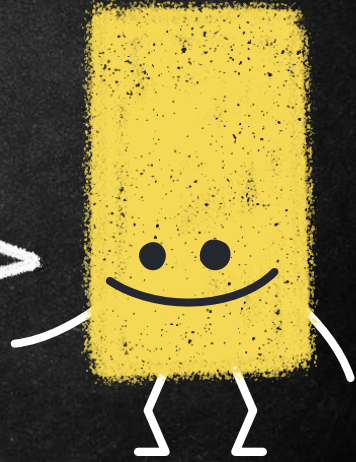
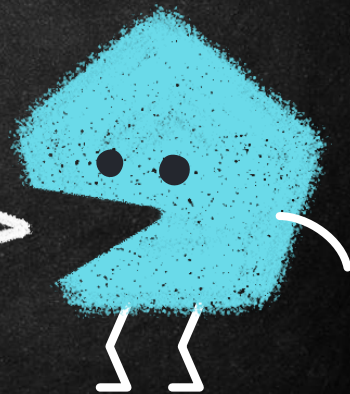
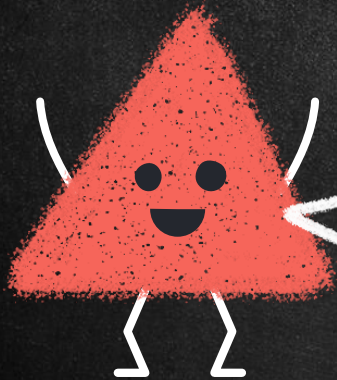


ALGORITHM  
STUDY FOR 2021  
WEEK 0



1.

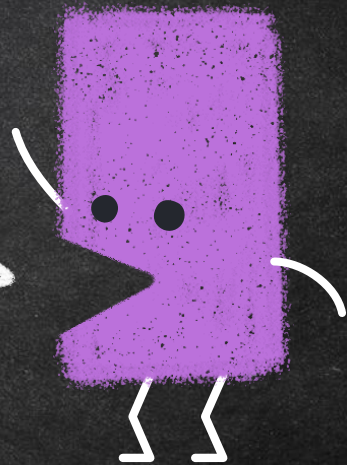
C++ REVIEW !!!





“

이번 스터디는 C++ 언어로  
진행이 됩니다. 자료구조 &  
알고리즘을 학습하기에 앞서  
중요한 내용들을 리뷰해봅시다!



# C++ REVIEW

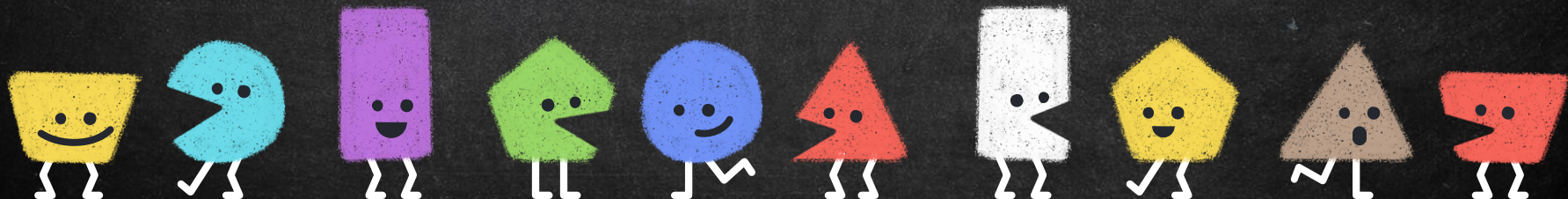






# POINTER

포인터가 무엇인지, 함수의 매개변수에 변수 값  
전달과 주소 값 전달의 차이를 정리합니다!



# POINTER

- ‘주소를 가리키는 것’을 의미함.
- &(주소 연산자) \*(역참조 연산자)
- 한 번 확인해보자!

`int a = 10;`

a라는 이름의 메모리 공간을 할당  
해당 메모리에 숫자 10 입력!

`int* p = &a;`

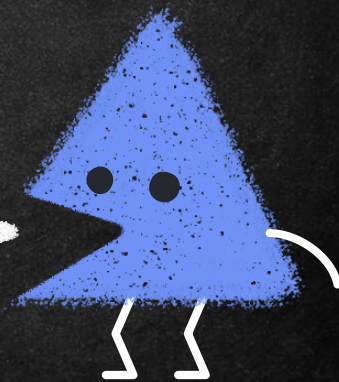
p라는 이름의 메모리 공간을 할당  
해당 메모리에 a의 주소값을 입력!

```
a: 10
&a: 0115FD70
p: 0115FD70
*p: 10
&p: 0115FD64
```

```
int a = 10;
int* p = &a;

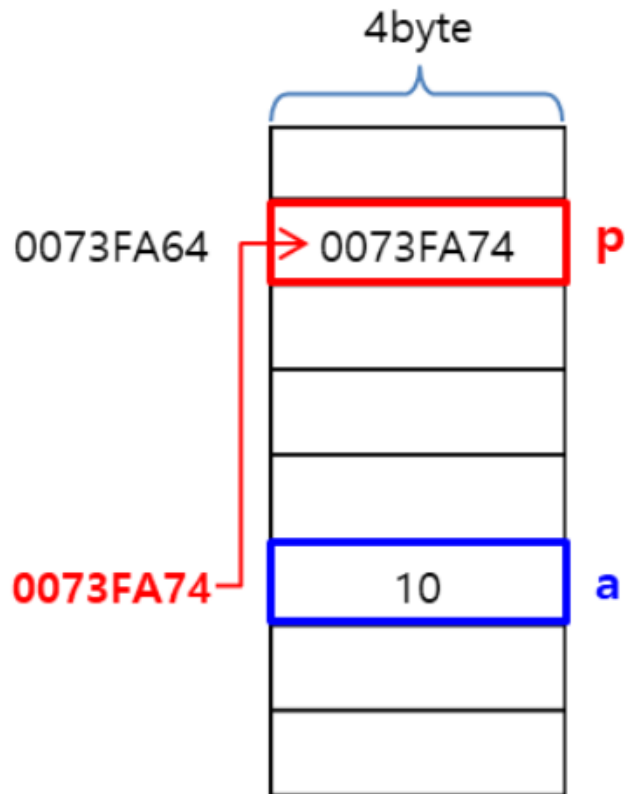
cout << "a: " << a << endl;
cout << "&a: " << &a << endl;
cout << "p: " << p << endl;
cout << "*p: " << *p << endl;
cout << "&p: " << &p << endl;

return 0;
```

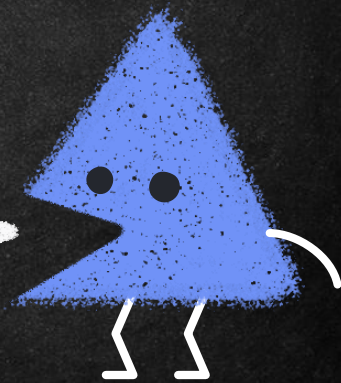




## 메모리 할당 구조!

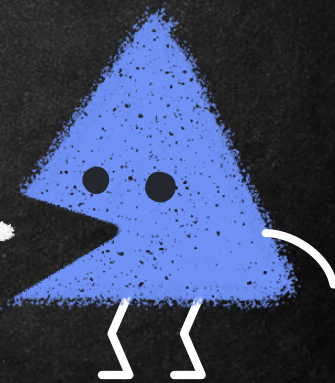


여기서 중요한 점은 **역참조**  
**연산자 \***를 통해 포인터  
변수가 가리키는 주소의 값을  
알 수 있다는 것이다!  
(전 슬라이드 참고할 것!!!)



## CALL BY REFERENCE VS VALUE

- 포인터의 역할에 대해 학습했다!
- 함수의 호출 방식에 대해 다음 Reference를 통해서 정리해보자!
- <https://codingplus.tistory.com/29>





```

1  #include <iostream>
2  using namespace std;
3
4  void callByVal(int a)
5  {
6      a = 20;
7  }
8
9  void callByRef(int* b)
10 {
11     *b = 20;
12 }
13
14 int main()
15 {
16     int a(10);
17     int b(10);
18
19     callByVal(a);
20     callByRef(&b);
21
22     cout << "a: " << a << endl;
23     cout << "b: " << b << endl;
24
25     return 0;
26 }

```

```

a: 10
b: 20

```

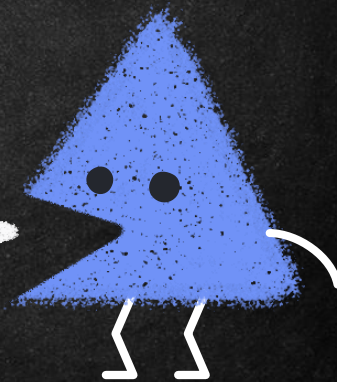
# 차이에 대한 인식!

## CallByVal 함수

-> a의 값을 받아서 메모리에 새로운 공간을 할당, 받은 값 10을 입력.  
새로운 메모리 공간의 10을 20으로 변환!

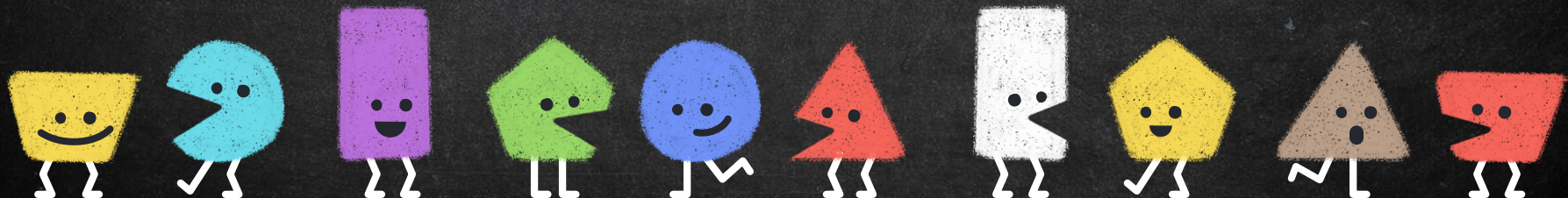
## CallByRef 함수

-> b의 주소값을 받아서 메모리에 새로운 공간 할당, b의 주소값을 입력.  
주소값이 가리키는 b변수를 찾아 20으로 변환!





배열이 무엇인지, 정적 배열과 동적 배열에 대해서  
정리합니다!





# ARRAY & POINTER

- 배열은 사실 포인터와 같은 역할!
- 포인터는 주소를 담는 변수!
- 즉, 배열 변수 또한 포인터와 동일하게 배열의 첫 index 주소를 가지고 있다!

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int arr[5] = { 1,2,3,4,5 };
6
7      cout << "arr[0]: " << arr[0] << endl;
8      cout << "arr: " << arr << endl;
9
10     return 0;
11 }

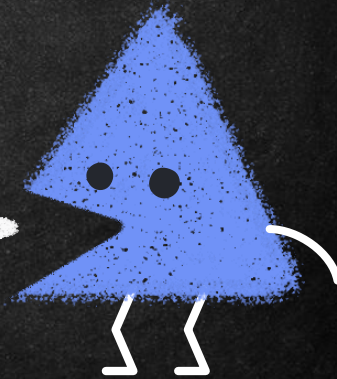
```

	0x0000	0x0004	0x0008	0x000C	0x0010
arr	1	2	3	4	5
	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]

```

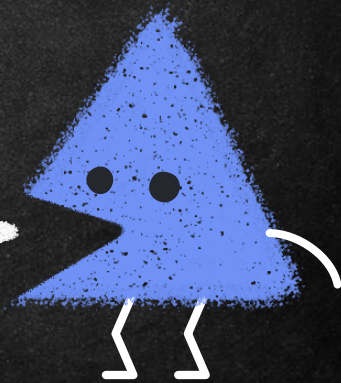
arr[0]: 1
arr: 0133F954

```



## 정적 배열 vs 동적 배열

- 처음에 배열의 크기를 지정하고 선언하는 정적 배열(Static Array)
- 배열의 크기를 유동적으로 조절할 수 있는 동적 배열(Dynamic Array)





```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int arr[3] = { 1,2,3 };
6
7      return 0;
8  }
9

```

## 정적 배열!

원래는 위에 사진처럼 그냥 크기를 지정해준 상태로 배열을 생성함.

```

#include <iostream>

using namespace std;

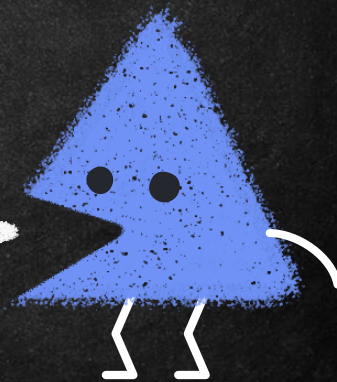
int main(){
    const int length = 5;
    int ary[length];

    return 0;
}

```

### <조심할 것>

아래 사진처럼 배열의 크기를 선언하는 변수 length를 만들 때, 이 값은 constant로 만들어주어야 한다!!!



```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int size;
6
7      cout << "size: ";
8      cin >> size;
9
10     int* dyary;
11     dyary = new int[size];
12     // 동적 배열 생성함.
13
14     // 입력 받기
15     for (int i = 0; i < size; i++)
16     {
17         cin >> dyary[i];
18     }
19     //배열 삭제하기
20     delete[] dyary;
21
22     return 0;
23 }

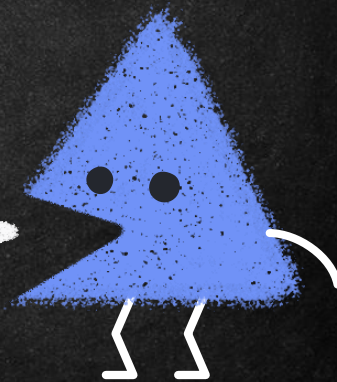
```

## 동적 배열!


사용자로부터 int형의 size를  
입력 받고,

For문을 통해서 동적으로  
생성된 배열에 값을 할당함.

마지막에 delete를 통해서  
삭제하는 것 잊지 말기!

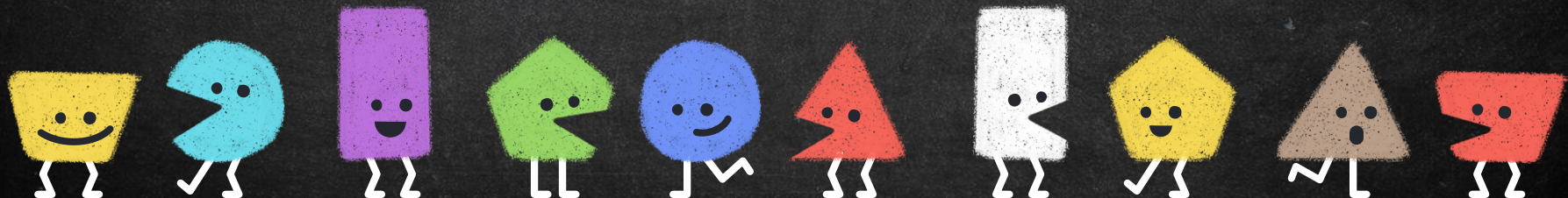






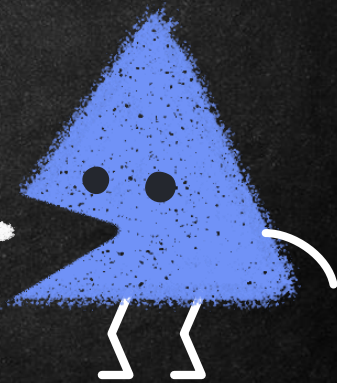
# STRUCT

구조체가 무엇인지, 사용법에 대해서 정리합니다!



# STRUCT

- 구조체는 여러 데이터 타입들을 한데 묶어 사용할 수 있는 일종의 데이터 묶음!
- 하나의 주제에 맞는 여러 데이터 타입을 묶어 한데 사용 가능해 편리!
- Linked List 구현 시 활용되므로 알아두기!
- **Code**와 함께 살펴봅시다!





```

5 struct Princess
6 {
7     string name;
8     string father;
9     string birthday = "알 수 없음";
10 }Goryeo[2];
11
12 int main() {
13     Princess jungmyung;
14     jungmyung.name = "정명공주";
15     jungmyung.father = "조선 선조";
16     jungmyung.birthday = "1603년 6월 27일";
17
18     Goryeo[0].name = "선정왕후";
19     Goryeo[0].father = "고려 성종";
20     Goryeo[1].name = "효정공주";
21     Goryeo[1].father = "고려 현종";

```

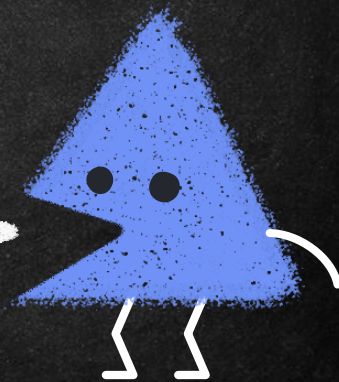
## 구조체 사용하기!

Princess라는 구조체 생성!

데이터로 이름, 아버지, 생일을  
가지고 있음! (type: string)

(11번 라인처럼 구조체를  
배열로 사용할 수 있게 미리  
지정 가능!)

Main함수에서 정명이라는  
구조체 생성 후 데이터 할당!  
고려라는 배열 구조체 크기에  
맞게 데이터 할당!

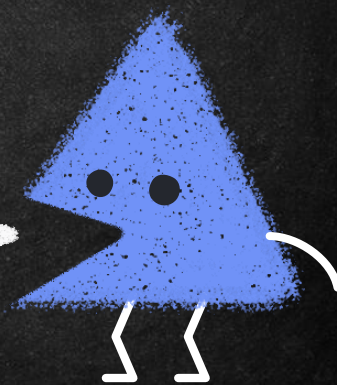


## 출력 결과 확인!

```
cout << " == 조선 공주 == " << endl;
cout << jungmyung.name << endl;
cout << jungmyung.father << endl;
cout << jungmyung.birthday << endl;

cout << " == 고려 공주 == " << endl;
cout << Goryeo[0].name << endl;
cout << Goryeo[0].father << endl;
cout << Goryeo[0].birthday << endl;
cout << Goryeo[1].name << endl;
cout << Goryeo[1].father << endl;
cout << Goryeo[1].birthday << endl;
```

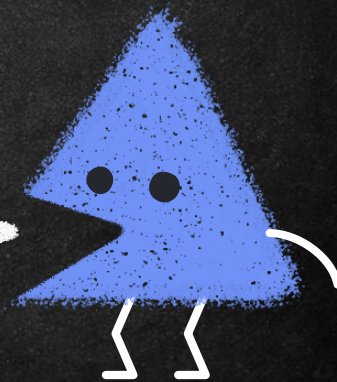
```
== 조선 공주 ==
정명공주
조선 선조
1603년 6월 27일
== 고려 공주 ==
선정왕후
고려 성종
알 수 없음
요정공주
고려 현종
알 수 없음
```





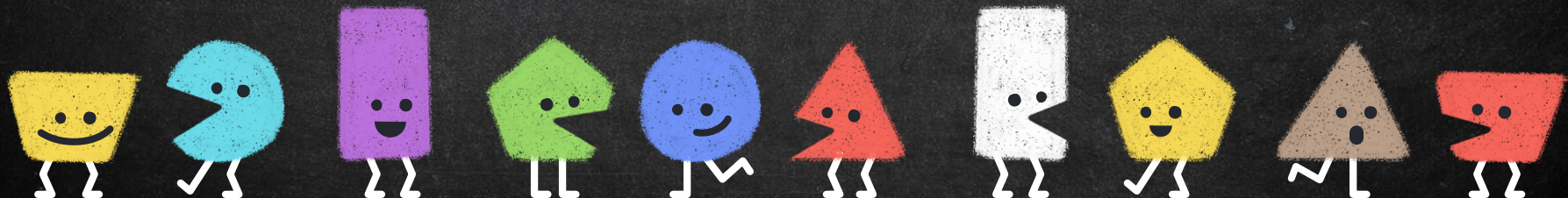
```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  struct Princess
6  {
7      string name;
8      string father;
9      string birthday;
10 };
11
12 void print(Princess* who)
13 {
14     cout << "jungso.name = " << who->name << endl;
15     cout << "jungso.father = " << who->father << endl;
16     cout << "jungso.birthday = " << who->birthday << endl;
17 }
18
19 int main() {
20     Princess jungso;
21     jungso.name = "정소공주";
22     jungso.father = "조선 태종";
23     jungso.birthday = "1412년";
24
25     print(&jungso);
26     return 0;
27 }
```

```
jungso.name = 정소공주
jungso.father = 조선 태종
jungso.birthday = 1412년
```





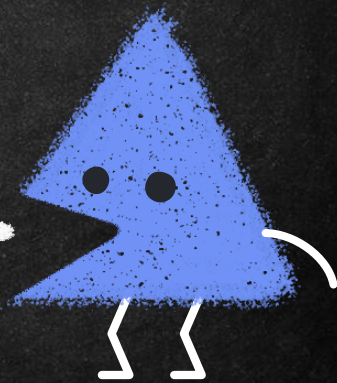
클래스가 무엇인지, 상속에  
대해서 정리합니다!





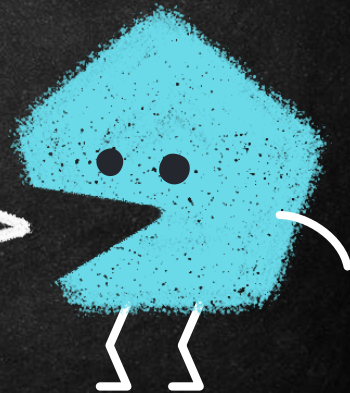
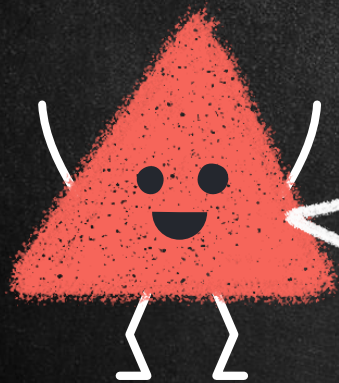
# CLASS

- 클래스는 매우 중요합니다!
- 객체지향에 관한 내용은 코드 몇 줄로 서술이 되지 않습니다.
- 다음 Reference를 통해서 이해해봅시다!
- <https://wikidocs.net/16468>
- 왼쪽에 (클래스의 기본부터 쪽 읽어보시면 많은 도움이 됩니다!!!)



2.

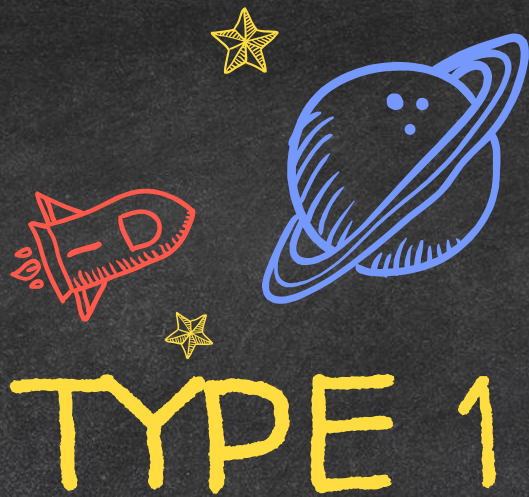
INPUTS → OUTPUTS!!!



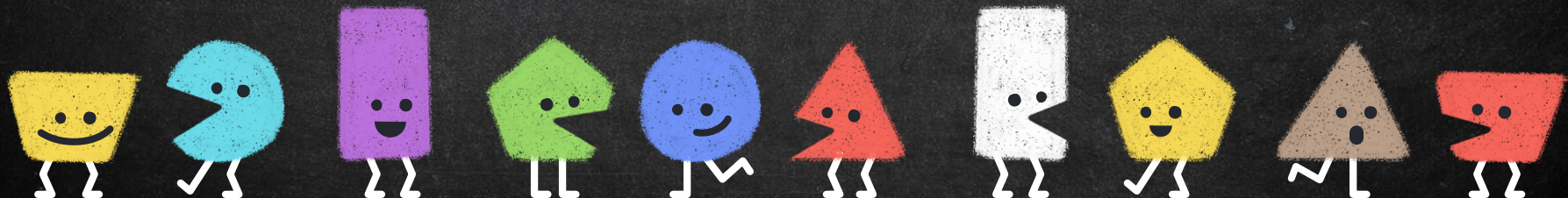


# INPUTS → OUTPUTS





입력 데이터의 개수가 주어진 경우에 대해  
정리하고, 문제를 풀어봅시다!





## 문제

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

## 입력

첫째 줄에 A와 B가 주어진다. ( $0 < A, B < 10$ )

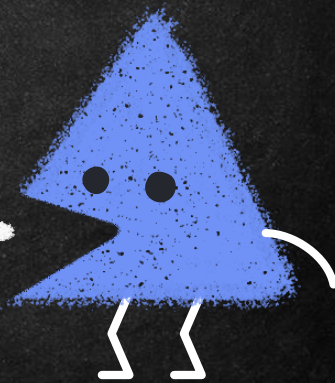
## 출력

첫째 줄에 A+B를 출력한다.

# 백준 1000번 !

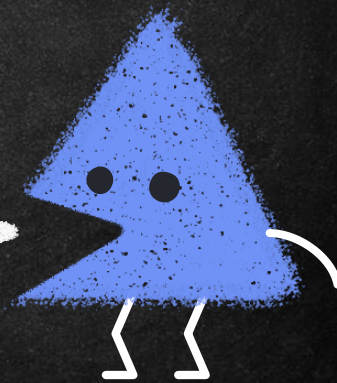
사용자로부터 몇 개의  
데이터를 입력 받아야  
하는지가 명시되어 있다.

문제에 제시된  
입력데이터만큼의 변수를  
생성하고,  
입력을 받은 후에 문제를 풀면  
된다!




# SOURCE CODE

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int a, b;
6      cin >> a >> b;
7      cout << a + b << endl;
8
9      return 0;
10 }
```

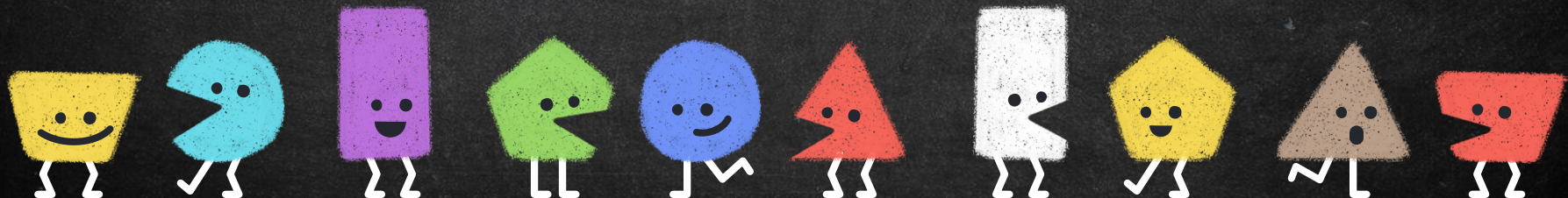






# TYPE 2

테스트 케이스의 개수가 주어진 경우에 대해  
정리하고, 문제를 풀어봅시다!



## 문제

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

## 입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다.

각 테스트 케이스는 한 줄로 이루어져 있으며, 각 줄에 A와 B가 주어진다. (0

## 출력

각 테스트 케이스마다 A+B를 출력한다.

## 예제 입력 1 복사

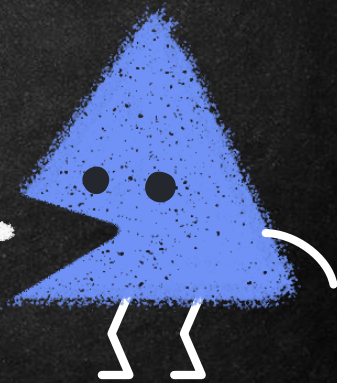
```
5
1 1
2 3
3 4
9 8
5 2
```

# 백준 10950번 !

사진처럼 테스트 케이스가  
주어진다. 예제에는 5개의  
테스트 케이스가 주어졌다.

핵심은 먼저 테스트 케이스의  
개수를 입력 받은 후에,

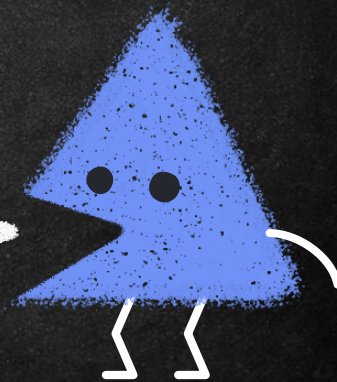
While문을 이용하여 테스트  
케이스의 개수를 줄여가며 각  
테스트 케이스의 정답을  
출력하는 것이다!

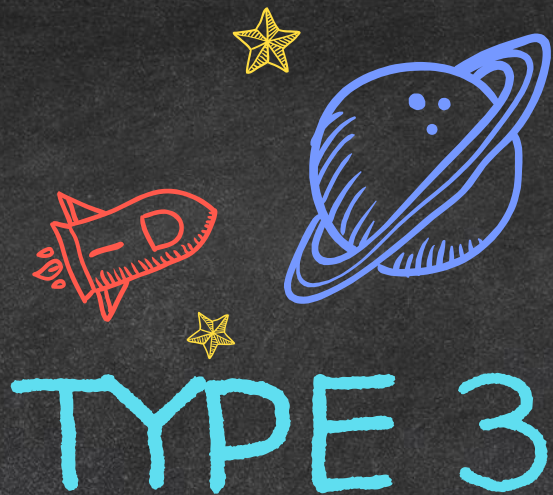




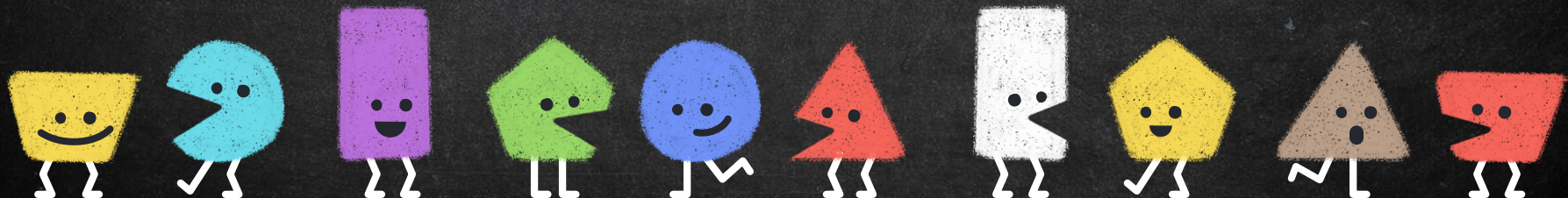
# SOURCE CODE

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int t;
6      // t는 테스트 케이스의 개수
7      cin >> t;
8      while (t--)
9      {
10         int a, b;
11         cin >> a >> b;
12         cout << a + b << endl;
13     }
14     return 0;
15 }
```





테스트 케이스의 개수를 모르는 경우에 대해  
정리하고, 문제를 풀어봅시다!





## 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.

## 출력

각 테스트 케이스마다 A+B를 출력한다.

## 예제 입력 1 복사

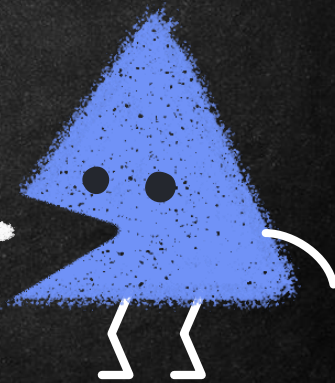
```
1 1
2 3
3 4
9 8
5 2
```

# 백준 10951번 !

사진처럼 테스트 케이스가  
여러 개의 ~라고 주어진다.

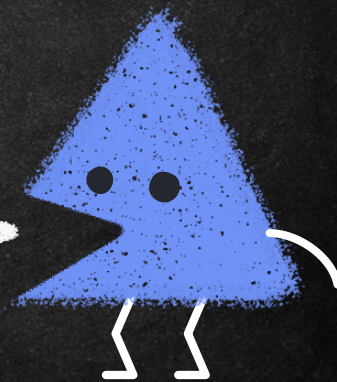
핵심은 주어지는 예시들을  
EOF까지 반복문을 활용하여  
값을 내는 것이다!

보통 다음과 같은 방식으로  
EOF까지 반복문을 사용한다!

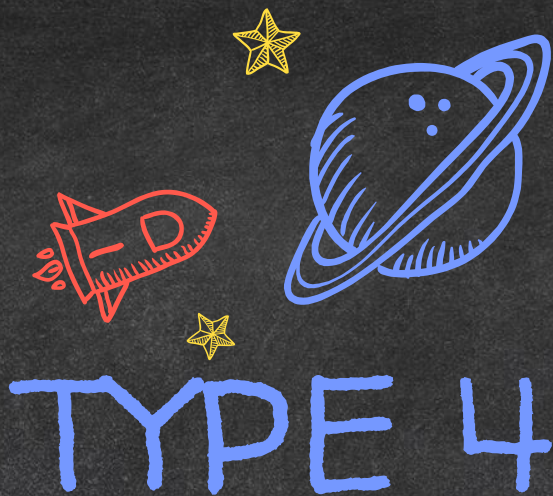


# SOURCE CODE

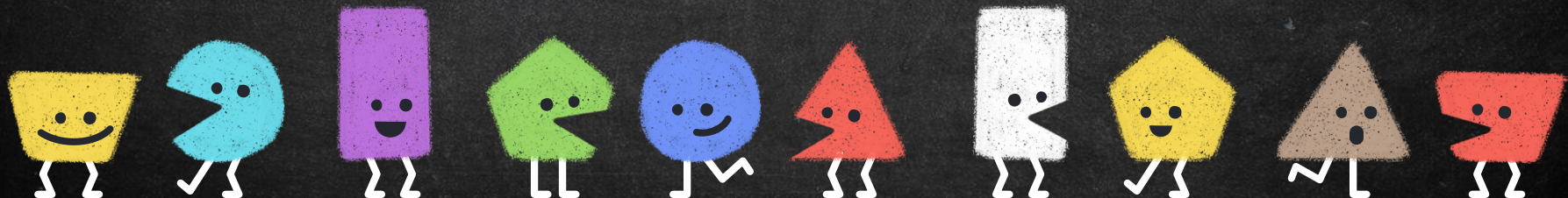
```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int a, b;
6      while (cin >> a >> b)
7      {
8          cout << a + b << endl;
9      }
10     return 0;
11 }
```







입력을 한 줄씩 받아야 하는 경우에 대해 정리하고,  
문제를 풀어봅시다!



# 백준 11718번 !

## 문제

입력 받은 대로 출력하는 프로그램을 작성하시오.

## 입력

입력이 주어진다. 입력은 최대 100줄로 이루어져 있고, 알파벳 소문자, 대문자, 공백, 숫자로만 이루어져 있다. 각 줄은 100글자를 넘지 않으며, 빈 줄은 주어지지 않는다. 또, 각 줄은 공백으로 시작하지 않고, 공백으로 끝나지 않는다.

## 출력

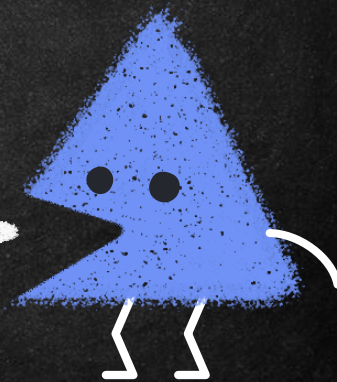
입력받은 그대로 출력한다.

### 예제 입력 1 복사

```
Hello  
Baekjoon  
Online Judge
```

### 예제 출력 1 복사

```
Hello  
Baekjoon  
Online Judge
```





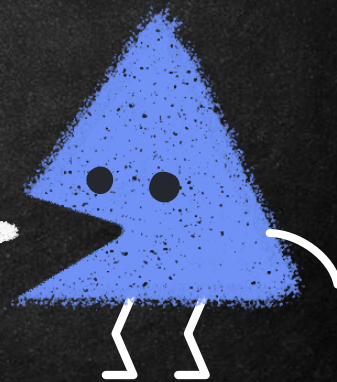
```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main(void)
6  {
7      string str;
8      while (true)
9      {
10         getline(cin, str);
11         if (str == "")
12             break;
13
14         cout << str << endl;
15     }
16     return 0;
17 }
18 }
```

## SOURCE CODE

한 줄씩 입력 받는 문제에서는  
getline 메소드를 이용한다.

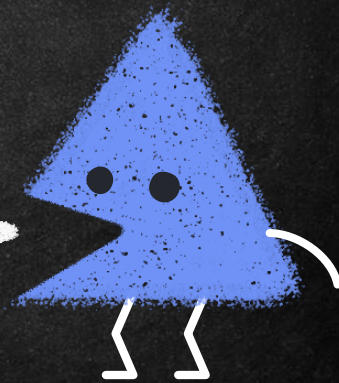
중간에 if 조건문을 활용해서  
입력받는 문자열이 없을 때에  
멈추도록 제어하고,

Getline을 이용하여 문자열  
변수에 사용자로부터  
입력받는다!



# GETLINE~~~

- Getline 메소드에 익숙하지 않으면 다음 reference를 참고한다!
- <https://www.cplusplus.com/reference/string/string/getline/?kw=getline>
- 모르는 게 있으면 여기에서 찾아보자!
- <https://www.cplusplus.com>





THANK YOU

