	100
출력 예	Input n : 100 Sum from 1 to 10 is 5050.

5.4 반복문 for

소스 5-14

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int i, no;
5     printf("A positive integer : ");
6     scanf("%d", &no);
7     printf("Results of for :");
8     for (i=1; i<=no; i +=2)
9        printf(" %d", i);
10    printf("\n");
11    printf("Results of while :");
12    i=1;
13    while (i<=no) {
14       printf(" %d", i);
15       i+=2;
16    }
17    printf("\n");
18    return 0;
19 }
```

입력 예	10
출력 예	A positive integer : 10 Results of for : 1 3 5 7 9 Results of while : 1 3 5 7 9

5.5 분기문

- 분기문(Jump Statements)이란?
  - ✓ 프로그램 수행 흐름을 코드 내 특정 위치로 분기(jump)하는 기능 수행
    - ➢ `break`, `continue`, `goto`, `return`
  - ✓ `break` 문
    - ➢ `switch` 문 학습하면서 `break` 문 사용, `switch` 블록을 빠져나가 다음 문장 수행을 위해 사용
  - ✓ `continue` 문
    - ➢ `break` 문과 달리 블록을 빠져나가는 것이 아니라 다음 반복으로 이동
  - ✓ `goto` 문
    - ➢ 사전 지정해 둔 프로그램 내 특정 위치로 프로그램 실행 흐름 변경
  - ✓ `return` 문 (Part. 8 참조)
    - ➢ 함수의 종료, `main()` 함수의 경우 프로그램의 종료

5.5 분기문

- `break` 문
  - ✓ 반복문 내부에서 반복을 종료하기 위해 `break` 문 사용
    - ➢ `break` 문을 포함하는 블록을 빠져나가는 기능
  - ✓ 반복문 내부에 반복문이 있는 중첩 반복문의 경우
    - ➢ `break` 문을 포함하는 가장 작은(가까운) 내부 반복문 종료

```
for(... ; ... ; ...)
{
   ... break;
   ...
}
next;
```

```
while( ... )
{
   ... break;
   ...
}
next;
```

```
do
{
   ... break;
   ...
} while ( ... );
next;
```

5.5 분기문

소스 5-16

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4    int n;
 5    while (1) {
 6       printf("Input an integer : ");
 7       scanf("%d", &n);
 8       if (n==0) break;
 9       printf("%d is inputed.\n", n);
10    }
11    printf("Terminated.");
12    return 0;
13 }
```

입력 예	5 -1 0
출력 예	Input an integer : 5 5 is inputed. Input an integer : -1 -1 is inputed. Input an integer : 0 Terminated.

5.5 분기문

- continue 문
  - ✓ 수행중인 반복을 중지하고 다음 반복으로 프로그램 진행 방향을 변경하는 명령어
    - ➢ 현재 실행 중인 반복 문장들은 무시, 다음 반복을 처음부터 실행
      - 즉, continue 아래에 위치하는 모든 반복 문장들은 실행되지 않음

```
for(num=0 ; num<10 ; num++)
{
   ...
   continue;
   ...
}
```

```
while( num<10 )
{
   ...
   continue;
   ...
}
```

```
do
{
   ...
   continue;
   ...
} while ( num<10 );
```

5.5 분기문

소스 5-17

```
 1 #include <stdio.h>
 2 #define MAX 50
 3 int main(void)
 4 {
 5    int i;
 6    printf("Even numbers from 1 to %d : ",MAX);
 7    for (i=1; i<=MAX; i++) {
 8       if (i % 2) continue;
 9       printf("%d ", i);
10    }
11    return 0;
12 }
```

출력 예

```
Even numbers from 1 to 50 : 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32
34 36 38 40 42 44 46 48 50
```


5.5 분기문

- goto 문
  - ✓ 프로그램상의 특정 위치로 프로그램 실행 흐름을 변경하는 명령어
  - ✓ goto 문으로 이동할 위치는 goto 문 바로 뒤에 레이블(label)을 적어 표시
  - ✓ 레이블
    - ➢ 식별자의 한 종류, 변수명과 같이 이름을 정하여 사용, 식별자 생성 규칙을 따름
    - ➢ 정의 시 반드시 이름 뒤 :(콜론)
    - ➢ switch 문에서 사용한 case: 와 default: 이 둘 또한 사전에 정의된 레이블
  - ✓ goto 문을 이용하면 편리하게 프로그램의 흐름을 바꿀 수 있다.
    - ➢ 여러개의 반복문으로 둘러쌓인 경우 한번에 모든 반복문 탈출 가능
    - ➢ 소스코드를 읽고 이해하는데 장애물, 일반적으로 goto 문 사용은 권장되지 않음

5.5 분기문

소스 5-18

```
1 #include <stdio.h>
2 int main(void)
3 {
4    int number = 0;
5 loop:
6    printf("Hello World!!!\n");
7    if (++number < 3) goto loop;
8    return 0;
9 }
```

출력 예

```
Hello World!!!
Hello World!!!
Hello World!!!
```


5.6 무한 반복문

- 무한 반복문 또는 무한 루프란?
  - ✓ 반복분 조건식이 항상 참이 되어 영원히 종료하지 않고 무한히 반복되는 반복문
    - ➢ 의도적으로 작성하지 않은 무한 반복문은 논리적 오류
    - ➢ 프로그래머가 의도적으로 무한 반복문을 만드는 경우
      - ex) 센티널 검사 수행 반복문
  - ✓ 무한 반복문 생성
    - ➢ 요점 – 적절한 종료 조건, 조건 만족할 때 반복문을 종료하고 빠져 나올 수 있어야 한다는 것

<무한 반복문 생성 예시>

```
for(;;) {
        ...
}
```

```
for( ;1; ) {
        ...
}
```

```
while(1) {
        ...
}
```

```
do {
        ...
} while(1);
```

5.6 무한 반복문

소스 5-19

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4    int select;
 5    do {
 6       printf("[1] Cola\n");printf("[2] Orange Juice\n");
 7       printf("[3] Mango Juice\n");printf("[4] Ginger Beer\n");
 8       printf("[5] Coffee\n");printf("Select beverage : ");
 9       scanf("%d", &select);
10       if (select <= 5 && select >= 1) break;
11    } while (1);
12    printf("Your choice is %d.\n", select);
13    return 0;
14 }
```

입력 예	6 5
출력 예	[1] Cola [2] Orange Juice [3] Mango Juice [4] Ginger Beer [5] Coffee Select beverage : 6 [1] Cola [2] Orange Juice [3] Mango Juice [4] Ginger Beer [5] Coffee Select beverage : 5 Your choice is 5.

5.6 무한 반복문

소스 5-20

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4    int sum = 0, n;
 5    while (1) {
 6       printf("Input 0 or positive value : ");
 7       scanf("%d", &n);
 8       if (n < 0) break;
 9       sum += n;
10    }
11    printf("Accumulated value : %d\n", sum);
12    return 0;
13 }
```

입력 예	10 20 5 7 -1
출력 예	Input 0 or positive value : 10 Input 0 or positive value : 20 Input 0 or positive value : 5 Input 0 or positive value : 7 Input 0 or positive value : -1 Accumulated value : 42

5.6 무한 반복문

소스 5-21

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4    int sum = 0, n;
 5    for (;;) {
 6       printf("Input 0 or positive value : ");
 7       scanf("%d", &n);
 8       if (n < 0) break;
 9       sum += n;
10    }
11    printf("Accumulated value : %d\n", sum);
12    return 0;
13 }
```

입력예

```
10
20
5
7
-1
```

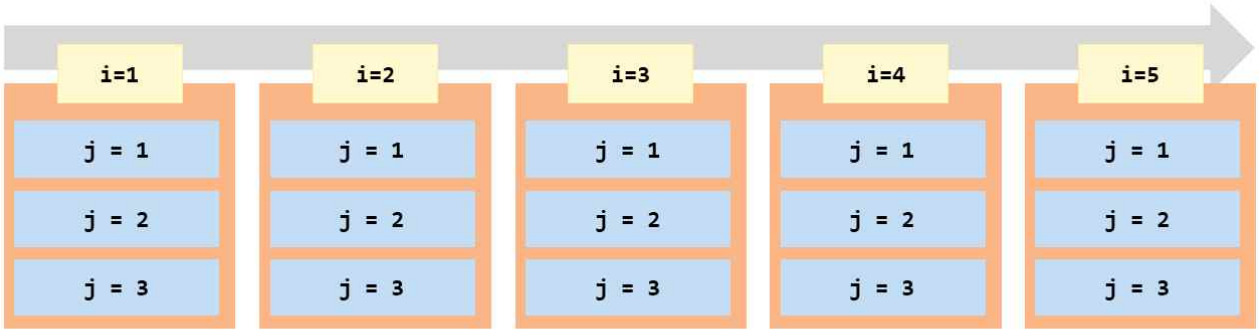
출력예

```
Input 0 or positive value : 10
Input 0 or positive value : 20
Input 0 or positive value : 5
Input 0 or positive value : 7
Input 0 or positive value : -1
Accumulated value : 42
```


5.7 중첩 반복문

- 중첩 반복문 또는 다중 반복문이란?
  - ✓ 반복문 내부에 또 다른 반복문이 있는 것
- 외부 반복문 5회, 내부 반복문 3회
  - ✓ 총 15회 반복 수행

```
for(i=1; i<=5;i++)
{
    for(j=1;j<=3;j++)
    {
        …
    }
}
```



5.7 중첩 반복문

소스 5-22

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int i, j;
5     for (i=1; i<=5; i++) {
6        for (j=1; j<=5; j++) {
7           printf("*");
8        }
9        printf("\n");
10    }
11    return 0;
12 }
```

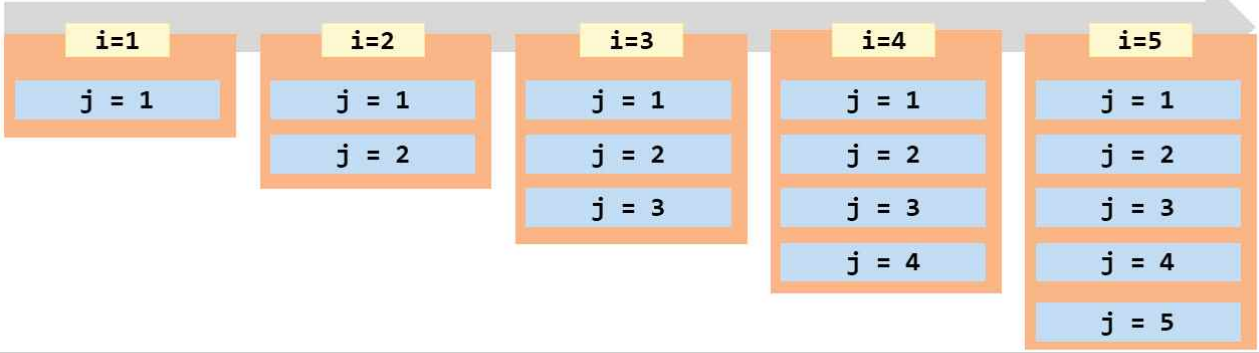
출력 예

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


5.7 중첩 반복문

- 외부 반복문 1~5회, 내부 반복문 1~i회
  - ✓총 15회 반복 수행

```
for(i=1; i<=5;i++)
{
    for(j=1;j<=i;j++)
    {
        …
    }
}
```



5.7 중첩 반복문

소스 5-23

```
1  #include <stdio.h >
2  int main(void)
3  {
4     int i, j;
5     for (i=1; i<=5; i++) {
6        for (j=1; j<=i; j++) {
7           printf("*");
8        }
9        printf("\n");
10    }
11    return 0;
12 }
```

출력 예

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


5.7 중첩 반복문

소스 5-24

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int i, j;
5     for(i=2; i<=9; i++) {
6        printf("##%2d단##\n", i);
7        for(j=1; j<=9; j++) {
8           printf("%d * %d = %-2d\n", i, j, i*j);
9        }
10       printf("\n");
11    }
12 }
```

출력 예

```
## 2단##
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
```

```
## 3단##
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
... 생략 ...
```

```
## 9단##
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

5.8 Q&A

Q 18. 왜 아래 loop는 항상 한 번만 실행될까?

```
for (i = start; i <end; i ++);
{
    printf("%d\n", i);
}
```

A. for를 쓴 문장의 마지막 뒤 세미콜론 때문에, for가 반복할 문장(statement)은 "null statement"가 된다. 중괄호로 둘러싼 블록은, for와 무관한, 그냥 다음 statement일 뿐이다.

5.8 Q&A

Q 19. 어떤 사람들은 goto가 악마와도 같다고 말하면서 절대로 쓰지 말라고 하는데, 너무 과장된 말이 아닐까?

A. 프로그래밍 스타일이란, 정형화된 것이 아니기 때문에 절대적인 규칙으로 제정될 수 없는 것이다. goto 문은 남용되면 유지/보수/관리하기 힘들 정도의 스파게티 코드를 만들어 낼 수 있다. 아무 생각없이 goto 문을 금지하는 것이 좋은 코드를 만들어 내는 것도 아니다. 물론 goto문을 전혀 쓰지 않고도 코드를 만들 수 있다. 대부분의 프로그래밍 스타일에 관한 "규칙"은 "안내지침"이란 단어로 보는게 더바람직 하다. 또한 이러한 규칙들이 왜 만들어졌는 가를 이해하는 것이 더 중요하다. 이러한 규칙에 대한 이해 없이, 무조건 어떤 구조를 배척하는 것은 원래 규칙이 의미하고자 한것과 오히려 반대되는 결과를 만들어낼 수 있다.

5.9 실습 및 과제

- 실습 및 과제 진행
  - DCU Code : http://code.cu.ac.kr
  - 5 주차 실습
    - 코딩 5-1, 코딩 5-2, 코딩 5-3, 코딩 5-4 ,
      코딩 5-5 , 코딩 5-6 , 코딩 5-7 (p196-222)
  - 5 주차 과제
    - 5장 프로그래밍 연습 1-9번 (p228-229)

3.7 참고 문헌

- 반복문 개요
  - https://ko.wikipedia.org/wiki/반복문
- 반복문 while
  - https://ko.wikipedia.org/wiki/While_루프
  - https://en.wikipedia.org/wiki/Sentinel_value
- 무한 반복문
  - https://namu.wiki/w/무한%20루프
- Q&A
  - http://cinsk.github.io/ko/cfaqs/index.html

E N D