초급 C언어

가천대학교
Gachon University

조건문과 관계 연산자, 논리연산자

if 문

- 주어진 조건을 만족할 경우, 처리할 내용이 있을 때 사용
- 형식

```
if( 조건식 ) {

        문장(들);

}
```

- 사용 방식 : 조건 식을 계산한 결과가 만족하면('참'이면), {...} 내부의 문장(들)을 실행한다.
  그 후 {...} 외부의 문장(들)을 실행한다.
  조건 식을 계산한 결과가 만족하지 않으면('거짓'이면), { ... } 외부의 문장(들)을 실행한다.

if else 문

- 조건의 참과 거짓에 따라 둘 중 하나를 선택해서 처리할 때 사용
- 형식)

```
if( 조건식 ) {
        문장(들);
}
else {
        문장(들);
}
```

- 사용 방식 : 조건 식을 계산한 결과가 만족하면('참'이면), 첫 번째 {...} 내부의 문장(들)을 실행한다.
  그 후 {...} 외부의 문장(들)을 실행한다.
  조건 식을 계산한 결과가 만족하지 않으면('거짓'이면), else {...} 내부의 문장(들)을 실행한다.
  그 후 {...} 외부의 문장(들)을 실행한다.

다중 if의 예(1)

- 다중 if
  - ✓ if else문에서 else 다음에 다시 if를 사용하여 또 다른 조건을 확인한다.
    ※ else if는 필요에 따라 여러 번 나올 수 있음
  - ✓ 마지막 else는 상황에 따라 생략될 수 있다.

- 형식)

```
if(조건식) {
        문장(들);
} else if(조건식) {
        문장(들);
}
....
else {                  //생략 가능
        문장(들);
}
```

- 사용 방식)

첫 번째 if의 조건식을 계산한 결과가 만족하면('참'이면),
해당 {...} 내부의 문장(들)을 실행한다.
그 후 {...} 외부의 문장(들)을 실행한다.(if문 종료)
첫 번째 if의 조건식을 계산한 결과가 만족하지 않으면('거짓'이면),
다음에 나오는 else if의 조건식을 계산하여 결과가 참이면,
해당 {...} 내부의 문장(들)을 실행한다.
그 후 {...} 외부의 문장(들)을 실행한다.(if문 종료)

...

모든 조건식에 만족한 결과가 없으면,
else {...} 내부의 문장(들)을 실행한다. (if문 종료) → 생략 가능

중첩 if의 예

- 중첩 if : if 안에 또 다른 if문을 사용

예) 나이가 18세 이하이면 청소년 요금, 65세 미만이면 성인 요금, 65세 이상이면 경로우대 요금을 적용

```
if (age <= 18) {
        printf("청소년 요금입니다.₩n");
}
else {

        if (age < 65) {
                printf("성인 요금입니다.₩n");
        }
        else {
                printf("경로우대 요금입니다.₩n");
        }
}
```

[실습] 다음과 같은 결과가 나오도록 코드를 작성하시오.

[결과] 첫 번째 실행
나이를 입력하시오:***18***
청소년 요금입니다.

[결과] 두 번째 실행
나이를 입력하시오:***19***
성인 요금입니다.

[결과] 세 번째 실행
나이를 입력하시오:***64***
성인 요금입니다.

[결과] 네 번째 실행
나이를 입력하시오:***65***
경로우대 요금입니다.

[조건]

- 입력 받은 나이에 따라 티켓 가격을 다음과 같이 출력
  - 18세 이하이면, "청소년 요금입니다." 출력
  - 19세 이상 65세미만이면, "성인 요금입니다." 출력
  - 65세 이상이면, "경로우대 요금입니다." 출력
- 중첩 if를 사용

switch문(1)

- 정수 값 또는 문자 값의 여러 조건들 중에서 해당되는 값만 선택하여 실행하는 조건문

예) 값이 1이면, '하나'를 출력
값이 2이면, '둘'을 출력
값이 3이면, '셋'을 출력
그외의 값이면, '해당사항 없음'을 출력

switch문(2)

형식)

```
switch (정수값 또는 문자값) {
  case 정수값1 또는 문자값1 :
     문장(들)1;
     break;

  case 정수값2 또는 문자값2 :
     문장(들)2;
      break;
. . .
  case 정수값n 또는 문자값n :
     문장(들)n;
     break;

   default :                  //생략 가능
     문장(들)n+1
     break;
}
```

예)

```
int num;
...
switch(num) {
    case 1:
            printf("하나₩n");
            break ;
    case 2:
            printf("둘₩n");
            break ;
    case 3:
            printf("셋₩n");
            break ;
     default:
            printf("해당사항없음.₩n");
            break;
}
```

사용방식

- switch(...)의 정수값 또는 문자값과 동일한 값의
  - ✓ case문이 있다면
    - -해당 case문 다음에 나열된 문장들을 break를 만날 때까지 수행한다.
    - -break를 만나면 switch문을 빠져나간다.
  - ✓ case문이 없다면
    - -default 문이 있다면 default문 다음에 나열된 문장들을 break를 만날 때까지 수행한다.
      break를 만나면 switch문을 빠져나간다.
    - -default 문은 생략 가능하므로, 이 문장이 없다면 switch문을 빠져나간다.

[실습] 다음과 같은 결과가 나오도록 코드를 작성하시오.

```
[결과] 첫 번째 실행
수식을 입력하시오(예: 2 + 5)
>>10 + 3
10 + 3 = 13

[결과] 두 번째 실행
수식을 입력하시오(예: 2 + 5)
>>10 - 3
10 - 3 = 7

[결과] 세 번째 실행
수식을 입력하시오(예: 2 + 5)
>>10 * 3
10 * 3 = 30

[결과] 네 번째 실행
수식을 입력하시오(예: 2 + 5)
>>10 / 3
10 / 3 = 3

[결과] 다섯 번째 실행
수식을 입력하시오(예: 2 + 5)
>>10 $ 3
지원되지 않는 연산자입니다.
```

[조건]

- 두 수와 덧셈, 뺄셈, 곱셈, 나눗셈의 기호 중 하나를 입력받아 해당 계산 결과를 출력
  ex. scanf_s("%d %c %d", &x, &op, 1, &y);
- 4칙 연산 기호가 아닌 경우, 지원되지 않는 연산자임을 출력
- switch 문을 사용,

조건식에 사용하는 관계 연산자

- x와 y사이의 관계를 표현하기 위한 연산자

의미	연산자	연산의 결과 (x는 1, y는 2인 경우)
x가 y보다 큰가?	x > y	거짓(0), 만족하지 않음
x가 y보다 크거나 같은가?	x >= y	거짓(0), 만족하지 않음
x가 y보다 작은가?	x < y	참(1), 만족함
x가 y보다 작거나 같은가?	x <= y	참(1), 만족함
x가 y와 같은가?	x == y	거짓(0), 만족하지 않음
x가 y와 같지 않은가?	x != y	참(1), 만족함

✓ **주의) x = y와 x == y 구분하기**

논리 연산자

- 여러 개의 조건을 조합하여 참(1)과 거짓(0)의 결과 값을 계산하는 연산자

의미	연산자
AND 연산(논리곱 연산) x와 y가 모두 참이면 연산 결과는 참, 그렇지 않으면 연산 결과는 거짓	x && y
OR 연산(논리합 연산) x나 y중에서 하나만 참이면 연산 결과는 참, 모두 거짓이면 연산 결과는 거짓	x || y
NOT 연산(논리부정 연산) x가 참이면 연산 결과는 거짓, x가 거짓이면 연산 결과는 참	!x

x	y	x && y	x || y	!x
거짓	거짓	거짓(0)	거짓(0)	참(1)
거짓	참	거짓(0)	참(1)	
참	거짓	거짓(0)	참(1)	거짓(0)
참	참	참(1)	참(1)	

논리 연산자의 예(3)

예) 다음의 결과값은?

!3 !참 ≡ !1 ***0***
!-3 !참 ≡ !1 ***0***

※ 조건으로 참과 거짓 판별할 때
→ 거짓은 0, 참은 0이 아닌 값

x	y
거짓(0)	거짓(0)
거짓(0)	참(0이 아닌 수)
참(0이 아닌 수)	거짓(0)
참(0이 아닌 수)	참(0이 아닌 수)

※ 논리 연산의 결과값
→ 거짓은 0, 참은 1

x && y	x || y	!x
거짓(0)	거짓(0)	참(1)
거짓(0)	참(1)	
거짓(0)	참(1)	거짓(0)
참(1)	참(1)	

연산자 우선순위(4)

- 여러 연산자가 함께 사용될 때 먼저 수행되는 순서가 정해져 있다.

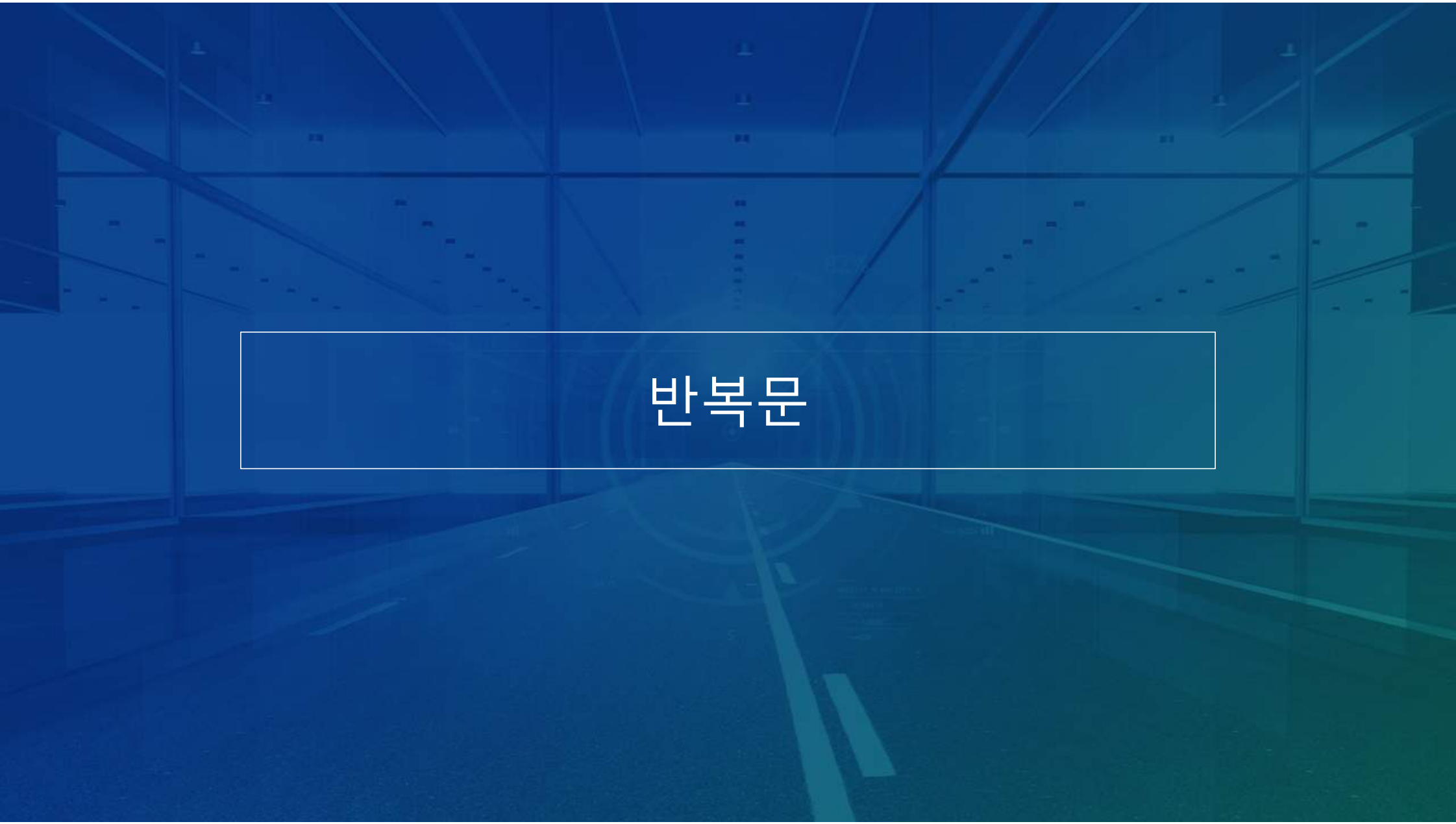
우선 순위	분류	연산자
1	괄호	( )
2	단항	(자료형)
3	산술	* / %
4		+ -
5	관계	> >= < <=
6		== !=
7	대입	=

연산자 우선순위(5)

- 여러 연산자가 함께 사용될 때 먼저 수행되는 순서가 정해져 있다.

우선 순위	분류	연산자
1	괄호	( )
2	단항	(자료형) !
3	산술	* / %
4		+ -
5	관계	> >= < <=
6		== !=
7	논리	&&
8		||
9	대입	=

- 기본적으로 논리 연산자는 관계 연산자보다 우선순위가 낮다.
  - ✓ **단, !(논리부정) 연산자는 관계 연산자보다 우선순위가 높다.**
  - ✓ **&&(논리곱) 연산자는 ||(논리합) 연산자보다 우선순위가 높다.**


반복문

for 문

- 정해진 횟수만큼 반복하는 구조로 반복 횟수를 알고 있을 때 사용

- 형식)

```
for ( 초기식; 조건식; 증감식 ) {
    문장(들);
}
```

- 사용 방식

① for문의 시작에서 초기식은 한번만 실행한다.

② 조건식의 계산 결과가 참이면
{...} 내부의 문장(들)을 실행한다.
증감식을 실행한다.
②로 돌아간다.

조건식의 계산 결과가 거짓이면
{...} 외부의 문장(들)이 실행된다.

for 문의 예

```
#include <stdio.h>
int main() {
   int i;

   for (i = 0; i<3; i = i + 1) {
      printf("[i]: %d     [i 의 세제곱]: %d₩n", i, i * i * i);
   }
   printf("계산 종료!!!₩n");

   return 0;
}
```

```
[결과]
[i]: 0     [i 의 세제곱]: 0
[i]: 1     [i 의 세제곱]: 1
[i]: 2     [i 의 세제곱]: 8
계산 종료!!!
```

초기식, i=0	조건식, i<3	for문 내부, i * i * i	증감식, i = i+1	for문 외부
1) i → 0				
	2) 0 < 3 (참)	3) 0 * 0 * 0 → 0	4) i → 1	
	5) 1 < 3 (참)	6) 1 * 1 * 1 → 1	7) i → 2	
	8) 2 < 3 (참)	9) 2 * 2 * 2 → 8	10) i → 3	
	11) 3 < 3 (거짓)			
				12) 계산 종료!!! 출력

중첩 for 문(1)

- 외부 for문의 { ... } 내 반복되는 문장 중 하나로 for문이 있는 형태

- 형식)

```
for ( 초기식; 조건식; 증감식 ) {      // 외부의 for문
    ...
    for ( 초기식; 조건식; 증감식 ) {      // 내부의 for문
        ...
    }
    ...
}
```

중첩 for 문(2)

- 사용 방식 :
  - ① 외부 for문의 시작에서 초기식은 한번만 실행한다.
  - ② 외부 for문의 조건식 계산 결과가 참이면

내부 반복문을 실행한다.
A) 내부 for문의 시작에서 초기식은 한번만 실행한다.
B) 내부 for문의 조건식 계산 결과가 참이면
내부 for문의 {...} 안쪽 문장(들)을 실행한다.
내부 for문의 증감식을 실행한다.
B)로 돌아간다.
내부 for문의 조건식 계산 결과가 거짓이면
내부 for문의 {...} 바깥 쪽 문장(들)을 실행한다.

외부 for문의 증감식을 실행한다.
②로 돌아간다.

외부 for문의 조건식 계산 결과가 거짓이면
외부 for문 {...} 바깥 쪽 문장(들)을 실행한다.

→ 외부 for문의 반복 문장에서 내부 for문을 만나면 내부 for문에 대한 반복 처리를 다 해 준 후 , 외부 for문의 증감식을 실행하고 다시 조건식을 판단한다.

중첩 for 문의 예(1)

```
// 중첩 for문
for (y = 3; y <6; y++) { //외부 for문
   for (x = 1; x <5; x++) {  //내부 for문
      printf("%d*%d=%2d    ", y, x, y * x);
   }
   printf("₩n");
}
```

y=3 y<6 (참)

초기식, x=1	조건식, x<5	for문 내부, y * x;	증감식, x++	for문 외부
1) x → 1				
	2) 1 < 5 (참)	3) 3*1 출력	4) x → 2	
	5) 2 < 5 (참)	6) 3*2 출력	7) x → 3	
	8) 3 < 5 (참)	9) 3*3 출력	10) x → 4	
	11) 4 < 5 (참)	12) 3*4 출력	13) x → 5	
	14) 5 < 5 (거짓)			15) printf("₩n);

y=4 y<6 (참)

초기식, x=1	조건식, x<5	for문 내부, y * x;	증감식, x++	for문 외부
1) x → 1				
	2) 1 < 5 (참)	3) 4*1 출력	4) x → 2	
	5) 2 < 5 (참)	6) 4*2 출력	7) x → 3	
	8) 3 < 5 (참)	9) 4*3 출력	10) x → 4	
	11) 4 < 5 (참)	12) 4*4 출력	13) x → 5	
	14) 5 < 5 (거짓)			15) printf("₩n);

중첩 for 문의 예(2)

```
// 중첩 for문
for (y = 3; y <6; y++) {//외부 for문
  for (x = 1; x <5; x++) { //내부 for문
    printf("%d*%d=%2d   ", y, x, y * x);
  }
  printf("₩n");
}
```

y=5 y<6 (참)

초기식, x=1	조건식, x<5	for문 내부, y * x;	증감식, x++	for문 외부
1) x → 1				
	2) 1 < 5 (참)	3) 5*1 출력	4) x → 2	
	5) 2 < 5 (참)	6) 5*2 출력	7) x → 3	
	8) 3 < 5 (참)	9) 5*3 출력	10) x → 4	
	11) 4 < 5 (참)	12) 5*4 출력	13) x → 5	
	14) 5 < 5 (거짓)			15) printf("₩n);

y=6 y<6 (거짓)

연산자 우선순위(6)

- 여러 연산자가 함께 사용될 때 먼저 수행되는 순서가 정해져 있다.

우선 순위	분류	연산자
1	괄호	( )
2	단항	(자료형) ! ++ --
3	산술	* / %
4		+ -
5	관계	> => < <=
6		== !=
7	논리	&&
8		||
9	대입	=

[실습] 다음과 같은 결과가 출력되도록 프로그램을 작성하시오.

[결과]

원하는 두 주사위의 합을 입력하세요 : ***7***

두 주사위 합이 7이(가) 되는 경우의 수를 찾아 봅니다.

```
--------------------
주사위A    주사위B
--------------------
1          6
2          5
3          4
4          3
5          2
6          1
```

[조건]

- 두 주사위의 합인 2~12를 입력받아 해당되는 경우의 수를 출력
- 중첩 for문을 사용

While문

- 형식)

```
while ( 조건식 ) {

    문장(들);
}
```

- 사용 방식

① 조건식의 계산 결과가 참이면
{...} 내부의 문장(들)을 실행한다.
①로 돌아간다.
조건식의 계산 결과가 거짓이면
{...} 외부의 문장(들)을 실행한다.

중첩 while문

- 외부 while문의 {...} 내 반복되는 문장 중 하나로 while문이 있는 형태

- 형식)

```
while ( 조건식 ) {  //외부 while문
    ...
  while ( 조건식 ) { //내부 while문
      ...
   }
   ...
}
```

- 사용 방식 :

※ 문법과 작동 방식이 중첩 for문과 유사

→ 외부 while문의 반복 문장에서 내부 while문을 만나면
내부 while문에 대한 반복 처리를 다 해 준 후,
외부 while문의 조건식을 판단한다.

→

① 외부 while문의 조건식 계산 결과가 참이면
내부 while문을 실행한다.
A) 내부 while문의 조건식 계산 결과가 참이면
내부 while문의 {...} 안쪽 문장(들)을 실행한다.
A)로 돌아간다.

내부 while문의 조건식 계산 결과가 거짓이면
내부 while문의 {...} 바깥 쪽 문장(들)을 실행한다.
①로 돌아간다.
외부 while문의 조건식 계산 결과가 거짓이면
{...} 외부의 문장(들)을 실행한다.

do while 문

- 반복 내용을 반드시 한 번은 실행한 후 계속할지 그만둘지 결정할 때 사용되는 반복문

예) 메뉴를 보여주고, 사용자가 프로그램을 계속 진행하거나 종료하는 메뉴를 선택할 수 있도록 하는 경우 등

- 형식)

```
do {
        문장(들);
} while( 조건식 );
```

- 사용 방식

① {...} 내부의 문장(들)을 반드시 한 번은 실행한다.
② 조건식의 계산 결과가 참이면
{...} 내부의 문장(들)을 실행한다.
②로 돌아간다.
조건식의 계산 결과가 거짓이면
{...} 외부의 문장(들)을 실행한다.

무한 반복(무한 loop)와 break문

- 무한 반복 : 반복문의 조건식 결과가 항상 참이어서 결코 거짓이 되지 않아 반복이 종료되지 않고 계속 반복되는 상태

예)

```
while (1) {  // 조건식에 상수 1이 있어서, 조건식이 항상 참
    …
}
```

- 무한 반복에서 탈출하려면, break문을 사용한다.

예) n이 음수일 때, break문으로 인해 반복문을 탈출

```
while (1) {
        …
        if (n  <  0){
            break;
        }
        …
}
```

break문과 continue문

- break문
  - 반복문 안에서 break를 만나면 반복문을 탈출

- continue문
  - 반복문 안에서 continue를 만나면 현재 반복을 멈추고 다음 반복 관련 문장을 진행

for 문

```
for ( i = 1 ; i < 10 ; i++ )
{
    if( i % 2 == 0 ){
        continue;}
    printf("%d ", i);
}
```

while 문

```
i = 1;
while ( i < 10 )
{
    if( i % 2 == 0 ){
        continue;}
    printf("%d ", i);
    i++;
}
```

do while 문

```
i = 1;
do {
    if( i % 2 == 0 ){
        continue;}
    printf("%d ", i);
    i++;
} while ( i < 10 );
```

[실습] 다음과 같은 결과가 출력되도록 프로그램을 작성하시오.

[결과]
*******메뉴*******
1. 라떼
2. 우유
3. 종료
****
메뉴를 선택하세요 : ***0***

*******메뉴*******
1. 라떼
2. 우유
3. 종료
****
메뉴를 선택하세요 : ***4***

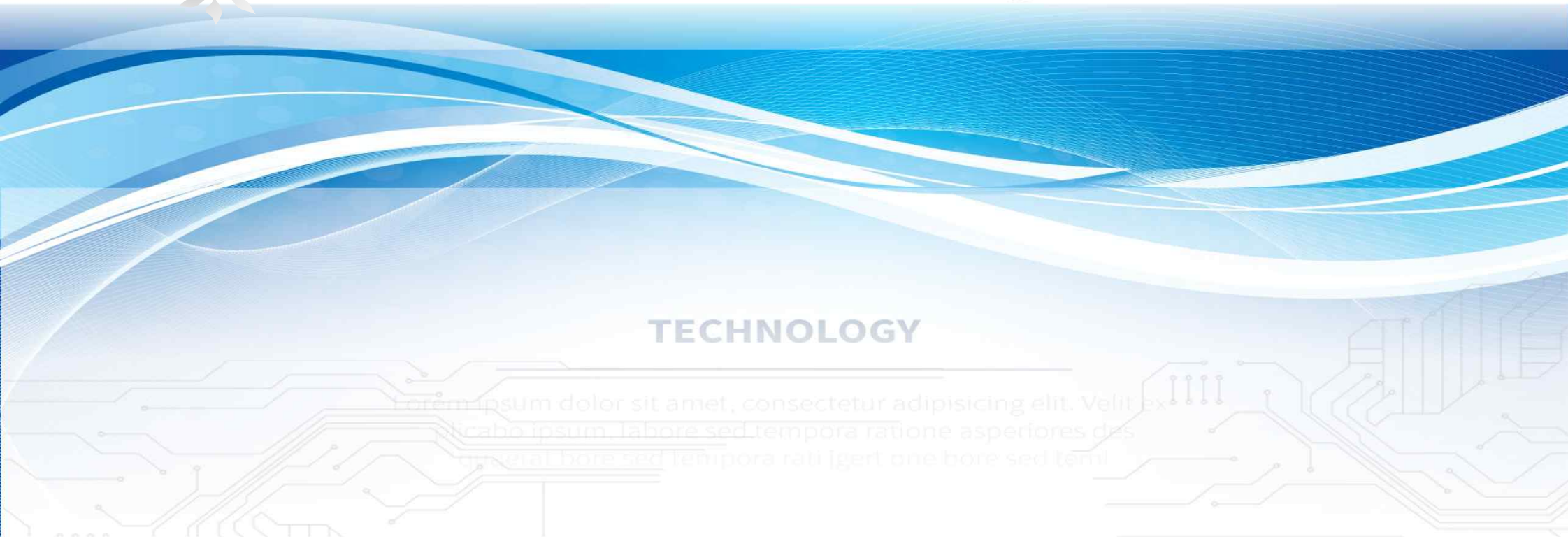
*******메뉴*******
1. 라떼
2. 우유
3. 종료
****
메뉴를 선택하세요 : ***1***
라떼를 선택했습니다.

*******메뉴*******
1. 라떼
2. 우유
3. 종료
****
메뉴를 선택하세요 : ***2***
우유를 선택했습니다.

*******메뉴*******
1. 라떼
2. 우유
3. 종료
****
메뉴를 선택하세요 : ***3***
메뉴 프로그램을 종료합니다.
계속하려면 아무 키나 누르십시오 . . .

[조건]
- 3번 종료 메뉴를 선택할 때까지 다음을 반복
  - 선택 입력 받기, ex. scanf_s("%d", &i);
  - 1~2번의 메뉴를 선택하면, 선택된 상품을 출력
  - 1~3번 외의 숫자를 입력하면, 다시 메뉴를 보여줌
- do ~ while문과 switch문을 사용

T h a n k y o u


TECHNOLOGY
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Velit ex
Incabo ipsum. labore sed tempora ratione asperiores des
labore sed tempora rati [gert one bore sed temi