

상수와 기본 자료형 조건문

20200413

기본 자료형

C언어의 기본 자료형

	자료형	크기	값의 표현범위
정수형	char	1바이트	-128이상 +127이하
	short	2바이트	-32,768이상 +32,767이하
	int	4바이트	-2,147,483,648이상 +2,147,483,647이하
	long	4바이트	-2,147,483,648이상 +2,147,483,647이하
	long long	8바이트	-9,223,372,036,854,775,808이상 +9,223,372,036,854,775,807이하
실수형	float	4바이트	$\pm 3.4 \times 10^{-37}$ 이상 $\pm 3.4 \times 10^{+38}$ 이하
	double	8바이트	$\pm 1.7 \times 10^{-307}$ 이상 $\pm 1.7 \times 10^{+308}$ 이하
	long double	8바이트 이상	double 이상의 표현범위

[표 05-1: C언어의 기본 자료형]

Sizeof 연산자

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

```
int main(void)
{
```

```
    char ch = 'a';
    int num = 10;
```

```
    // 변수의 크기 확인
```

```
    printf("변수 ch의 크기: %d\n", sizeof(ch)); // 1
```

```
    printf("변수 num의 크기: %d\n", sizeof(num)); // 4
```

```
    // 자료형의 크기도 얼마인지 확인 가능
```

```
    printf("char의 크기: %d\n", sizeof(char)); // 1
```

```
    printf("int의 크기: %d\n", sizeof(int)); // 4
```

```
    return 0;
```

```
}
```

Sizeof()는 함수가 아니라 연산자다!!

★ 정수를 처리하기 위한 일반적인 자료형 ★

```
#include <stdio.h>

int main(void)
{
    // char에 관한 변수 정의
    char num1 = 1, num2 = 2;
    char res_char = 0;

    // short에 관한 변수 정의
    short num3 = 300, num4 = 400;
    short res_short = 0;

    // 각 char 자료형의 크기
    printf("sizeof num1: %d\n", sizeof(num1));
    printf("sizeof num2: %d\n\n", sizeof(num2));

    // 각 short 자료형의 크기
    printf("sizeof num3: %d\n", sizeof(num3));
    printf("sizeof num4: %d\n\n", sizeof(num4));

    // 계산하는 순간??
    printf("sizeof char add: %d\n", sizeof(num1 + num2));
    printf("sizeof short add: %d\n\n", sizeof(num3 + num4));

    // res라는 변수에 값을 담은 후의 크기
    res_char = num1 + num2;
    res_short = num3 + num4;
    printf("sizeof res_char: %d\n", sizeof(res_char));
    printf("sizeof res_short: %d\n\n", sizeof(res_short));

    return 0;
}
```



왜 더하는 순간 sizeof()는
우리가 아는 결과대로
나오지 않을까??



일반적으로 CPU가 처리하기에 가장 적합한 크기의
정수 자료형을 int로 정의한다. 따라서 int형의 연산속도가
다른 자료형의 연산속도에 비해 동일하거나 더 빠르다.

int형 연산이 CPU가 성능을 내기에 가장 좋은 연산!!

따라서 int보다 작은 크기의 데이터도

int형 데이터로 바뀌어서 연산이 진행된다!!

연산의 대상이 되는 변수를 선언하는 경우에는,
특히 연산의 횟수가 빈번한 경우에는 저장되는 값의 크기가
작더라도 int형 변수를 선언하는 것이 좋다!!

변수에 1이라는 값을 저장하고 싶은데

그렇다고 우리가 char 형식의 자료형을 쓰는 것은 아니잖아유?

char num = 1; <-> int num = 1;

그럼 char형 변수와 short형 변수는 왜 있는거야??

데이터의 양이 많아서 연산속도(int형 변수)보다
데이터의 크기를 줄이는 것이 더 중요한 데이터들!!

정수 자료형에서는 int를 보편적으로 선택하듯이,
실수 자료형에서도 보편적으로 선택하는 자료형은 없나유?

실수를 처리하기 위한 일반적인 자료형

실수 자료형	소수점 이하 정밀도	바이트 수
float	6자리 정밀도 너무 낮음	4
double	15자리	8
long double	18자리 크기 너무 커서 부담 多	12

[표 05-2: 실수 자료형의 정밀도]

Practice 1

#실수를 대표하는 자료형 #윤희남 교수님 책 예제

다음과 같은 수식의 값을 계산하여 화면에 출력하는 프로그램을 작성 하시오. 지수 표기법을 사용하여 변수들을 초기화한다.

$$3.32 * 10^{-3} + 9.76 * 10^{-8}$$

출력 결과

0.003320

$3.32 * 10^{-3}$ 는 $3.32e-3$ 으로 표기하면 된다.

Summary

1. 정수를 대표하는 자료형은 `int` (because of 연산속도)
2. 실수를 대표하는 자료형은 `double` (because of 정밀도)

unsigned

- 정수 자료형의 이름 앞에만 unsigned를 붙일 수 있다.
- unsigned가 붙으면, 부호 비트(MSB)도 데이터의 크기를 표현하는데 사용이 된다.
- 따라서 표현할 수 있는 값이 0 이상의 범위로 두 배가 된다.

ASCII Code

문자는 이렇게 표현되는 거구나!

```
int main(void)
{
    char ch1 = 'A';
    char ch2 = 'C';
}
```



```
int main(void)
{
    char ch1 = 65;
    char ch2 = 67;
}
```



Practice 2

#아스키 코드 #%c의 사용법

다음 코드를 분석하시오.

하지만 일반적으로
문자는 char에,
정수는 int에 넣는다!

```
#include <stdio.h>

int main(void)
{
    char ch1 = 'A' // 문자를 char에 저장 (익숙)
    char ch2 = 65; // 정수를 char에 저장
    int ch3 = 'Z' // 문자를 int에 저장
    int ch4 = 90; // 정수를 int에 저장 (익숙)

    printf("%c %d\n", ch1, ch1);
    printf("%c %d\n", ch2, ch2);
    printf("%c %d\n", ch3, ch3);
    printf("%c %d\n", ch4, ch4);

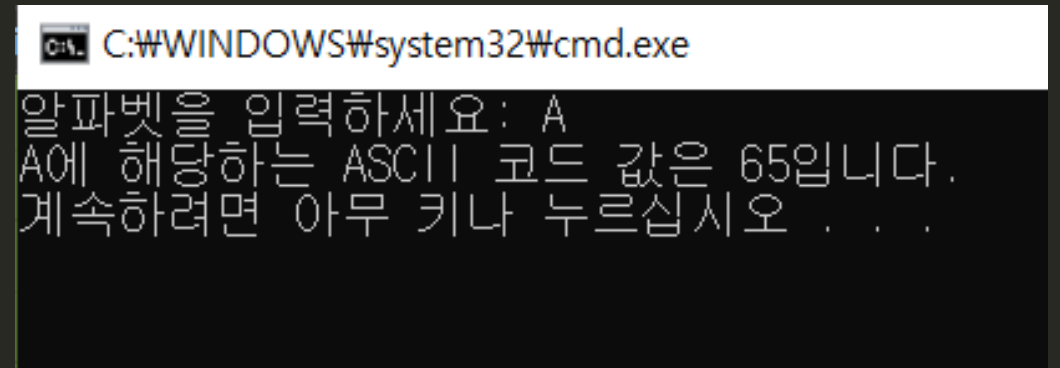
    return 0;
}
```

Practice 3

#아스키 코드 #scanf #%c #문자 하나를 %c와 %d 둘 다 출력

프로그램 사용자로부터 알파벳 문자 하나를 입력 받는다. 그리고 이에 해당하는 아스키 코드 값을 출력하는 코드를 작성해보자.

예를 들어, 사용자가 문자 A를 입력하면 정수 65를 출력해야 한다.



```
C:\WINDOWS\system32\cmd.exe
알파벳을 입력하세요: A
A에 해당하는 ASCII 코드 값은 65입니다.
계속하려면 아무 키나 누르십시오 . . .
```

Summary

1. 정수를 대표하는 자료형은 `int` (because of 연산속도)
2. 실수를 대표하는 자료형은 `double` (because of 정밀도)
3. `char` 형에서 문자로 입력 -> 그에 대한 아스키 코드 값으로 대체

const 상수

```
int main(void)
{
    const int MAX = 100;
    const double PI = 3.14;

    MAX = 500; // 값의 변경 불가!! 컴파일 에러 발생
}
```

대입 연산 중 자동 형 변환

```
#include <stdio.h>

int main(void)
{
    double num1 = 245;
    int num2 = 3.1415;
    int num3 = 129;
    char ch = num3;

    // 결과가 어떻게 나올까요??
    printf("245를 double 형으로: %f\n", num1);
    printf("3.1415를 int 형으로: %d\n", num2);
    printf("char 보다 큰 정수를 넣었을 때: %d\n", num3);

    return 0;
}
```

정수의 승격에 의한 형 변환

```
#include <stdio.h>

int main(void)
{
    // char에 관한 변수 정의
    char num1 = 1, num2 = 2;
    char res_char = 0;

    // short에 관한 변수 정의
    short num3 = 300, num4 = 400;
    short res_short = 0;

    // 각 char 자료형의 크기
    printf("sizeof num1: %d\n", sizeof(num1));
    printf("sizeof num2: %d\n\n", sizeof(num2));

    // 각 short 자료형의 크기
    printf("sizeof num3: %d\n", sizeof(num3));
    printf("sizeof num4: %d\n\n", sizeof(num4));

    // 계산하는 순간??
    printf("sizeof char add: %d\n", sizeof(num1 + num2));
    printf("sizeof short add: %d\n\n", sizeof(num3 + num4));

    // res라는 변수에 값을 담은 후의 크기
    res_char = num1 + num2;
    res_short = num3 + num4;
    printf("sizeof res_char: %d\n", sizeof(res_char));
    printf("sizeof res_short: %d\n\n", sizeof(res_short));

    return 0;
}
```


자료형 불일치로 발생하는 형 변환

```
#include <stdio.h>

int main(void)
{
    int num1 = 3, num2 = 4;
    double divRes = 0;

    divRes = num1 / num2;
    printf("나눗셈의 결과: %f\n", divRes);

    return 0;
}
```

★ 버퍼 ★

```
#include <stdio.h>

int main(void)
{
    int num;
    char ch;

    printf("숫자를 입력해보세요\n");
    scanf("%d", &num);
    printf("문자를 한번 입력해보세요\n");
    scanf("%c", &ch);
    printf("총 2번 입력받을 수 있으셨나요??\n");

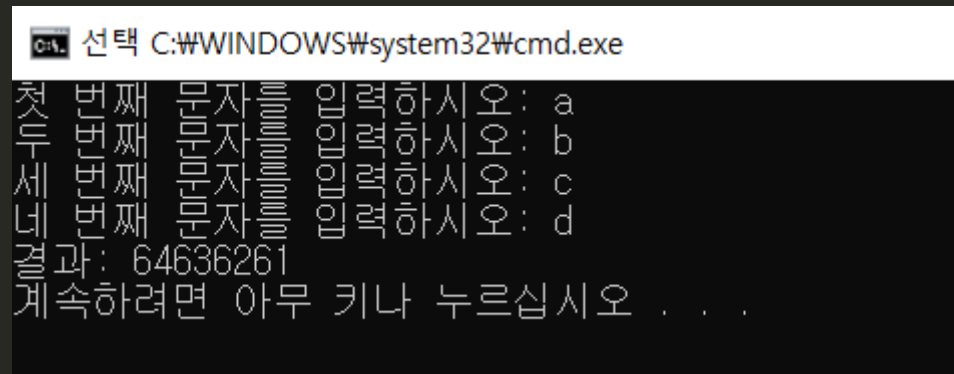
    return 0;
}
```

Practice 4 #비트 연산자 응용 #시프트 연산자 #버퍼 #unsigned int

비트 이동 연산을 이용하여 문자 4개를 받아서 하나의 unsigned int형의 변수 안에 저장하는 프로그램을 작성하라.

첫 번째 문자는 비트 0부터 비트 7까지 저장되고, 두 번째 문자는 비트 8부터 비트 15까지, 세 번째 문자는 비트 16에서 비트 23까지, 네 번째 문자는 비트 24부터 비트 31까지 저장된다.

결과로 생성되는 정수 값은 16진수로 출력하도록 한다. 비트 이동 연산과 비트 OR 연산을 사용하라.



```
선택 C:\WINDOWS\system32\cmd.exe
첫 번째 문자를 입력하시오: a
두 번째 문자를 입력하시오: b
세 번째 문자를 입력하시오: c
네 번째 문자를 입력하시오: d
결과: 64636261
계속하려면 아무 키나 누르십시오 . . .
```

Practice 4

#비트 연산자 응용 #시프트 연산자 #버퍼 #unsigned int

31

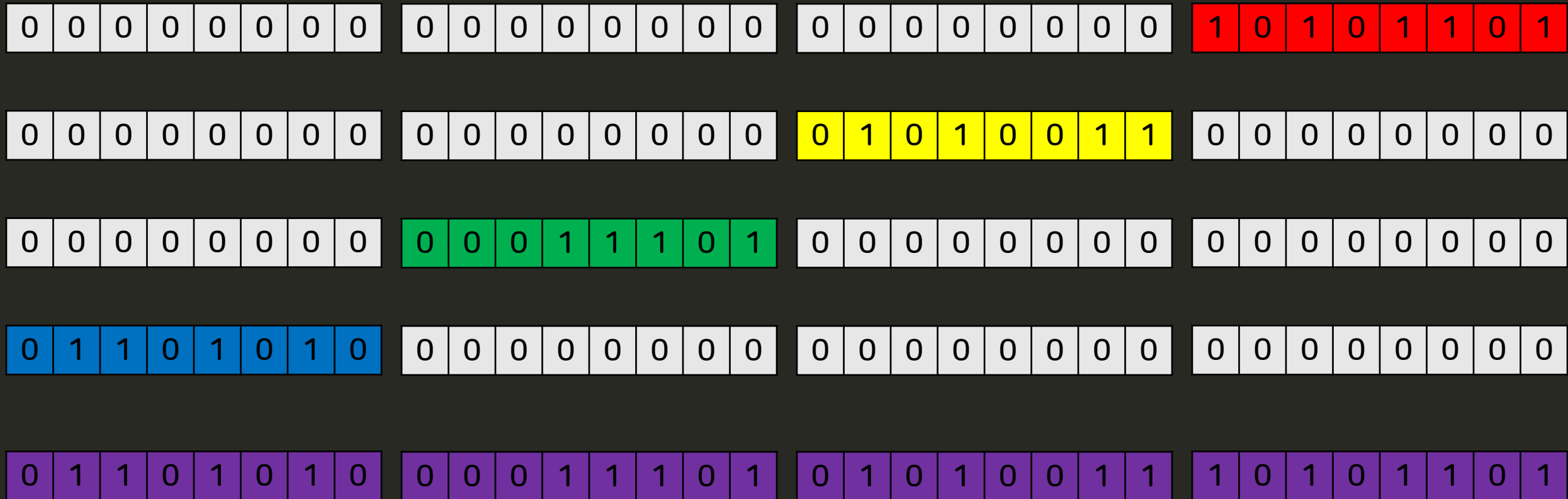
0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0

Practice 4 #비트 연산자 응용 #시프트 연산자 #버퍼 #unsigned int

31

0



Summary

1. 정수를 대표하는 자료형은 `int` (because of 연산속도)
2. 실수를 대표하는 자료형은 `double` (because of 정밀도)
3. `char` 형에서 문자로 입력 -> 그에 대한 아스키 코드 값으로 대체
4. 버퍼문제는 자료형이 `char` 일 때 발생, 버퍼를 비우자!!

조건에 따른 흐름의 분기

if statement

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

```
int main(void)
{
```

```
    int num = 0;
    printf("정수를 입력하세요: ");
    scanf("%d", &num);
```

```
    if(num<0) // num이 0보다 작으면 아래의 문장을 실행
        printf("입력 값은 0보다 작다.\n");
```

```
    if(num>0) // num이 0보다 크면 아래의 문장을 실행
        printf("입력 값은 0보다 크다.\n");
```

```
    if(num==0) // num이 0이면 아래의 문장을 실행
        printf("입력 값은 0이다.\n");
```

```
    return 0;
```

```
}
```


if-else statement

```
#include <stdio.h>

int main(void)
{
    int num = 0;
    printf("정수를 입력하세요: ");
    scanf("%d", &num);

    if(num<0)
        printf("입력 값은 0보다 작다.\n");

    else
        printf("입력 값은 0보다 작지 않다.\n");

    return 0;
}
```

if-else if-else statement

```
#include <stdio.h>

int main(void)
{
    int num = 0;
    printf("정수를 입력하세요: ");
    scanf("%d", &num);

    if(num<0) // num이 0보다 작으면 아래의 문장을 실행
        printf("입력 값은 0보다 작다.\n");

    else if(num>0) // num이 0보다 크면 아래의 문장을 실행
        printf("입력 값은 0보다 크다.\n");

    else // num이 0이면 아래의 문장을 실행
        printf("입력 값은 0이다.\n");

    return 0;
}
```

그냥 다 if로 쓰냐 혹은 else if를 사용하느냐

```
#include <stdio.h>

int main(void)
{
    int num = 0;
    printf("정수를 입력하세요: ");
    scanf("%d", &num);

    if(num<0) // num이 0보다 작으면 아래의 문장을 실행
        printf("입력 값은 0보다 작다.\n");

    if(num>0) // num이 0보다 크면 아래의 문장을 실행
        printf("입력 값은 0보다 크다.\n");

    if(num==0) // num이 0이면 아래의 문장을 실행
        printf("입력 값은 0이다.\n");

    return 0;
}
```

다 if로 쓰는 경우

VS

```
#include <stdio.h>

int main(void)
{
    int num = 0;
    printf("정수를 입력하세요: ");
    scanf("%d", &num);

    if(num<0) // num이 0보다 작으면 아래의 문장을 실행
        printf("입력 값은 0보다 작다.\n");

    else if(num>0) // num이 0보다 크면 아래의 문장을 실행
        printf("입력 값은 0보다 크다.\n");

    else // num이 0이면 아래의 문장을 실행
        printf("입력 값은 0이다.\n");

    return 0;
}
```

else if를 사용하는 경우

Practice 5 `#if-else statement` `#scanf`

계산기 프로그램을 작성하세요.

처음에는 계산할 연산자에 대한 숫자 값을 받고, 연산자를 결정한 다음,
두 개의 숫자를 입력한 후 결과가 나오도록 해야 합니다.

※ 정수형 나눗셈 값, 예외 처리 체크

```
C:\WINDOWS\system32\cmd.exe
숫자를 입력하세요.(1은 덧셈, 2는 뺄셈, 3은 곱셈, 4는 나눗셈입니다): 3
두 개의 실수를 입력하세요: 6 2
결과는 12입니다.
계속하려면 아무 키나 누르십시오 . . .
```

Summary

1. 정수를 대표하는 자료형은 `int` (because of 연산속도)
2. 실수를 대표하는 자료형은 `double` (because of 정밀도)
3. `char` 형에서 문자로 입력 -> 그에 대한 아스키 코드 값으로 대체
4. 버퍼문제는 자료형이 `char` 일 때 발생, 버퍼를 비우자!!
5. `if`는 모두 실행!! `if-else`는 해당 조건 만족하면 바로 탈출!!

조건 연산자(Conditional Operator)

```
res = ( num1 > num2 ) ? (num1) : (num2);
```

참일 경우,
num1을 반환

거짓일 경우,
num2을 반환

참 고

조건 연산자는 영 어색한데요.

조건 연산자는 간단한 if~else문을 대신할 수 있는 연산자로 C 프로그래머들이 선호하는 연산자이다. 사실 처음 접하면 매우 어색한 연산자 중 하나인데, 익숙해지고 또 자주 사용하다 보면, 코드를 간결히 하는데 한몫 제대로 한다는 것을 알게 될 것이다.

조건 연산자(Conditional Operator)

```
#include <stdio.h>

int main(void)
{
    int num = 0;
    int result = 0; // 결과 값을 저장할 변수
    printf("정수를 입력하세요: ");
    scanf("%d", &num);

    if(num >= 0)
        result = num;
    else
        result = -num;

    printf("입력한 정수의 절댓값은 %d입니다.\n", result);

    return 0;
}
```

```
#include <stdio.h>

int main(void)
{
    int num = 0;
    int result = 0; // 결과 값을 저장할 변수
    printf("정수를 입력하세요: ");
    scanf("%d", &num);

    result = (num >= 0) ? (num) : (-num);

    printf("입력한 정수의 절댓값은 %d입니다.\n", result);

    return 0;
}
```

조건 연산자(Conditional Operator)

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

```
int main(void)
```

```
{
```

```
    int num = 0;
```

```
    int result = 0; // 결과 값을 저장할 변수
```

```
    printf("정수를 입력하세요: ");
```

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

```
    (num >= 0) ? printf("절댓값: %d", num); : printf("절댓값: %d", -num);
```

```
    return 0;
```

```
}
```


Practice 6 `#if-else statement` `#scanf` `#조건 연산자`

```
C:\WINDOWS\system32\cmd.exe
두 개의 정수를 입력하세요: 4 16
두 수의 차는 12입니다.
계속하려면 아무 키나 누르십시오 . . .
```

두 개의 정수를 입력 받아서 두 수의 차를 출력하는 프로그램을 구현해 보자.

단, 무조건 큰 수에서 작은 수를 뺀 결과를 출력해야 한다.

예를 들어서 입력된 두 수가 순서에 상관 없이 12와 5라면 7이 출력되어야 하고,

입력된 두 수가 순서에 상관 없이 4와 16이라면 12가 출력되어야 한다.

즉, 출력 결과는 무조건 0 이상이 되어야 한다.

(단, if-else 연산을 사용해서 코드를 구현한 다음, 제대로 동작이 되면 멘토에게
검사를 맡은 후, 조건 연산자를 사용해서 다시 한 번 구현해보자.)

Summary

1. 정수를 대표하는 자료형은 `int` (because of 연산속도)
2. 실수를 대표하는 자료형은 `double` (because of 정밀도)
3. `char` 형에서 문자로 입력 -> 그에 대한 아스키 코드 값으로 대체
4. 버퍼문제는 자료형이 `char` 일 때 발생, 버퍼를 비우자!!
5. `if`는 모두 실행!! `if-else`는 해당 조건 만족하면 바로 탈출!!
6. 조건 연산자 지금은 많이 안 쓰이지만, 사용 방법이랑 목적만 기억

Mid-Term

1. 개념

전공 책 꼼꼼하게 읽기 (분명히 모르는 것 有)

2. 문제 풀이

다양한 문제 많이 풀어 보기 (정형화.. 유형 정리..?)

감 잃지 않기 (C언어 시험 2-3일 전.. but 개인차 있을 수 有)