작성자 : 김기희 (cupid072@naver.com)

제어문

- 학습목표
  - 문장의 종류를 기술할 수 있다.
  - 제어문의 종류를 기술할 수 있다.
  - 다양한 형태의 if 문을 기술하고 설명할 수 있다.
  - for 문의 형태를 기술하고 수행 순서를 설명할 수 있다.
  - while 문과 do ~ while 문의 형태를 기술하고 수행 순서를 설명할 수 있다.
  - switch 문의 형태를 기술하고 수행 절차를 설명할 수 있다.
  - continue, break, goto 문에 대해 설명할 수 있다.
  - 제어문을 사용하여 주어진 요구사항을 만족하는 프로그램을 작성할 수 있다.

문장(Statement)의 종류

분류		구문 예
선행처리지시문		#include, #define, ...
선언문		변수의 선언, 함수의 선언, 구조체 자료형의 선언
제어문	선택(택일문)	if, if ~ else, if ~ else if, switch ~ case
	반복문	for, while, do~while
	분기문	break, continue, goto
기타		공문 (Empty Statement)

조건문

- 조건식에 따라 실행 여부를 결정한다.
- if 뒤에 괄호()를 사용하여 조건식을 기술한다.
- 조건식 뒤에는 조건식이 참 일 때 수행할 if 절이 뒤 따른다.
- 조건식에 참일 때 수행할 if 절이 여러 라인으로 구성된 복문인 경우 중괄호{}를 사용하여 실행 범위를 지정한다.

```
if (조건식) {
    문장;
}
```

```
if (조건식) 문장;
```

```
if (조건식)
    문장;
```

※ 들여쓰기는 문장의 흐름과는 아무런 상관이 없다.

조건문

- if 절 뒤에 else 절이 뒤 따르는 경우 조건식이 참 일 때 if 절을 수행하며
- if 조건식이 거짓일 경우 else 절을 수행한다.

```
if (조건식) {
    조건이 참일 때 문장;
}
else {
    조건이 거짓일 때 문장;
}
```

```
if (조건식) 조건이 참일 때 문장;
else 조건이 거짓을 때 문장;

if (조건식)
    조건이 참일 때 문장;
else
    조건이 거짓일 때 문장;
```

※ 들여쓰기는 문장의 흐름과는 아무런 상관이 없다.

조건문

- if 절 뒤에는 else if 절이 뒤 따르는 경우 순차적으로 조건식을 평가한다.

```
if (조건식1) {                  조건식1을 평가한다.
    문장1;                      조건식1이 참이면 문장1을 수행한 후
}                               전체 if 문을 빠져나가 문장m을 수행한다.
else if (조건식2) {             이전 if 절의 조건식이 거짓이면 조건식2를 평가한다.
    문장2;                      조건식2가 참이면 문장2를 수행한 후
}                               전체 if 문을 빠져나가 문장m을 수행한다.
else if (조건식n) {             이전 if 절의 조건식이 거짓이면 조건식n을 평가한다.
    문장n;                      조건식n이 참이면 문장n을 수행한 후
}                               전체 if 문을 빠져나가 문장m을 수행한다.
문장m
```

조건문

- else 절은 기술할 수도 있고 기술하지 않을 수도 있으며 기술한다면 무조건 마지막 절로 기술되어야 한다.

```
if (조건식1) {                  조건식1을 평가한다.
    문장1;                      조건식1이 참이면 문장1을 수행한 후
}                               전체 if 문을 빠져나가 문장m을 수행한다.
else if (조건식2) {             이전 if 절의 조건식이 거짓이면 조건식2를 평가한다.
    문장2;                      조건식2가 참이면 문장2를 수행한 후
}                               전체 if 문을 빠져나가 문장m을 수행한다.
else {                          이전 if 절의 조건식이 거짓이면
    문장n;                      문장n을 수행한 후
}                               전체 if 문을 빠져나가 문장m을 수행한다.
문장m
```

조건문

- 조건문의 if 조건식은 논리식이 되어야 한다. (권장)

```
int a = 10;
.....

if (a) {                    // if의 값이 0이 아니면 조건을 만족하지만 산술식이므로 NG
    .....
}

if (a != 0) {               // if 조건식의 값이 논리식이므로 OK
    .....
}
```

반복문 - 구간반복

- 구간 반복

  값의 범위 만큼 반복 수행하는 제어문

```
for (초기식 ; 조건식 ; 증감식) {
    반복 수행할 문장;
}
```

```
for (초기식 ; 조건식 ; 증감식) 반복 수행할 문장;
for (초기식 ; 조건식 ; 증감식)
    반복 수행할 문장;
```

- 반복문 제어를 위해 사용하는 변수를 루프 카운터(loop counter)라고 한다.
- 초기식 : 루프 카운터를 초기화
- 조건식 : 루프 카운터의 값을 평가하여 반복 여부를 결정(조건식이 참인 경우 반복 수행)
- 증감식 : 루프 카운터의 값을 변경한다.

for 문 연습

- 1부터 n 까지의 짝수 값 만을 출력하는 프로그램을 작성해 봅시다.

```
#include <stdio.h>
int main() {
    int n = 10;

    // TODO
    // for 반복문을 사용하여 1부터 10까지 출력하는 코드 작성

    return 0;
}
```

반복문 - 구간반복

- for 문을 반복하기 위한 초기식, 조건식, 증감식을 반드시 기술해야 하는 것은 아니다.

  for (;;) 무한반복실행

- 무한반복이 필요한 경우 for 문 보다는 while문을 사용한다. (권장)

  while (1) 무한반복실행

- for 절 내에서는 루프 카운터의 값을 조작하지 않도록 한다. (권장)
- for 문이 종료된 후 for 문에 사용된 루프 카운터는 참조하지 않도록 한다. (권장)

반복문 - 조건반복

- 조건 반복은 주어진 조건이 참인 동안 반복 수행하는 제어문이다.
- while 뒤에 조건식이 기술되며 조건식 뒤에는 while 절이 뒤 따른다.

```
while (조건식) {
    조건식이 참인 동안 반복 수행;
}
```

- while 문은 선비교 후실행 제어문으로 조건식 평가에 따라 while 절이 한 번도 수행되지 않을 수 있다.

while 반복문 연습

- 정수 n 이 주어졌을 때 소수인가 아닌가를 확인하는 프로그램을 작성해 봅시다.
- 소수일 경우 : "true" 출력
- 소수가 아닌 경우 : "false" 출력

```
#include <stdio.h>

int main() {
    int n = 7;
    // TODO
    // n의 값이 소수인가 아닌가를 확인하는 코드를 작성해보세요.

    return 0;
}
```

반복문 - 조건반복

- do ~ while문은 조건을 만족하는 동안 문장을 반복 수행한다.
- 선비교 후실행의 while 문과는 달리 do ~ while 문은 선실행 후비교이다.

```
do {
   반복 수행 문장;
} while (조건식);
```

- do ~ while 문은 선실행 후비교 제어문으로 조건식에 관계 없이 무조건 한 번은 수행된다.

반복문의 조건식

- for 문의 조건식, while문의 조건식, do ~ while 문의 조건식은 논리식이 되어야 한다. (권장)
  다음의 반복문은 배열 원소의 값이 0이 아닌 동안 반복 수행한다.

```
int arr[] = { 9, 5, 4, 8, 2, 1, 7, 3, 6, 0 };

for (int i=0 ; arr[i] ; i++) {                    // for 문의 조건식이 논리식이 아니므로 NG
      printf("arr[%d] : %d\n", i, arr[i]);
}

int i = 0;
while (arr[i]) {                                  // while 문의 조건식이 논리식이 아니므로 NG
      printf("arr[%d] : %d\n", i, arr[i]);
      i++;
}
```

※ 위의 코드는 문법적으로 에러는 아니지만 Misra-C 제약을 위반한다.

택일문

- 택일문은 조건식의 값에 따라 다른 문장을 수행한다.

```
switch (조건식) {
    case 값1:   문장1;    break;
    case 값2:   문장2;    break;
    case n:     문장n;    break;
    default:    문장m;
}
```

- 조건식은 반드시 정수 값 또는 문자 값을 나타내는 식이어야 한다.
- case 값은 반드시 정수 리터럴 또는 문자 리터럴 이어야 한다.
- case 절의 break 문이 반드시 기술되어야 하는 것은 아니다.
- 동일한 값을 가지는 case 절이 있어서는 안되며 default 절 또한 한 번만 기술되어야 한다.
- default 절은 switch 절 내의 어느곳에서나 기술할 수 있으며 반드시 기술되어야 하는 것은 아니다.
- default 절을 기술하는 경우 마지막 절로 기술할 것을 권장한다.

택일문의 case 값

- case 값으로 리터럴 상수의 사용

```
int jumsu = 96;

switch (jumsu / 10) {
    case 10:
        printf("A");
        break;
    case 9:
        printf("B");
        break;
    ......
    default:
        printf("D");
        break;
}
```

- case 값으로 사용자 정의 상수 사용

```
int jumsu = 96;
const int A = 10;
const int B = 9;

switch (jumsu / 10) {
    case A:          // 정수 리터럴 또는 문자 리터럴이 아니므로 NG
        printf("A");
        break;
    case B:
        printf("B");
        break;
    ......
    default:
        printf("D");
        break;
}
```

택일문의 fall through

- fall through는 허용하지 않도록 한다.

```
int jumsu = 96;

switch (jumsu / 10) {
    case 10:
        printf("A");
    case 9:
        printf("A");
    default:
        printf("B");
}
```

jumsu가 100 인 경우 "AAB" 출력
jumsu가 90 이상인 경우 "AB" 출력
jumsu가 90 미만인 경우 "B" 출력

※ fall through는 문법적인 에러는 아니지만 허용하지 않을 것을 권장한다.

- 예외적으로 fall through를 허용하는 경우

```
int jumsu = 96;

switch (jumsu / 10) {
    case 10:
    case 9:
        printf("A");
        break;
    default:
        printf("B");
        break;
}
```

jumsu가 90 이상인 경우 "A" 출력
jumsu가 90 미만인 경우 "B" 출력

택일문의 default 절

- switch 문의 default 절은 어느 곳에나 기술할 수 있으나 기술 시 가장 마지막 절로 기술할 것을 권장한다.
- switch 문의 default 절을 반드시 기술해야 하는 것은 아니지만 기술 할 것을 권장한다.
- switch 문의 default 절에서 처리할 내용이 없을 경우 공문(empty statement)로 기술한다.

```
switch (condition) {
    case value:
    ......
    default:                // switch 문의 default 절은 가장 마지막 절로 기술한다.
        ;                   // 특별히 수행할 내용이 없는 경우 공문으로 처리한다.
        break;
}
```

분기문

- break 문
  - break 문은 반복문(for, while, do~while)과 택일문(switch) 내에서 사용된다.
  - 반복문이나 택일문 내에서 break 문을 만나면 실행 제어는 break 문이 포함된 가장 안쪽의 반복문이나 택일문 블록을 탈출한다.

- continue문
  - continue문은 반복문(for, while, do ~ while) 내에서 사용된다.
  - 반복문 내에서 continue 문을 만나면 실행제어는 continue 문이 포함된 가장 안쪽 반복문의 조건식으로 이동한다.
  - while 절 내의 continue 문은 실행제어를 while 조건식으로 이동시킨다.
  - for 절 내의 continue 문은 실행제어를 for 문의 증감식으로 이동시킨다.

분기문

- goto 문
  - 같은 블록 내의 특정 label(레이블)로 제어를 옮길 때 사용한다.
  - goto 문과 label은 동일한 함수 내에 존재해야 한다.

```
int a = 0;
start:
printf("a : %d\n", a);
a++
if (a < 10) goto start;
```

- goto 문은 되도록이면 사용하지 않도록 한다.

중첩 제어문

- 중첩 제어문은 제어문 내에 제어문이 기술되는 형식을 말한다.

```
if (condition1) {
    if (condition2) {
        ......
    }
}

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

```
if (condition1 && condition2) {
    ......
}


// 10 * 10회 반복 실행
```

중첩 제어문 연습 #1

- 다음과 같이 *를 출력하는 프로그램을 작성해 보세요.

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

중첩 제어문 연습 #2

- 다음과 같이 *를 출력하는 프로그램을 작성해 보세요.

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

중첩 제어문 연습 #3

- 다음과 같이 *를 출력하는 프로그램을 작성해 보세요. (가로방향 20개, 세로방향 10개)

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