

3장 연산자, 반복문

2장까지는 C언어 IDE 툴로 Visual Studio를 사용함
3장은 C언어 IDE 툴로 VS Code를 사용할 예정

1. 이 장에서 만드는 프로그램
2. 연산자
3. 반복문의 종류
4. 이중 반복문 사용하기
5. 프로젝트: 피라미드를 쌓아라



VS Code에서 C언어 설치

- “MinGW”검색하여 설치(C언어 컴파일러)
 - <https://sourceforge.net/projects/mingw> 다운로드 설치
 - MinGW Installation Manager에서

- ✓ Mingw-developer-toolkit
- ✓ Mingw32-base
- ✓ Mingw32-gcc-g++
를 mark for Installation

✓ [Installation] > [Apply Changes] 설치 진행




✓ 시스템 환경 변수 편집

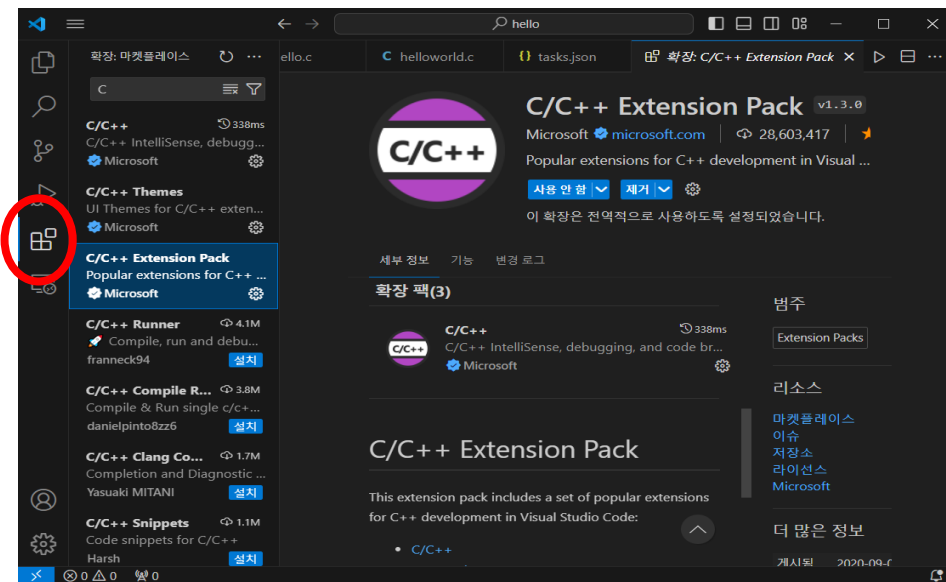
✓ [환경변수] > 시스템 변수 PATH에 c:/MinGW/bin추가

✓ Cmd창에 실행

- gcc --version
- g++ --version
- gdb --version

- VS code(user) 설치 후 확장 설치
 - Korean Language Pack 설치
 - C/C++ Extension Pack 설치
 - Code Runner 설치
(ctrl+alt+n 실행)
(ctrl+alt+m 실행중지)

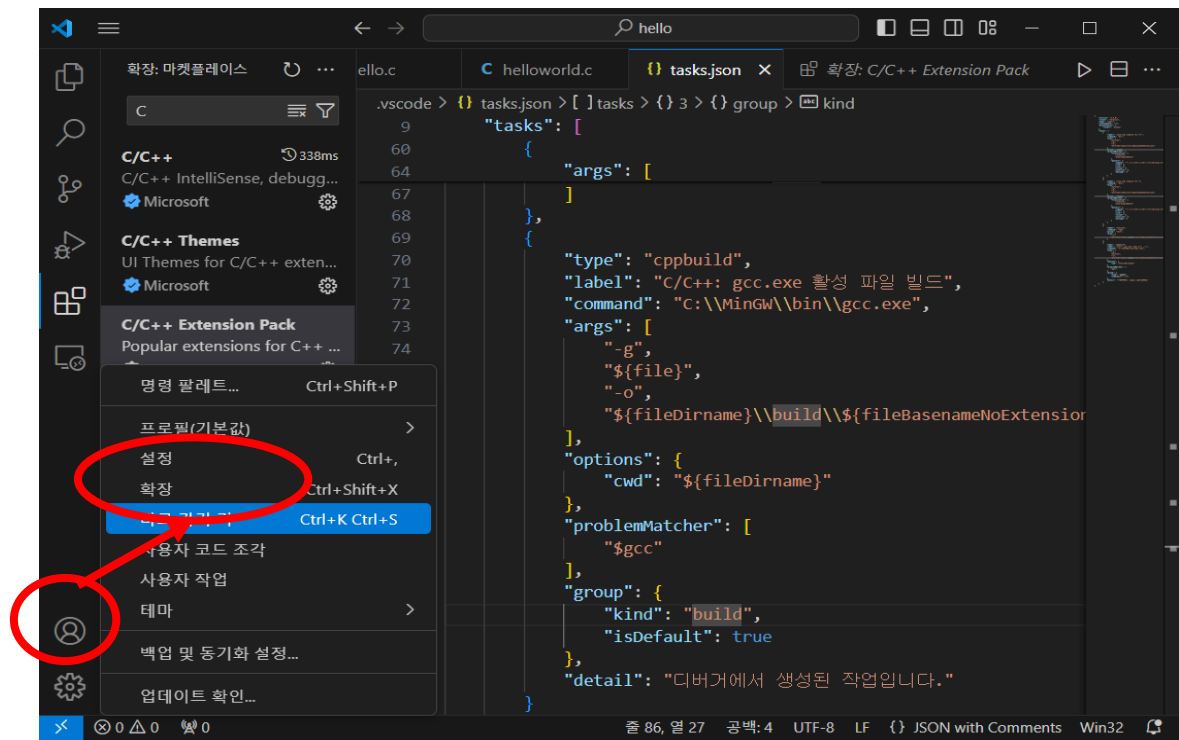
Package	Class	Installed Version	Repl
 mingw-developer-toolkit	bin		2013
 mingw32-base	bin		2013
<input type="checkbox"/> mingw32-gcc-ada	bin	6,3,0	
<input type="checkbox"/> mingw32-gcc-fortran	bin	6,3,0	
 mingw32-gcc-g++	bin	6,3,0	
<input type="checkbox"/> mingw32-gcc-objc	bin	6,3,0	





VS Code에서 C언어 설치

- VS code 폴더(ch03_loop) 열기한 후 C 소스파일을 하나 놓는다.
- [터미널] > [기본빌드 작업구성] > C/C++ gcc.exe 활성화 파일 빌드를 선택하면 tasks.json파일이 생성되는데 교안폴더의 내용으로 수정
- VS code 폴더(ch03_loop) 열기한 후 **교안의 .vscode폴더를 복사해도 무방**
- [바로가기] 선택후





-



1. 이 장에서 만드는 프로그램

원하는 구구단 단수의 범위를 입력받아 출력하는 프로그램을 구현

실행결과						—	□	×
구구단을 몇단부터 몇단까지 출력할지 최소구구단수와 최대구구단수						>2	7	
2 x 1 = 2	3 x 1 = 3	4 x 1 = 4	5 x 1 = 5	6 x 1 = 6	7 x 1 = 7			
2 x 2 = 4	3 x 2 = 6	4 x 2 = 8	5 x 2 = 10	6 x 2 = 12	7 x 2 = 14			
2 x 3 = 6	3 x 3 = 9	4 x 3 = 12	5 x 3 = 15	6 x 3 = 18	7 x 3 = 21			
2 x 4 = 8	3 x 4 = 12	4 x 4 = 16	5 x 4 = 20	6 x 4 = 24	7 x 4 = 28			
2 x 5 = 10	3 x 5 = 15	4 x 5 = 20	5 x 5 = 25	6 x 5 = 30	7 x 5 = 35			
2 x 6 = 12	3 x 6 = 18	4 x 6 = 24	5 x 6 = 30	6 x 6 = 36	7 x 6 = 42			
2 x 7 = 14	3 x 7 = 21	4 x 7 = 28	5 x 7 = 35	6 x 7 = 42	7 x 7 = 49			
2 x 8 = 16	3 x 8 = 24	4 x 8 = 32	5 x 8 = 40	6 x 8 = 48	7 x 8 = 56			
2 x 9 = 18	3 x 9 = 27	4 x 9 = 36	5 x 9 = 45	6 x 9 = 54	7 x 9 = 63			



2. C언어 연산자

순위	명칭	연산자	결합성
①	1 차 연 산 자	()	좌결합성 →
②	단 항 연 산 자	! ++ --	우결합성 ←
③	이 항 연 산 자	승법연산자	→
④		가법연산자	
⑤		관계(비교)연산자	
⑥		비트곱연산자	
⑦		비트합연산자	
⑧		논리곱연산자	
⑨		논리합연산자	
⑩	조건(3항)연산자	? :	
⑪	할 당 연 산 자	= += -= *= /= %=	←



2. 연산자

```
#include <stdio.h>
// 산술연산자 : + - * / %(나머지연산자)
int main(void){
    int n1 = 33, n2=10;
    printf("%d %c %d = %d\n", n1, '/', n2, n1/n2); // 정수와 정수끼리 계산결과는 정수
    printf("%d %c %d = %.2lf\n", n1, '/', n2, (double)n1/n2);
    printf("%d %c %d = %d\n", n1, '%', n2, n1%n2);
    // 나머지 연산자의 다른 용도
    // 짝수/홀수 판별, 배수 판별 용도
    // 모든 정수에 나머지 연산자 2를 적용했을 때 값이 0이면 짝수, 1이면 홀수.
    // 배수를 판별할 때에는 나머지의 값이 0인지 확인합니다
    if ( (n1%2) == 0 )
        printf("n1은 짝수입니다\n");
    else
        printf("n1은 홀수입니다\n");
    if ( (n1%5) == 0 )
        printf("n1은 5의 배수입니다\n");
    else
        printf("n1은 5의 배수가 아닙니다\n");
}
```




2. 연산자

```
#include <stdio.h>
// 증감연산자 : ++(1증가), --(1감소)

int main(void){
    int n1 = 10;
    printf("++n1 = %d\n", ++n1); // n1을 1 증가후, 현재 줄 수행
    printf("n1++ = %d\n", n1++); // 현재 줄 수행한 후 1 증가
    printf("n1 = %d\n", n1 );
}
```



2. ++ 연산자

- ++ 연산자가 변수 앞에 있을 때(전위) : 먼저 1 증가 연산을 한 후에 출력 작업 수행
- ++ 연산자가 변수 뒤에 있을 때(후위) : 먼저 출력 작업을 수행한 후에 다음 줄로 넘어가기 전에 1 증가 연산 수행

그림 3-2 ++ 연산자의 연산 순서

++b(전위)

```
printf("b는 %d\n", ++b);  
① b += 1  
② printf("b는 %d\n", b);
```

b++(후위)

```
printf("b는 %d\n", b++);  
② b += 1  
① printf("b는 %d\n", b);
```



2. 연산자

```
#include <stdio.h>
#include <stdbool.h>
// 동등비교연산자(관계연산자) : ==(같다), !=(다르다), >, >=, ....
// 삼항연산자 (조건식)? (조건이참일경우 취할 값) : (조건이거짓일경우취할값)
int main(void){
    int n1 = 10, n2=5;
    bool result;
    result = n1>=n2;
    printf("%d %s %d 는 %s\n", n1, ">=", n2, result? "true":"false");
    result = n1==n2;
    printf("%d %s %d 는 %s\n", n1, "==", n2, result? "true":"false");
    result = n1!=n2;
    printf("%d %s %d 는 %s\n", n1, "!=", n2, result? "true":"false");

    result = !(n1!=n2); // 논리연산자(반대를 의미)
    printf("!(n1!=n2)는 %s\n", result? "true":"false");
}
```



2. 연산자

```
#include <stdio.h>
```

```
int main(void){  
    int i=1, j=10, h=10;  
    printf("&&(and) : (i<j) && (++j>h) 는 %s\n", ((i<j) && (++j>h))? "true":"false");  
    printf("j = %d\n", j);  
  
    // &&(AND) 연산의 경우 좌항이 false인 우항의 값은 보지도 않고 결과를 false로  
    printf("&&(and) : (i>j) && (++j>h) 는 %s\n", ((i>j) && (++j>h))? "true":"false");  
    printf("j = %d\n", j);  
  
    // ||(OR) 연산의 경우 좌항이 true면 우항의 값은 보지도 않고 결과를 true로  
    printf("||(OR) : (i<j) || (++j>h) 는 %s\n", ((i<j) || (++j>h))? "true":"false");  
    printf("j = %d\n", j);  
}
```



2. 연산자

```
#include <stdio.h>
```

```
int main(void){  
    int n1 = 10; // 0 ~ 0 1 0 1 0  
    int n2 = 6;  // 0 ~ 0 0 1 1 0  
    // -----  
    //      &  : 0 ~ 0 0 0 1 0 => 2  
    //      |   : 0 ~ 0 1 1 1 0 => 14  
    //      ^   : 0 ~ 0 1 1 0 0 => 12  
    printf("n1 & n2 = %d\n", (n1&n2));  
    printf("n1 | n2 = %d\n", (n1|n2));  
    printf("n1 ^ n2 = %d\n", (n1^n2));  
    // 시프트 연산자 1bit 왼쪽 시프트는 2배  
    printf("n1 << 1 = %d\n", (n1<<1));  
    printf("n1 >> 1 = %d\n", (n1>>1));  
}
```



2. 연산자

```
#include <stdio.h>
// 할당(대입)연산자 : =, +=, -=, *=, /=, %=
int main(void){
    int n1 = 10;
    n1 += 10; // n1 = n1+10
    printf("n1 = %d\n", n1);
    n1 *= 10; // n1 = n1*10
    printf("n1 = %d\n", n1);
    int n2, n3;
    n1 = n2 = n3 = 10;
    printf(" n1=%d\n n2=%d\n n3=%d\n", n1, n2, n3);
}
```




3. 반복문의 종류

(1) for 문

형식 **for** (선언; 조건; 증감) {
 // 수행할 문장
 }

3.3.1 for.c

```
int main(void) {  
    for (int i = 1; i <= 10; i++) {  
        printf("Hello World %d\n", i);  
    }  
    return 0;  
}
```

실행결과

```
Hello World 1  
Hello World 2  
Hello World 3  
Hello World 4  
Hello World 5  
Hello World 6  
Hello World 7  
Hello World 8  
Hello World 9  
Hello World 10
```




3. 반복문의 종류

(1) for 문

- 변수를 선언하고 초기화한 후 변수의 조건을 확인
- 조건에 해당하면 중괄호 안 문장을 수행
- 변수의 값을 증감한 후 다시 조건을 확인

그림 3-3 for 문의 작동 순서

```
for (int i = 1; i <= 10; i++) {  
    printf("Hello World %d\n", i);  
}
```

① 선언 ②⑤ 조건 확인 ④ 증가
③⑥ 문장 수행



3. 반복문의 종류

(2) while 문

형식 선언;
`while` (조건) {
 // 수행할 문장(증감 포함)
}

3.3.2 while.c

```
int main(void) {  
    int i = 1;  
    while (i <= 10) {  
        printf("Hello World %d\n", i);  
    }  
    return 0;  
}
```

실행결과

```
Hello World 1  
Hello World 2  
Hello World 3  
Hello World 4  
Hello World 5  
Hello World 6  
Hello World 7  
Hello World 8  
Hello World 9  
Hello World 10
```



3. 반복문의 종류

(2) while 문

- while 문 위에 변수를 선언하고 초기화
- 조건을 확인해 문장 수행
- 수행하고 나서 다시 조건으로 돌아가 조건에 맞으면 다시 문장 수행

그림 3-4 while 문의 작동 순서

```
int i = 1;           ① 선언
while (i <= 10) {    ②⑤ 조건 확인
    printf("Hello World %d\n", i++);  ③⑥ 문장 수행  ④⑦ 증가
}
```



3. 반복문의 종류

(3) do-while 문

형식 선언;
do {
 // 수행할 문장
} while (조건);

3.3.3 do_while.c

```
int main(void) {  
    int i = 1;  
    do {  
        printf("Hello World %d\n", i++);  
    } while (i <= 10);  
    return 0;  
}
```

실행결과

```
Hello World 1  
Hello World 2  
Hello World 3  
Hello World 4  
Hello World 5  
Hello World 6  
Hello World 7  
Hello World 8  
Hello World 9  
Hello World 10
```



3. 반복문의 종류

(3) do-while 문

- do 뒤의 중괄호 안에 수행할 문장을 작성
- while 뒤에 문장을 수행할 조건을 작성
- do-while 문 앞에 변수 선언
- 주의 : do-while 문의 조건 뒤에 세미콜론을 빼먹으면 안 된다!

그림 3-5 do-while 문의 작동 순서

```
int i = 1;
do {
    printf("Hello World %d\n", i++);
} while (i <= 10);
```

① 선언 ②⑤ 문장 수행 ③⑥ 증가 ④⑦ 조건 확인



3. 3.5_구구단quiz.c

- 원하는 구구단수를 입력받아 구구단을 출력하는 프로그램을 구현하시오



4. 실행 중단하기

(1) break 문

- 조건을 만족하든 안 하든 상관없이 for 문을 탈출

4.3.1 break.c

```
int main(void) {  
    for (int i = 1; i <= 30; i++) {  
        if (i >= 6) {  
            printf("나머지 학생은 집에 가세요.\n");  
            break;  
        }  
        printf("%d번 학생은 조별 발표를 준비하세요.\n", i);  
    }  
    return 0;  
}
```

실행결과

```
1번 학생은 조별 발표를 준비하세요.  
2번 학생은 조별 발표를 준비하세요.  
3번 학생은 조별 발표를 준비하세요.  
4번 학생은 조별 발표를 준비하세요.  
5번 학생은 조별 발표를 준비하세요.  
나머지 학생은 집에 가세요
```



4. 실행 중단하기

(2) continue 문

- 이번 반복만 종료한 후 다음 반복으로 넘어감

4.3.2 continue.c

```
int main(void) {  
    for (int i = 1; i <= 30; i++) {  
        if (i >= 6 && i <= 10) {  
            if (i == 7) {  
                printf("%d번 학생은 결석입니다.\n", i);  
                continue;  
            }  
            printf("%d번 학생은 조별 발표를 준비하세요.\n", i);  
        }  
    }  
    return 0;  
}
```

실행결과

6 번 학생은 조별 발표를 준비하세요.
7 번 학생은 결석입니다.
8 번 학생은 조별 발표를 준비하세요.
9 번 학생은 조별 발표를 준비하세요.
10 번 학생은 조별 발표를 준비하세요.



5. 이중 반복문 사용하기

- ⊖ 이중 반복문(중첩 반복문) : for 문 안에 for 문을 겹치게 작성하는 것
- ⊗ 첫 번째 for 문 안에 두 번째 for 문을 추가

3.4 이중반복문.c

```
int main(void) {  
    for (int i = 1; i <= 3; i++) {  
        printf("첫 번째 반복문 : %d\n", i);  
        for (int j = 1; j <= 5; j++) {  
            printf("    두 번째 반복문 : %d\n", j);  
        }  
    }  
    return 0;  
}
```

빈칸 4개 넣기

실행결과

```
첫 번째 반복문 : 1  
    두 번째 반복문 : 1  
    두 번째 반복문 : 2  
    두 번째 반복문 : 3  
    두 번째 반복문 : 4  
    두 번째 반복문 : 5  
첫 번째 반복문 : 2  
    두 번째 반복문 : 1  
    두 번째 반복문 : 2  
    두 번째 반복문 : 3  
    두 번째 반복문 : 4  
    두 번째 반복문 : 5  
첫 번째 반복문 : 3  
    두 번째 반복문 : 1  
    두 번째 반복문 : 2  
    두 번째 반복문 : 3  
    두 번째 반복문 : 4  
    두 번째 반복문 : 5
```



5. 이중 반복문 사용하기

(1) 실습 1: 구구단 출력하기

⊖ for 문의 기본 형태를 작성

⊖ for 문 안 출력 문장 아래에 두 번째 for 문을 i 대신 j로 선언

⊗ 출력할 값 작성

3.4.1 구구단출력하기.c

```
int main(void) {  
    for (int i = 2; i <= 9; i++) { ----- ❶ 첫 번째 for 문  
        printf("%d단 출력\n", i);  
        for (int j = 1; j <= 9; j++) { ----- ❷ 두 번째 for 문  
            printf("%d × %d = %d\n", i, j, i * j); ❸ 구구단 출력  
        }  
    }  
    return 0;  
}
```

실행결과

2단 출력

2 × 1 = 2

2 × 2 = 4

2 × 3 = 6

(중략)

9 × 7 = 63

9 × 8 = 72

9 × 9 = 81



6. 프로젝트: 원하는 범위의 구구단 출력

scanf함수(vscode)나 scanf_s함수(visualStudio)를 통해, 원하는 구구단 단수의 범위를 입력받아 아래와 같이 구구단을 출력하시오

실행결과						—	□	×
구구단을 몇단부터 몇단까지 출력할지 최소구구단수와 최대구구단수						>>2 7		
2 x 1 = 2	3 x 1 = 3	4 x 1 = 4	5 x 1 = 5	6 x 1 = 6	7 x 1 = 7			
2 x 2 = 4	3 x 2 = 6	4 x 2 = 8	5 x 2 = 10	6 x 2 = 12	7 x 2 = 14			
2 x 3 = 6	3 x 3 = 9	4 x 3 = 12	5 x 3 = 15	6 x 3 = 18	7 x 3 = 21			
2 x 4 = 8	3 x 4 = 12	4 x 4 = 16	5 x 4 = 20	6 x 4 = 24	7 x 4 = 28			
2 x 5 = 10	3 x 5 = 15	4 x 5 = 20	5 x 5 = 25	6 x 5 = 30	7 x 5 = 35			
2 x 6 = 12	3 x 6 = 18	4 x 6 = 24	5 x 6 = 30	6 x 6 = 36	7 x 6 = 42			
2 x 7 = 14	3 x 7 = 21	4 x 7 = 28	5 x 7 = 35	6 x 7 = 42	7 x 7 = 49			
2 x 8 = 16	3 x 8 = 24	4 x 8 = 32	5 x 8 = 40	6 x 8 = 48	7 x 8 = 56			
2 x 9 = 18	3 x 9 = 27	4 x 9 = 36	5 x 9 = 45	6 x 9 = 54	7 x 9 = 63			