5. 제어구조

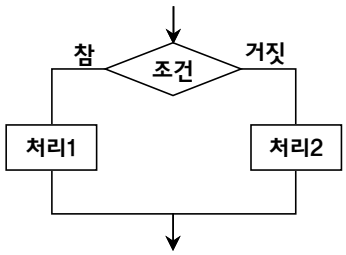
C의 제어구조는 다음과 같다.

선택	if switch
반복	for while do - while
분기	goto break continue return

(1) if ~ else

[형식] if (조건)

```
{
    처리1
}
else
{
    처리2
}
```



① 조건이 **참**이면 '**처리1**'이 실행되고, **거짓**이면 '**처리2**'가 실행된다.
② 처리할 문장이 하나이면 블록 { }은 생략할 수 있다.
③ else 이하는 생략할 수 있다

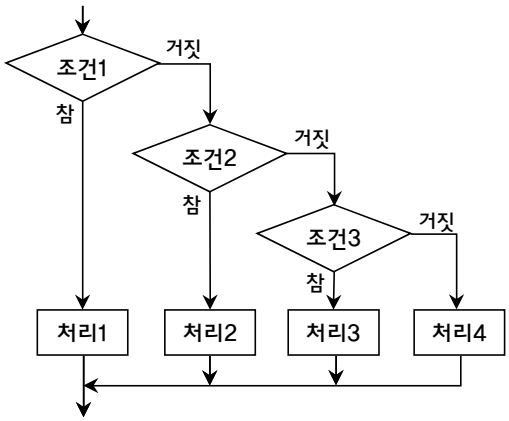
```
#include<stdio.h>
void main(){
  int  a = 1, b = 100;
  if( a < 5 )              //조건은 참
    b++;
  else
    b--;
  printf("실행결과 : %d\n", b);
}
```

실행결과 : 101

◉ if ~ else if ~ else 문

여러 개의 조건을 이용해서 특정 부분을 **처리**할 때 사용할 수 있다.

```
[형식] if (조건1)
          처리1
       else if (조건2)
          처리2
       else if (조건3)
          처리3
       else
          처리4
```



- 처음 if와 마지막 else 사이에는 여러 개의 'else if'가 올 수 있다.
- else 이하는 생략할 수 있다

예제

```
#include<stdio.h>
void main ( )
{
  int sco = 85;
  if( sco >= 90 )
    puts("수");
  else if( sco >= 80 )        //조건이 참이 되어
    puts("우");               //이 부분이 수행된다.
  else if( sco >= 70 )
    puts("미");
  else if( sco >= 60 )
    puts("양");
  else
    puts("가");
}
```

실행 결과 : 우

(2) switch ~ case

```
[형식] switch(수식)
       {
          case 값1  :  처리 1
                       break ;
          case 값2  :  처리 2
                       break ;
                ⋮
          default   :  처리 n
       }
```

- 수식의 값과 일치하는 경우의 값이 **있으면**, '**처리**' 부분을 수행하고,
- 수식의 값과 일치하는 경우의 값이 **없으면**, **default**의 '**처리 n**'을 수행한다.
- **default**문의 위치는 마지막이 아닌 아무 곳이나 올 수 있다.
- 단지, **관례적**으로 **default**문의 위치를 **마지막**에 둔다.
- break를 만나면 switch 블록을 탈출한다.
- 만약, **break**가 **하나도 없으면** 수식의 값에 해당하는 경우부터 **모두 수행**한다.
- 즉, **switch** 블록은 수식의 값에 해당하는 경우부터 **break가 있는 곳까지 수행**한다.

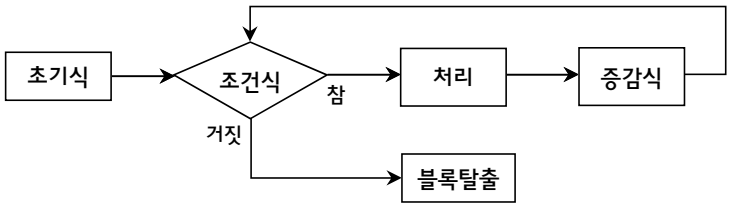
[예]

```
#include<stdio.h>
void main( ) {
   int key = 2, r = 10;
   switch( key )
   {
     case 1 : r = r + 1;
              break;
     case 2 : r = r + 2;        //수식의 값 key=2이므로 수행이 시작되고
     case 3 : r = r + 3;
              break;            //break;문이 있으므로 switch 블록을 탈출
     default : r = r + 4;
              break;
   }
   printf("출력 결과 : %d\n", r);
}
```

출력 결과 : 15

(3) for

[형식] for (초기식; 조건식; 증감식)
{
처리
}



① 조건식이 **거짓**이면 블록을 **탈출**한다.
② for 블록은 다음과 같이 반복 수행된다.

예제

1에서 10까지 정수 합 구하기

```
#include<stdio.h>
void main()
{
  int i, hap = 0;
  for(i=1; i<=10; i++)
    hap += i;
  printf("실행결과 : %d\n", hap);
}
실행결과 : 55
```

[Tip] 제어구조 for문에 미치는 문장이 하나이면 블록으로 묶지 않아도 된다.
블록을 하지 않으면 첫 번째 세미콜론(;)까지가 블록으로 간주된다.

◉ 중첩된 for문

for문 안에 또 다른 for문이 올 수 있다.

```
n = 0 ;
for(i = 0;  i < 2;  i++)
{
      for( j = 0;  j < 3;  j++)
      {
          n++;
      }
}
```

Inner

Outer

• 위의 프로그램 구조에서 제어변수 i, j와 n의 값은 다음과 같이 변화된다.

실행 순서	i	j	n
1	–	–	0
2	0	0	1
3		1	2
4		2	3
5		3	
6	1	0	4
7		1	5
8		2	6
9		3	
10	2		

→ 제어변수 j=3이면 조건 j<3이 거짓이므로 Inner loop를 벗어난다.

→ 제어변수 i=2이면 조건 i<2이 거짓이므로 Outer loop를 벗어난다.

for 문에 사용되는 "초기식, 조건식, 증감식"은 **선택적으로 기술**할 수 있다.

〈for문의 여러 가지 사용 형태〉

① for(; ;) → 세미콜론만 있으면 '**무한루프**'가 된다.**(;은 생략할 수 없다)**

② for(; 1 ;) → 조건식만 있다. 상수 1이므로 항상 참이 된다. 역시 '무한루프'이다.

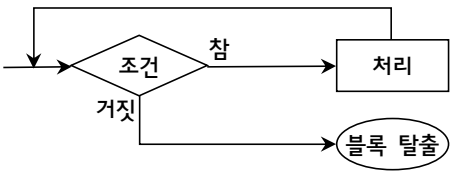
③ for(i=5; i<9 ;) → 증감식이 생략되어 있다.

④ for(i=5. a++; ;) → 초기식만 있다. 역시 '무한루프'이다.

(4) while

[형식] while(조건식)
{
처리
}

① 먼저, 조건식을 판단한 후 **참**이면 블록 내부 '**처리**' 부분을 실행한다.
② 조건식이 거짓이면 블록을 탈출한다.
③ 조건이 처음부터 **거짓**이면 블록 내부는 **전혀** 실행되지 않는다.



예제

1에서 10까지 정수 합 구하기

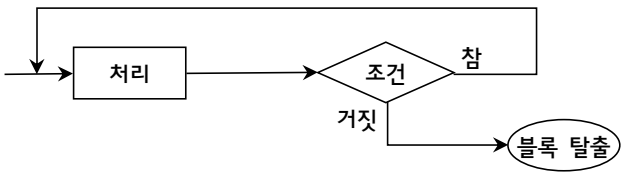
```
#include<stdio.h>
void main()
{
  int  i=0, hap=0;
  while( i < 10 )                    //i = 10이면 블록을 탈출한다.
  {
    i++;
    hap += i;
  }
    printf("실행결과 : %d\n", hap);
}
```

실행결과 : 55

(5) do ~ while

[형식] do
{
처리
}while(조건식);

① 먼저, 블록 내부 '**처리**'를 한번 실행한 후 조건식을 판단한다.
② 조건식이 거짓이면 블록을 탈출한다.



[예] 1에서 100까지 정수의 합

```
#include<stdio.h>
void main()
{
  int i=0, hap=0;
  do{
    i++;
    hap += i;
  }while(i<100);
  printf("실행결과 : %d\n", hap);
}
```

실행결과 : 5050

◉ **제어구조 while과 do~while의 차이점**

- 두 제어구조의 차이는 **'조건 판단이 먼저냐? 블록 내부 실행이 먼저냐?'**에 있다.
- while 구조에서는 블록 내부가 한 번도 실행되지 않을 수 있으며,
- do~while 구조에서는 블록 내부가 최소한 한 번은 실행된다.

(6) break / continue

break	• for, while, do-while, switch에서 블록을 **강제적**으로 **탈출**한다. • if와는 무관
continue	• for에서 실행순서를 **증감식**으로 분기시킨다. • while, do-while에서 실행순서를 **조건식**으로 분기시킨다. • if와는 무관 • switch에 continue를 사용하면 **오류** 발생(사용 불가)

(7) goto

• goto는 프로그램 실행순서를 특정 위치로 **이동**시킬 때 사용한다.

[예] goto

```
#include<stdio.h>
void main(){
  int  i=0, hap=0;
  AA : i++;
       hap += i;
  if ( i < 10 ) goto AA;
  printf("실행결과 : %d\n", hap);
}
```

기출문제 분석

1. 〈보기〉 C 프로그램의 실행 결과는? [2018년 서울 9급]

```
------<보기>--------------------------------
 #include <stdio.h>
 int main( )
 {
   int a=0, b=1;
   switch(a){
     case 0 : printf("%d \ n", b++); ; break;
     case 1 : printf("%d \ n", ++b); ; break;
     default : printf("0 \ n", b); ; break;
   }
   return 0;
 }
-----------------------------------------------
```

① 0　　② 1

③ 2　　④ 3

♤ C 프로그램 실행 결과

```
// 프로그램 분석
 int main( )
 {
   int a=0, b=1;
   switch(a){                                  //a=0
     case 0 : printf("%d \ n", b++); ; break;  //a=0이므로, case 0이 실행되고 블록 탈출
     case 1 : printf("%d \ n", ++b); ; break;
     default : printf("0 \ n", b); ; break;
   }
   return 0;
 }
```

• printf("%d \ n", b++); //b=1이므로, 1이 출력되고 b는 나중에 1 증가한다.

• b++에서 연산자 ++가 변수 뒤에 있음

정답 : ②

2. 다음은 어느 학생이 C 언어로 작성한 학점 계산 프로그램이다. 출력 결과는? [2021년 국가 9급]

```
--------------------------------
#include <stdio.h>
int main()
{
  int score = 85;
  char grade;
  if (score >= 90) grade='A';
  if (score >= 80) grade='B';
  if (score >= 70) grade='C';
  if (score < 70) grade='F';
  printf("학점 : %c\n", grade);
  return 0;
}
--------------------------------
```

① 학점 : A
② 학점 : B
③ 학점 : C
④ 학점 : F

♤ C 언어

```
// 제어구조 if문이 여러 개 사용된 경우
  int main()
  {
    int score = 85;
    char grade;
    if (score >= 90) grade='A';
    if (score >= 80) grade='B';     // score = 85이므로, grade='B'
    if (score >= 70) grade='C';     // score = 85이므로, grade='C'
    if (score < 70) grade='F';
    printf("학점 : %c\n", grade);   // 출력 : 학점 : C
    return 0;
  }
```

• 사용된 if문은 각각 별개이다.

정답 : ③

3. 숫자 0부터 n까지, n을 포함한 합을 구하는 함수를 C 언어로 구현하고자 한다. 이때 n은 0보다 크거나 같은 경우만 고려한다. 다음에 주어진 프로그램에서 /* 공란 */으로 표시된 곳에 채워져야 할 코드로 가장 옳은 것은? [2022년 군무원 9급]

```
------------------
int sum(int n)
{
  int result = 0;

  /* 공란 */
  return result;
}
------------------
```

① for(int i=0; i++; i<n) result = result + i;
② for(int i=0; i++; i<=n) result = result + i;
③ for(int i=0; i<=n; i++) result = result + i;
④ for(int i=0; i<n; i++) result = result + i;

♤ C 언어

```
// 숫자 0부터 n까지, n을 포함한 합을 구하는 함수
int sum(int n)
{
  int result = 0;

  for(int i=0; i<=n; i++) result = result + i;     // 0부터 n까지 합
  return result;
}
int main(void)
{
  int y;
  y = sum(10);
  printf("%d\n", y);          //출력 : 55
  return 0;
}
```

정답 : ③

4. 다음 C 프로그램의 실행 결과로 옳은 것은? [2021년 지방 9급]

```
-----------------------------------------------
 #include <stdio.h>
 int main()
 {
   int count, sum = 0;
   for (count = 1; count <= 10; count++)
   {
     if ((count % 2) == 0)
       continue;
     else
       sum += count;
   }
   printf("%d\n", sum);
 }
-----------------------------------------------
```

① 10 ② 25
③ 30 ④ 55

♤ C 프로그램 - 홀수 합 구하기

```
// 프로그램 분석 - 홀수 합 구하기
 int main()
 {
   int count, sum = 0;
   for (count = 1; count <= 10; count++)
   {
     if ((count % 2) == 0)      // 홀수, 짝수 판별
       continue;                // 짝수이면, 증감식(count++)으로 분기
     else
       sum += count;            // 홀수이면, sum = 1 + 3 + 5 + 7 + 9 = 25
   }
   printf("%d\n", sum);         // 출력 : 25
 }
```

정답 : ②

5. C 프로그램의 실행 결과로 옳은 것은? [2018년 계리직]

```
--------------------------------
#include<stdio.h>
int main( )
{
  int i, sum=0;
  for(i=1; i<=10; i+=2)
  {
    if(i%2 && i%3) continue;
    sum += i;
  }
  printf("%d \n", sum);
  return 0;
}
--------------------------------
```

① 6　　② 12
③ 25　　④ 55

♤ C 프로그램

// 프로그램 분석

```
int main( )
{
  int i, sum=0;
  for(i=1; i<=10; i+=2)          //i = 1, 3, 5, 7, 9 순으로 증가
  {
    if(i%2 && i%3) continue;     //i 값이 2 또는 3의 배수이면 0이 되어 조건은 거짓
    sum += i;                    //sum = 3 + 9 = 12 (조건이 거짓인 경우에 누적)
  }
  printf("%d \n", sum);
  return 0;
}
```

• 명령어 continue;는 for문의 증감식으로 제어를 이동시킨다.

정답 : ②

6. 다음 C 프로그램의 출력 값은? [2017년 국가 9급]

```
#include <stdio.h>
void funCount();
int main(void)
{
  int num;
  for(num=0; num<2; num++)
    funCount();
  return 0;
}
void funCount()
{
  int num=0;
  static int count;
  printf("num = %d, count = %d\n", ++num, count++);
}
```

① num = 0, count = 0 num = 0, count = 1
② num = 0, count = 0 num = 1, count = 1
③ num = 1, count = 0 num = 1, count = 0
④ num = 1, count = 0 num = 1, count = 1

♤ C에서 자동변수와 정적변수

- int num=0; → 자동변수 : 자동변수는 호출시마다 초기화가 된다.
- static int count; → 정적변수 : 정적변수는 초기화가 0으로 **한 번만** 된다.
  ↳ static int count = 0;과 같다. 초기화를 생략한 경우

// 프로그램 진행 과정

num	funCount();	printf("num = %d, count = %d\n", ++num, count++);
0	num = 0, count = 0	num = 1, count = 0
1	num = 0, count = 1	num = 1, count = 1

- ++num : num값 1 증가 후에 출력
- count++ count값 출력 후에 1 증가

정답 : ④

7. 〈보기〉 C 프로그램의 실행 결과로 화면에 출력되는 숫자가 아닌 것은? [2018년 서울 9급]

```
------<보기>-----------------
 #include <stdio.h>
 int my(int i, int j)
 {
   if (i<3) i = j = 1;
   else {
     i = i - 1;
     j = j - i;
     printf("%d, %d, ", i, j);
     return my(i, j);
   }
 }
 int main(void)
 {
   my(5, 14);
   return 0;
 }
 ----------------------------
```

① 1 ② 3
③ 5 ④ 7

♤ C 프로그램 출력 결과

- 이런 유형의 문제는 다음처럼 변수 테이블을 그려서 풀어야 한다.
- 진행 과정은 다음과 같다.

i	j	printf("%d, %d, ", i, j);
5	14	
4	10	**4, 10,**
3	7	**3, 7,**
2	5	**2, 5,**
1	1	

- 출력 결과 : 4, 10, 3, 7, 2, 5,

정답 : ①

8. 다음 C 프로그램의 실행 결과로 옳은 것은? [2016년 계리직]

```
-------------------------------
#include <stdio.h>
int main()
{
    int a = 120, b = 45;
    while( a != b ){
        if( a > b) a = a - b;
        else b = b - a;
    }
    printf("%d", a);
}
-------------------------------
```

① 5 ② 15
③ 20 ④ 25

♤ C 프로그램 – 최대공약수 알고리즘

// 변수 테이블

a	b	a = a – b	b = b – a
120	45	120 – 45 = 75	
75		75 – 45 = 30	
30			45 – 30 = 15
	15	30 – 15 = 15	
15			

• 이런 문제는 변수 테이블을 그려서 변수 값이 변하는 것을 추적하는 것이 좋다.

정답 : ②