

C Programming

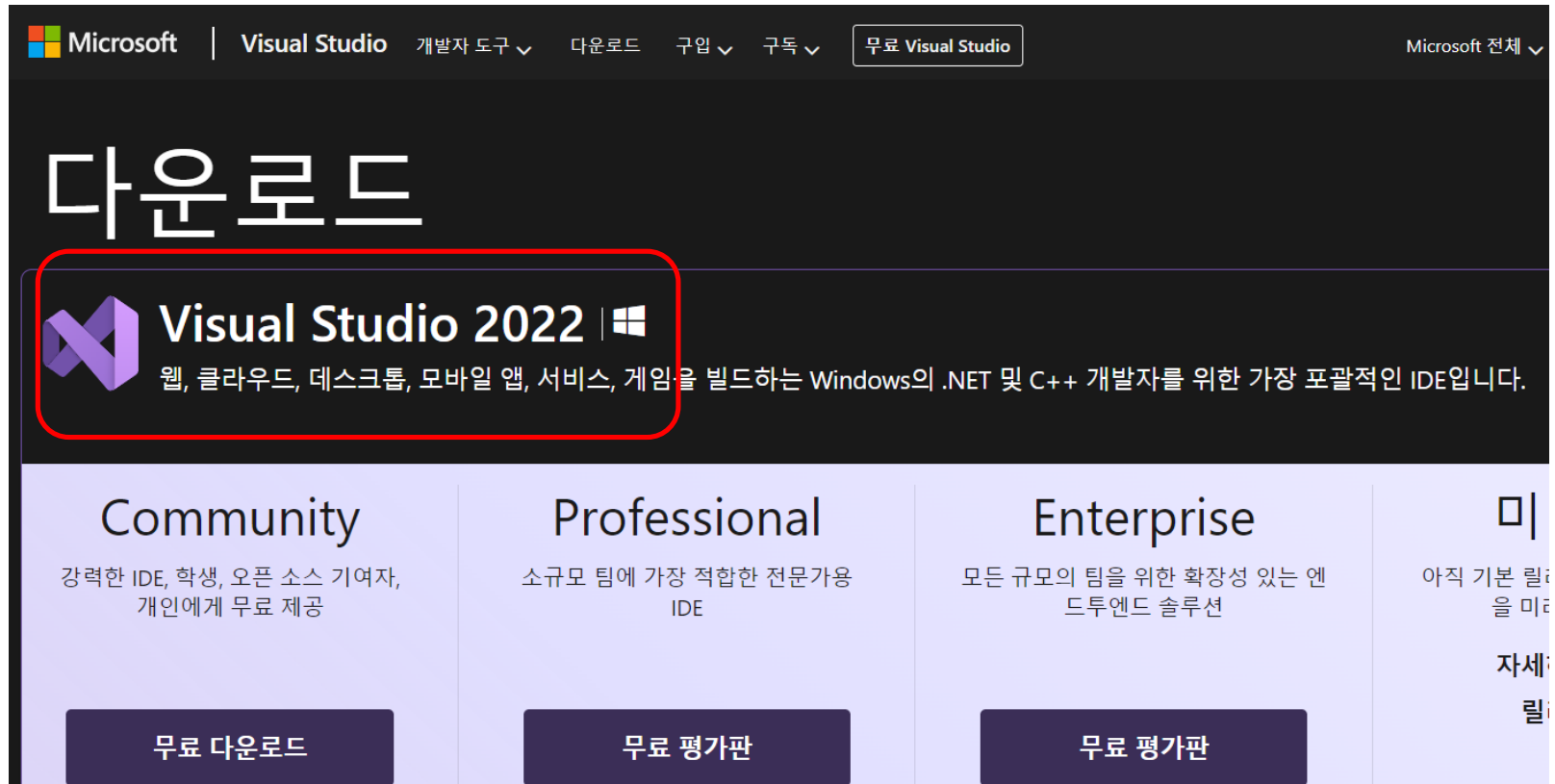
#2/3 ver 0.1

Yongseok Chi

Reference

1. Reference

- (1) Preprocessor, msdn.microsoft.com/en-us/library/d9x1s805.aspx
- (2) 비주얼 스튜디오 <https://visualstudio.microsoft.com/ko/downloads/>
- (3) 강의자료 **Visual Studio 2013** version
- (4) Editor : Notepad++ 설치 <https://notepad-plus-plus.org/downloads/>
- (5) code 비교 : beyond compare <https://www.scootersoftware.com/>
- (6) project 분석 : source insight <https://www.sourceinsight.com/>



[Program note]

- scanf, **scanf_s**

: 키보드 버퍼에 남은 데이터로 인해 오류 발생의 경우

→ **fflush(stdin);** 를 사용

예)

```
printf("첫번째 수를 입력하세요 : ");  
scanf("%d", &a);  
printf("계산할 연산자를 입력하세요 : ");  
scanf("%c", &ch); ●  
printf("두번째 수를 입력하세요 : ");  
scanf("%d", &b);
```

```
printf("첫번째 수를 입력하세요 : ");  
scanf("%d", &a);  
fflush(stdin);  
printf("계산할 연산자를 입력하세요 : ");  
scanf("%c", &ch);  
printf("두번째 수를 입력하세요 : ");  
scanf("%d", &b);
```

→ 실행 결과 아래와 같이 이상한
경우가 발생한다면,

```
첫번째 수를 입력하세요 : 55  
계산할 연산자를 입력하세요 :  
두번째 수를 입력하세요 :
```

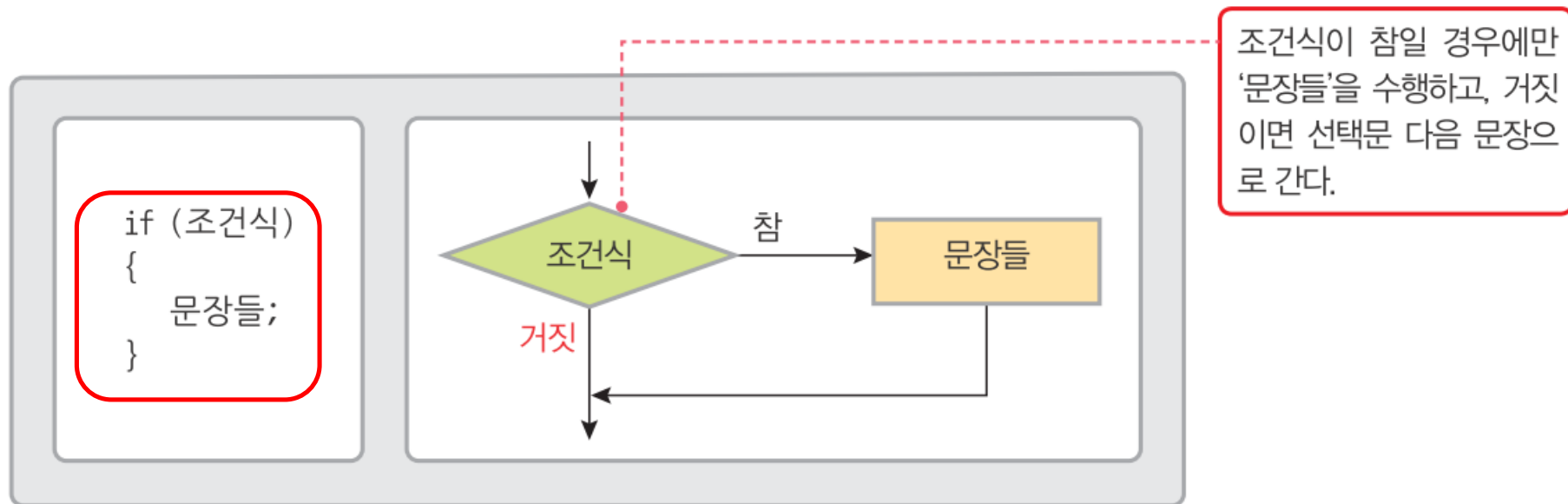
오류, **scanf("%c", &ch);** 실행 안 된다면,
fflush(stdin); 를 사용하여 버퍼 비우기



```
첫번째 수를 입력하세요 : 55  
계산할 연산자를 입력하세요 : /  
두번째 수를 입력하세요 : 11
```

6. 제어문

6.1 if 문 (단순 if 문)



x 값이 5와 10 사이인지?

```
if ((5 <= x) && (x <= 10))
    printf("5와 10 사이의 수입니다.\n");
```

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

```
int main()
{
```

```
    int x;
    scanf_s("%d", &x);
```

```
    if ((5 <= x) && (x <= 10))
        printf("5와 10 사이의 수입니다.\n");
```

```
    return 0;
```

```
}
```

```
7
5와 10 사이의 수입니다.
계속하려면 아무 키나 누르십시오
```

6. 제어문

6.1 if 문 (단순 if 문) ... 예제

현재의 온도를 입력받아 출력하시오. 단 입력 받은 온도가 0이하의 경우에는 영하라는 메시지를 표현하시오

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

```
int main()  
{
```

```
    int temperate;
```

```
    printf("현재 온도는? ");
```

```
    scanf_s("%d", & temperate);
```

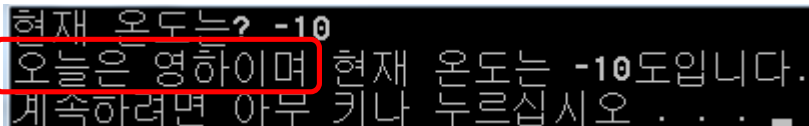
```
    if (temperate < 0)
```

```
        printf("오늘은 영하이며 ");
```

```
    printf("현재 온도는 %d도입니다.\n", temperate );
```

```
    return 0;
```

```
}
```



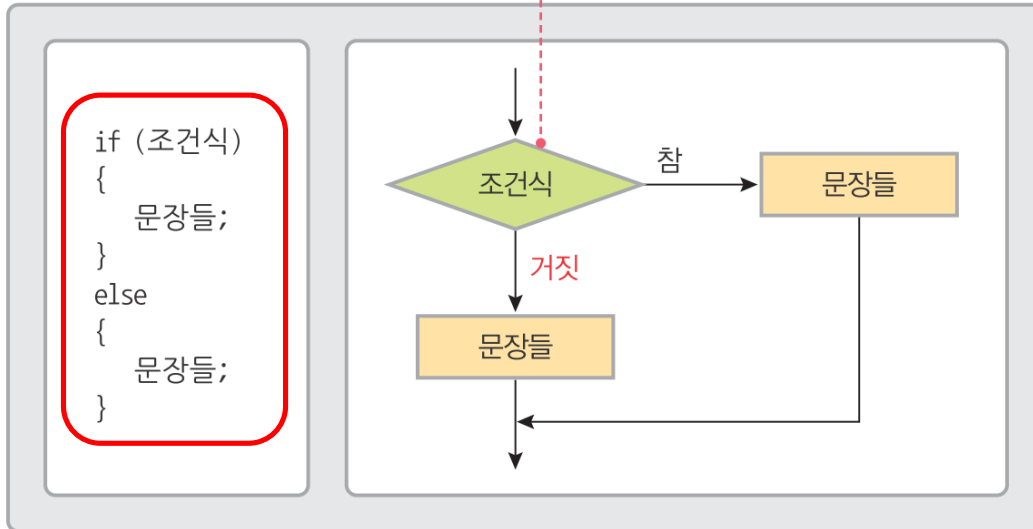
```
현재 온도는? -10  
오늘은 영하이며 현재 온도는 -10도입니다.  
계속하려면 아무 키나 누르십시오 . . .
```

```
현재 온도는? 10  
현재 온도는 10도입니다.  
계속하려면 아무 키나 누르십시오 . . .
```

6. 제어문

6.2 if 문 (if ~ else 문)

조건식의 결과가 참 또는 거짓인지에 따라 실행되는 '문장들'이 다르다.



70 이상이면 '합격!', 아니면 '불합격!' 출력

```
if (score >= 70)
    printf("합격!Wn");
else
    printf("불합격!Wn");
```



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

```
int main()
{
```

```
    int score ;
    scanf_s("%d", & score);
```

```
    { if (score >= 70) printf("합격!Wn");
      else printf("불합격!Wn");
    }
    return 0;
```

```
}
```



```
80
합격!
계속하려면 아무 키나 누르십시오
```

```
60
불합격!
계속하려면 아무 키나 누르십시오
```

6. 제어문

6.2 if 문 (if ~ else 문) ... 예제

사용자가 입력한 값에 따라 이 자리가 "홀수입니다." 또는 '짝수입니다.' 둘 중 하나가 되게.

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

```
int main()
{
    int num;

    printf("정수를 입력하십시오 : ");
    scanf_s("%d", &num);
    printf("입력한 수 %d는(은) ", num);

    if (num % 2 == 0)
        printf("짝수입니다.\n");
    else
        printf("홀수입니다.\n");

    return 0;
}
```

→ 디버그 - 디버깅하지 않고 시작

```
정수를 입력하십시오 : 4
입력한 수 4는(은) 짝수입니다.
계속하려면 아무 키나 누르십시오 . . .
```

```
정수를 입력하십시오 : 7
입력한 수 7는(은) 홀수입니다.
계속하려면 아무 키나 누르십시오 . . .
```

6. 제어문

6.2 if 문 (if ~ else 문) ... 주의

; 을 붙이지 않아야 한다.

```
if (age >= 20) ; ●
```

```
    adult++;
```

```
else ; ●
```

```
    child++;
```

바로 뒤에 ; 을 붙이면 컴파일러는 조건식이 참일 때
처리할 내용이 없는 빈 문장인 ; 으로 해석함
→ else 없는 if문이 끝난 것으로 해석되어
뒤의 else로 인해 에러가 됨

else 뒤에 조건식을 단독으로 사용 불가

```
if (age >= 20)
```

```
    adult++;
```

```
else (age < 20)
```

```
    child++;
```

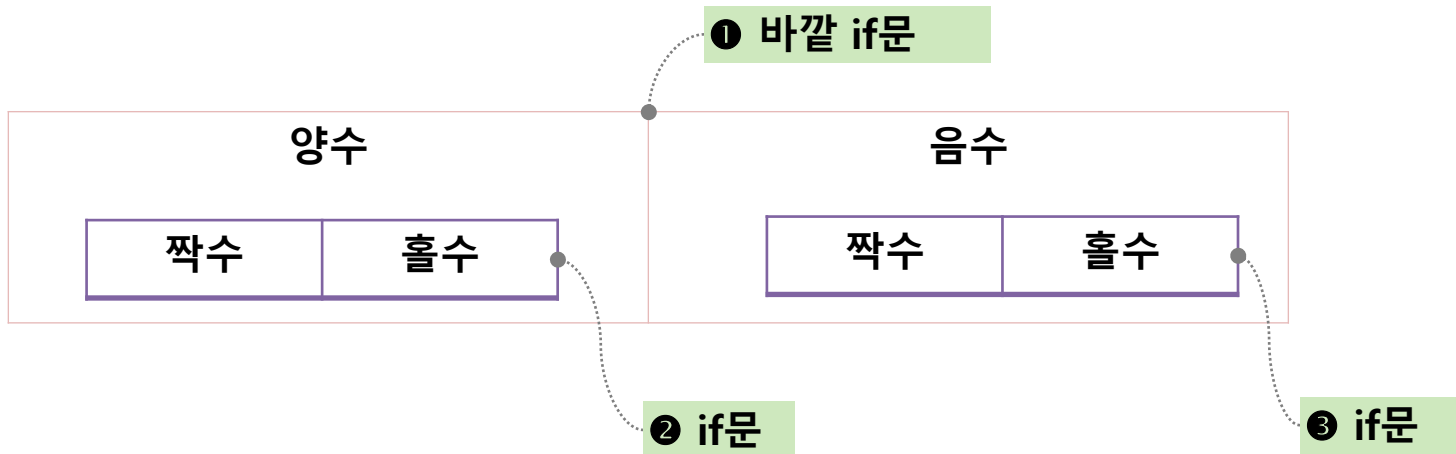

6. 제어문

6.3 if 문 (내포된 if ~ else 문)

: if문 안에 또 다른 if문을 사용

: 입력된 정수 num에 대해

➔ 양의 짝수, 양의 홀수, 음의 짝수, 음의 홀수 중 해당 사항을 출력하기



6. 제어문

6.3 if 문 (내포된 if ~ else 문)

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

```
int main()
```

```
{
```

```
    int num;
```

```
    printf("정수를 입력하시오 : ");
```

```
    scanf_s("%d", &num);
```

```
    printf("입력한 수 %d는(은) ", num);
```

```
    if (num >= 0) {
```

// num이 양수면

```
        if (num % 2 == 0) printf("양의 짝수입니다.\n");
```

```
        else printf("양의 홀수입니다.\n");
```

```
    }
```

```
    else {
```

// num이 음수면

```
        if (num % 2 == 0) printf("음의 짝수입니다.\n");
```

```
        else printf("음의 홀수입니다.\n");
```

```
    }
```

```
    return 0;
```

```
}
```

→ 디버그 - 디버깅하지 않고 시작

6. 제어문

6.3 if 문 (내포된 if ~ else 문)

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

```
int main()  
{
```

```
    int num;
```

```
    printf("정수를 입력하시오 : ");
```

```
    scanf_s("%d", &num);
```

```
    printf("입력한 수 %d는(은) ", num);
```

```
    if (num >= 0) {
```

```
        if (num % 2 == 0) printf("양의 짝수입니다.\n");
```

```
        else printf("양의 홀수입니다.\n");
```

```
    }
```

```
    else {
```

```
        if (num % 2 == 0) printf("음의 짝수입니다.\n");
```

```
        else printf("음의 홀수입니다.\n");
```

```
    }
```

```
    return 0;
```

```
}
```

C:\Windows\system32\cmd.exe

정수를 입력하시오 : 5
입력한 수 5는(은) 양의 홀수입니다.
계속하려면 아무 키나 누르십시오 . . .

C:\Windows\system32\cmd.exe

정수를 입력하시오 : 8
입력한 수 8는(은) 양의 짝수입니다.
계속하려면 아무 키나 누르십시오 . . .

C:\Windows\system32\cmd.exe

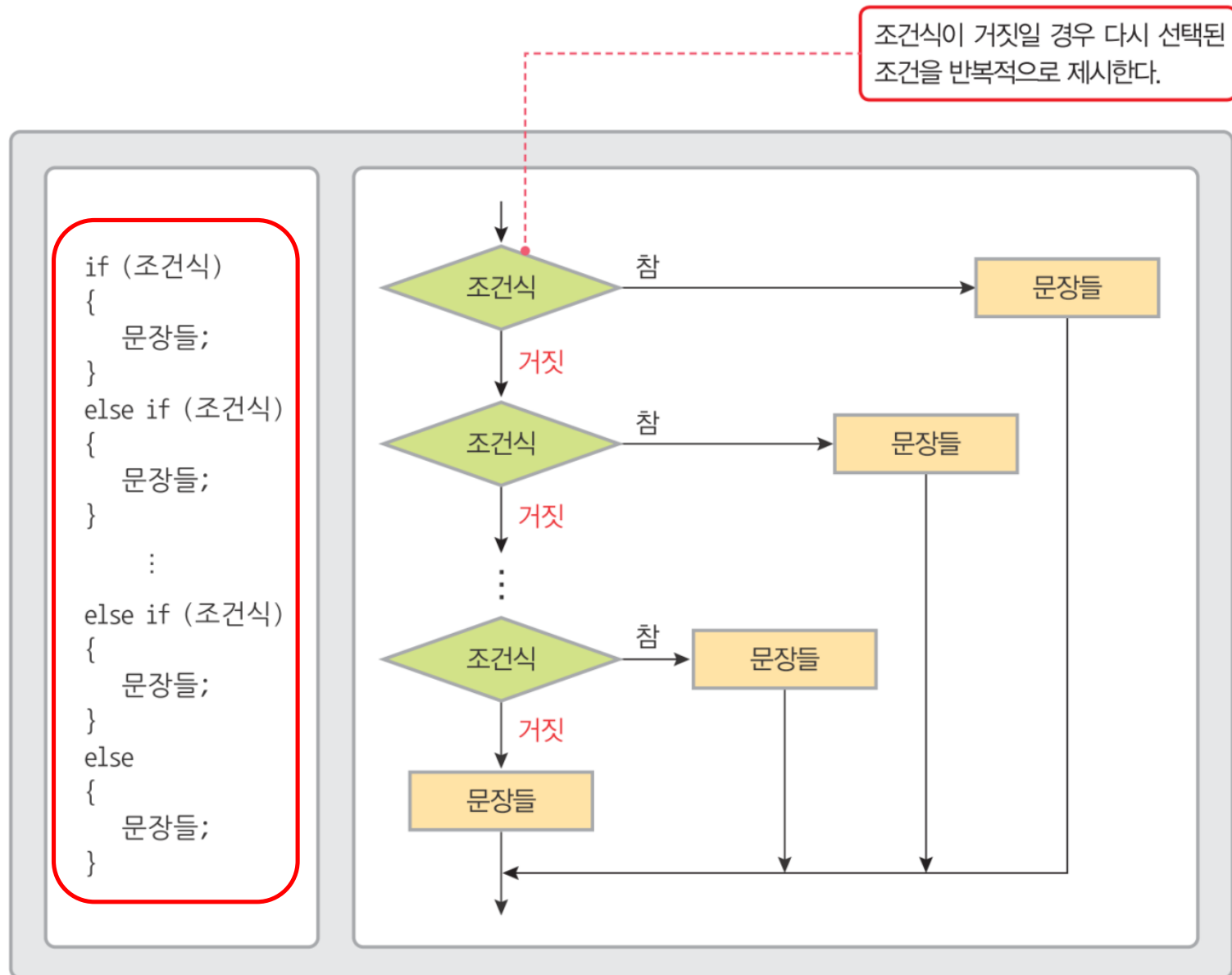
정수를 입력하시오 : -255
입력한 수 -255는(은) 음의 홀수입니다.
계속하려면 아무 키나 누르십시오 . . .

C:\Windows\system32\cmd.exe

정수를 입력하시오 : -368
입력한 수 -368는(은) 음의 짝수입니다.
계속하려면 아무 키나 누르십시오 . . .

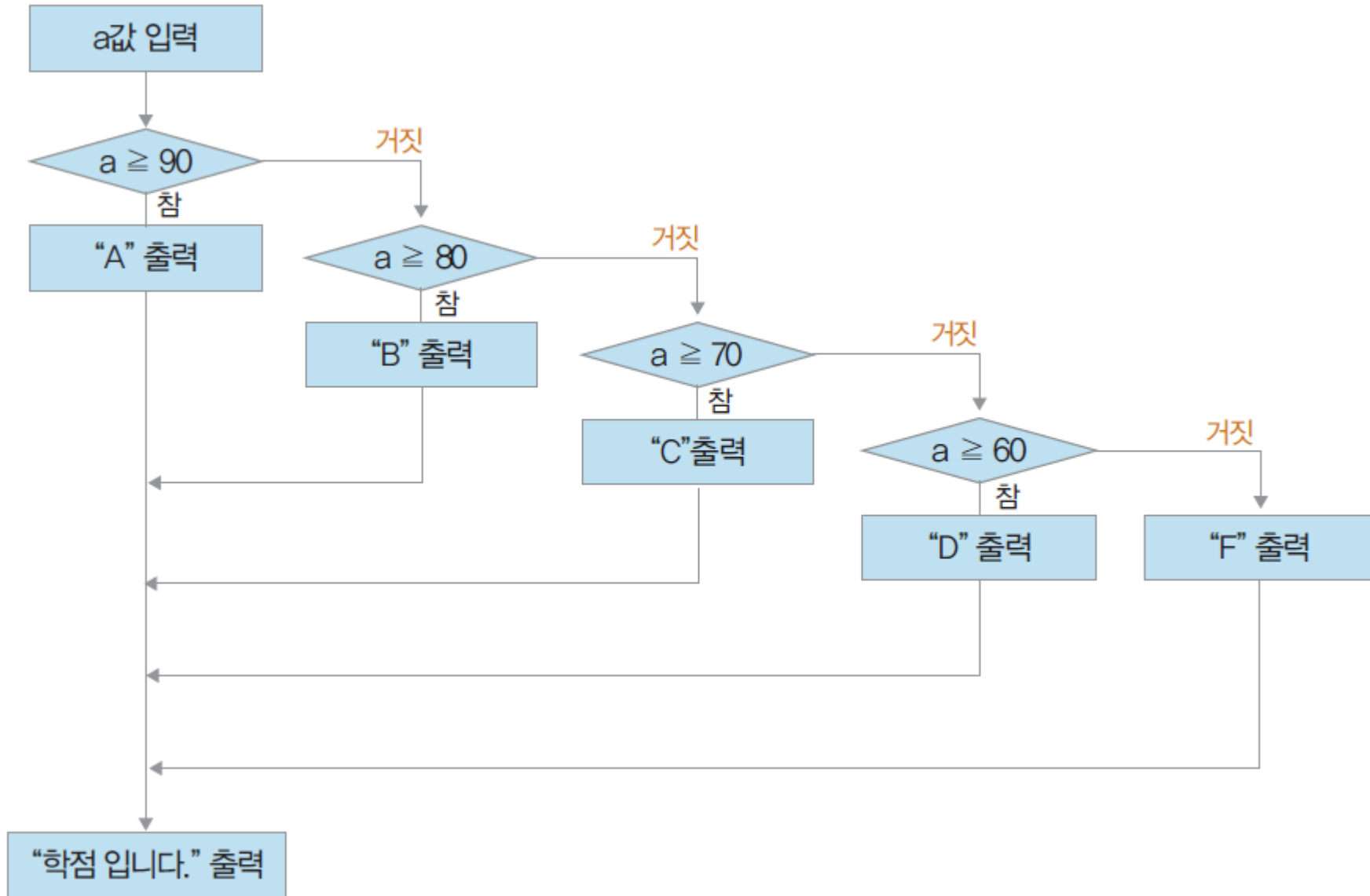
6. 제어문

6.4 if 문 (다중 if ~ else 문)



6. 제어문

6.4 if 문 (다중 if ~ else 문) 예제 프로그램



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

```
int main()
```

```
{
```

```
    int score;           // 점수를 저장할 변수
```

```
    char grade;          // 학점에 해당하는 문자 1개를 저장할 변수
```

```
    printf("점수를 입력하시오 : ");
```

```
    scanf_s("%d", &score);
```

```
    if (score >= 90)
```

```
        grade = 'A';
```

```
    else if (score >= 80)
```

```
        grade = 'B';
```

```
    else if (score >= 70)
```

```
        grade = 'C';
```

```
    else if (score >= 60)
```

```
        grade = 'D';
```

```
    else grade = 'F';
```

```
    printf("학점 : %c\n", grade);
```

```
    return 0;
```

```
}
```

→ 디버그 – 디버깅하지 않고 시작

```
점수를 입력하시오 : 99
학점 : A
계속하려면 아무 키나 누르십시오 . . .
```

6. 제어문

6.5 if 문 예제 (계산기 만들기)

: if문을 사용하기

: 계산할 첫번째 값 입력받기

: +, -, x, / 에 번호 붙여 출력하기

: 입력된 +, -, x, / 확인하기

: 계산할 두번째 값 입력받기

: 계산하여 출력하기

```
첫번째 계산할 값을 입력하세요 ==> 1000  
<1>덧셈 <2>뺄셈 <3>곱셈 <4>나눗셈 ==> 2  
두번째 계산할 값을 입력하세요 ==> 122  
1000 - 122 = 878
```

➔ 프로그램 하시오

➔ 디버그 - 디버깅하지 않고 시작

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

```
int main()
{
    int a, b;
    int result;
    int k;

    printf("첫번째 계산할 값을 입력하세요 == > ");
    scanf_s("%d", &a);
    printf("<1>덧셈 <2>뺄셈 <3>곱셈 <4>나눗셈 == > ");
    scanf_s("%d", &k);
    printf("두번째 계산할 값을 입력하세요 == > ");
    scanf_s("%d", &b);

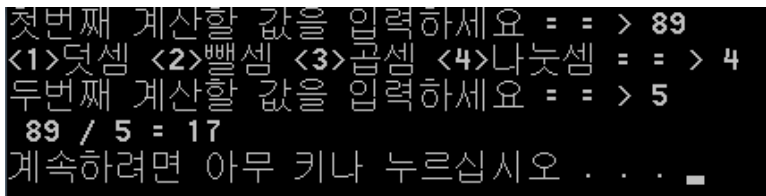
    if (k == 1) {
        result = a + b ;
        printf(" %d + %d = %d \n", a, b, result);
    }
```

```
    if (k == 2) {
        result = a - b;
        printf(" %d - %d = %d \n", a, b, result);
    }

    if (k == 3) {
        result = a * b ;
        printf(" %d * %d = %d \n", a, b, result);
    }

    if (k == 4) {
        result = a / b ;
        printf(" %d / %d = %d \n", a, b, result);
    }

    return 0;
}
```



정확한 소수점 표현하려면?


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

```
int main()  
{
```

```
    int a, b;
```

```
    int result;
```

```
    int k;
```

```
    printf("첫번째 수를 입력하세요 : ");
```

```
    scanf("%d", &a);
```

```
    fflush(stdin);
```

```
    printf("계산할 연산자를 입력하세요 : ");
```

```
    scanf("%c", &k);
```

```
    printf("두번째 수를 입력하세요 : ");
```

```
    scanf("%d", &b);
```

```
첫번째 수를 입력하세요 : 5  
계산할 연산자를 입력하세요 : /  
두번째 수를 입력하세요 : 1  
5 / 1 = 5.000000 입니다.  
계속하려면 아무 키나 누르십시오 . . .
```

```
        if (k == '+')
```

```
            printf("%d + %d = %d 입니다. \n", a, b, a + b);
```

```
        else if (k == '-')
```

```
            printf("%d - %d = %d 입니다. \n", a, b, a - b);
```

```
        else if (k == '*')
```

```
            printf("%d * %d = %d 입니다. \n", a, b, a*b);
```

```
        else if (k == '/')
```

```
            printf("%d / %d = %f 입니다. \n", a, b, a / (float)b);
```

```
        else if (k == '%')
```

```
            printf("%d %% %d = %d 입니다. \n", a, b, a%b);
```

```
        else printf("연산자를 잘못 입력했습니다. \n");
```

```
    return 0;
```

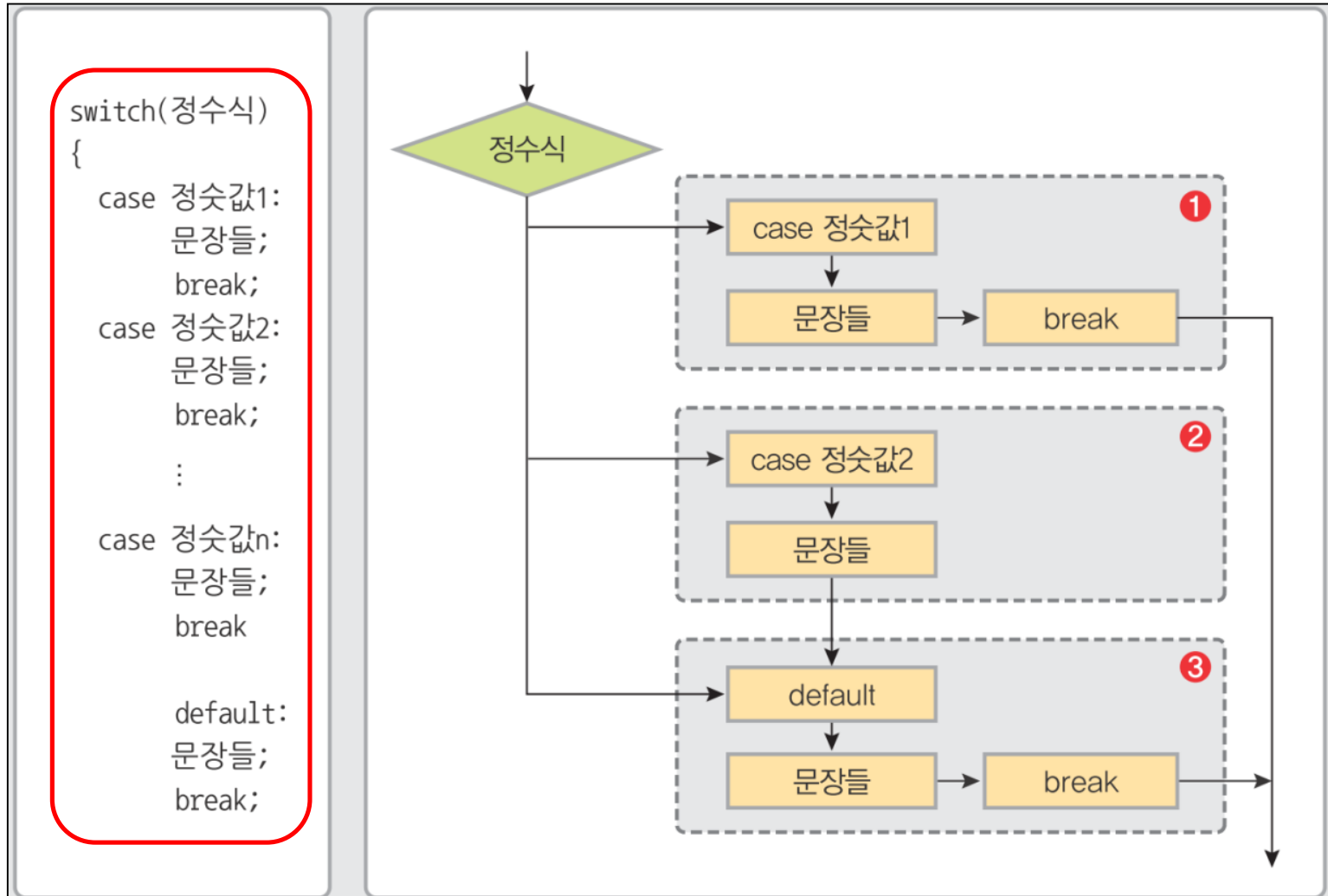
```
}
```

6. 제어문

6.6 switch문

: switch문은 정숫값을 갖는 정수식을 사용.

: 정수식이 갖는 값에 따라 서로 다른 처리를 요구하는 다중 처리에 유용



6. 제어문

6.6 switch문

: switch문은 정숫값을 갖는 정수식을 사용.

: 정수식이 갖는 값에 따라 서로 다른 처리를 요구하는 다중 처리에 유용

switch (식)

{

case 값 1: 문장 1; break;

case 값 2: 문장 2; break;

.

.

.

case 값 n: 문장 n; break;

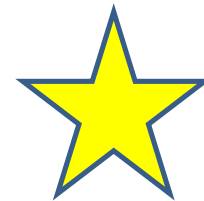
default: break;

}

결과값이 정수형 이여야 함
(정수형으로 변환 가능한 것도 가능)

식의 결과값과 일치하는 case 값이 없을 때 default 뒤의
내용을 실행한다.

생략 가능 → 그러나 여러분은 반드시 기재할 것



6. 제어문

6.6 switch문

: 예) 결혼 여부를 묻고 기혼자 또는 미혼자 수를 1 증가하기

```
switch (married)
```

```
{
```

```
case 1: printf("기혼자 %d\n");
```

```
    married++;
```

```
    break;
```

```
case 2: printf("미혼자 %d\n");
```

```
    single++;
```

```
    break;
```

```
default: break;
```

```
}
```

break가 없으므로
계속 아래(다음)로 실행하게 됨
→ 결국 1이거나 2이거나 의미



그러므로 반드시 반드시
break 사용할 것

6. 제어문

6.6 switch문 예제(1)

: 입력 받은 점수의 학점 출력하기 : switch문 이용

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

```
int main() {
```

```
    int score;
```

```
    char grade;
```

```
    printf("점수를 입력하시오. : ");
```

```
    scanf_s("%d", &score);
```

```
    switch (score / 10) {
```

```
        case 9: grade = 'A'; break;
```

```
        case 8: grade = 'B'; break;
```

```
        case 7: grade = 'C'; break;
```

```
        case 6: grade = 'D'; break;
```

```
        default: break;
```

```
    }
```

```
    printf("학점 : %c\n", grade);
```

```
점수를 입력하시오. : 90
학점 : A
계속하려면 아무 키나 누르십시오 . . .
```

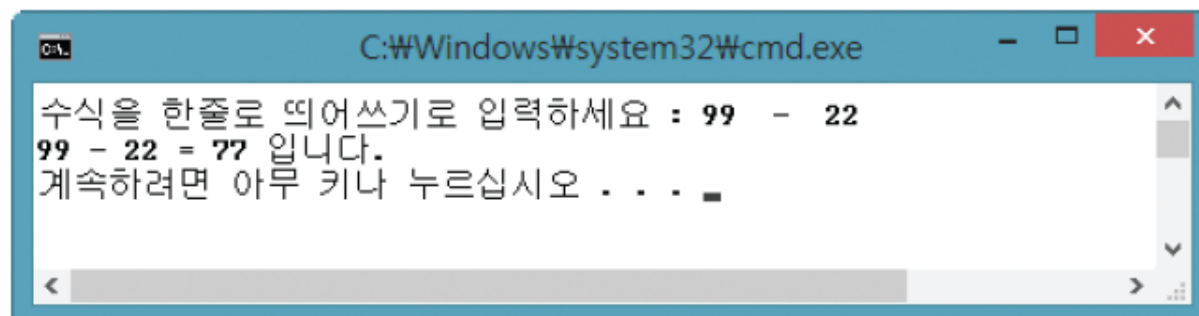
6. 제어문

6.6 switch문 예제(2)

: 계산기 만들기

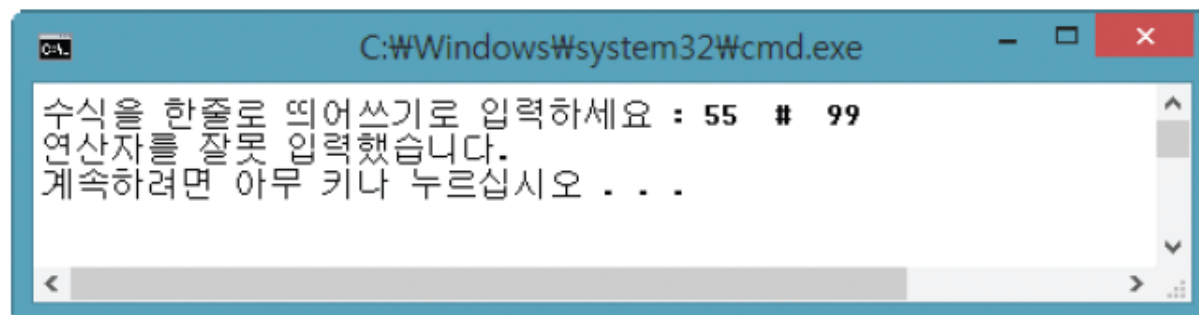
예제설명 수식을 띄어쓰기로 한 줄에 입력한 후 switch~case문을 활용하여 두 수의 +, -, *, /, % 연산을 수행하는 프로그램이다.

실행결과



```
C:\Windows\system32\cmd.exe

수식을 한줄로 띄어쓰기로 입력하세요 : 99 - 22
99 - 22 = 77 입니다.
계속하려면 아무 키나 누르십시오 . . .
```



```
C:\Windows\system32\cmd.exe

수식을 한줄로 띄어쓰기로 입력하세요 : 55 # 99
연산자를 잘못 입력했습니다.
계속하려면 아무 키나 누르십시오 . . .
```

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

```
int main()  
{
```

```
    int a, b;
```

```
    char ch;
```

```
    printf("수식을 한줄로 띄어쓰기로 입력하세요 : ");
```

```
    scanf_s("%d %c %d", &a, &ch, &b);
```

```
    switch(ch) {
```

```
        case '+' : printf("%d + %d = %d 입니다. \n", a, b, a+b);
```

```
            break;
```

```
        case '-' : printf("%d - %d = %d 입니다. \n", a, b, a-b);
```

```
            break;
```

```
        case '*' : printf("%d * %d = %d 입니다. \n", a, b, a*b);
```

```
            break;
```

```
        case '/' : printf("%d / %d = %d 입니다. \n", a, b, a/b);
```

```
            break;
```

```
        case '%' : printf("%d %% %d = %d 입니다. \n", a, b, a%b);
```

```
            break;
```

```
        default : break;
```

```
    } }
```

결과 어떤지요????

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

```
int main()
{
    int a, b;
    char ch;

    printf("수식을 한줄로 띄어쓰기로 입력하세요 : ");
    scanf_s("%d %c %d", &a, &ch, 1, &b);
    switch (ch) {
        case '+' : printf("%d + %d = %d 입니다. \n", a, b, a+b);
                    break;
        case '-' : printf("%d - %d = %d 입니다. \n", a, b, a-b);
                    break;
        case '*' : printf("%d * %d = %d 입니다. \n", a, b, a*b);
                    break;
        case '/' : printf("%d / %d = %d 입니다. \n", a, b, a/b);
                    break;
        case '%' : printf("%d %% %d = %d 입니다. \n", a, b, a%b);
                    break;
        default : break;
    } }
```

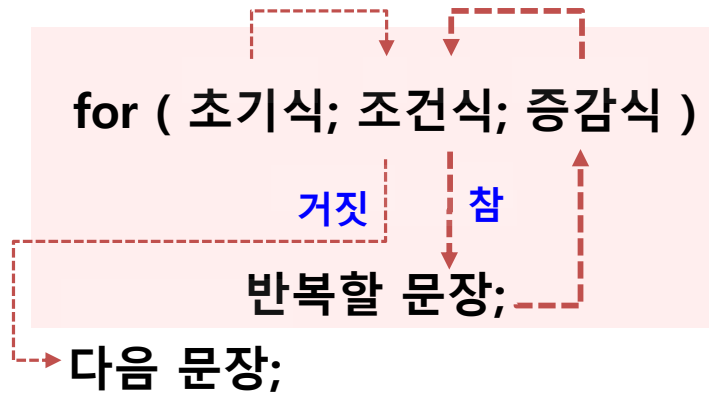
scanf_s 사용의 경우
: 문자의 개수를 알려야 함.
숫자 1 이라고...

```
수식을 한줄로 띄어쓰기로 입력하세요 : 5555 % 11
5555 % 11 = 0 입니다.
계속하려면 아무 키나 누르십시오 . . .
```


6. 제어문

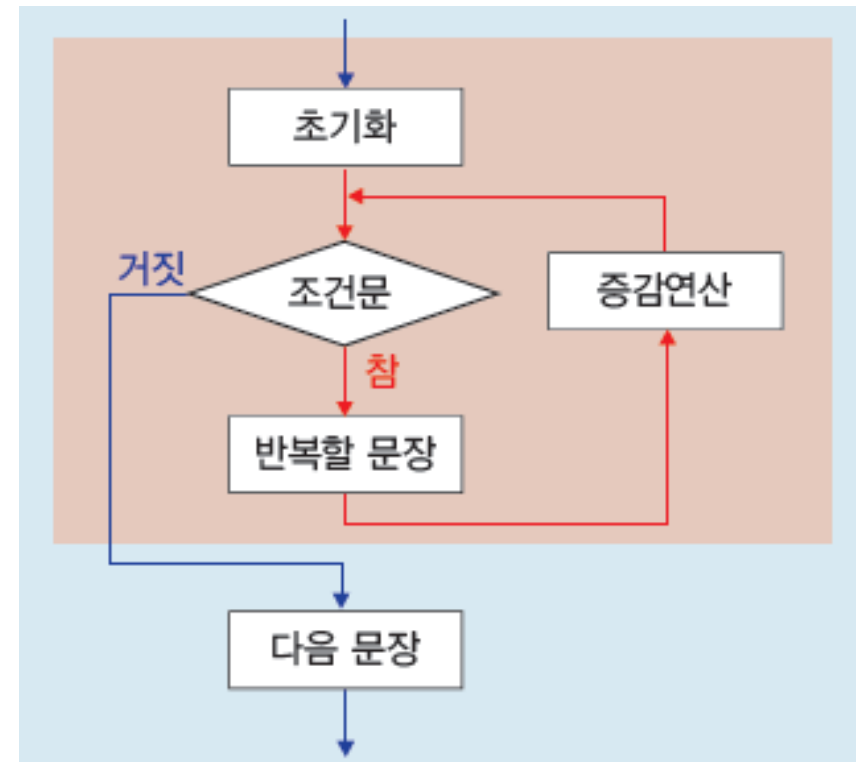
6.7 for문

: 동일한 내용을 특정 횟수만큼 반복 처리할 때 사용



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

다음 문장;



6. 제어문

6.7 for문

: for문 실행과정

```
for (i=1; i<=5; i++)  
    printf("1번 \n");
```

for문 실행 과정

<u>i</u> 값	<u>i</u> <=5	반복할 문장
1	참	printf("1번 \n");
2	참	printf("1번 \n");
3	참	printf("1번 \n");
4	참	printf("1번 \n");
5	참	printf("1번 \n");
6	거짓	for문을 끝냄

for (i=1; i<=5; i++)

'i를 1부터 5까지 1씩 증가시키며 반복'

6. 제어문

6.7 for문

: 1에서 5까지의 합 구하기

```
1 sum = 0;
```

```
2 sum += 1;
```

```
3 sum += 2;
```

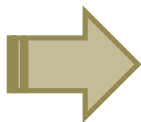
```
4 sum += 3;
```

```
5 sum += 4;
```

```
6 sum += 5;
```

```
7
```

```
8 printf("결과: %d", sum);
```



```
1 sum = 0 ;
```

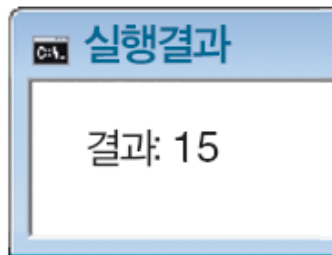
```
2 for (i=1; i<=5; i++)
```

```
3     sum += i;
```

```
4
```

```
5 printf("결과: %d", sum);
```

반복 동안 변하는
값은 제어 변수를
활용




6. 제어문

6.7 for문

: 주의해야 할 문법

```
1 for (i=1; i<=5; i++) ;  
2     printf("C Language \n");
```



```
for (i=1, i<=5, i++)
```



: 응용

```
for (sum=0, i=1; i<=100; i++)    // 1에서 100까지의 합 구하기  
    sum += i;
```

```
for (sum=0, i=0; i<=100; i+=3)  // 1과 100 사이에 있는 3의 배수 합 구하기  
    sum += i;
```

```
for (i=100; i>=1; i--)          // 100에서 1까지 거꾸로 출력하기  
    printf("%3d \n", i);
```

6. 제어문

6.7 for문

: 주의해야 할 문법

초기식, 조건식, 증감식은 모두 생략 가능하나 ;은 생략 불가능

`for (; ;)`

→ 무한 루프 → 이러한 표현은 지양할 것

→ while 문을 사용

// 1 + 3 + 5 + ... + n이 처음으로 1000을 넘는 n 찾기

```
sum = 0;
```

```
for (i=1; ; i=i+2)
```

```
{
```

```
    sum += i;
```

```
    if (sum > 1000)
```

```
        break;
```

```
}
```

```
printf("1부터 %d까지의 홀수의 합은 %d", i, sum);
```

1 다음에 3을 더해야 하므로 i 값을 2씩 증가시킨다.

현재 sum의 값이 1000보다 커지면 반복을 끝내야 한다.

break문을 만나 for문을 빠져 나온다.

6. 제어문

6.7 for문 ... 중첩

: for문 안에 또 다른 for문이 포함된 것

```
for (j=1; j<=3; j++)  
    printf("%d \n", j);
```

```
for (j=1; j<=3; j++)  
    printf("%d \n", j);
```

```
for (i=1; i<=2; i++) {  
    for (j=1; j<=3; j++)  
        printf("%d \n", j);  
}
```

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

```
int main()  
{  
    for (int i=1; i<=2; i++) {  
        printf("i=%d일 때 : ", i);  
        for (int j=1; j<=3; j++)  
            printf("j=%d ", j);  
        printf("\n");  
    }  
}
```

for 문에 변수 선언
또는
for 문 밖에 선언 **int i, j;**

➔ 결과 값을 미리 생각해 보기

6. 제어문

6.7 for문 ... 중첩

: for문 안에 또 다른 for문이 포함된 것

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

```
int main()
{
    for (int i=1; i<=2; i++) {
        printf("i=%d일 때 : ", i);

        for (int j=1; j<=3; j++)
            printf("j=%d ", j);

        printf("\n");
    }
}
```

```
i=1일 때 : j=1 j=2 j=3
i=2일 때 : j=1 j=2 j=3
계속하려면 아무 키나 누르십시오 . . .
```

6. 제어문

6.8 for문 ... 퀴즈 결과 출력하기

- (1) 입력 : 학생 10명의 퀴즈 점수
- (2) 전체 평균, 통과자수(70이상), 탈락자수 출력

#분석

```
for (i=1; i<=10; i++)
```

```
{
```

```
    // 학생 1명 마다 처리할 내용
```

```
    ① 퀴즈 점수 입력 받기
```

```
    ② 입력된 점수를 sum에 더하기
```

```
    ③ 점수가 70점 이상이면 통과자수를 1 증가, 아니면 탈락자수 1 증가
```

```
}
```

```
④ avg = sum / 10
```

➔ 프로그램 하시오

➔ 발표 준비


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

```
int main() {
```

```
    int i, quiz, sum, pass, fail;
```

```
    double avg;
```

```
    sum = 0; // 누적용 변수를 0으로 초기화
```

```
    pass = 0;
```

```
    fail = 0;
```

```
    /* 10명의 점수를 입력받아 */
```

```
    for (i=1; i<=10; i++) {
```

```
        printf("%d번의 퀴즈 점수는? ", i);
```

```
        scanf_s("%d", &quiz);
```

```
        sum += quiz;
```

```
        if (quiz >= 70) pass++;
```

```
        else fail++;
```

```
    }
```

```
    /* sum으로부터 평균 구하기 */
```

```
    avg = (double)sum / 10;
```

```
    printf("=====Wn");
```

```
    printf("평균:%.2lf점Wn", avg);
```

```
    printf("통과자:%2d명Wn", pass);
```

```
    printf("탈락자:%2d명Wn", fail);
```

```
    return 0;
```

```
}
```

// i번 학생의 퀴즈 점수 입력

// 점수를 sum에 누적하기

// 70점 이상이면 통과

// 70점 미만이면 탈락

➔ 디버그 - 디버깅하지 않고 시작

```
1번의 퀴즈 점수는? 70
2번의 퀴즈 점수는? 80
3번의 퀴즈 점수는? 50
4번의 퀴즈 점수는? 90
5번의 퀴즈 점수는? 100
6번의 퀴즈 점수는? 10
7번의 퀴즈 점수는? 50
8번의 퀴즈 점수는? 90
9번의 퀴즈 점수는? 70
10번의 퀴즈 점수는? 80
=====
평균: 69.00점
통과자: 7명
탈락자: 3명
계속하려면 아무 키나 누르십시오 . . .
```

6. 제어문

6.9 for문 ... 특정 단의 구구단 출력하기

(1) 입력 : 1~20 숫자를 입력 받아, 숫자에 해당되는 구구단 출력하기

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

```
int main()
```

```
{
```

```
    int i, n;
```

```
    printf("출력을 원하는 단은?");
```

```
    scanf_s("%d", &n);
```



```
    return 0;
```

```
}
```

→ 프로그램 하시오

6. 제어문

6.9 for문 ... 특정 단의 구구단 출력하기

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

```
int main()
```

```
{
```

```
    int i, n;
```

```
    printf("출력을 원하는 단은?");
```

```
    scanf_s("%d", &n);
```

```
    for (i = 1; i <= 20; i++)
```

```
        printf("%d * %d = %2d\n", n, i, n*i);
```

```
    return 0;
```

```
}
```

출력을 원하는 단은?19

19 * 1 = 19

19 * 2 = 38

19 * 3 = 57

19 * 4 = 76

19 * 5 = 95

19 * 6 = 114

19 * 7 = 133

19 * 8 = 152

19 * 9 = 171

19 * 10 = 190

19 * 11 = 209

19 * 12 = 228

19 * 13 = 247

19 * 14 = 266

19 * 15 = 285

19 * 16 = 304

19 * 17 = 323

19 * 18 = 342

19 * 19 = 361

19 * 20 = 380

계속하려면 아무 키나 누르십시오 .

6. 제어문

6.10 for문 ... 주사위의 경우의 수 구하기

(1) 2개의 주사위를 던졌을 때,

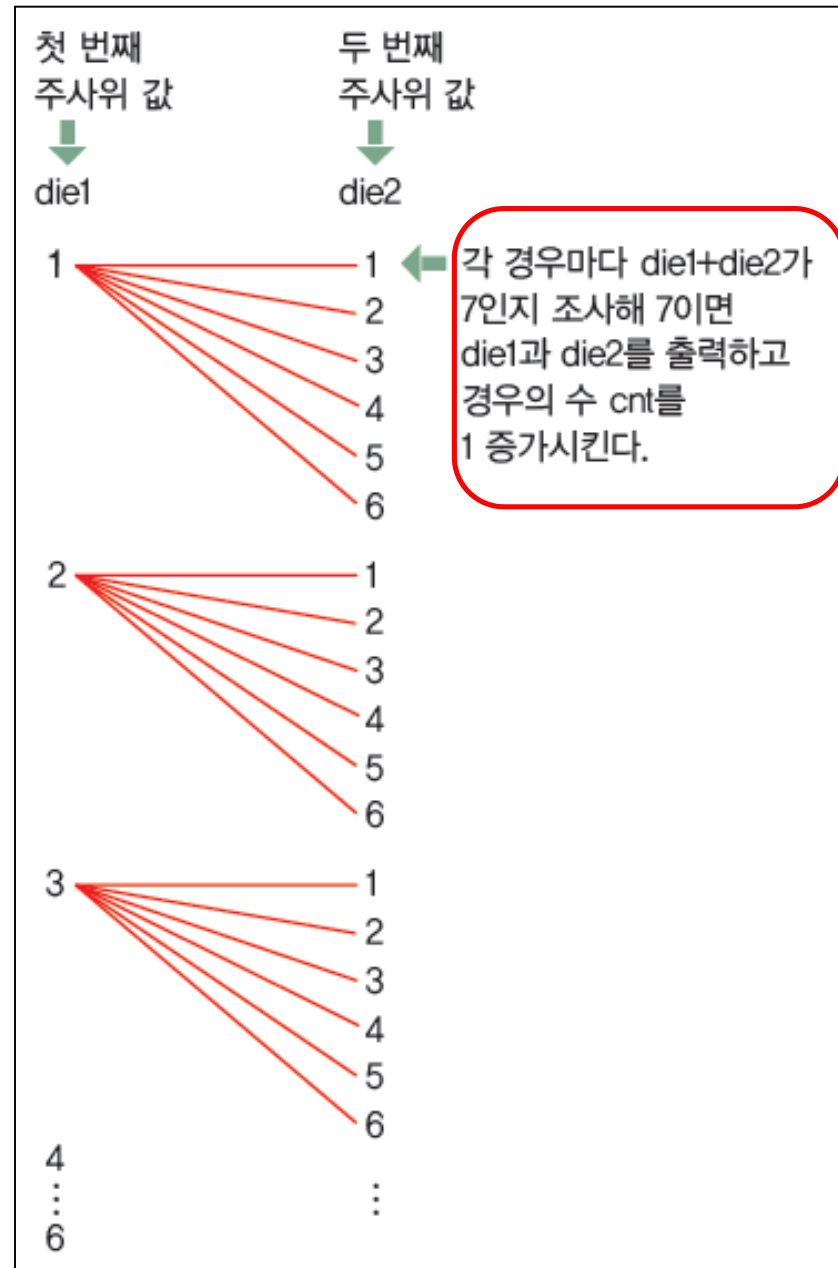
합이 7이 되는 경우를 출력하시오

→ 주사위의 가능한 값을 모두 조사하기 위해

주사위 하나 마다 for문을 사용

→ 프로그램 하시오

→ 프로그램후, 발표 준비



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

```
int main() {
    int dice1, dice2;
    int cnt=0;

    printf(" =====\n");
    printf(" 주사위1 주사위2 \n");
    printf(" =====\n");

    for (dice1 = 1; dice1 <= 6; dice1++)
    {
        for (dice2 = 1; dice2 <= 6; dice2++) {
            if (dice1 + dice2 == 7) {
                printf("Wt%d Wt%d \n", dice1, dice2);
                cnt++;
            }
        }
    }

    printf(" =====\n");
    printf("Wt총 %d가지\n", cnt);

    return 0;
}
```

=====
주사위1 주사위2
=====
1 6
2 5
3 4
4 3
5 2
6 1
=====
총 6가지
계속하려면 아무 키나 누르십시오

6. 제어문

6.11 for문 ... 중첩된 for 문으로 알파벳 출력하기

■ ASCII (16진수, char형만 이해)

10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자
0	0x00	NULL	22	0x16	STN	44	0x2C	,	66	0x42	B	88	0x58	X	110	0x6E	n
1	0x01	SOH	23	0x17	ETB	45	0x2D	-	67	0x43	C	89	0x59	Y	111	0x6F	o
2	0x02	STX	24	0x18	CAN	46	0x2E	.	68	0x44	D	90	0x5A	Z	112	0x70	p
3	0x03	ETX	25	0x19	EM	47	0x2F	/	69	0x45	E	91	0x5B	[113	0x71	q
4	0x04	EOT	26	0x1A	SUB	48	0x30	0	70	0x46	F	92	0x5C	\	114	0x72	r
5	0x05	ENQ	27	0x1B	ESC	49	0x31	1	71	0x47	G	93	0x5D]	115	0x73	s
6	0x06	ACK	28	0x1C	FS	50	0x32	2	72	0x48	H	94	0x5E	^	116	0x74	t
7	0x07	BEL	29	0x1D	GS	51	0x33	3	73	0x49	I	95	0x5F	_	117	0x75	u
8	0x08	BS	30	0x1E	RS	52	0x34	4	74	0x4A	J	96	0x60	`	118	0x76	v
9	0x09	HT	31	0x1F	US	53	0x35	5	75	0x4B	K	97	0x61	a	119	0x77	w
10	0x0A	₩n	32	0x20	SP	54	0x36	6	76	0x4C	L	98	0x62	b	120	0x78	x
11	0x0B	VT	33	0x21	!	55	0x37	7	77	0x4D	M	99	0x63	c	121	0x79	y
12	0x0C	FF	34	0x22	"	56	0x38	8	78	0x4E	N	100	0x64	d	122	0x7A	z
13	0x0D	₩r	35	0x23	#	57	0x39	9	79	0x4F	O	101	0x65	e	123	0x7B	{
14	0x0E	SO	36	0x24	\$	58	0x3A	:	80	0x50	P	102	0x66	f	124	0x7C	
15	0x0F	SI	37	0x25	%	59	0x3B	;	81	0x51	Q	103	0x67	g	125	0x7D	}
16	0x10	DLE	38	0x26	&	60	0x3C	<	82	0x52	R	104	0x68	h	126	0x7E	~
17	0x11	DC1	39	0x27	'	61	0x3D	=	83	0x53	S	105	0x69	i	127	0x7F	DEL
18	0x12	DC2	40	0x28	(62	0x3E	>	84	0x54	T	106	0x6A	j			
19	0x13	DC3	41	0x29)	63	0x3F	?	85	0x55	U	107	0x6B	k			
20	0x14	DC4	42	0x2A	*	64	0x40	@	86	0x56	V	108	0x6C	l			
21	0x15	NAK	43	0x2B	+	65	0x41	A	87	0x57	W	109	0x6D	m			

6. 제어문

6.11 for문 ... 중첩된 for 문으로 알파벳 출력하기

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

```
int main() {  
    char start, ch;  
  
    for (start = 'z'; start >= 'a'; start--)  
    {  
        // 각 행 시작 부분의 연속된 빈 칸 출력하기  
        for (ch = 'a'; ch < start; ch++)  
            printf(" ");  
  
        // 빈 칸 뒤에 알파벳을 start부터 'z'까지 출력하기  
        for (ch = start; ch <= 'z'; ch++)  
            printf("%c", ch);  
  
        printf("\n"); // 알파벳 출력 후 행 바꾸기  
    }  
  
    return 0;  
}
```

1칸

```
z  
yz  
xyz  
wxyz  
vwxyz  
uvwxyz  
tuvwxyz  
stuvwxyz  
rstuvwxyz  
qrstuvwxyz  
pqrstuvwxyz  
opqrstuvwxyz  
nopqrstuvwxyz  
mnopqrstuvwxyz  
lmnopqrstuvwxyz  
klmnopqrstuvwxyz  
jklmnopqrstuvwxyz  
ijklmnopqrstuvwxyz  
hijklmnopqrstuvwxyz  
ghijklmnopqrstuvwxyz  
fghijklmnopqrstuvwxyz  
efghijklmnopqrstuvwxyz  
defghijklmnopqrstuvwxyz  
cdefghijklmnopqrstuvwxyz  
bcdefghijklmnopqrstuvwxyz  
abcdefghijklmnopqrstuvwxyz  
계속하려면 아무 키나 누르십시오
```

6. 제어문

6.12 while 문

: 반복 횟수는 모르지만 어떤 조건을 만족하는 동안은 계속 반복할 때.

: 조건식이 참일 경우 반복

```
while ( 조건식 )
```

```
    반복할 문장;
```

```
다음 문장;
```

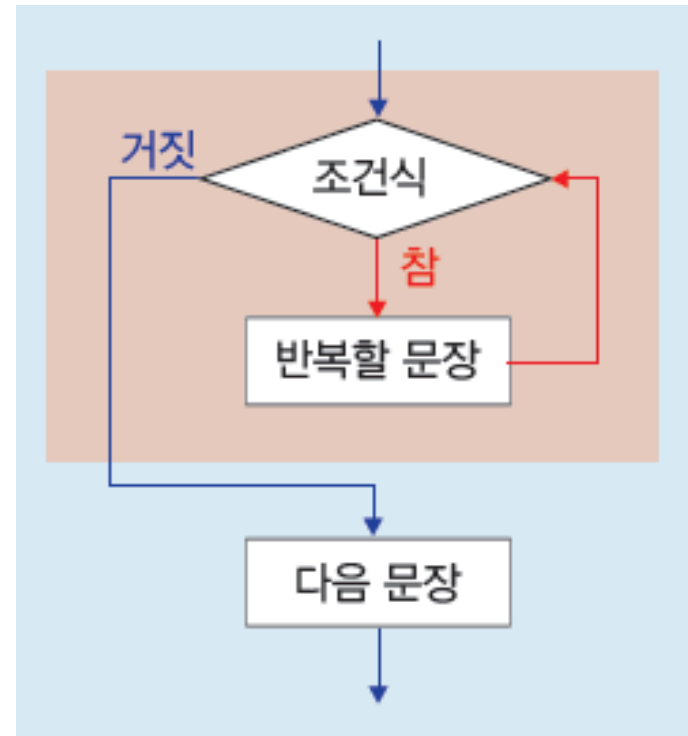
```
while ( 조건식 )
```

```
{
```

```
    반복할 여러 문장;
```

```
}
```

```
다음 문장;
```



6. 제어문

6.12 while 문

: 예제

```
sum = 0, i = 0;  
while (sum <= 100)  
{  
    i++;  
    sum = sum + i;  
}
```

참일 경우 계속 반복

즉, sum 값이 101되면 while 문에서 벗어남

: 예제

```
sum = 0, i = 1;  
while (1)  
{  
    sum = sum + i;  
    if (sum > 100)  
        break;  
    i++;  
}
```

값이 1 이므로 무한루프

무한루프에서 벗어날 수 있음

6. 제어문

6.12 do ~ while 문

: MISRA 규정에 의해 오류 발생 확률이 높아 사용하지 않음 (강의 생략)

- Stabilization of SW : MISRA-C (Motor Industry Software Reliability Association)

: MISRA

"Guidelines for the use of the C language in vehicle based software" 영국산업신뢰성협회

: MISRA-C :

2004는 현재 높은 소프트웨어 신뢰성이 필요한 항공, 자동차, 우주, 원전 사업 등에 사용.

- 총 141개의 룰이 있으며, 121개의 필수 룰과 20개의 권고 룰로 구성

- "Environment" 부터 "Run-time failure"까지 21개의 그룹으로 구성

- | | |
|--------------------------------|---------------------------------|
| • Environment | • Pointer type conversions |
| • Language extensions | • Expressions |
| • Documentation | • Control statement expressions |
| • Character sets | • Control flow |
| • Identifiers | • Switch statements |
| • Types | • Functions |
| • Constants | • Pointers and arrays |
| • Declarations and definitions | • Structures and unions |
| • Initialisation | • Preprocessing directives |
| • Arithmetic type conversions | • Standard libraries |
| | • Run-time failures |

6. 제어문

6.12 MISRA-C

: Environment Rule

Rule1.1 All code shall conform to **ISO 9899:1990 “Programming languages – C”**, amended and corrected by ISO/IEC 9899/COR1:1995, ISO/IEC 9899/AMD1:1995, and ISO/IEC 9899/COR2:1996. 모든 코드는 ISO/IEC 9899:1990, Programming language – C (ISO/IEC 9899/COR 1:1995, ISO/IEC 9899/AMD 1:1995 및 ISO/IEC 9899/COR 2:1996에 의해서 추가 및 개정된 것)를 따라야 한다.

Rule1.2 No reliance shall be placed on undefined or unspecified behavior.

미정의 또는 미규정 된 동작에 의존해서는 안 된다.

Rule1.3 Multiple compilers and/or languages shall only be used if there is a common defined interface standard for object code to which the languages/compilers/assemblers conform. 여러 개의 언어나 컴파일러를 사용할 경우, object code에 대한 공통적인 인터페이스가 있어야 한다.

Rule1.4 The compiler/linker shall be checked to ensure that 31 character significance and case sensitivity are supported for external identifiers.

컴파일러나 링커가 외부 식별자에 대하여 최대 31문자까지 지정이 가능하고, 대문자/소문자의 구분을 지원하고 있는지 검토되어야 한다.

Rule1.5 Floating-point implementations should comply with a defined floating-point standard. 부동소수점의 사용은 정의된 부동소수점 규격에 따라야 한다.

6. 제어문

6.12 MISRA-C

: Language extensions

Rule2.1 Assembly language shall be encapsulated and isolated.

어셈블리 언어는 캡슐화되어 격리되어야 한다.

Rule2.2 Source code shall only use `/* ... */` style comments.

소스 코드에는 `/*...*/` 스타일의 주석 표기법 만을 사용해야 한다.

Rule2.3 The character sequence `/*` shall not be used within a comment.

주석의 시작은 `/*`에서 `*/`로 끝나야 한다. 중간에 `/*`이 포함되어 있으면 안된다.

Rule2.4 Sections of code should not be “Commented out?”

코드의 일부를 코멘트아웃 해서는 안된다. 필요 시 조건부 컴파일(`#if...#endif`)를 사용한다.

6. 제어문

6.12 MISRA-C

S.No	MISRA Rule#	Description
1	MISRA rule 5	Only those characters and escape sequences which are defined in the ISO C standard shall be used.
2	MISRA rule 7	Trigraphs shall not be used.
3	MISRA rule 8	Multibyte characters and wide string literals shall not be used.
4	MISRA rule 9	Comments shall not be nested.
5	MISRA rule 10	Sections of code should not be 'commented out'.
6	MISRA rule 13	The basic types of char, int, long, float and double should not be used, but specific length equivalents should be typedefed for the specific compiler (and microprocessor).
7	MISRA rule 14	The type char shall always be declared as unsigned char or signed char.
8	MISRA rule 16	The underlying bit representations of floating point numbers shall not be used in any way by the programmer.
9	MISRA rule 17	typedef names shall not be reused.
10	MISRA rule 19	Octal constants (other than zero) shall not be used.
11	MISRA rule 20	All object and function identifiers shall be declared before use.
12	MISRA rule 21	Identifiers in an inner scope shall not use the same name as an identifier in an outer scope and therefore hide that identifier.
13	MISRA rule 22	Declarations of objects should be at function scope unless a wider scope is necessary.
14	MISRA rule 23	All declarations at file scope should be static where possible.
15	MISRA rule 25	An identifier with external linkage shall have exactly one external definition.
16	MISRA rule 26	If objects or functions are declared more than once they shall have compatible declarations.
17	MISRA rule 27	External objects should not be declared in more than one file.
18	MISRA rule 29	The use of a tag shall agree with its declaration.
19	MISRA rule 30	All automatic variables shall have been assigned a value before being used.
20	MISRA rule 31	Braces shall be used to indicate and match the structure in the non-zero initialisation of arrays and structures.
21	MISRA rule 32	In an enumerator list, the "=" construct shall not be used to explicitly initialise list members other than the first, unless all items are explicitly initialised.
22	MISRA rule 33	The right hand operand of a && or operator shall not contain side effects.
23	MISRA rule 34	The operands of a logical && or shall be primary expressions.
24	MISRA rule 35	Assignment operators shall not be used in expressions which return Boolean values.
25	MISRA rule 37	Bitwise operations shall not be performed on signed integer types.
26	MISRA rule 38	The right hand operand of a shift operator shall lie between zero and one less than the width in bits of the left hand operand (inclusive).
27	MISRA rule 39	The unary minus operator shall not be applied to an unsigned expression.
28	MISRA rule 40	The sizeof operator should not be used on expressions that contain side effects.
29	MISRA rule 43	Implicit conversions which may result in a loss of information shall not be used.
30	MISRA rule 45	Type casting from any type to or from pointers shall not be used.
31	MISRA rule 46	The value of an expression shall be the same under any order of evaluation that the standard permits.
32	MISRA rule 48	Mixed precision arithmetic should use explicit casting to generate the desired result.
33	MISRA rule 50	Floating point variables shall not be tested for exact equality or inequality.
34	MISRA rule 52	There shall be no unreachable code.
35	MISRA rule 56	The goto statement shall not be used.
36	MISRA rule 59	The statements forming the body of an if, else if, else, while, do... While or for statement shall always be enclosed in braces.
37	MISRA rule 61	Every non-empty case clause in a switch statement shall be terminated with a break statement.
38	MISRA rule 62	All switch statements should contain a final default clause.
39	MISRA rule 64	Every switch statement shall have at least one case.
40	MISRA rule 65	Floating point variables shall not be used as loop counters.
41	MISRA rule 68	Functions shall always be declared at file scope.
42	MISRA rule 69	Functions with variable numbers of arguments shall not be used.
43	MISRA rule 70	Functions shall not call themselves, either directly or indirectly.
44	MISRA rule 71	Functions shall always have prototype declarations and the prototype shall be visible at both the function definition and call.
45	MISRA rule 75	Every function shall have an explicit return type.
46	MISRA rule 76	Functions with no parameters shall be declared with parameter type void.
47	MISRA rule 78	The number of parameters passed to a function shall match the function prototype.
48	MISRA rule 79	The values returned by void functions shall not be used.
49	MISRA rule 80	Void expressions shall not be passed as function parameters.
50	MISRA rule 81	const qualification should be used on function parameters which are passed by reference, where it is intended that the function will not modify the parameter.
51	MISRA rule 83	For functions with non-void return type
52		i, there shall be one return statement for every exit branch (including the end of program)
53		ii, each return shall have an expression
54		iii, the return expression shall match the declared return type.
55	MISRA rule 84	For functions with void return type, return statements shall not have an expression.
56	MISRA rule 85	Functions called with no parameters should have empty parentheses.
57	MISRA rule 87	#include statements in a file shall only be preceded by other preprocessor directives or comments.
58	MISRA rule 88	Non-standard characters shall not occur in header file names in #include directives.
59	MISRA rule 89	The #include directive shall be followed by either a <filename> or "filename" sequence.
60	MISRA rule 91	Macros shall not be #defined and #undef'd within a block.
61	MISRA rule 94	A function-like macro shall not be 'called' without all of its arguments.
62	MISRA rule 95	Arguments to a function-like macro shall not contain tokens that look like preprocessing directives.
63	MISRA rule 96	In the definition of a function-like macro the whole definition, and each instance of a parameter, shall be enclosed in parentheses.
64	MISRA rule 98	There shall be at most one occurrence of the # or ## pre-processor operators in a single macro definition.
65	MISRA rule 99	All uses of the #pragma directive shall be documented and explained.
66	MISRA rule 100	The defined pre-processor operator shall only be used in one of the two standard forms.
67	MISRA rule 102	No more than 2 levels of pointer indirection should be used.
68	MISRA rule 103	Relational operators shall not be applied to pointer types except where both operands are of the same type and point to the same array, structure or union.
69	MISRA rule 104	Non-constant pointers to functions shall not be used.
70	MISRA rule 105	All the functions pointed to by a single pointer to function shall be identical in the number and type of parameters and the return type.
71	MISRA rule 106	The address of an object with automatic storage shall not be assigned to and object which may persist after the object has ceased to exist.
72	MISRA rule 107	The null pointer shall not be de-referenced.

Abbreviation	description	Criteria
N1	Total Number of operator	
N2	Total number of operands	
STMT	Number of statement	<= 50
VG	Cyclomatic Number (VG)	<= 20
AVGS	Average size of statement	<= 9
GOTO	Number of GOTO statement	0
RETU	Number of Return Statement	1
NBCALLING	Number of caller	<= 10
PATH	Number of path	<= 1000
PARA	Number of function parameter	<= 5
ap_cg_cycle	call Graph recursions	0
	dead code	0
DRCT_CALLS	Number of calls direct	<= 7
LEVL	Number of Level	<= 4



6. 제어문

6.13 프로그램 연습

- : 정수 n 을 입력받아,
 n 이하의 모든 홀수의 합과 짝수의 합을 출력 (**while문**)
- : 짝수 저장 변수 `even`, 홀수 저장 변수 `odd`, 최대값 n
 - **$(odd \leq n) \&\& (even \leq n)$**
 - `even = 2` 초기화, `odd = 1` 초기화
 - 프로그램 하시오
 - 프로그램 후, 발표 준비

```
정수 n을 입력하시오 : 555
정수 1에서 555이하 홀수들의 합은 77284이고 짝수들의 합은 77006입니다.
계속하려면 아무 키나 누르십시오 . . .
```

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

```
int main() {  
    int n, odd, even, odd_sum = 0, even_sum = 0;  
  
    printf("정수 n을 입력하시오 : ");  
    scanf_s("%d", &n);  
  
    odd = 1; even = 2;  
  
    while (odd <= n && even <= n)  
    {  
        odd_sum += odd;  
        even_sum += even;  
        odd += 2;  
        even += 2;  
    }  
  
    // n이 홀수라면 while문에서 마지막 odd 값(n)을 더하지 않고 while문을 빠져나오므로  
    // while문을 끝낸 직후 이를 체크하여 odd_sum에 마지막 odd를 더해야 함  
  
    if (odd == n) odd_sum += odd;  
  
    printf("정수 1에서 %d이하 홀수들의 합은 %d이고 ", n, odd_sum);  
    printf("짝수들의 합은 %d입니다.\n", even_sum);  
    return 0;  
}
```

→ 디버그 – 디버깅하지 않고 시작

```
정수 n을 입력하시오 : 555  
정수 1에서 555이하 홀수들의 합은 77284이고 짝수들의 합은 77006입니다.  
계속하려면 아무 키나 누르십시오 . . .
```

7. Preprocessor(전처리문)

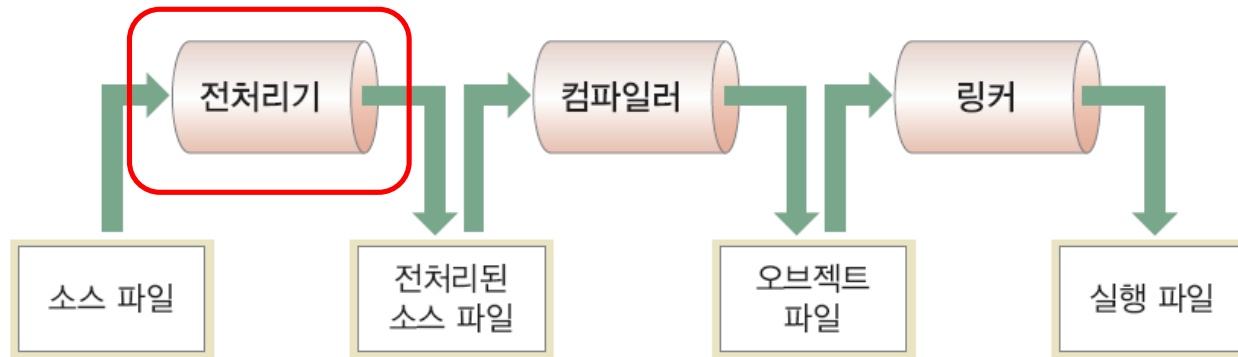
7.1 Preprocessor

◆ 전처리기(preprocessor)

: 소스(source) 프로그램을 오브젝트(object) 프로그램으로

컴파일하기 전에 수행되는 프로그램

: 소스 파일이 컴파일 될 수 있도록 준비하는 역할



◆ 전처리기 지시자

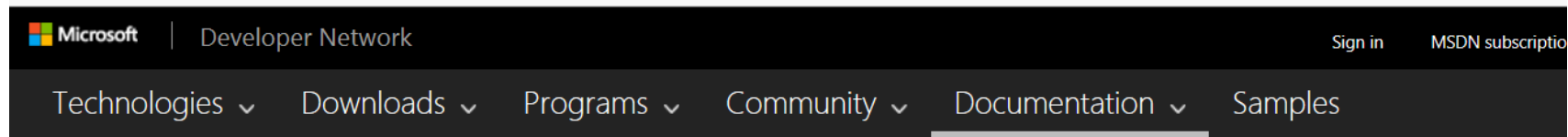
: 전처리기에게 내리는 지시자: # 으로 시작

: 끝에 ; 을 붙이지 않음

: 한 행에 한 지시자만 쓸 수 있음

7. Preprocessor(전처리문)

<https://msdn.microsoft.com/en-us/library/3sxhs2ty.aspx>



... > C/C++ Language and Standard Libraries > C/C++ Preprocessor Reference > Preprocessor ▾

Phases of Translation

▾ Preprocessor Directives

#define Directive

#error Directive

#if, #elif, #else, and #endif Directives

#ifdef and #ifndef Directives

▸ #import Directive

#include Directive

#line Directive

Null Directive

#undef Directive

#using Directive

▸ Preprocessor Operators

▸ Macros

Preprocessor Directives

Visual Studio 2015 | Other Versions ▾

Preprocessor directives, such as **#define** and **#ifdef**, are typically used to make source programs easy to change and easy to compile in different execution environments. Directives in the source file tell the preprocessor to perform specific actions. For example, the preprocessor can replace tokens in the text, insert the contents of other files into the source file, or suppress compilation of part of the file by removing sections of text. Preprocessor lines are recognized and carried out before macro expansion. Therefore, if a macro expands into something that looks like a preprocessor command, that command is not recognized by the preprocessor.

Preprocessor statements use the same character set as source file statements, with the exception that escape sequences are not supported. The character set used in preprocessor statements is the same as the [execution character set](#). The preprocessor also recognizes negative character values.

The preprocessor recognizes the following directives:

#define	#error	#import	#undef
#elif	#if	#include	#using
#else	#ifdef	#line	#endif
#ifndef	#pragma		

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ #define

: 지정한 기호 상수를 프로그래머가 정의한 치환 문자열로 대체

: 단순히 치환만 함

형식 매크로 상수
#define 매크로명 치환할 값

공백이 있으면 안됨
대문자 사용이 관례

예)

```
#define MAX 100  
#define NUM MAX-1
```

// MAX를 100으로 정의
// NUM을 MAX-1 → 100-1로 정의

예)

```
#define PI 3.1415  
area = PI * r * r;
```

예)

```
#define MSG "잘못된 입력입니다."  
if (i<0) printf(MSG);
```

예)

```
#define PRT printf  
PRT("Hello!");
```

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ #define

: 프로그램 가독성 증가, 프로그램 수정 용이

예) 100을 1000으로 변경하려면? 매크로 상수 사용시 매크로 정의만 수정하면 됨

```
// 매크로 사용 않은 경우  
for (i=1; i<=100; i++)  
:  
avg = sum / 100;
```



```
// 매크로 사용 경우  
#define MAX 100  
for (i=1; i<=MAX; i++)  
:  
avg = sum / MAX;
```

예) 정수 1(논리값 참에 해당) 보다는 TRUE라는 단어가 이해하기 쉬움

```
// 매크로 사용 않은 경우  
if (leapyear == 1)  
    printf("윤년");
```



```
// 매크로 사용 경우  
#define TRUE 1  
  
if (leapyear == TRUE)  
    printf("윤년");
```

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ #define

: 프로그램 가독성 증가

→ 비트 연산자 $\&$, $|$, \wedge , \sim 대신 AND, OR, XOR, NOT 을 사용하기

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

```
#define AND &
```

```
#define OR |
```

```
#define XOR ^
```

```
#define NOT ~
```

```
int main() {
```

```
    int x = 5, y = 6;
```

```
    printf("x & y = %d \n", x AND y);
```

```
    printf("x | y = %d \n", x OR y);
```

```
    printf("x ^ y = %d \n", x XOR y);
```

```
    printf("~x = %d \n", NOT x);
```

```
    return 0;
```

```
}
```

```
x & y = 4
x | y = 7
x ^ y = 3
~x = -6
```

계속하려면 아무 키나 누르십시오 .

→ 디버그 - 디버깅하지 않고 시작

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ #define

: 매크로 함수

형식 매크로 함수

#define 매크로 함수명(인수1, 인수2, ...) 치환할 내용

예

```
#define ADD(x, y) ((x) + (y))  
#define SQUARE(x) ((x) * (x))
```

연산자 우선순위 문제로
잘못된 결과가 발생할 수 있으므로
인수마다 ()로 묶어줘야 안전함

한 칸 띄어쓰기

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ #define

: 매크로 함수

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

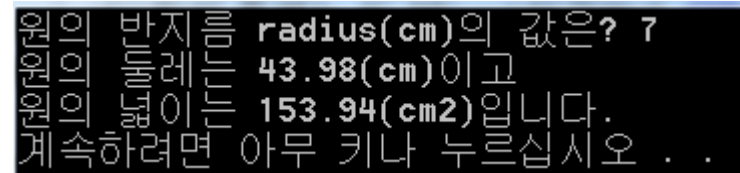
```
#define PI 3.141592
```

```
#define CIRCUM(r) (2 * PI * (r))
```

```
#define AREA(r) (PI * (r) * (r))
```

```
int main() {  
    int radius;  
  
    printf("원의 반지름 radius(cm)의 값은? ");  
    scanf_s("%d", &radius);  
    printf("원의 둘레는 %.2f(cm)이고\n", CIRCUM(radius) );  
    printf("원의 넓이는 %.2f(cm2)입니다.\n", AREA(radius) );  
  
    return 0;  
}
```

→ 디버그 – 디버깅하지 않고 시작



```
원의 반지름 radius(cm)의 값은? 7  
원의 둘레는 43.98(cm)이고  
원의 넓이는 153.94(cm2)입니다.  
계속하려면 아무 키나 누르십시오 . . .
```

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ #include

: 특정 파일을 현재 파일에 포함하기 위해 사용

```
#include <stdio.h>
#include <stdlib.h>
#include "C:\user\userlib.h"
#include "test.cpp"
```

라이브러리 헤더 파일

← 사용자 정의 헤더 파일









































































































































































































































← 사용자 정의 파일

: Visual studio에서 include 파일 위치

C:\Program Files\Microsoft Visual Studio\2022\Community\SDK\ScopeCppSDK\vc15\SDK\include\wucrt

➔ stdio.h 파일 열어보기

VCWinclde

 cliext	 cmath	 ctype.h	 io.h	 ppl.h	 string	 xfacet
 CodeAnalysis	 codecvt	 cwchar	 iomanip	 pplcancellation_token.h	 string.h	 xfunctional
 cvt	 collection.h	 cwctype	 ios	 pplconcrct.h	 stringstream	 xhash
 Manifest	 comdef.h	 delayhlp.cpp	 iosfwd	 pplinterface.h	 swprintf.inl	 xiosbase
 msclr	 comdefsp.h	 delayimp.h	 iostream	 ppltasks.h	 system_error	 xkeycheck.h
 sys	 comip.h	 deque	 iso646.h	 process.h	 tchar.h	 xlocale
 thr	 complex	 direct.h	 istream	 queue	 thread	 xlocbuf
 agents.h	 complex.h	 dos.h	 iterator	 random	 time.h	 xlocinfo
 agile.h	 comutil.h	 dvec.h	 ivec.h	 ratio	 time.inl	 xlocinfo.h
 algorithm	 concrt.h	 eh.h	 limits	 regex	 tmmintrin.h	 xlocmes
 allocators	 concrtm.h	 emmintrin.h	 limits.h	 rtapi.h	 tuple	 xlocmon
 ammintrin.h	 ConcurrencySal.h	 errno.h	 list	 safeint.h	 type_traits	 xlocnum
 amp.h	 concurrent_priority_queue.h	 exception	 listing.inc	 safeint_internal.h	 typeindex	 xloctime
 amp_graphics.h	 concurrent_queue.h	 excpt.h	 locale	 sal.h	 typeinfo	 xmemory
 amp_math.h	 concurrent_unordered_map.h	 fcntl.h	 locale.h	 scoped_allocator	 typeinfo.h	 xmemory0
 amp_short_vectors.h	 concurrent_unordered_set.h	 fenv.h	 malloc.h	 search.h	 unordered_map	 xmmmintrin.h
 amprt.h	 concurrent_vector.h	 filesystem	 map	 set	 unordered_set	 xrefwrap
 arm_neon.h	 condition_variable	 float.h	 math.h	 setjmp.h	 use_ansi.h	 xstddef
 armintr.h	 conio.h	 forward_list	 mbctype.h	 setjmpex.h	 utility	 xstring
 array	 crtdbg.h	 fpieee.h	 mbstring.h	 share.h	 vdefs.h	 xtgmath.h
 assert.h	 crtdefs.h	 fstream	 memory	 signal.h	 valarray	 xtr1common
 atomic	 crtversion.h	 functional	 memory.h	 smmintrin.h	 varargs.h	 xtree
 bitset	 crtwrn.h	 future	 minmax.h	 srv.h	 vcclr.h	 xutility
 cassert	 csetjmp	 fvec.h	 mm3dnow.h	 sstream	 vccorlib.h	 xxamp.h
 ccomplex	 csignal	 gcroot.h	 mmintrin.h	 stack	 vector	 xxamp_inl.h
 cctype	 cstdarg	 hash_map	 mutex	 stdarg.h	 vsqcapture.h	 xxatomic
 cerrno	 cstdbool	 hash_set	 new	 stdbool.h	 wchar.h	 ymath.h
 cfenv	 cstdddef	 immintrin.h	 new.h	 stddef.h	 wctype.h	 yvals.h
 cfloat	 cstddint	 initializer_list	 nmmintrin.h	 stdexcept	 wmmmintrin.h	
 chrono	 cstdio	 internal_concurrent_hash.h	 numeric	 stdexcept.h	 wtime.inl	
 cinttypes	 cstdlb	 internal_split_ordered_list.h	 omp.h	 stdint.h	 xatomic.h	
 ciso646	 cstring	 intrin.h	 ostream	 stdio.h	 xatomic0.h	
 climits	 ctgmth	 inttypes.h	 pgobootrun.h	 stdlib.h	 xcomplex	
 clocale	 ctime	 invkprxy.h	 pmmmintrin.h	 streambuf	 xdebug	

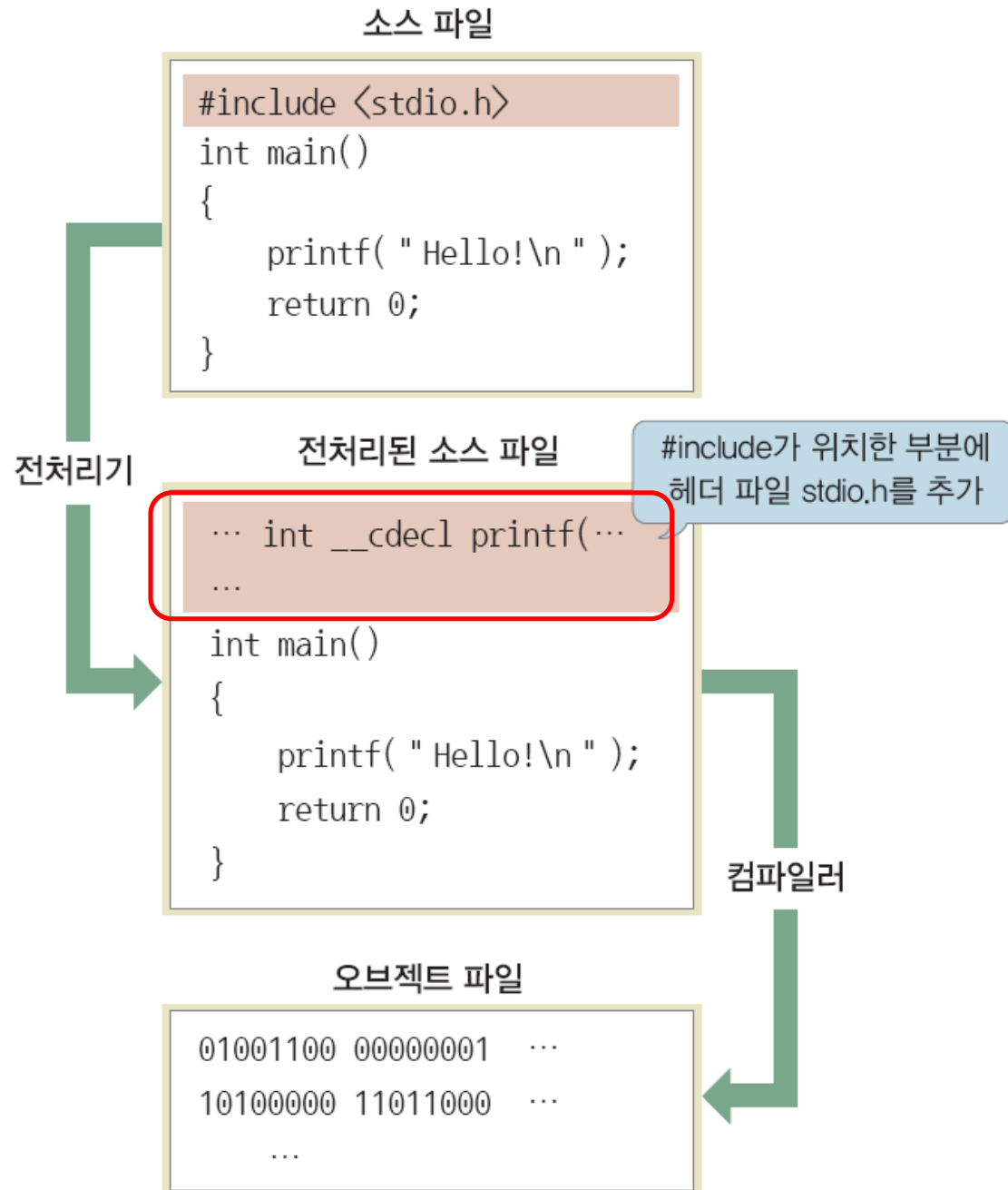
7. Preprocessor(전처리문)

7.1 Preprocessor

◆ **#include**

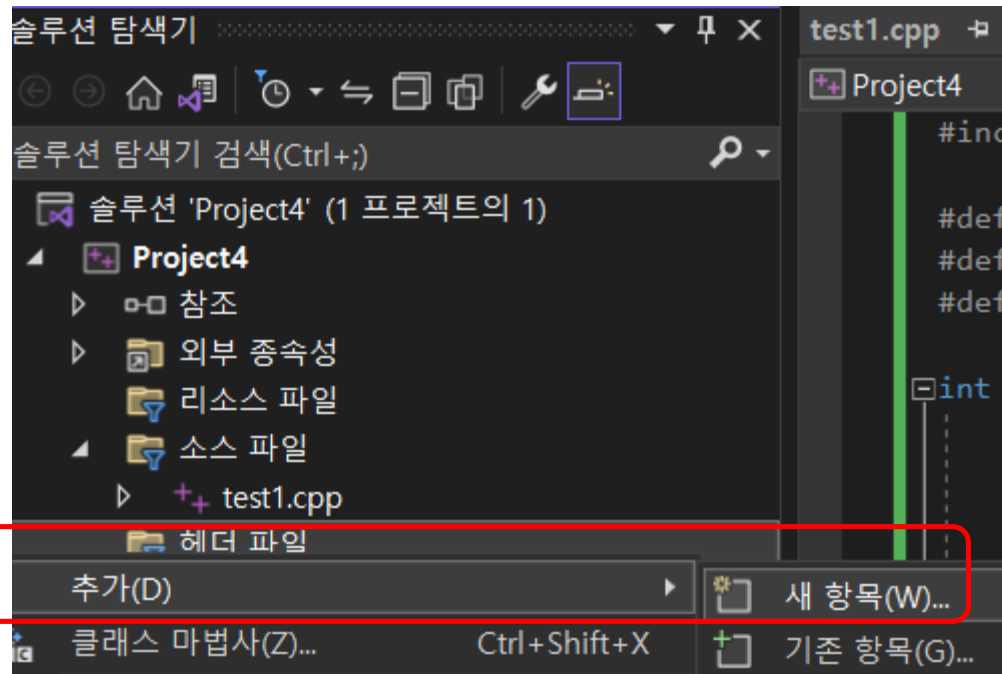
: 처리과정

#include <stdio.h>

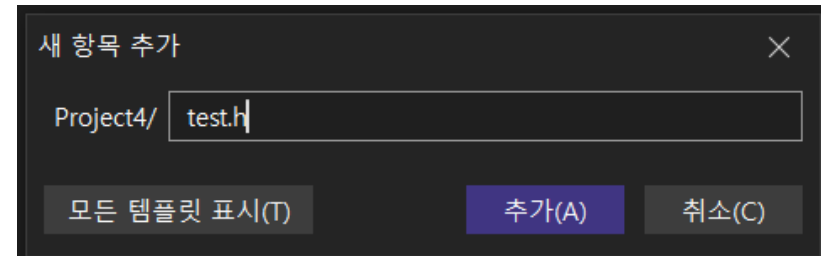


7. Preprocessor(전처리문)

◆ #include → header 파일 만들기

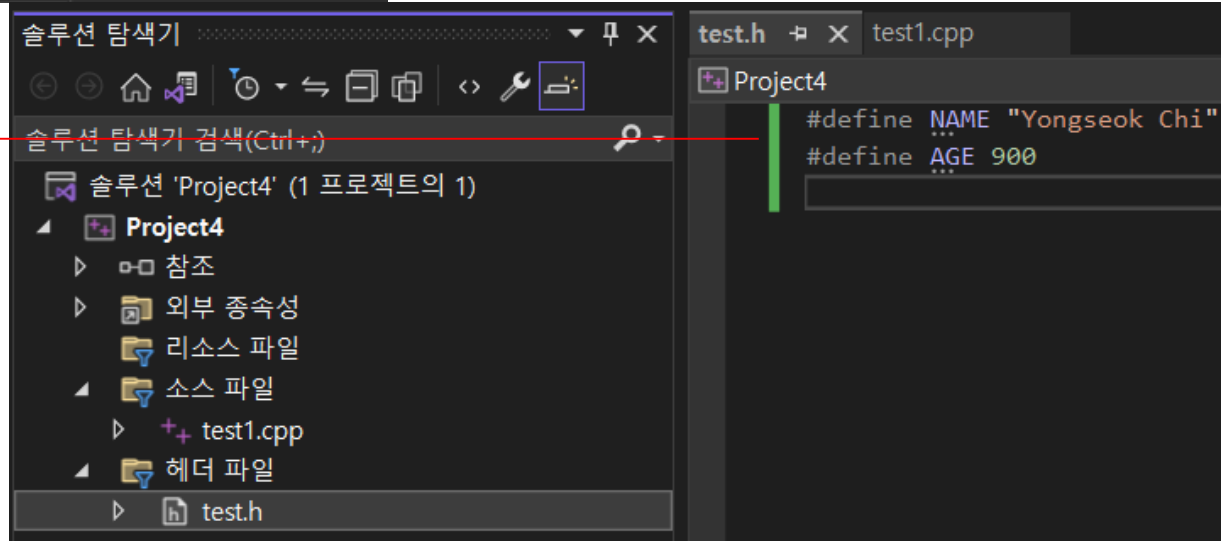


- (1) 새로운 프로젝트 만들기
- (2) 헤더파일에 아래와 같이 만들기
- (3) test.h 저장하기

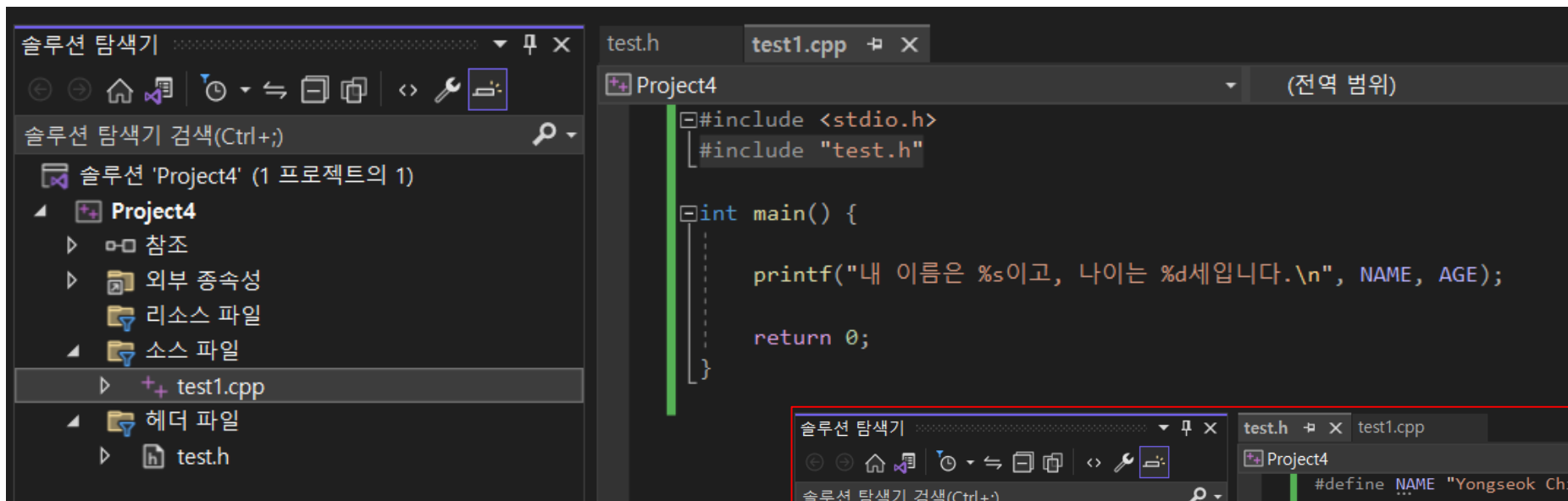


test.h 에 아래 define 문 선언하기

```
#define NAME "Yongseok Chi"
#define AGE 900
```



◆ #include → header 파일 만들기



C 파일에 아래 내용 만들기

#include <stdio.h>

#include "test.h"

int main() {

printf("내 이름은 %s이고, 나이는 %d세입니다.\\n", NAME, AGE);

return 0;

}

사용자 정의 header 파일은 " " 를 사용

내 이름은 Yongseok Chi이고, 나이는 900세입니다.
계속하려면 아무 키나 누르십시오 . . .

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ 조건부 Compiler 지시자

- 특정 조건을 만족할 때만 특정 코드를 프로그램에 삽입함
: 전처리를 마치면 필요한 코드만 삽입됨으로써,
선택된 코드만 컴파일 된다.

- #if
- #ifdef
- #ifndef
- #undef

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ 조건부 Compiler 지시자

- 조건식 결과값에 따라 특정 문장을 선택적으로 소스 파일에 삽입

형식 #if를 이용한 조건부 컴파일

```
#if 조건식
    문장 1;
#else
    문장 2;
#endif
```

조건식이 참 → '문장 1' 을 삽입
거짓 → '문장 2' 를 삽입

- 프로그램이 한국, 미국, 유럽의 세 가지 버전으로 존재 시
국가 별로 서로 다른 헤더 파일이 포함되게

```
#if (NATION == 1)
    #include "korea.h"
#elif (NATION == 2)
    #include "usa.h"
#else
    #include "europe.h"
#endif
```

또 다른 조건문을 검사하려면
#else와 #if를 결합한 #elif 문을 사용

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ 조건부 Compiler 지시자

형식 #ifdef를 이용한 조건부 컴파일

```
#ifdef 매크로명  
문장;
```

매크로명의 매크로가 정의되어 있다면 소스 코드에 포함할 내용

```
#else  
문장 2;
```

- 매크로가 정의되어 있지 않을 때 포함될 내용
- 필요 없다면 생략 가능
- #elif 는 사용 불가

```
#endif
```

예

```
#ifdef ADD  
    printf("%d", x + y);  
#else  
    printf("%d", x - y);  
#endif
```

형식 #undef를 이용한 매크로 정의 해제

```
#undef 매크로명
```

- 매크로 정의를 해제
- 이전에 정의된 매크로 정의를 무효화하고 새로 정의할 때 사용

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ 조건부 Compiler 지시자 : #ifdef, #undef

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

```
#define PI 3.141592
```

```
#define R 5
```

```
int main() {
```

```
    double area;
```

```
    #ifdef PI
```

```
        printf("PI = 3.141592Wn");
```

```
    #endif
```

```
    #undef R
```

// R의 정의 해제

```
    #define R 3
```

// R을 3으로 재정의

```
    area = PI * R * R;
```

```
    printf("Radius = %dWnArea = %.2fWn", R, area);
```

```
    return 0;
```

```
}
```

```
PI = 3.141592
```

```
Radius = 3
```

```
Area = 28.27
```

계속하려면 아무 키나 누르십시오

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ 조건부 Compiler 지시자

#ifndef

: “if not defined”의 약어

: 매크로가 정의되어 있지 않은 경우에만

#ifndef ~#endif 사이의 문장을 소스 파일에 삽입하여 컴파일 되게 함

7. Preprocessor(전처리문)

7.1 Preprocessor

◆ 실습

: test2.h 만들기 (내용은 아래와 같이)

```
#define SQUARE(x) ((x) * (x))
```

```
#define CUBE(x) ((x) * (x) * (x))
```

↓ 한 칸 띄어쓰기

: test2.c 만들어 실행하기

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

```
#include "test2.h"
```

```
int main() {
```

```
    int n;
```

```
    printf("입력 : ");
```

```
    scanf_s("%d", &n);
```

```
    printf("%d의 제곱 = %d, %d의 세제곱 = %d\n", n, SQUARE(n), n, CUBE(n));
```

```
    return 0;
```

```
}
```

→ 빌드 - 솔루션빌드

→ 디버그 - 디버깅하지 않고 시작

```
입력 : 9
9의 제곱 = 81, 9의 세제곱 = 729
계속하려면 아무 키나 누르십시오 .
```

7. Preprocessor(전처리문)

7.2 실습 **#undef**

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

```
#define SIZE 5
```

```
#define PRINT(i) for(i = 0; i < SIZE; i++) printf("*"); printf("\n");
```

```
int main()
```

```
{
```

```
    int i = 0;
```

```
    PRINT(i);
```

```
    #undef SIZE
```

```
    #define SIZE 10
```

```
    PRINT(i);
```

```
    #undef SIZE
```

```
    #define SIZE 15
```

```
    PRINT(i);
```

```
    return 0;
```

```
}
```

➔ 디버그 – 디버깅하지 않고 시작

7. Preprocessor(전처리문)

7.2 실습 #undef

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

```
#define SIZE 5
```

```
#define PRINT(i) for(i = 0; i < SIZE; i++) printf("*"); printf("\n");
```

```
int main()
```

```
{
```

```
    int i = 0;
```

```
    PRINT(i);           // * 5번 출력 그리고 new line
```

```
    #undef SIZE
```

```
    #define SIZE 10
```

```
    PRINT(i);           // * 10번 출력 그리고 new line
```

```
    #undef SIZE
```

```
    #define SIZE 15
```

```
    PRINT(i);           // * 15번 출력 그리고 new line
```

```
    return 0;
```

```
}
```

```
*****
```

```
*****
```

```
*****
```

```
계속하려면 아무 키나 누르십시오 . . .
```

7. Preprocessor(전처리문)

7.3 실습 #define

: 인생시계

: 평균 수명 80세를 24시간으로 판단하여, 나는 과연 24시간 중에 얼마나 흘러가고 있나?

→ 80세 : (24h x 60min) = 내 나이(입력) : ?min

?min = (24h x 60min) x 내 나이 / 80세

?min = 18min x 내 나이

```
printf("인생시계 : %d시 %d분\n", LIFEHOUR(age), LIFEMINUTE(age));
```

#define 에 LIFEHOUR(age), LIFEMINUTE(age) 적용하시오

→ 프로그램 하시오

→ 디버그 - 디버깅하지 않고 시작

7. Preprocessor(전처리문)

7.3 실습 #define

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

```
#define LIFEHOUR(age) ((age * 18) / 60)
```

// 인생시계의 "시"로 변환

```
#define LIFEMINUTE(age) ((age * 18) % 60)
```

// 인생시계의 "분"로 변환

```
int main()
```

```
{
```

```
    int age;
```

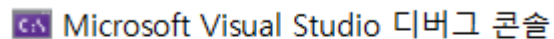
```
    printf("나이 : ");
```

```
    scanf_s("%d", &age);
```

```
    printf("인생시계 : %d시 %d분\n", LIFEHOUR(age), LIFEMINUTE(age));
```

```
    return 0;
```

```
}
```



Microsoft Visual Studio 디버그 콘솔

```
나이 : 27
인생시계 : 8시 6분
```

7. Preprocessor(전처리문)

7.4 실습

: 제곱미터 단위로 계산된 넓이를 평수로 변환하기

```
#include <stdio.h>

#define PYEONG 0.3025                // 1평 = 0.3025제곱미터

int main()
{
    float area;                      // 제곱미터 단위의 면적

    printf("넓이(제곱미터) : ");
    scanf_s("%f", &area);
    printf("넓이(평) : %.2lf\n", area * PYEONG);

    return 0;
}
```




→ 디버그 – 디버깅하지 않고 시작

```
넓이(제곱미터) : 150
넓이(평) : 45.38
계속하려면 아무 키나 누르십시오 .
```


7. Preprocessor(전처리문)

7.4 실습

: 단위 변환 프로그램은 만드시오

 단위  

통합검색 이미지 어학사전 ☐ 백과사전 ☐ 블로그 사이트 뉴스 웹문서 더보기

관련 [가스사용량 단위](#) [숫자 단위](#) [단위 종류](#) [단위환산](#) [여러가지 단위](#) [단위 표기](#) [수량 단위](#)
[mm 단위변환](#) [마이크로미터 단위](#) [금액 단위](#) [단위 표시](#) [규격](#) [땅넓이 단위](#) [갤론](#) [금액 단위 천문](#) 

단위변환

길이

넓이

무게

연비

부피

압력


온도


속도

데이터량

시간

을(를)





변환

1 밀리미터(mm)

0.000001 킬로미터(km)

0.001094 야드(yd)

0.00055 간(間)

5.399568e-7 해리(海里)

0.1 센티미터(cm)

0.03937 인치(inch)

6.213712e-7 마일(mile)

0.000009 정(町)

0.001 미터(m)

0.003281 피트(ft)

0.0033 자(尺)

0.000003 리(里)

8. 배열(array)

8.1 배열(array)

변수: 한 개의 자료 저장

배열: 여러 자료 저장 → 여러 변수의 모임

→ 형태가 같은 값 여러 개를 연속된 기억장소(주소)에 같은 이름(배열명)으로 저장

quiz[0]	quiz[1]	quiz[2]	quiz[3]	quiz[4]
8	9	10	8	7

→ 1차원, 2차원, 3차원 등의 다차원 배열

8. 배열(array)

8.2 1차원 배열(array)

```
int age[1000];           // 1000개 32bits(int) 값을 저장하는 배열
double average[100];     // 100개의 64bits(double) 값을 저장하는 배열
char name[10];           // 9개의 문자를 저장하는 배열 (마지막에 null문자 생략됨)
```

[사용방법]

```
int quiz[5] = {8, 9, 10, 8, 7}; →
```

↓ ↓ ↓ ↓ ↓
0 1 2 3 4 자리

quiz[0]	quiz[1]	quiz[2]	quiz[3]	quiz[4]
8	9	10	8	7

```
int sum[100] = {0};
```

→ 부족한 data는 자동으로 0으로 지정됨

```
int error[] = {0, 1, 0, 2, 3, 0, 1};
```

→ 배열 원소수를 명시 않으면 {} 안의 값 개수가 원소수로 결정됨

8. 배열(array)

8.2 1차원 배열(array)

[주의]

```
int a[3] = {21, 25, 20};
```

```
printf("%d", a[3]);
```

→ **a[3]**은 a 배열에 포함된 기억장소가 아니므로 프로그램 실행 결과를 예측할 수 없게 됨

→ 즉, 위에 배열은 **a[0], a[1], a[2]** 자리만 있음



8. 배열(array)

8.2 1차원 배열(array)

[사용 - 효율성]

첨자만 달라진다.

```
printf("%3d ", quiz[0]);  
printf("%3d ", quiz[1]);  
printf("%3d ", quiz[2]);  
printf("%3d ", quiz[3]);  
printf("%3d ", quiz[4]);
```



배열 원소 수를 활용

```
for (i=0; i<5; i++)  
    printf("%3d ", quiz[i]);
```

for의 제어변수 활용

8. 배열(array)

8.3 1차원 배열(array) 연습

입력: 5명의 퀴즈 점수

출력: 평균, 평균 미만 학생 수, 각 학생의 평균 점수와의 차이

1. 퀴즈 점수 5개를 1차원 배열 quiz에 입력하기 (for문)
2. 평균 avg 구하기
 - ① $sum \leftarrow quiz[0] + quiz[1] + quiz[2] + quiz[3] + quiz[4]$ (for문)
 - ② $avg \leftarrow sum/5$
3. 각 학생에 대해 ($avg > quiz[i]$)일 때마다 count를 1 증가
→ 평균 미만 수 count 구하기 (for문)
4. 각 학생에 대해 차이점수 ($quiz[i]-avg$) 출력하기 (for문)

8. 배열(array)

8.3 1차원 배열(array) 연습 (먼저, 배열부터 이해하기)

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

```
#define SIZE 5
```

```
int main()
```

```
{
```

```
    int quiz[SIZE];                // 정수 SIZE개 저장 배열 선언
```

```
    int i, count, sum;
```

```
    double avg;
```

```
    printf("%d명의 점수를 순서대로 입력하세요.\n\n", SIZE);
```

```
    for (i=0; i<SIZE; i++) {
```

```
        printf("%d번의 퀴즈 점수는? ", i + 1);
```

```
        scanf_s("%d", &quiz[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

5명의 점수를 순서대로 입력하세요.

1번의 퀴즈 점수는? 90

2번의 퀴즈 점수는? 70

3번의 퀴즈 점수는? 95

4번의 퀴즈 점수는? 65

5번의 퀴즈 점수는? 100

계속하려면 아무 키나 누르십시오 . . .

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

```
#define SIZE 5
```

```
int main()
```

```
{  
    int quiz[SIZE];  
    int i, count=0, sum=0;  
    double avg;
```

```
    printf("%d명의 점수를 순서대로 입력하세요.WnWn", SIZE);  
    for (i=0; i<SIZE; i++) {  
        printf("%d번의 퀴즈 점수는? ", i + 1);  
        scanf("%d", &quiz[i]);  
    }
```

```
    for (i=0; i<SIZE; i++) {  
        sum = sum + quiz[i];  
    }  
    avg = (double)sum / SIZE;
```

```
    for (i=0; i<SIZE; i++) {  
        if (quiz[i] < avg) count++;  
    }
```

```
    printf("=====\n");  
    printf("번호 점수 평균과의 차이 Wn");  
    printf("=====\n");
```

```
    for (i=0; i<SIZE; i++) {  
        printf("%2d %2d %5.1fWn", i+1, quiz[i], quiz[i] - avg);  
    }
```

```
    printf("=====\n");  
    printf("평균 : %.1f점 Wn", avg);  
    printf("=====\n");  
    printf("평균미만 : %d명 Wn", count);  
  
    return 0;  
}
```

5명의 점수를 순서대로 입력하세요.

1번의 퀴즈 점수는? 100
2번의 퀴즈 점수는? 90
3번의 퀴즈 점수는? 60
4번의 퀴즈 점수는? 77
5번의 퀴즈 점수는? 65

=====
번호 점수 평균과의 차이
=====

1	100	21.6
2	90	11.6
3	60	-18.4
4	77	-1.4
5	65	-13.4

=====
평균 : 78.4점

=====

평균미만 : 3명
계속하려면 아무 키나 누르십시오 . . .

8. 배열(array)

8.3 1차원 배열(array) 연습

입력: 25명의 퀴즈 점수

출력: 평균, 평균 미만 학생 수, 각 학생의 평균 점수와의 차이

앞 예제를 바탕으로 프로그램해보기 “define문의 중요성”

8. 배열(array)

8.4 1차원 배열(array) 연습2(최소값)

다섯 물질의 어는 점 5개 중 가장 낮은 어는 점 구하기 (최소값 구하기)

최솟값을 min에 구하기

f[0]	3
f[1]	0
f[2]	-30
f[3]	-20
f[4]	-1

① f[0]을 min의 초깃값으로 지정

② f[1]이 현재 min보다 작다면 min을 f[1]로 변경

③ f[2]가 현재 min보다 작다면 min을 f[2]로 변경

④ f[3]이 현재 min보다 작다면 min을 f[3]으로 변경

⑤ f[4]가 현재 min보다 작다면 min을 f[4]로 변경

처음 min은 3이지만,...

3

1. min = 배열의 첫 원소로 정한 후,
2. 나머지 원소와 min을 **비교해** 더 작은 값을 min으로 저장하기 (**for문**)
if (min > 배열 원소)
 min = 배열 원소;


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

```
#define N 5
```

```
int main()
```

```
{
```

```
    int f[N] = {3, 0, -30, -20, -1};           // 배열을 선언  
    int i, min;
```

```
    min = f[0];                                // 최소값 초기화
```

```
    for (i=1; i<N; i++) {                      // 최소값 찾기
```

```
        if (f[i] < min) min = f[i];
```

```
    }
```

```
    printf("어는 점 목록:");                  // 배열 내용 출력
```

```
    for (i=0; i<N; i++) {
```

```
        printf("%4d", f[i]);
```

```
    }
```

```
    printf("\n가장 낮은 어는 점: %d\n", min); // 최소값 출력
```

```
    return 0;
```

```
}
```

→ 디버그 – 디버깅하지 않고 시작

```
어는 점 목록:   3   0 -30 -20  -1  
가장 낮은 어는 점: -30  
계속하려면 아무 키나 누르십시오 . . .
```

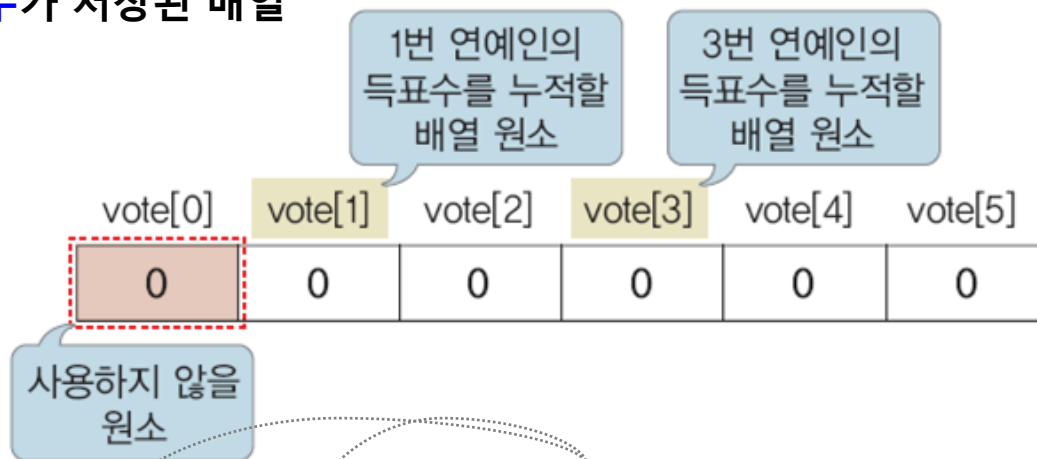
8.5 1차원 배열(array) 연습3

(1) 응답자 30명의 연예인 5명(1,2,3,4,5 : 0은 없음)에 대한 선호도 결과가 저장된 배열

→ `int survey[30] = {1, 3, 2, 5, 3, 2, 1, ..., 1, 4, 2, 3};`

(2) 각 연예인 별 총 득표수가 저장된 배열

→ `int vote[6] = {};`



(3) 해당 연예인의 득표수 누적

→ 예, 설문 조사 응답이 3 → `vote[3]++`

(4) 모든 응답마다 확인하여 연예인 득표수 누적하기

→ `for (i=0; i<PERSONS; i++) {`

`vote[survey[i]]++`

`}`

survey[0]	survey[1]	survey[2]	...		survey[29]
1	3	2			3
vote[0]	vote[1]	vote[2]	vote[3]	vote[4]	vote[5]
0	0	0	0	0	0

```
#include <stdio.h>

#define PERSONS 30          // 설문 조사 응답수
#define STARS 6             // 연예인 번호와 vote 배열의 첨자를 일치시키기 위해 연예인 수+1

int main()
{
    int survey[PERSONS] = {1, 3, 2, 5, 3, 2, 1, 2, 3, 4, 5, 2, 3, 3, 2,
                           1, 4, 5, 2, 3, 5, 1, 3, 4, 2, 3, 1, 4, 2, 3};
    int vote[STARS] = {};
    int i;

    for (i=0; i<PERSONS; i++)
    {
        vote[survey[i]]++;           // 연예인(survey[i])의 득표수를 1 증가
    }

    printf("연예인 득표수\n");
    printf("===== \n");

    for (i=1; i<STARS; i++) {
        printf(" %d번   %d표\n", i, vote[i]);
    }

    return 0;
}
```

- ➔ 결과 값을 종이에 미리 작성하시오
- ➔ 디버그 - 디버깅하지 않고 시작
- (좋은 예제는 아니지만 배열공부 도움)

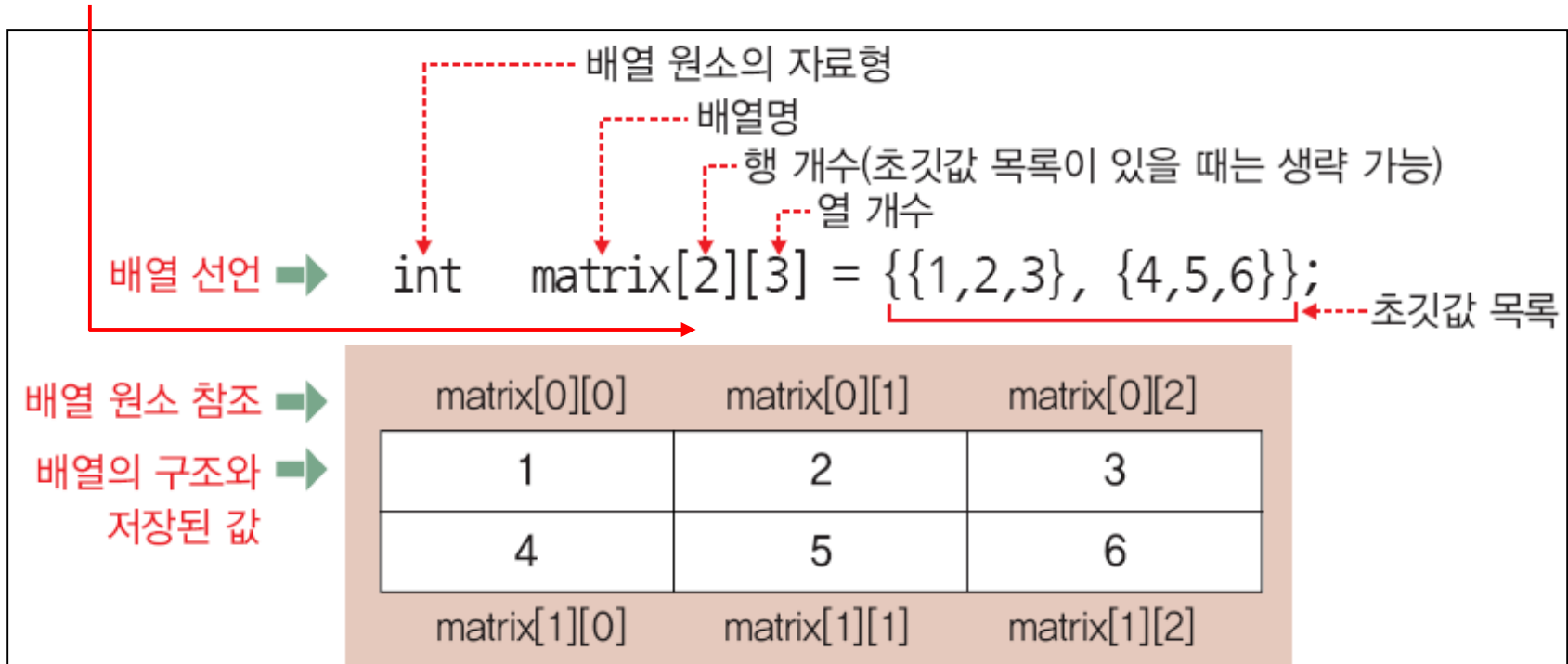
연예인 득표수	
=====	
1번	5표
2번	8표
3번	9표
4번	4표
5번	4표
계속하려면 아무 키나 누르십시오 .	

9. 다차원 배열(array)



9.1 2차원 배열

- 표 형태의 구조로 표현되는 자료들을 저장하는 데 사용
- 예, 2행(line, row) 3열(column)의 2차원 배열 matrix



9. 다차원 배열(array)

9.1 2차원 배열

- 예, 3행 4열의 2차원 배열 matrix

3행

4열

```
int test[3][4] = {{9, 8, 9, 10}, {10, 9, 10, 9}, {9, 7, 9, 8}};
```

	1차	2차	3차	4차
학생 1	9	8	9	10
학생 2	10	9	10	9
학생 3	9	7	9	8

학생의 퀴즈 점수 4개가 저장된 1차원 배열을 세 개 모아둔 것과 같다.

9. 다차원 배열(array)

9.1 2차원 배열

- 선언과 동시에 초기화하기

형식 2차원 배열을 선언과 동시에 초기화하기

자료형 배열명[행수][열수] = {{1행 초깃값 목록}, {2행 초깃값 목록}, ..., {마지막 행 초깃값 목록}};

`int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};`

→ 순서대로 한 행씩 초기화

<code>matrix[0][0]</code>	<code>matrix[0][1]</code>	<code>matrix[0][2]</code>
1	2	3
4	5	6
<code>matrix[1][0]</code>	<code>matrix[1][1]</code>	<code>matrix[1][2]</code>

`int matrix[][3] = {{1, 2, 3}, {4, 5, 6}};`

→ 초기값 목록이 있으면 → **행의 개수 생략 가능**

『주의』 **열 개수 생략 불가**

`int matrix[][3] = {1, 2, 3, 4, 5, 6};`

↓
↓
6개

열수가 3이므로 `int matrix[2][3]`으로 해석됨 => 권장하지 않음

9. 다차원 배열(array)

9.1 2차원 배열

- 2차원 배열 출력(프로그램)하기

2차원 배열은 1차원 배열의 배열

→ 배열 처리는 일반적으로 이중으로 중첩된 반복문을 이용

matrix 배열을 다음과 같이 출력하기

matrix[0][0]	matrix[0][1]	matrix[0][2]
1	2	3
4	5	6
matrix[1][0]	matrix[1][1]	matrix[1][2]



```
int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

```
1  for (row=0; row<2; row++)
2  {
3      for (col=0; col<3; col++)
4          printf("%3d", matrix[row][col]);
5      printf("\n");
6  }
```

배열의 행 개수

배열의 열 개수

한 행을 출력



1	2	3
4	5	6

→ 결과 값을 종이에 미리 작성하시오
(이중 for문의 결과를 순서대로 작성)

9. 다차원 배열(array)

9.1 2차원 배열

- 2차원 배열의 **행 단위 입력** ★

1	2	3
4	5	6

▶ 한 행씩 순서대로 입력 받기



0행 0열? 1
0행 1열? 2
0행 2열? 3
1행 0열? 4
1행 1열? 5
1행 2열? 6

첫째 행의 입력
둘째 행의 입력

같은 행 입력 시

행 첨자는 같고 열 첨자만 변함

```
int matrix[2][3];
```

```
for (row=0; row<2; row++)
```

```
{
```

```
    for (col=0; col<3; col++)
```

```
    {
```

```
        printf("%d행 %d열? ", row, col);
```

```
        scanf("%d", &matrix[row][col]);
```

```
    }
```

```
}
```


9. 다차원 배열(array)

9.1 2차원 배열

- 2차원 배열의 **열** 단위 입력

1	2	3
4	5	6

한 열씩 순서대로 입력 받기



.....
0열 0행? 1
0열 1행? 4 } 첫째 열 입력
1열 0행? 2
1열 1행? 5 } 둘째 열 입력
2열 0행? 3
2열 1행? 6 } 셋째 열 입력

같은 행 입력 시

열 첨자는 같고 행 첨자만 변함

```
int matrix[2][3];
for (col=0; col<3; col++)
{
    for (row=0; row<2; row++)
    {
        printf("%d행 %d열? ", row, col);
        scanf("%d", &matrix[row][col]);
    }
}
```

9. 다차원 배열(array)

9.2 2차원 배열 프로그램 예제

문제

입력 : 2×3 행렬 A와 B

출력 : A와 B를 더한 행렬 C

분석

행렬은 행과 열 수가 같아야 덧셈이 가능

→ 세 행렬 A, B, C 모두 동일 크기의 2차원 배열 이용

$$\begin{array}{ccc} 2 \times 3 \text{ 행렬 A} & + & 2 \times 3 \text{ 행렬 B} & = & 2 \times 3 \text{ 행렬 C} \\ \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} & + & \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix} & = & \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{pmatrix} \end{array}$$

덧셈 규칙 $c_{ij} = a_{ij} + b_{ij}$ ($i=1 \sim 2, j=1 \sim 3$)

• 2차원 배열 처리

→ 중첩 for문

9. 다차원 배열(array)

9.2 2차원 배열 프로그램 예제

[해결 과정]

- Ⓐ 행렬 A를 행 단위로 입력받기 (중첩 for문)
- Ⓑ 행렬 B를 행 단위로 입력받기 (중첩 for문)
- Ⓒ 행렬 C = 행렬 A + 행렬 B (중첩 for문)

$C[i][j] = A[i][j] + B[i][j]$ (행 $i = 0 \sim 1$, 열 $j = 0 \sim 2$)

[프로그램 실행 결과]

- Ⓓ 결과 행렬 C를 행 단위로 출력하기 (중첩 for문)

➔ 혼자, 2차원 배열을 정리하면서 프로그램 하시오

```
행렬 A 입력
1 0 1 11
1 0 2 13
1 0 3 14
2 0 1 15
2 0 2 16
2 0 3 17
```

```
행렬 B 입력
1 0 1 21
1 0 2 22
1 0 3 23
2 0 1 24
2 0 2 25
2 0 3 26
```

```
행렬 A      +      행렬 B      =      행렬 C
[ 11 13 14 ]   [ 21 22 23 ]   [ 32 35 37 ]
[ 15 16 17 ]   [ 24 25 26 ]   [ 39 41 43 ]
```

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

```
#define M 2          // 행 개수
```

```
#define N 3          // 열 개수
```

```
int main() {
```

```
    int A[M][N], B[M][N], C[M][N];    // M×N 행렬
```

```
    int i, j;
```

```
    printf("M행렬 A 입력 M\n");        // 행렬 A 입력
```

```
    for (i=0; i<M; i++) {
```

```
        for (j=0; j<N; j++) {
```

```
            printf("%d행 %d열? ", i+1, j+1);
```

```
            scanf_s("%d", &A[i][j]);
```

```
        } //end of for(N)
```

```
    } // end of for(M)
```

```
    printf("M행렬 B 입력 M\n");        // 행렬 B 입력
```

```
    for (i=0; i<M; i++) {
```

```
        for (j=0; j<N; j++) {
```

```
            printf("%d행 %d열? ", i+1, j+1);
```

```
            scanf_s("%d", &B[i][j]);
```

```
        } //end of for(N)
```

```
    } // end of for(M)
```

행\열	1	2	3	입력
1	1	2	3	?
2	1	2	3	?

[11 13 14]
[15 16 17]

실행時,
scanf 문제시 아래 사용
fflush(stdin);

행\열	1	2	3	입력
1	1	2	3	?
2	1	2	3	?

[21 22 23]
[24 25 26]

```
for (i=0; i<M; i++) {           // 두 행렬의 합 구하기
```

```
    for (j=0; j<N; j++) {
```

```
        C[i][j] = A[i][j] + B[i][j];
```

```
    } //end of for(N)
```

```
} // end of for(M)
```

```
/* 덧셈 결과 출력하기 */
```

```
printf("Wn 행렬 A    + Wt 행렬 B    = Wt 행렬 C\n");
```

```
for(i=0; i<M; i++) {
```

```
    /* 행렬 A의 한 행 출력 */
```

```
    printf(" [ ");
```

```
    for(j=0; j<N; j++) {
```

```
        printf("%2d ", A[i][j]);
```

```
    } // end of for(N)
```

```
    printf("] Wt");
```

```
    /* 행렬 B의 한 행 출력 */
```

```
    printf(" [ ");
```

```
    for(j=0; j<N; j++) {
```

```
        printf("%2d ", B[i][j]);
```

```
    } // end of for(N)
```

```
    printf("] Wt");
```

```
[ 11 13 14 ]
[ 15 16 17 ]
```

A

```
[ 21 22 23 ]
[ 24 25 26 ]
```

B

```
[ 32 35 37 ]
[ 39 41 43 ]
```

C

```
/* 행렬 C의 한 행 출력 */
```

```
printf(" [ ");
```

```
for(j=0; j<N; j++) {
```

```
    printf("%2d ", C[i][j]);
```

```
} // end of for(N)
```

```
printf("] Wn");
```

```
} // end of for(M)
```

```
return 0;
```

```
} // end of main
```

디버그 – 디버깅하지 않고 시작

9. 다차원 배열(array)

9.2 2차원 배열 프로그램 예제 결과

```
행렬 A 입력
1행 1열 ? 11
1행 2열 ? 13
1행 3열 ? 14
2행 1열 ? 15
2행 2열 ? 16
2행 3열 ? 17

행렬 B 입력
1행 1열 ? 21
1행 2열 ? 22
1행 3열 ? 23
2행 1열 ? 24
2행 2열 ? 25
2행 3열 ? 26

행렬 A      +      행렬 B      =      행렬 C
[ 11 13 14 ]   [ 21 22 23 ]   [ 32 35 37 ]
[ 15 16 17 ]   [ 24 25 26 ]   [ 39 41 43 ]
```

↓
int **A[2][3]**

↓
int **B[2][3]**

↓
int **C[2][3]**

9. 다차원 배열(array)

9.3 char형 1차원 배열을 이용한 문자열 처리

◆ 문자열 상수

“ ”로 묶어 놓은 연속된 문자들: “Seoul”, “Korea”, “Hong Gil Dong”

→ 문자열의 끝에는 **널(null)**이라는 ‘\0’ 문자가 있음.

문자열의 끝에 널문자 ‘\0’가
자동으로 포함되므로
실제로는 6바이트가 된다.

문자열 상수 “Seoul”

'S'	'e'	'o'	'u'	'l'	'\0'
-----	-----	-----	-----	-----	------

◆ 문자열 저장을 위한 char형 1차원 배열

문자열에 포함할 최대 문자수로
(실제 문자수 + 널문자 1)개

배열 선언 →

`char s[6] = "Seoul";`

`char s[6] = {'S', 'e', 'o', 'u', 'l'};`

배열 원소 참조 →

문자열 참조 →

	s[0]	s[1]	s[2]	s[3]	s[4]	s[5]
s →	'S'	'e'	'o'	'u'	'l'	'\0'

1차원 배열명 s는 문자열을
의미하며 실제로는 배열의
시작 주소에 해당한다.

문자열의 끝을
나타내는 널문자

9. 다차원 배열(array)

복습

9.3 char형 1차원 배열을 이용한 문자열 처리 : 문자 전용 입력 함수 `getchar`

```
변수 = getchar()
```

: `getchar` 함수 실행시 키보드에서 누른 문자가 `getchar()`의 결과 값이 됨

➔ 이 결과 값(문자 1개)이 변수에 대입(저장)됨

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

```
int main()
{
    unsigned char grade;

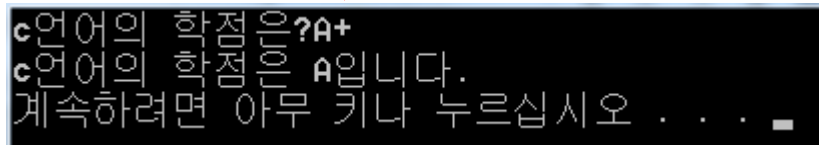
    printf("C 언어의 학점은? ");

    grade = getchar();

    printf("C 언어의 학점은 %c입니다.\n", grade);

    return 0;
}
```

↓ 문자 1개만 해당함



```
c언어의 학점은?A+
c언어의 학점은 A입니다.
계속하려면 아무 키나 누르십시오...
```


9. 다차원 배열(array)

9.3 char형 1차원 배열을 이용한 문자열 처리 : 문자열 전용 입력 함수 **gets, gets_s**

: getchar은 공백, 탭 등이 포함된 즉, 문자열을 한꺼번에 입력 받을 수 없음

→ "Hong GilDong"을 입력하면 "H"만 입력 됨

→ 행 단위 입력 함수 gets 또는 **gets_s**를 사용



```
gets(문자열을 저장할 변수);  
↳ 엔터키를 입력하기 전까지의 모든 문자가 변수에 저장
```

#include <stdio.h>

```
int main()  
{  
    char address[30];  
    printf("주소는? ");  
    gets_s(address);  
    printf("입력한 주소: %s", address);  
    return 0;  
}
```

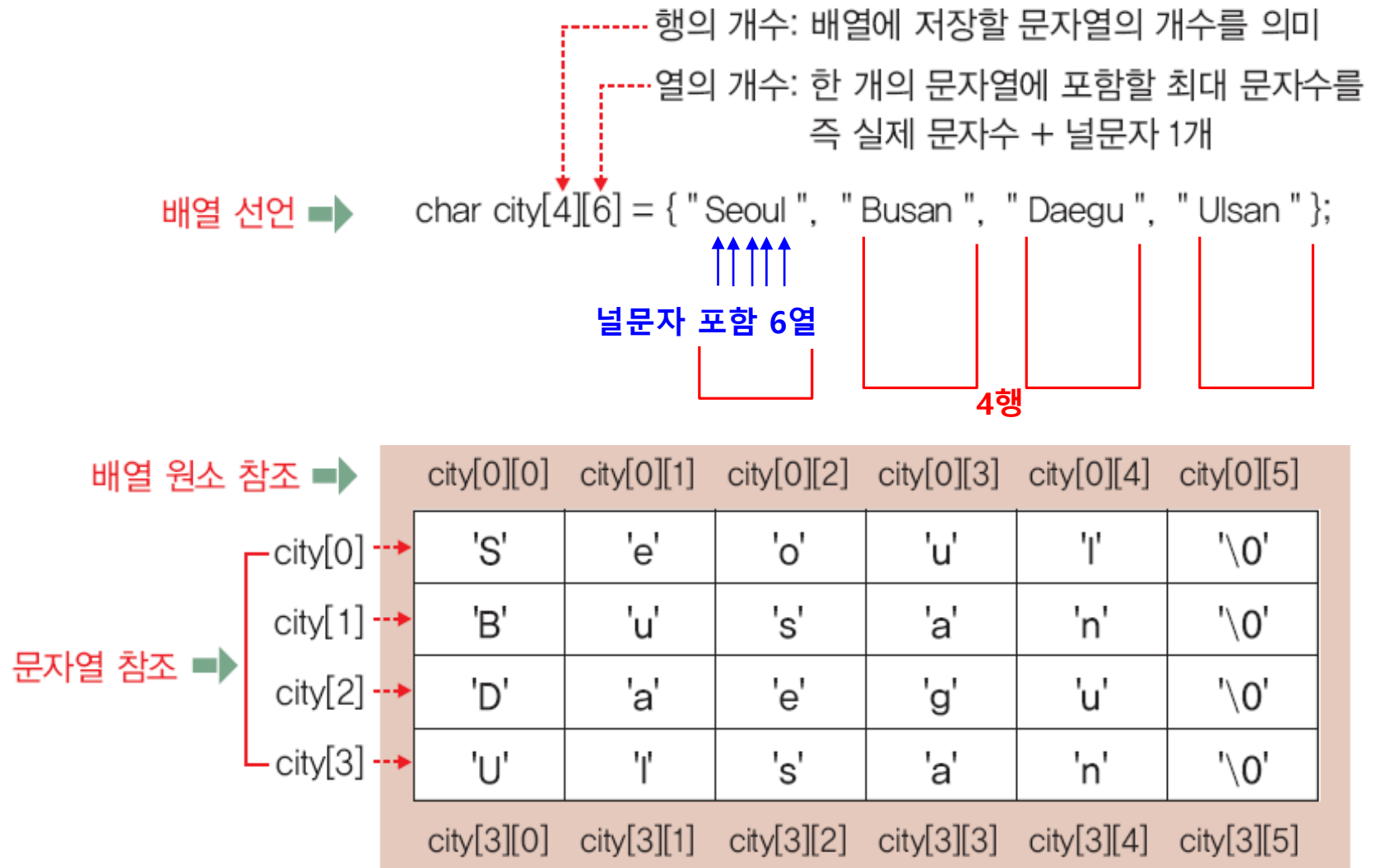
최대 29개까지

```
주소는?부산시 사상구 주례로 47  
입력한 주소는 : 부산시 사상구 주례로 47계속하려면 아무 키나 누르십시오 . . .
```

9. 다차원 배열(array)

9.4 char형 2차원 배열을 이용한 여러 개의 문자열 처리

◆ 여러 문자열 저장 : char형 2차원 배열 이용



9. 다차원 배열(array)

9.4 char형 2차원 배열을 이용한 여러 개의 문자열 처리

: 4개의 영어 도시명을 출력 후 한글 도시명을 입력

```
char city[4][6] = {"Seoul", "Busan", "Daegu", "Ulsan"};
```

```
char city_kor[4][5];
```

// city 배열의 도시명의 한글 저장

```
for (i=0; i<4; i++) {
```

```
    printf("%s\n", city[i]);
```

// puts(city[i]);와 동일

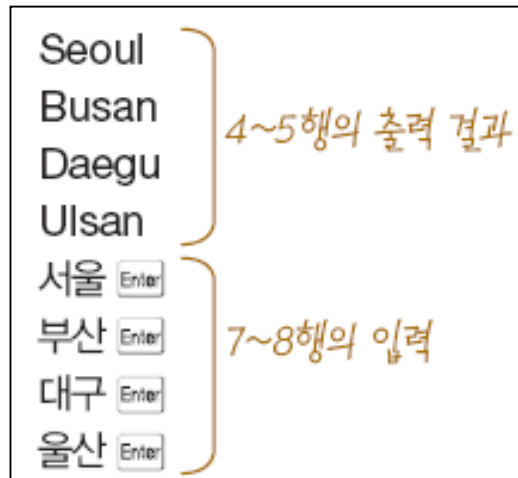
```
}
```

```
for (i=0; i<4; i++) {
```

```
    scanf_s("%s", city_kor[i]);
```

// gets(city_kor[i]) , 한글 2Byte

```
}
```



9. 다차원 배열(array)

9.5 3차원 배열

배열의 면 개수(초깃값 목록이 있다면 면 개수만 생략 가능)
면에 포함된 행 개수
각 행에 포함된 열 개수

배열 선언 → `int n[2][3][4] = { { {20, 22, 30, 35}, {12, 19, 35, 22}, {10, 9, 15, 20} }, { {20, 22, 30, 35}, {12, 19, 35, 22}, {12, 15, 9, 25} } }`

1면의 초깃값 목록

배열 원소 참조 → `n[1][2][0]` `n[1][0][1]` `n[1][0][2]` `n[1][0][3]`

		20	22	30	35	
첫째 면	20	22	30	35		둘째 면
	12	19	35	22	22	
	10	9	15	20	25	
	<code>n[0][2][0]</code>	<code>n[0][2][1]</code>	<code>n[0][2][2]</code>	<code>n[0][2][3]</code>		

```

#include <stdio.h>

#define L 2          // 3차원 배열의 면 개수
#define M 3          // 행 개수 3
#define N 4          // 열 개수 5

int main() {
    int t[L][M][N] = { { { 20, 22, 30, 35 }, { 12, 19, 35, 22 }, { 10, 9, 15, 20 } },
                        { { 20, 22, 30, 35 }, { 12, 19, 35, 22 }, { 12, 15, 9, 25 } } };

    int i, j, k;

    for (i = 0; i < L; i++) {                // 면 개수만큼 반복
        printf("Wn %d면 WnWn", i + 1);
        for (j = 0; j < M; j++) {            // 행 개수만큼 반복
            for (k = 0; k < N; k++) {         // 열 개수만큼 반복
                printf("%3d ", t[i][j][k]);
            }
            printf("Wn");
        }
    }

    return 0;
}

```

1면				
20	22	30	35	
12	19	35	22	
10	9	15	20	
2면				
20	22	30	35	
12	19	35	22	
12	15	9	25	

→ 디버그 – 디버깅하지 않고 시작