2023/05/01

# APPDONG
# C언어 멘토링

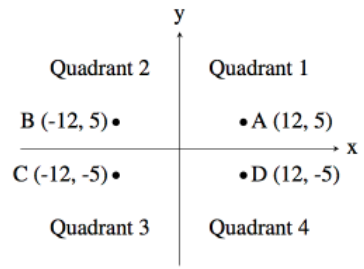
3주차
멘토 : 김민수

# Contents

# 조건문 2-4 : 사분면 고르기

## 문제

흔한 수학 문제 중 하나는 주어진 점이 어느 사분면에 속하는지 알아내는 것이다. 사분면은 아래 그림처럼 1부터 4까지 번호를 갖는다. "Quadrant n"은 "제n사분면"이라는 뜻이다.




예를 들어, 좌표가 (12, 5)인 점 A는 x좌표와 y좌표가 모두 양수이므로 제1사분면에 속한다. 점 B는 x좌표가 음수이고 y좌표가 양수이므로 제2사분면에 속한다.

점의 좌표를 입력받아 그 점이 어느 사분면에 속하는지 알아내는 프로그램을 작성하시오. 단, x좌표와 y좌표는 모두 양수나 음수라고 가정한다.

## 입력

첫 줄에는 정수 x가 주어진다. (−1000 ≤ x ≤ 1000; x ≠ 0) 다음 줄에는 정수 y가 주어진다. (−1000 ≤ y ≤ 1000; y ≠ 0)

## 출력

점 (x, y)의 사분면 번호(1, 2, 3, 4 중 하나)를 출력한다.

### 예제 입력 1 복사

```
12
5
```

### 예제 출력 1 복사

```
1
```

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

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

    if (x > 0 && y > 0) printf("1");
    else if (x < 0 && y>0) printf("2");
    else if (x < 0 && y < 0) printf("3");
    else printf("4");

    return 0;
}
```

사분면 결정 조건을 if-else문으로 구현

# 조건문 2-5 : 알람 시계

## 문제

상근이는 매일 아침 알람을 듣고 일어난다. 알람을 듣고 바로 일어나면 다행이겠지만, 항상 조금만 더 자려는 마음 때문에 매일 학교를 지각하고 있다.

상근이는 모든 방법을 동원해보았지만, 조금만 더 자려는 마음은 그 어떤 것도 없앨 수가 없었다.

이런 상근이를 불쌍하게 보던 창영이는 자신이 사용하는 방법을 추천해 주었다.

바로 "45분 일찍 알람 설정하기"이다.

이 방법은 단순하다. 원래 설정되어 있는 알람을 45분 앞서는 시간으로 바꾸는 것이다. 어차피 알람 소리를 들으면, 알람을 끄고 조금 더 잘 것이기 때문이다. 이 방법을 사용하면, 매일 아침 더 잤다는 기분을 느낄 수 있고, 학교도 지각하지 않게 된다.

현재 상근이가 설정한 알람 시각이 주어졌을 때, 창영이의 방법을 사용한다면, 이를 언제로 고쳐야 하는지 구하는 프로그램을 작성하시오.

## 입력

첫째 줄에 두 정수 H와 M이 주어진다. ($0 \le H \le 23$, $0 \le M \le 59$) 그리고 이것은 현재 상근이가 설정한 놓은 알람 시간 H시 M분을 의미한다.

입력 시간은 24시간 표현을 사용한다. 24시간 표현에서 하루의 시작은 0:0(자정)이고, 끝은 23:59(다음날 자정 1분 전)이다. 시간을 나타낼 때, 불필요한 0은 사용하지 않는다.

## 출력

첫째 줄에 상근이가 창영이의 방법을 사용할 때, 설정해야 하는 알람 시간을 출력한다. (입력과 같은 형태로 출력하면 된다.)

### 예제 입력 1 복사

```
10 10
```

### 예제 출력 1 복사

```
9 25
```

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main() {
    int H, M;
    int answer;

    scanf("%d %d", &H, &M);

    answer = H * 60 + M - 45;

```

시간을 모두 "분"으로 먼저 바꿔준다.

그 후 "45분"을 감산한다.

```
13      if (answer < 0) {
14          printf("23 %d", answer + 60);
15      }
16      else {
17          printf("%d %d", answer / 60, answer % 60);
18      }
19
20      return 0;
21  }
```

24시간 표기법으로 출력을 해야하기 때문에
계산 결과가 음수가 나오는 경우
시는 무조건 "23시", 분은 계산 "결과 + 60분"

시 = 나누기 60, 분 = 나머지 연산 60

# 조건문 2-6 : 오븐 시계

## 문제

KOI 전자에서는 건강에 좋고 맛있는 훈제오리구이 요리를 간편하게 만드는 인공지능 오븐을 개발하려고 한다. 인공지능 오븐을 사용하는 방법은 적당한 양의 오리 훈제 재료를 인공지능 오븐에 넣으면 된다. 그러면 인공지능 오븐은 오븐구이가 끝나는 시간을 분 단위로 자동적으로 계산한다.

또한, KOI 전자의 인공지능 오븐 앞면에는 사용자에게 훈제오리구이 요리가 끝나는 시각을 알려 주는 디지털 시계가 있다.

훈제오리구이를 시작하는 시각과 오븐구이를 하는 데 필요한 시간이 분단위로 주어졌을 때, 오븐구이가 끝나는 시각을 계산하는 프로그램을 작성하시오.

## 입력

첫째 줄에는 현재 시각이 나온다. 현재 시각은 시 A (0 ≤ A ≤ 23) 와 분 B (0 ≤ B ≤ 59)가 정수로 빈칸을 사이에 두고 순서대로 주어진다. 두 번째 줄에는 요리하는 데 필요한 시간 C (0 ≤ C ≤ 1,000)가 분 단위로 주어진다.

## 출력

첫째 줄에 종료되는 시각의 시와 분을 공백을 사이에 두고 출력한다. (단, 시는 0부터 23까지의 정수, 분은 0부터 59까지의 정수이다. 디지털 시계는 23시 59분에서 1분이 지나면 0시 0분이 된다.)

## 예제 입력 1 복사

```
14 30
20
```

## 예제 출력 1 복사

```
14 50
```

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main() {
    int A, B, C;
    int answer;

    scanf(" %d %d", &A, &B);
    scanf(" %d", &C);

    answer = A * 60 + B + C;

```

앞의 알람 시계 문제와 같은 방식으로 접근

시간을 모두 "분"으로 먼저 바꿔준다.
그 후 C를 더해준다.

```
14      if (answer / 60 > 23) {
15          printf("%d %d", answer / 60 - 24, answer % 60);
16      }
17      else {
18          printf("%d %d", answer / 60, answer % 60);
19      }
20
21      return 0;
22  }
```

answer / 60으로 "시 " 를 구했을 때
23을 초과한다면 "-24"를 해준다.
(C의 범위가 0~1000이고 24시간 표기법으로
출력해야하기 때문)

시 = 나누기 60, 분 = 나머지 연산 60

# 조건문 2-7 : 주사위 세개

## 문제

1에서부터 6까지의 눈을 가진 3개의 주사위를 던져서 다음과 같은 규칙에 따라 상금을 받는 게임이 있다.

1. 같은 눈이 3개가 나오면 10,000원+(같은 눈)×1,000원의 상금을 받게 된다.
2. 같은 눈이 2개만 나오는 경우에는 1,000원+(같은 눈)×100원의 상금을 받게 된다.
3. 모두 다른 눈이 나오는 경우에는 (그 중 가장 큰 눈)×100원의 상금을 받게 된다.

예를 들어, 3개의 눈 3, 3, 6이 주어지면 상금은 1,000+3×100으로 계산되어 1,300원을 받게 된다. 또 3개의 눈이 2, 2, 2로 주어지면 10,000+2×1,000 으로 계산되어 12,000 원을 받게 된다. 3개의 눈이 6, 2, 5로 주어지면 그중 가장 큰 값이 6이므로 6×100으로 계산되어 600원을 상금으로 받게 된다.

3개 주사위의 나온 눈이 주어질 때, 상금을 계산하는 프로그램을 작성 하시오.

## 입력

첫째 줄에 3개의 눈이 빈칸을 사이에 두고 각각 주어진다.

## 출력

첫째 줄에 게임의 상금을 출력 한다.

### 예제 입력 1 복사

```
3 3 6
```

### 예제 출력 1 복사

```
1300
```

### 예제 입력 2 복사

```
2 2 2
```

### 예제 출력 2 복사

```
12000
```

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main() {
    int x, y, z;

    scanf("%d %d %d", &x, &y, &z);

    if (x == y && y == z) {
        printf("%d", 10000 + x * 1000);
    }
```

같은 눈이 3개인 경우 - (1)
같은 눈이 2개인 경우 - (2)
모두 다른 눈인 경우 - (3)

총 3가지 경우를 고려해서 if-else문 설계

같은 눈이 3개인 경우

출력은 10000원 + (같은 눈) * 1000원

```
13  else if (x == y) {
14      printf("%d", 1000 + x * 100);
15  }
16  else if (y == z) {
17      printf("%d", 1000 + y * 100);
18  }
19  else if (z == x) {
20      printf("%d", 1000 + z * 100);
21  }
```

같은 눈이 2개인 경우

출력은 1000원 + (같은 눈) * 100원

```
22      else if (x > y && x > z) {
23          printf("%d", x * 100);
24      }
25      else if (y > x && y > z) {
26          printf("%d", y * 100);
27      }
28      else {
29          printf("%d", z * 100);
30      }
31
32      return 0;
33  }
```

모두 다른 눈인 경우
x값이 가장 클 때 / y값이 가장 클 때 / z값이 가장 클 때
총 3가지 경우를 나눠서 if문 설정

출력은 (가장 큰 눈) * 100원

# 반복문 3-9 : 별 찍기 - 1

## 문제

첫째 줄에는 별 1개, 둘째 줄에는 별 2개, N번째 줄에는 별 N개를 찍는 문제

## 입력

첫째 줄에 N(1 ≤ N ≤ 100)이 주어진다.

## 출력

첫째 줄부터 N번째 줄까지 차례대로 별을 출력한다.

## 예제 입력 1 복사

```
5
```

## 예제 출력 1 복사

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

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main() {
    int n;
    int i, j;

    scanf("%d", &n);

    for (i = 1; i <= n; i++) {
        for (j = 1; j <= i; j++) {
            printf("*");
        }
        printf("₩n");
    }

    return 0;
}
```

출력 예시)

*
**
***
****
...

별이 하나씩 증가하면서 층층이 쌓이는 형식

i : 층(높이), j : 별 개수
어떤 층의 별 개수는 그 층의 숫자(높이)와 같다

별을 출력하고 나면 개행문자를 출력해서
층을 쌓는다.

# 반복문 3-10 : 별 찍기 - 2

## 문제

첫째 줄에는 별 1개, 둘째 줄에는 별 2개, N번째 줄에는 별 N개를 찍는 문제

하지만, 오른쪽을 기준으로 정렬한 별(예제 참고)을 출력하시오.

## 입력

첫째 줄에 N(1 ≤ N ≤ 100)이 주어진다.

## 출력

첫째 줄부터 N번째 줄까지 차례대로 별을 출력한다.

## 예제 입력 1 복사

```
5
```

## 예제 출력 1 복사

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

앞의 문제 별 찍기 1번은 별을 왼쪽정렬하여 출력이었다면
별 찍기 2번은 별을 오른쪽정렬하여 출력을 해야한다.

```
11      for (i = 1; i <= n; i++) {
12          for (j = n; j > i; j--) {
13              printf(" ");
14          }
15
16          for (j = 1; j <= i; j++) {
17              printf("*");
18          }
19
20          printf("₩n");
21      }
22
23      return 0;
24  }
```

오른쪽 정렬을 위해서 공백을 규칙에 따라 출력하고 나서 별을 출력한다.
어떤 층의 공백 개수는 "n - 그 층의 숫자(높이)"

어떤 층의 별 개수는 그 층의 숫자(높이)와 같다
-> 앞의 문제와 별 출력 규칙은 동일하다.

# 반복문 3-11 : A + B - 5

## 문제

두 정수 A와 B를 입력받은 다음, A+B를 출력하는 프로그램을 작성하시오.

## 입력

입력은 여러 개의 테스트 케이스로 이루어져 있다.

각 테스트 케이스는 한 줄로 이루어져 있으며, 각 줄에 A와 B가 주어진다. (0 < A, B < 10)

입력의 마지막에는 0 두 개가 들어온다.

## 출력

각 테스트 케이스마다 A+B를 출력한다.

## 예제 입력 1 복사

```
1 1
2 3
3 4
9 8
5 2
0 0
```

## 예제 출력 1 복사

```
2
5
7
17
7
```

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main() {
    int A, B;

    while (1) {
        scanf("%d %d", &A, &B);

        if (A + B == 0) break;

        printf("%d\n", A + B);
    }
}
```

수행 횟수(반복 횟수)가 정해지지 않은 경우
조건식에 참값(일반적으로 1)을 넣어서
무한반복으로 설정

입력 마지막에 0 두 개가 들어온다고 되어있으므로
탈출 조건을 "A + B == 0"으로 설정

# 반복문 3-12 : A + B - 4

## 문제

두 정수 A와 B를 입력받은 다음, A+B를 출력하는 프로그램을 작성하시오.

## 입력

입력은 여러 개의 테스트 케이스로 이루어져 있다.

각 테스트 케이스는 한 줄로 이루어져 있으며, 각 줄에 A와 B가 주어진다. (0 < A, B < 10)

## 출력

각 테스트 케이스마다 A+B를 출력한다.

## 예제 입력 1 복사

```
1 1
2 3
3 4
9 8
5 2
```

## 예제 출력 1 복사

```
2
5
7
17
7
```

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main() {
    int A, B;

    while (scanf("%d %d", &A, &B) != -1) {
        printf("%d₩n", A + B);
    }

    return 0;
}
```

탈출 조건을 문제에서 따로 명시하지 않음.
테스트케이스의 수도 주어지지 않음.

scanf();의 반환값을 조건식으로 설정.

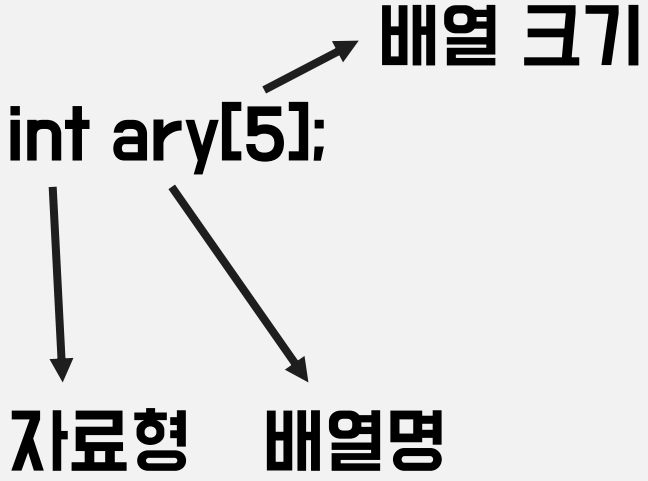
scanf(); 함수는 읽어들인 데이터의 개수를 반환한다.
scanf("%d %d", &A, &B); <- 이 경우에는 2를 반환.

그러나, 읽어들일 데이터가 없는 경우
즉, EOF(End of File) 상태가 된 경우
데이터를 읽으려고 하면 -1을 반환시킨다.

# 배열(Array)

특정한 데이터를 모아서 관리하기 위해 사용되는 데이터 타입

같은 형태의 많은 데이터를 메모리에 연속적으로 할당한다.
특정 타입의 변수들의 집합이라고 생각하면 된다.




int ary[5] = {1, 2, 3, 4, 5};

# 배열(Array)

배열의 자료형이 int 타입인 경우 int 타입 데이터만 저장된다.
(double 타입은 double 타입만, char 타입은 문자열을 저장하기 위해 사용)

배열은 인덱스(index)로 데이터에 접근할 수 있다.
(인덱스는 항상 0부터 시작한다.)

배열 각 인덱스의 공간은 배열의 자료형의 크기를 가진다.

배열을 선언할 때 배열의 크기는 항상 상수값이어야 한다. (변수 x)

# 배열(Array)

## 1. 1차원 배열

배열 초기화 방법

1) int arr[5] = {};
2) int arr[5] = {0};
3) int arr[5] = {0, };

-> 모두 0으로 초기화

4) int arr[5] = {-1, };

-> 첫번째 원소만 -1로 초기화
나머지 원소들은 0으로 초기화

5) int arr[] = {1, 2, 3, 4, 5};

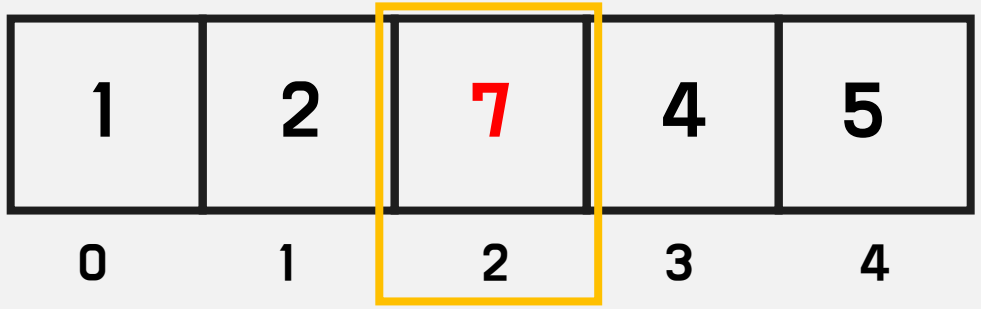
-> 원하는 값을 미리 지정 가능
(이 때, 배열 크기 명시 안해도 상관x)

# 배열(Array)

## 1. 1차원 배열

배열 원소 값 활용하는 방법 : 인덱스로 접근하기
int arr[5] = {1, 2, 3, 4, 5};

arr[2] = 7;  // 대입연산



# 배열(Array) - 배열 원소 출력하기

```c
#include <stdio.h>

int main() {
    int arr[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

    printf("Array 3번째 원소 %d₩n", arr[2]);

    return 0;
}
```

인덱스는 0부터 시작하는 것을 꼭 기억하자.

```
Microsoft Visual Studio 디버그 콘솔
Array 3번째 원소 3

C:\Users\minsu\Desktop\studyC\Project1\
).
이 창을 닫으려면 아무 키나 누르세요...
```

# 배열(Array) - 평균 점수 구하기

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main() {
    int score[5];
    int i, total = 0;
    double avg = 0.0;

    for (i = 0; i < 5; i++) {
        scanf("%d", &score[i]);
    }
```

배열 원소 입력 받기

# 배열(Array) - 평균 점수 구하기

```
13
14      for (i = 0; i < 5; i++) {
15          total += score[i];
16          printf("%4d", score[i]);
17      }
18      printf("₩n");
19
20      avg = (double)total / 5;
21
22      printf("평균 점수 : %.1f₩n", avg);
23
24      return 0;
25  }
```

배열 원소 모두 더하기
배열 원소 출력

평균 계산

# 배열(Array)

## 2. 다차원 배열

어떤 배열이 1차원 배열을 원소로 가진다 -> 2차원 배열
2차원 배열을 원소로 가진다 -> 3차원 배열
… n차원 배열

2차원 배열 선언
(배열의 타입) (배열의 이름) [x][y];

-> 2개의 인덱스로 자료를 관리

3차원 배열 선언
(배열의 타입) (배열의 이름) [x][y][z];

-> 3개의 인덱스로 자료를 관리

# 배열(Array)

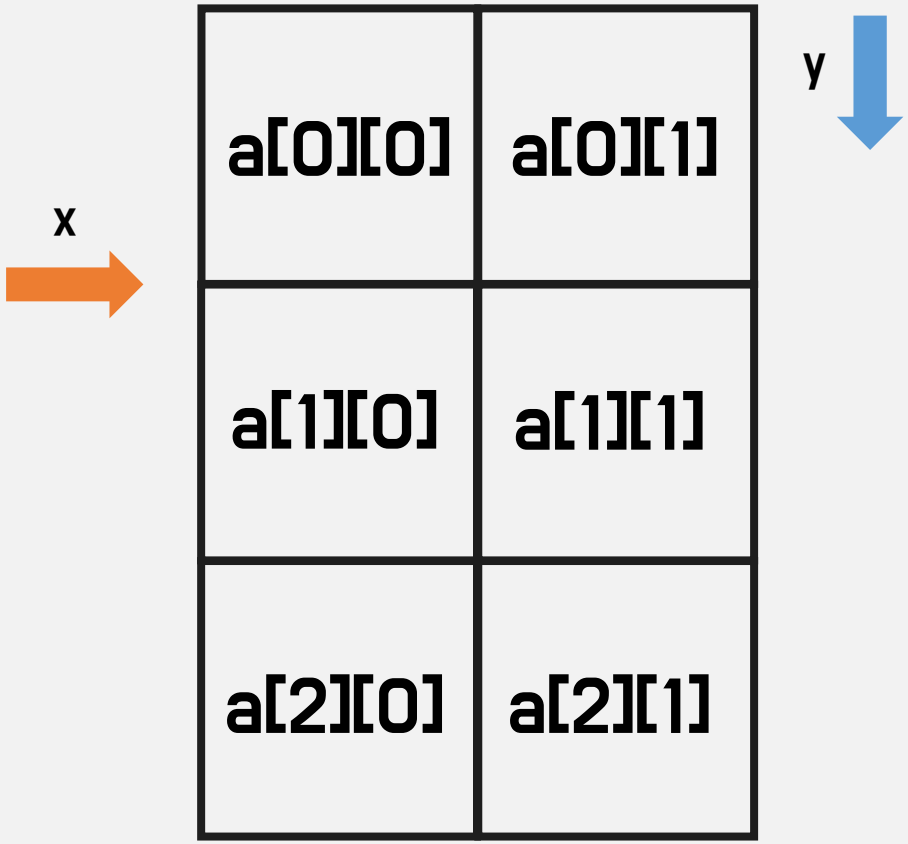
## 3. 2차원 배열 정의

1) int arr[2][3] = {1, 2, 3, 4, 5, 6};
2) int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};
3) int arr[][3] = {1, 2, 3, 4, 5, 6};

-> 다차원 배열의 경우 맨 앞의 크기를 제외한 나머지 크기들은 정확하게 명시해야 한다.

# 배열(Array)

int a[3][2];  -> 각 원소는 원소가 2개인 (크기가 2인) 1차원 배열이다.




컴퓨터
메모리 공간

| a[0][0] |
| --- |
| a[0][1] |
| a[1][0] |
| a[1][1] |
| a[2][0] |
| a[2][1] |

# 배열(Array) - 여러 학생의 평균 점수 구하기

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main() {
    int score[3][4] = { 0, };
    double avg[3] = { 0.0 };

    for (int i = 0; i < 3; i++) {
        printf("학생 %d 네 과목 점수 입력 >> ", i + 1);

        for (int j = 0; j < 4; j++) {
            scanf(" %d", &score[i][j]);
            avg[i] += score[i][j];
        }
    }

```

# 배열(Array) - 여러 학생의 평균 점수 구하기

```
18      for (int i = 0; i < 3; i++) {
19          avg[i] /= 4;
20          printf("학생 %d의 평균 : %lf₩n", i + 1, avg[i]);
21      }
22  }
```

```
Microsoft Visual Studio 디버그 콘솔
학생 1 네 과목 점수 입력 >> 10 20 30 40
학생 2 네 과목 점수 입력 >> 11 12 13 14
학생 3 네 과목 점수 입력 >> 90 95 66 44
학생 1의 평균 : 25.000000
학생 2의 평균 : 12.500000
학생 3의 평균 : 73.750000

C:\Users\minsu\Desktop\studyC\Project1\x64\D
).
이 창을 닫으려면 아무 키나 누르세요...
```

# 포인터 기초

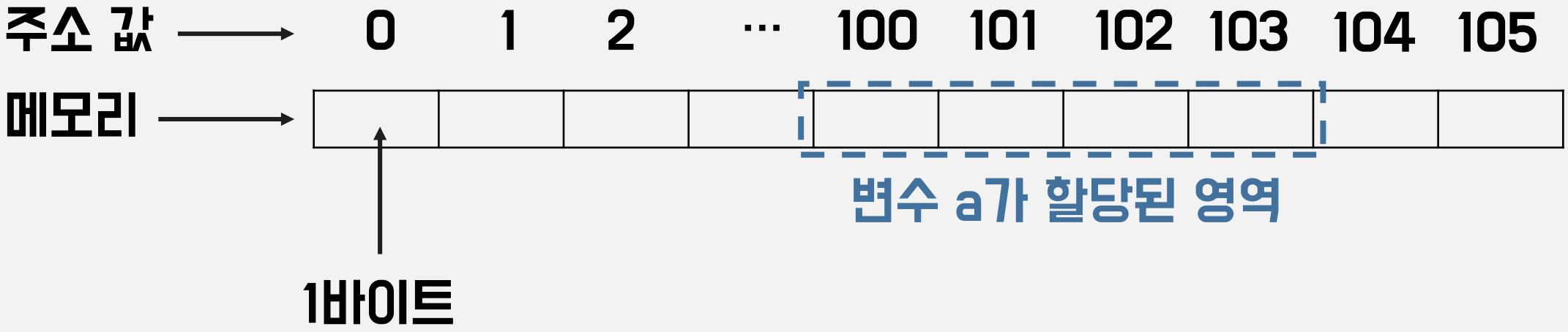
메모리의 주소

데이터를 저장 해놓는 메모리의 위치를 주소 값으로 식별한다

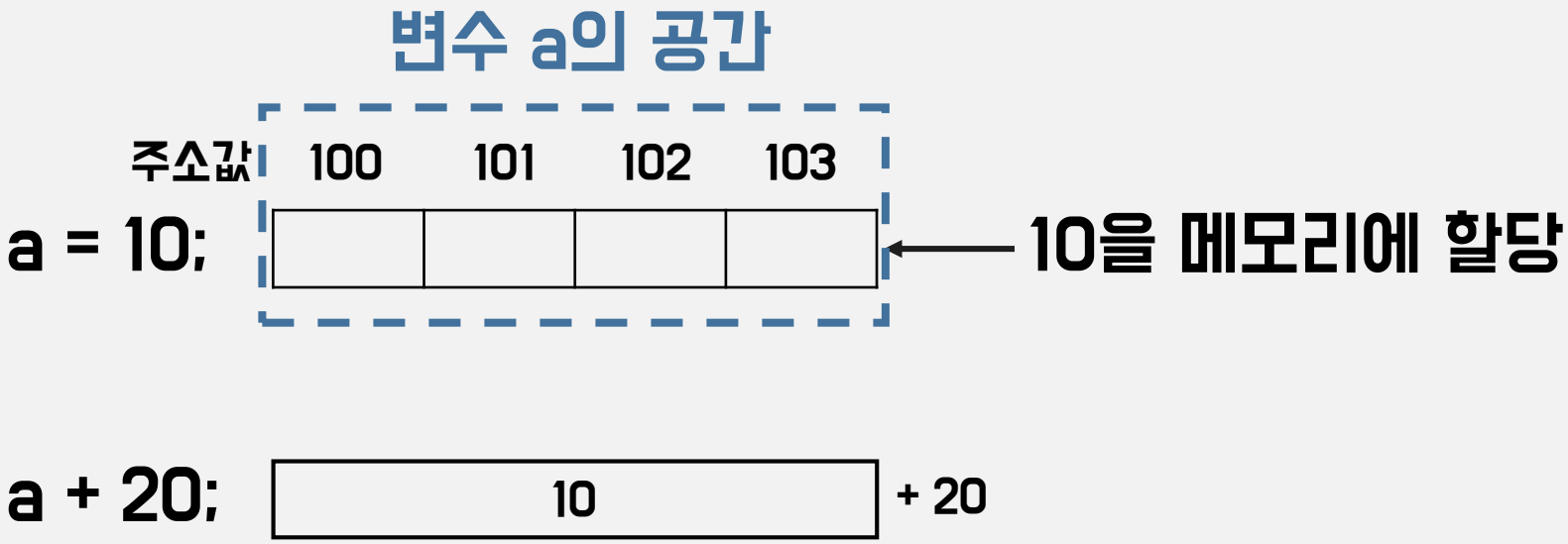
주소 값은 0부터 시작하고 바이트 단위로 1씩 증가하며
2바이트 이상의 크기를 갖는 변수는 여러 개의 주소 값에 걸쳐 할당된다

# 포인터 기초

int a;



# 포인터 기초




a = 10; -> 메모리의 100번지에서 103번지까지 4바이트 공간에 10을 저장한다
a + 20; -> 메모리 100번지부터 103번지까지 4바이트에 저장된 값과 20을 더하는 연산을 수행한다

# 포인터 기초

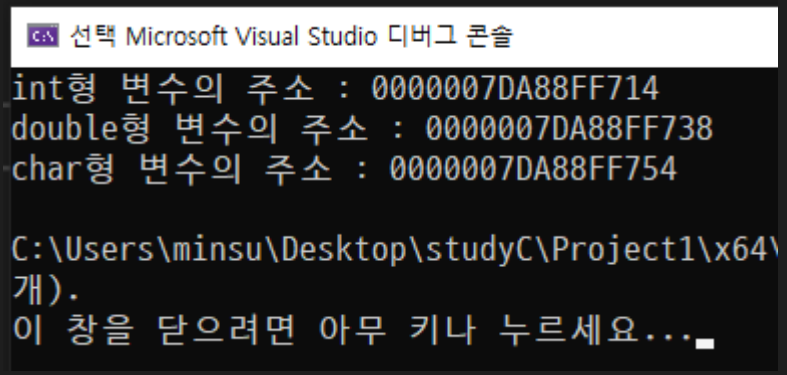
주소 연산자 &

여기서 말하는 "주소"는 변수가 할당된 메모리 공간의 시작 주소를 의미한다
주소 연산자를 사용해서 주소 값을 알 수 있다

주소는 보통 16진수로 표현하기 때문에 주소를 출력할 때는 %p를 사용해서 출력하면 된다

# 포인터 기초

```c
#include <stdio.h>

int main() {
    int a;
    double b;
    char c;

    printf("int형 변수의 주소 : %p₩n", &a);
    printf("double형 변수의 주소 : %p₩n", &b);
    printf("char형 변수의 주소 : %p₩n", &c);

    return 0;
}
```



# 포인터 기초

간접 참조 연산자 *

메모리의 주소를 필요할 때마다 계속 주소 연산자로 구하는 것 보다는
한 번 구한 주소를 저장해서 사용하면 편하다

이 때, 포인터를 선언해서 변수의 메모리 주소를 할당하면 된다

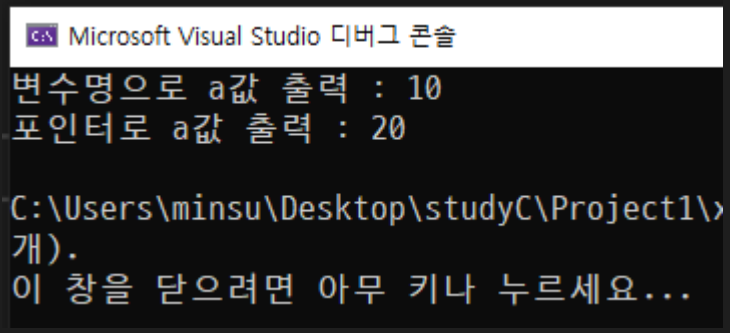
포인터 자체는 변수의 메모리 주소 값을 가리키고 있는데
간접 참조 연산자(*)로 변수의 데이터 값에 접근할 수 있다

# 포인터 기초

```c
#include <stdio.h>

int main() {
    int a = 10;
    int* pa;

    pa = &a;

    printf("변수명으로 a값 출력 : %d₩n", a);
    *pa = 20;

    printf("포인터로 a값 출력 : %d₩n", *pa);

    return 0;
}
```

포인터 변수에 a의 주소 할당

간접참조로 a값 변경



# 포인터 기초

int a;
int *pa;

pa = &a;

값을 입력할 때
scanf("%d", &a); 대신에 scanf("%d", pa); 사용 가능

sizeof 연산자로 포인터 크기를 확인하면 가리키는 자료형과 관계없이
크기가 같다 -> 환경에 따라 다른데 보통 4바이트 또는 8바이트

포인터 변수의 자료형과 가리킬려는 변수의 자료형은 일치해야함
int a; -> int *pa;  double b; -> double *pb;

# 포인터 기초

## 포인터 상수 표현 (const) 1

```
int a, b;
const int *pa = &a;

*pa = 20;     // 에러

pa = &b;      // 가능
```

pa가 가리키는 변수 a는 pa를 간접참조하여 변경할 수 없다는 것을 의미

단, 다른 변수를 다시 가리키는 것은 가능

# 포인터 기초

## 포인터 상수 표현 (const) 1

```
int a, b;
int *const pa = &a;

pa = &b;      // 에러

*pa = 20;     // 가능
```

pa값을 변경할 수 없다는 것을 의미
(다른 변수를 가리킬 수 없음)

단, 가리키는 변수의 값을 간접 참조하여
변경할 수는 있음

# 문제 해결

## 백준 단계별로 풀어보기 4단계 (https://www.acmicpc.net/step)

## "1차원 배열" 1번~4번

| 단계 | 제목 | 설명 |
|---|---|---|
| 1 | 입출력과 사칙연산 | 입력, 출력과 사칙연산을 연습해 봅시다. Hello World! |
| 2 | 조건문 | if 등의 조건문을 사용해 봅시다. |
| 3 | 반복문 | for, while 등의 반복문을 사용해 봅시다. |
| 4 | 1차원 배열 | 배열을 사용해 봅시다. |
| 5 | 문자열 | 문자열을 다루는 문제들을 해결해 봅시다. |
| 6 | 심화 1 | 지금까지의 프로그래밍 문법으로 더 어려운 문제들을 풀어봅시다. |
| 7 | 2차원 배열 | 배열 안에 배열이 있다면 어떨까요? 2차원 배열을 만들어 봅시다. |

## 해결 못한 문제는 다음 멘토링 시간 전까지 해오기

끝