배열생성 방법

목적

```
// 동일한 데이터 타입의 변수가 여러개 필요할때

// 배열을 생성 방법
// 1. stack에 할당하는 방법(지역 변수)
//    1-1. 일단은 배열의 데이터 타입(int 같은)을 적는다.
//    1-2. 배열의 이름이 될 변수명을 적는다.
//    1-3. 배열임을 알리기 위해 []을 변수 옆에 적어준다.
//    1-4. 필요하다면 배열의 값들을 초기화한다.
//         (이때 원소로 지정한 숫자에 따라 배열의 길이가 지정된다)
//    * 가변으로 구성하고 싶다면 new를 사용한다.
```

```
//============for 또는 while 을 통해 배열을 하면 편리하다==========
```

배열 생성 ※주의 숫자 1의 배열 위치는 0이다. 배열 시작 번호는 0번

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

5회 반복

```
for (int i = 0; i < 5; i++) {
    // System.out.printf("arr[%d] = %d\n", i, arr[i]);
    System.out.println(arr[i]);
}
int i = 0;

while (i < 5) {
    System.out.println(arr[i]);
    i++;
}
```

출력값

```
1
2
3
4
5
```

스위치문 생성방법

```
5           Boolean isTrue = true;
6           //참 혹은 거짓 선언 규칙
7           int num;
8
9           Scanner sc = new Scanner(System.in);
10
11          while (isTrue) {
12
13              System.out.println("저희 상점에 방문해주셔서 감사합니다. 물건을 골라주세요");
14              num = sc.nextInt();
15              switch (num){
16                  case 0:
17                      System.out.println("탈출합니다.");
18                      isTrue = false;
19                      break;
20                  case 1:
21                      System.out.println("비누를 장바구니에 담았습니다");
22                      break;
23                  case 2:
24                      System.out.println("신발을 장바구니에 담았습니다");
25                      break;
26                  case 3:
27                      System.out.println("에어팟을 장바구니에 담았습니다");
28                      break;
29                  default:
30                      System.out.println("프로그램을 종료합니다.");
31                      isTrue = false;
32                      break;
33
```

반복문 실행

입력 받는 값

입력 값에 따라 출력

break 설정하지 않을시 다음 case도 출력한다.

default는 예상치 못한 값에 출력 한다.

출력값

```
저희 상점에 방문해주셔서 감사합니다. 물건을 골라주세요
1
비누를 장바구니에 담았습니다
저희 상점에 방문해주셔서 감사합니다. 물건을 골라주세요
11
프로그램을 종료합니다.
```

Challenge1 문제 풀이

```java
public static void main(String[] args) {
    int bigFrontCnt = 0, smallFrontcut = 0;
    // A가 500개
    // B가 30개
    // if (case A || case B)
    // 합격 케이스 500 + 불합격 500 + 추가검사 500개
    // if (case B || case A)
    // 합격 케이스 30개 + 불합격 970개 + 추가검사 970개

    // 1 ~ 1000까지의 숫자중 2의 배수는 A
    // 1 ~ 1000까지의 숫자중 33의 배수는 B
    for (int i = 0; i <= 1000; i++) {
        if (((++bigFrontCnt != 0) && (i % 2 == 0)) ||
                ((++bigFrontCnt != 0) && (i % 33 == 0))) {
            ; //아무것도 안한다는 뜻
            if (((++smallFrontcut != 0) && (i % 33 == 0)) ||
                    ((++smallFrontcut != 0) && (i % 2 == 0))) {
                ;
            }
            System.out.println("케이스수가 많은 A가 앞 에있는 경우" + bigFrontCnt);
            System.out.println("케이스수가 적은 B가 앞 에있는 경우" + smallFrontcut);
        }
    }
}
```

고정값

조건

데이터1001개

A가 앞에 있는조건

B가 앞에 있는조건

OR연산은 앞에 조건을 만족하면 뒤에 조건은 생략된다.

2^6~2^9 비트 출력

문제 조건 : 1000개의 데이터가 있다. A가 500개 ,B가 30개 있고 둘 중 하나의 케이스를 판정하고자 한다. 어떻게 하면 가장 효율적으로 이 케이스들을 찾아낼 수 있을까 고민해보자

답 :

OR연산자는 앞에 많은 케이스가 오는게 적게 연산한다.

AND연산자는 앞에 적은 케이스가 오는게 적게 연산한다.

출력값

```
케이스수가 많은 A가 앞 에있는 경우1501
케이스수가 적은 B가 앞 에있는 경우1971
```

실수로 반복값을 1001개로 설정해서 1개 더나왔다.

Challenge2 문제 풀이

문제 조건 : 5, 6, 7, 8, 9, 10이 모두 출력되어야 하고 7, 8, 9, 10이 출력되어야 한다.
증복이 발생하면 안됨.

개념 포인트 : int는 4바이트 이다. 4byteX8bit/1byte = 32개의 bit 칸이 있다
출력하는 변수는 각각 6개와 4개가 있다. 32개 공간을 순서대로 사용하자고 생각하면 편하다.

답 :

FIRST_BIAS의 값이 5인 이유가 문제의 답과 연결되있다고 생각한다.
FIRST의 값을 출력하는 범위는 5~10이다. 즉 2^5~2^10에 저장될것이다.
하지만 쉬프트 연산자로 -5를 시켜 2^0~2^5까지 값을 이동 시켜 저장시켜 증복되는 Second 값이 들어갈 자리를 만들었다.
Second 값또한 -1 쉬프연산을 통해 비트에 빈공간없이 채운다.
출력값에 bit 합계를 보면 알수있다.

```java
final int FIRST_BIAS = 5;
final int SECOND_BIAS = 1;
final int FIRST_RANGE = 6;
final int FIRST_OFFSET = 5;
final int SECOND_RANGE = 4;
final int SECOND_OFFSET = 7;
final int BIN = 1;
int testBit = 0;
int randNum;

for (int i = 0; i < FIRST_RANGE; i++) {
    randNum = (int)(Math.random() * FIRST_RANGE + FIRST_OFFSET);

    while ((testBit & (BIN << (randNum - FIRST_BIAS))) != 0) {
        randNum = (int)(Math.random() * FIRST_RANGE + FIRST_OFFSET);
    }
    System.out.printf("5 ~ 10 randNum = %d\n", randNum);
    testBit |= (BIN << (randNum - FIRST_BIAS));
}

System.out.println("testBit = " + testBit);
for (int i = 0; i < SECOND_RANGE; i++) {
    randNum = (int)(Math.random() * SECOND_RANGE + SECOND_OFFSET);

    while ((testBit & (BIN << (randNum - SECOND_BIAS))) != 0) {
        randNum = (int)(Math.random() * SECOND_RANGE + SECOND_OFFSET);
    }
    System.out.printf("7 ~ 10 randNum = %d\n", randNum);
    testBit |= (BIN << (randNum - SECOND_BIAS));
}
System.out.println("testBit = " + testBit);
```

고정값

변수 지정

6회 출력
랜덤 값 출력

2^0~2^5 비트 출력

4회 출력
랜덤 값 출력

2^6~2^9 비트 출력

출력값

```
5 ~ 10 randNum = 9
5 ~ 10 randNum = 10
5 ~ 10 randNum = 6
5 ~ 10 randNum = 8
5 ~ 10 randNum = 5
5 ~ 10 randNum = 7
testBit = 63
7 ~ 10 randNum = 9
7 ~ 10 randNum = 7
7 ~ 10 randNum = 8
7 ~ 10 randNum = 10
testBit = 1023
```

1번째 범위2^0~2^5 = 63
2번째 범위2^6~2^9 = 960
63 + 960 = 1023