(디지털컨버전스) 스마트 콘텐츠와 웹 융합 응용SW개발자 양성과정

2021년 5월 14일
[6일차 복습]

- 수강생 : 김 민 규
- 강의장 : 강 남 C
- 수강 기간 : 2021. 05. 07 ~ 2021. 12. 08
- 수강 시간 : 15:30 ~ 22:00
- 이상훈 강사님 | 이은정 취업담임

▶ 내용 : 비트연산자를 이용한 중복회피 문제

```java
public class a_비트연산자 {
    //비트연산자를 이용한 중복회피 문제
    public static void main(String[] args) {
        final int Bin = 1;
        int testBit = 0;
        int randNum;

        for(int i = 0; i < 10; i++) {
            randNum =(int)(Math.random() * 10);

            //i는 돌아가는 횟수. 즉 0~9까지 총 10회를 돌리게됨. 랜덤수
            // 0~9 까지를 랜덤으로 10회출력

            while ((testBit & (Bin << randNum)) !=0) {
                randNum = (int) (Math.random() * 10);

                // 0이므로 while문은 돌지 않는다 > or 관계연산자에서 입력되지 않은 2^n이 0~9까지 중복되지않되게 입력된다
                // souf를 통해 randNum의 값이 출력된다.
            }
            System.out.printf("randNum = %d\n", randNum);

            testBit |= (Bin << randNum);
        }
    }
}

```

주요 내용

-
testBit & (Bin 〈〈 randNum)) != 0
0이 아니라면 계속 loop
= 0일때까지 계속 loop
중복된 변수값을 확인하며,
0~9까지 모두 출력되었을 경우 0 값이
나오면서 while loop가 종료됨

testBit |= (Bin 〈〈 randNum)
계속 randNum 값을 출력하여
그 중 중복없는 값을 testBit에 입력
0~9까지 모두 출력되면 while loop가
종료됨
--

▶ 내용 : 변수 스코핑

```java
public class b_변수스코핑 {
    public static void main(String[] args) {
        System.out.println("안녕");
    }

    // i 변수는 for 문 내부에서만 사용할 수 있으므로 아래와 같이 사용이 불가하다!
    // System.out.println("i = " + i);

    int j;

        //for (j = 0; j < 3; j++) {
        //System.out.println("하이");

    }
```

주요 내용

스코핑(스코프 = 유효범위)
변수를 사용할 수 있는 범위?구간?
괄호를 벗어난 구간은 변수 사용이 불가
--
즉 for문 안에서 선언된 i의 값은 for문의 중괄호 값 안에서만 print가능하고
중괄호 밖에서는 오류가 발생하게됨.

중괄호 안에서만 유효한 값
--
사용되는 시점에서의 유효범위를 사용하는 것이 아니라
정의된 시점에서의 유효범위를 사용하는 것

정적 스코프(static scope)
렉시컬 스코프(lexical scope)라고 불림
--
지역변수 : 지역변수는 중괄호({}) 안에서 선언된 변수를 말함.
지역변수는 선언된 지역 내에서만 유효

전역변수 : 전역변수는 특정 지역(중괄호) 밖에서 선언된 변수를 말함

전역변수는 언뜻 편히보일 수 있지만,
프로그램구조의 복잡성과 메모리차지로
신중히 사용하는 것이 좋음.

▶ 내용 : 배열없는 중복회피 문제

```java
public class c_배열없는중복회피문제 {
    public static void main(String[] args) {
        final int Bin = 1;
        int testBit = 0;
        int randNum;

        for (int i = 0; i < 6; i++) {
            randNum = (int) (Math.random() * 6 + 5);

            //-5는 2^5 - 5라 2^0부터 시작하여 순서대로 사용하게함.(비트 32개있으므로 최대한의 효율을 위한방법)
            // bit를 'randNum - 5'만큼 왼쪽으로 이동시킨다
            while ((testBit & (Bin << (randNum - 5))) != 0) {
                randNum = (int) (Math.random() * 6 + 5);
            }

            System.out.printf("5~10 무작위: %d\n", randNum);
            testBit |= (Bin << (randNum - 5));
        }
        for (int i = 0; i < 4; i++) {
            randNum = (int) (Math.random() * 4 + 7);

            //-1는 2^7 - 1라 2^6부터 시작하여 순서대로 사용하게(비트 32개있으므로 최대한의 효율을 위한방법)
            while ((testBit & (Bin << randNum - 1)) != 0) {
                randNum = (int) (Math.random() * 4 + 7);
            }

            System.out.printf("7~10 무작위: %d\n", randNum);
            testBit |= (Bin << (randNum - 1));
        }

    }
}
```

주요 내용

--

For (int i = 0; i<6; i++) {
0~5까지 출력, 즉 6회 출력
randNum = (int)(Math.random() *6+5)
0+5 ~ 5+5 = 5 ~10까지의 수 랜덤출력

--

while((testBit&(Bin<<(randNum-5)!=0)
randNum의 비트를 -5만큼 이동시킨다.
즉, 5부터 시작되는 randNum의 비트를
0부터 시작하게 만들어 0~32까지 순서대로
프로그램을 효율적으로 운영하게 만듦

--

while((testBit&(Bin<<(randNum-1)!=0)
randNum의 비트를 -1만큼 이동시킨다.
즉, 2^7부터 시작되는 비트를 6부터
시작하게 만든다.

--

▶ 내용 : 쉬프트연산 비트변동

```java
public class d_쉬프트연산비트변동 {
        public static void main(String[] args) {
            System.out.println("쉬프트 연산에 사용되는 int 데이터 타입은 32비트임을 잊지맙시다!");

            int num = 14;
            int shiftBit = 4;

            System.out.printf("%d << %d = %d\n", num, shiftBit, num << shiftBit);

            // 14 ===> 1110
            // 14 << 4
            // 0000 1110
            // 1110 0000 ===> 224
            // 4칸 이동( 14를 2진법으로 바꾼 후 왼쪽으로 4칸 이동 shiftBit)
        }
    }
```

주요 내용

--
Num 〈〈 shftBit는
Num(14)를 2진법으로 바꾼 후
왼쪽으로 shiftBit(4)칸 이동시킨다.

14 => 1110
4칸 왼쪽으로 이동
1110 0000 => 224
--

▶ 내용 : Switch문과 bolean의 개념

```java
import java.util.Scanner;

public class aa {
    public static void main(String[] args) {
        import java.util.Scanner;
    public class e_switch문bolean개념 {
        public static void main(String[] args) {
            System.out.println("저희 상점에 방문해주셔서 감사합니다. 물건을 골라주세요.");
            Boolean isTrue = true;

            Scanner scan = new Scanner(System.in);
            int num;
            while(isTrue) {
                System.out.print("숫자를 눌러 물건을 담으세요: ");
                num = scan.nextInt();

                switch(num) {

                    case 0:
                        System.out.println("탈출합니다.");
                        isTrue = false;
                        break;

                    case 1:
                        System.out.println("비누를 장바구니에 담았습니다.");
                        break;
                    case 2:
                        System.out.println("신발을 장바구니에 담았습니다.");
                        break;
                    case 3:
                        System.out.println("에어팟을 장바구니에 담았습니다.");
                        break;
                    default:
                        System.out.println("그런건 없습니다!");
                        break;
                    //0,1,2,3 이외의 숫자를 담으면 없음. else와 유사?
                    // default는 말 그대로 기본값으로 예상치 못한 입력시 출력되는 것.
                }
```

주요 내용

--

Boolean
참과 거짓을 표현하는 데이터타입
True,false/1,0

--

Switch
입력된 키보드값에 따라 적절한 처리를 하게 한다.
〉〉 Switch (num) 〉〉 숫자형데이터
Ex) case 0, case 1
0, 1을 입력 시 해당 case를 실행

〉〉 Switch (str) 〉〉 문자열 데이터
Ex) case (“hi”)
Hi를 입력 시 해당 case를 실행

num의 경우 숫자형만 인식가능
또는 홑따옴표를 이용한 문자 낱개 인식가능

즉, switch의 데이터타입과
Case의 데이터타입이 일치해야함.

--

Break는 더 이상 내려가지 않고
해당시점에서 종료할 수 있게 해줌

--

Case에 입력된 숫자, 문자열을 입력안하면
Default로 처리. Else와 유사한것 같음.

▶ 내용 : Continue 활용

```java
public class g_Continue {
    public static void main(String[] args) {
        for(int i = 0; i < 10; i++) {
            if( i % 2 == 0) {
                continue;
                //continue로 인하여 아래 진행해야할 코드가 있더라도
                // 무조건 for loop의 최상단으로 이동하게됨.

                // i가 2의배수일 때 증감식이 계속 진행되게됨?
                // 2의배수는 왜 출력이 안되었는지?
                // 출력되는 부분이 if조건식은 제외되고 출력되는 것인지?
            }
            System.out.println("i = " + i);
        }
    }
}

```

주요 내용

--
If문에 해당되는 i를 2로 나눠 나머지가 0 일때 탈출을하게 되지만,
Continue로 인하여 최상단으로 이동하여 다음반복이 실행하게됨.

즉 2의배수마다 탈출하여
2의 배수값은 출력되지 않음.

Continue로 인하여 2의 배수를 제외한 수가 모두 출력되는 것
--

▶ 내용 : Quiz 10

```java
public class h_Quiz10Challenge {
    //10. 관계 연산자 문제(Challenge - 질문 노트 포함)
    //1000개의 데이터가 있다.
    //여기서 C에 해당하는 데이터는 30개 있다.
    //F에 해당하는 데이터는 500개 있다.
    //B에 해당하는 데이터는 240개 있다.
    //A에 해당하는 데이터는 700개 있다.
    //D에 해당하는 데이터는 350개 있다.
    //C 혹은 F 둘 중 하나의 케이스를 판정하고자 한다.
    //어떻게 하면 가장 효율적으로 이 케이스들을 찾아낼 수 있을까 고민해보자!
    public static void main(String[] args) {

        int bigFrontCnt = 0, smallFrontCnt = 0;

        for(int i = 1; i <= 1000; i++) {
            if(((++bigFrontCnt != 0) && (i % 2 == 0)) || ((++bigFrontCnt != 0) && (i % 33 == 0))) {
                ;
            }
            if (((++smallFrontCnt != 0) && (i % 33 == 0 )) || ((++smallFrontCnt != 0) && (i % 2 == 0))) {
                ;
            }
        }
        System.out.println("큰놈이 앞에 있을 때: " + bigFrontCnt);
        System.out.println("작은놈이 앞에 있을 때: " + smallFrontCnt);

    }
}
```

주요 내용

--
Or연산자는 앞에 많은 수가
And연산자는 앞에 적은 수가
오는게 효율적

And 연산자
1) 좌변의 값이 true인지 확인
2) True라면 우변은 평가하고 값을 반환
3) False라면 우변을 평가하지 않고 좌변을 반환

따라서 확률이 낮거나, 낮은 수를 앞에 두는것이 효율적

Or연산자
ex) a가 500개 b가 30개 등 총 1000개
A | B
일치 500개, 불일치 500개, 추가검사500개

B | A
일치 30개, 불일치 970개, 추가검사 970개
--
옆의 문제를 살펴보면
For문에서 첫번째 if문
500개 검사(일치) + 500개(불일치) + 500개(재검사)
두번째 if문
30개(일치)+ 970개(불일치)+970개(재검사)

▶ 내용 : Quiz 25

```java
public class i_Quiz25 {
    public static void main(String[] args) {
        //25. 복습 문제
        //1 ~ 100 까지의 숫자중 2의 배수는 모두 더한다.
        //여기서 5의 배수는 모두 뺀다.
        //11의 배수는 더한다.
        //중복이 발생할 경우엔 무시한다.
        //모든 값을 처리한 이후 결과값은 무엇인지 프로그래밍해보자!

        int sum = 0;

        for(int i = 1; i <= 100; i++) {
            if(i % 2 == 0 && i % 5 == 0 && i % 11 == 0) {
                System.out.println("110의 배수 = "+ i);
            } else if (i % 11 == 0 && i % 5 == 0) {
                System.out.println("55의 배수 = " + i);
                // 11과 5의 공배수일 경우만 출력
                // 11의배수 5의배수에서의 55는 해당 조건에서만 출력된다. 아래도 같음.
                // 따라서 공배수가 겹쳐져 출력되지 않음.
            } else if (i % 11 == 0 && i % 2 == 0) {
                System.out.println("22의 배수 = " + i);
            } else if (i % 5 == 0 && i % 2 == 0) {
                System.out.println("10의 배수 = " + i);
            } else if (i % 11 == 0) {
                System.out.println("11의 배수 = " + i);
                sum += i;
            } else if (i % 5 == 0) {
                System.out.println("5의 배수 = " + i);
                sum -= i;
            } else if (i % 2 == 0) {
                System.out.println("2의 배수 = " + i);
                sum += i;

                // sum += i가 아래항목들에만 들어간 이유는?
                // 2, 5, 10의 배수들을 출력

            }
        }
        System.out.println("최종 결과 = " + sum);
```

주요 내용

--
If문에서
110, 55, 22와 같은
2, 5, 11의 공배수들을 걸러낸다.

2, 5, 11 각자 배수들은
따로 출력이된다.

즉 공배수들은 중복으로 출력되지 않도록
&&사용되어 걸러짐.

Sum += i;
2, 10의 각자 배수들의 합 출력

Sum -= i;
5의 배수값을 모두 뺀다.

--

▶ 내용 : 배열

```java
public class j_배열 {
    public static void main(String[] args) {
        int arr[] = {1, 2, 3, 4, 5};
        for(int i = 0; i < 5; i++) {
            // 배열의 인덱스(방)은 0부터 시작. i = 0 ~4까지가 인덱스. 따라서 0~4 : 1~5를 출력하게됨
            // for와 while 문등의 반복문과 혼합구성에 있어 매우 탁월
            System.out.printf("arr[%d] = %d\n", i, arr[i]);

        }
    }
}

//데이터타입의 변수가 많을수록 효율적, 효과적 관리가 가능함.
// 배열을 만드는 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 사용해야 하는데 이것은 다음주에 학습하도록 한다.

```

주요 내용

--
배열에서 위치를 가리키는 숫자 = 인덱스
인덱스는 언제나 0부터 시작하게 됨.

Arr[] = {1, 2, 3, 4};
일 경우 0~3까지의 인덱스를 갖게 됨.
--

▶ 내용 : Quiz 27

```java
import java.util.Scanner;

public class k_Quiz27 {
    //27. 복습 문제(챌린지 문제 - 배열 사용 x)
    //아래와 같은 형태의 숫자 배치가 있다.
    //1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...   피보나치수열
    //사용8을자가 15를 입력하면 15번째 값을,  입력하면 8번째 값을 구하도록 프로그래밍 해보자!
    //(n을 입력하면 n 번째 값을 구하도록 프로그래밍 해보자 ~)
    public static void main(String[] args) throws InterruptedException {

        System.out.println("구하고싶은 n번째 값을 입력하시오");
        Scanner scan = new Scanner(System.in);
        //1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...   피보나치수열

        int a = 0, b = 1;
        int c = 0;
        int n;

        n = scan.nextInt();


        for (int i = 0; i < n ; i++) {
            a = b;
            b = c;
            c = a + b;


        }
        System.out.printf("%d번째 피보나치 수열값은 = %d\n", n, c);
```

결과값

```
구하고싶은 n번째 값을 입력하시오
8
8번째 피보나치 수열값은 = 21

Process finished with exit code 0
```

주요 내용

-

-