연산자와 제어문

연산자

- 산술 연산자 : 수학적인 연산
  - +, -, *, /, %, =
  - +=, -=, *=, /=, %= : a += 6 → a = a + 6을 줄여쓴 것
  - ++, — : 각각 1을 더하거나 뺄 때 사용
    - 전치와 후치 : 연산 순서를 바꿈. 전치인 경우에는 그 줄의 산술 계산이 시작되기 전, 변수의 값을 변화시키고 후치인 경우에는 그 줄의 산술 계산이 끝난 뒤 변수의 값을 변화시킨다.
- 비교 연산자
  - >, <, ==, > =, < =, ! = : 같다의 경우에는 대입 연산자와 구별하기 위해 ==을 사용함
- 논리 연산자
  - &&(And), ||(Or), !(Not)

```
#include <stdio.h>

int main(){
    int i_a = 3, i_b = 3;

    i_a += 6;
    i_b = i_b + 6;
    printf("i_a += 6 연산의 결과 %d는 i_b = i_b + 6 연산의 결과 %d와 같다\n", i_a, i_b);

    int i_c = 0;
    printf("전치 %d\n", ++i_c);
    i_c = 0;
    printf("후치 %d\n", i_c++);

    int i_d = 3, i_e = 5, i_f = 5;
    bool b_p = i_d == i_e;
    bool b_q = (i_d < i_e) && (i_d != i_f);
    bool b_r = (i_d < i_e) || (i_d != i_f);
    printf("%d %d %d\n", b_p, b_q, b_r);

    return 0;
}
```

```
실행 결과

i_a += 6 연산의 결과 9는 i_b = i_b + 6 연산의 결과 9와 같다
전치 1
후치 0
0 1 1
```

if

- 제어문 (조건문) : 프로그램은 위에서부터 아래로 진행되지만, 어떠한 조건에 의해 문장을 건너뛰는 등 프로그램의 흐름을 바꾸는 역할을 하는 문장들을 제어문이라 함
- if(조건식) { 실행문 } 으로 구성되며, 조건식을 만족하지 못했을 경우 if else ( ), else ( ) 로 넘어가 실행되기도 함
- if문 안에 if문이 존재할 수 있으며, 2개의 조건식을 가지는 효과가 있음

```
#include <stdio.h>

int main(){
    int i_a;
    scanf("%d", &i_a);

    if(i_a % 2 == 0){
        if(i_a % 3 == 0){
            printf("%d는 2의 배수이자 3의 배수", i_a);
        }
        else{
            printf("%d는 2의 배수지만 3의 배수는 아님", i_a);
        }
    }
    else{
        if (i_a % 3 == 0) {
            printf("%d는 2의 배수는 아니지만 3의 배수", i_a);
        }
        else{
            printf("%d는 2의 배수와 3의 배수가 아님", i_a);
        }
    }

    return 0;
}
```

```
입력
```

```
6

실행 결과

6는 2의 배수이자 3의 배수
```

Switch와 Goto

- switch - case : 제한적인 상황에서 사용할 수 있는 if 문과 같음
  - 보통 입력받는 값이 예측 가능하고 한정된 상황에서 사용.
  - 각각의 입력값에 case를 대응하여 실행함
- goto : 함수 내에서 미리 지정해둔 곳으로 점프할 수 있는 함수
  - 개발자 입장에서는 편리할 수 있으나, 코드가 직관적이지 않게 꼬이고, 남발할 경우 해독이 힘들어 현재는 사용을 지양하고 있음.

```
#include <stdio.h>

int main(){
    int i_choise;
    makeChoise:

    printf("새 게임 : 1\n불러오기 : 2\n설정 : 3\n크레딧 : 4\n");
    scanf("%d", &i_choise);

    switch (i_choise) {
        case 1:
            printf("새 게임\n");
            break;

        case 2:
            printf("불러오기\n");
            break;

        case 3:
            printf("설정\n");
            break;

        case 4:
            printf("크레딧\n");
            break;

        default:
            printf("잘못 입력하셨습니다\n");
            goto makeChoise;
            break;
    }
```

```
    return 0;
}
```

```
실행 결과

새 게임 : 1
불러오기 : 2
설정 : 3
크레딧 : 4
0
잘못 입력하셨습니다
새 게임 : 1
불러오기 : 2
설정 : 3
크레딧 : 4
2
불러오기
```

while

- while : 반복문, while (조건식) { 실행문 } 으로 구성되어 조건식이 true라면 실행문을 반복하여 계속 실행시킴.
    - 조건식이 true라면 무한 반복이 가능함
- do - while : do { 실행문 } while (조건식); 으로 구성되어 do의 실행문이 실행된 뒤, while의 조건식이 true라면 다시 do로 돌아감.

```
#include <stdio.h>

int main(){
    int i_a = 0;

    do{
        i_a++;
    }while (i_a <= 10);

    printf("%d\n", i_a);
    return 0;
}
```

```
실행 결과

11
```

for

- for : for (초기화식 ; 조건식 ; 증감식) { 실행문 } 으로 구성되어 제한된 횟수만큼 반복할 수 있음. 따라서 이 식을 응용하면, while과 같이 사용할 수 있음.
  - 일정하게 증가 / 감소되는 함수가 필요할 때,
  - 배열의 모든 원소에 순차적으로 접근하고 싶을 때,
  - 특정 횟수만큼 작업을 반복하고 싶을 때 사용됨
  - if문과 같이 중첩하여 사용할 수 있음.
- break : 반복문 한 개를 빠져나오고 싶을 때 사용
- continue : 반복문의 시작으로 돌아갈 때 사용

```
#include <stdio.h>

int main(){
    int n, sum = 0;
    scanf("%d", &n);

    //일정하게 증가/감소되는 함수가 필요할 때
    for(int i = 1; i <= n; i *= 2){
        printf("%d ", i);
    }
    printf("\n\n");

    //배열의 모든 원소에 순차적으로 접근하고 싶을 때
    int i_a[10] = {0};
    for(int i = 0; i < 10; i++){
        i_a[i] += i;
        printf("%d ", i_a[i]);
    }
    printf("\n\n");

    //특정 횟수만큼 작업을 반복하고 싶을 때
    for(int i = 1; i <= n; i++){
        printf("%d 번만큼 문장을 출력 (%d / %d)\n", n, i, n);
    }
    printf("\n");

    //초기화식 ; 조건식; 증감식에는 하나 이상의 요소가 들어갈 수 있음
    for(int i = 1; i <= n; sum += i++);
    printf("1부터 %d까지의 합 : %d\n\n", n, sum);

    //break : 반복문 하나를 빠져나옴
    for(int i = n - 1; i >= 1; i--){
        if(n % i == 0){
            printf("%d는 %d의 자기 자신을 제외한 가장 큰 약수\n\n", i, n);
            break;
```

```
        }
    }

    //continue : 반복문의 시작으로 돌아갈 때 사용
    for (int i = 1; i <= n; i++) {
        if(i % 3 == 0){
            printf("짝 ");
            continue;
        }
        printf("%d ", i);
    }
    printf("\n\n");

    //중첩 for 문
    for(int i = 0; i < n; i++){
        for (int j = 0; j < i; j++) {
            printf("#");
        }
        printf("\n");
    }

    return 0;
}
```

```
입력

9

실행 결과

1 2 4 8

0 1 2 3 4 5 6 7 8 9

9 번만큼 문장을 출력 (1 / 9)
9 번만큼 문장을 출력 (2 / 9)
9 번만큼 문장을 출력 (3 / 9)
9 번만큼 문장을 출력 (4 / 9)
9 번만큼 문장을 출력 (5 / 9)
9 번만큼 문장을 출력 (6 / 9)
9 번만큼 문장을 출력 (7 / 9)
9 번만큼 문장을 출력 (8 / 9)
9 번만큼 문장을 출력 (9 / 9)

1부터 9까지의 합 : 45

3는 9의 자기 자신을 제외한 가장 큰 약수

1 2 짝 4 5 짝 7 8 짝


#
##
###
####
```

```
#####
######
#######
########
```

Section 2. 종합문제

- 시험점수를 입력받아 등급을 나누어주는 프로그램을 작성해 보세요

```
#include <stdio.h>

int main(){
    int i_score;
    scanf("%d", &i_score);

    if(i_score < 0 || i_score > 100){
        printf("잘못 입력하셨습니다.\n");
    }
    else{
        if(i_score >= 90){
            printf("A\n");
        }
        else if(i_score >= 80){
            printf("B\n");
        }
        else if(i_score >= 70){
            printf("C\n");
        }
        else if(i_score >= 60){
            printf("D\n");
        }
        else{
            printf("E\n");
        }
    }

    return 0;
}
```

```
입력

76

실행 결과

C
```

- 자연수를 입력받아 약수를 출력하는 프로그램을 작성해 보세요.

```
#include <stdio.h>

int main(){
    int i_n;
    scanf("%d", &i_n);

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

    return 0;
}
```

```
입력

12

실행 결과

1 2 3 4 6 12
```

- n을 입력받아 1부터 n까지의 숫자 중 일의 자릿수가 3, 6, 9인 경우 *을 출력하는 프로그램을 작성해 보세요.

```
#include <stdio.h>

int main(){
    int i_n;
    scanf("%d", &i_n);

    for(int i = 1; i <= i_n; i++){
        if((i % 10) % 3 == 0 && i % 10 != 0){
            printf("* ");
            continue;
        }
        printf("%d ", i);
    }

    return 0;
}
```

입력

17

실행 결과

1 2 * 4 5 * 7 8 * 10 11 12 * 14 15 * 17