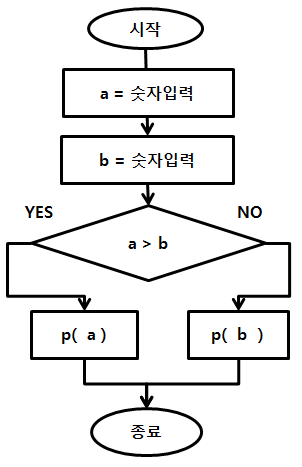
조건문을 사용해서 둘중에 하나 원하는 결과를 출력하는 프로그램을 구현해 보자. 사용자에게 두 수 a, b를 입력 하여 둘 중 큰수를 화면에 출력하는 순서도와 의사 코드이다. a가 크면 a가 출력 되고 b가 크면 b가 출력 된다. 순서도와 의사 코드를 어떻게 구현 하였는지 자세히 확인해 보고 이후 나온 순서도와 코드에 사용해 보자.

의사코드

a=숫자입력

b=숫자입력

if(a>b){

p(a)

}else{

p(b)

}

문제 1. 입력한 숫자가 10보다 큰 수인지 아닌지 출력하는 순서도와 코드를 만들어 보자.

문제 2. 입력 받은 숫자가 양수인지 0인지 음수인지 판단하는 순서도와 코드를 만들어 보자.

문제 3. 입력받은 3개의 숫자 중 가장 큰 수를 출력하는 순서도와 코드를 구현하시오.

%연산의 사용 용도

1.어떤 숫자가 배수인지 아닌지 확인 할 때 사용한다.

100이라는 숫자가 5의 배수라면 100을 5로 나눈 나머지는 0이된다. 5의 배수란 5을 곱해서 나온수 이기 때문에 어떤수 a가 5의 배수라면 a를 5로 나눈 나머지가 0이면 어떤수 a는 5의 변수이다. a%5==0 이라면 a는 5의 배수이다.

2. 어떤 숫자가 배수인지 아닌지 확인 할 때 사용한다.

어떤 숫자가 2로 나누어지면 짝수이다. 2,4,6,8…. 모두 2로 나눠진다. a%2==0이라면 a는 짝수 이다.

3. 잔돈 계산 할때 사용한다.

5700원를 500원짜리로 잔돈을 거슬러 주고 싶다면 5700원을 500으로 나눈 몫이 결과가 된다. 5700/500==11이므로 5700은 500원 11개를 거슬러주고 200원이 남는다. 이때 200원은 5700원을 500원으로 나눈 나머지이므로 5700%500==200과 같다.

4. 초를 시분초로 변경할 때 사용한다.

10000초가 몇시간인지 알고 싶으면 1시간은 3600초이므로 10000/3600==2 여서 2시간 하고 2800초가 남는다. 여기서 2800초를 구하고 싶다면 10000%3600==2800 하면된다.

문제 1) a 원을 동전으로 바꾸면 500원짜리 몇 개 100원짜리 몇 개가 되는가?

문제 2) 500원짜리 n개 100원짜리 m개 총 얼마인가?

문제 3) n초를 입력 받아 시분초로 바꿔보자.

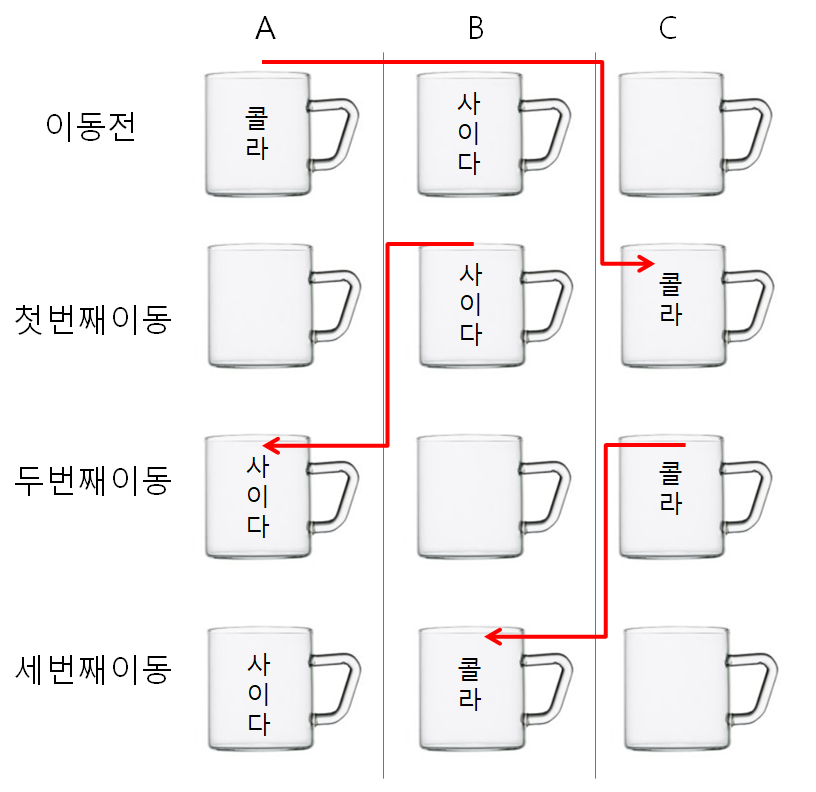
문제 4) n시간 m분 l초는 총 몇초인가?

교환방법

두 수 a=5, b=4가 있고 두수를 교환하여 a=4, b=5가 되게 만들려면 어떻게 해야 할 것인지를 고민해 보자. a=b, b=a 라고 하면 될거 같지만 막상 해보면 a=5,b=4,a=b 에서 a,b값은 이미 둘다 4가 되어 버려서 다음 b=a 이후 a, b 값은 여전히 4가 된다. 따라서 a=4,b=4가 되어 우리가 원하는 값이 아니다. 어떻게 하면 우리가 원하는 결과가 될것인지 고민해 보자.

임시 저장 공간 c를 선언하여 c=a,a=b,b=c 와 같이 하면 된다.

잘 생각해 보면 이 문제의 해답은 변수 2개 가지고 처리할 수 없다. 임시로 값을 저장 할 수 있는 변수 하나를 C로 하나더 선언해서 교환하여야 교환이 가능하다.

왼쪽 이미지에는 ABC 세개의 컵이 있다. 처음에는 A 컵에 콜라가 담겨져 있고 B 컵에 사이다가 담겨져 있는데 두 컵이 내용물을 교환해서 a 컵에는 사이다가 B 컵에는 콜라가 담기게 하려고 한다. 이 동전 시점에 A 컵 B 컵에 내용물이 담겨져 있는 상태여서 내용물을 교환할 수 없기 때문에 새로운 컵 씨를 가지고 왔다.

첫 번째 이동해서 A 컵에 들어있는 콜라를 비워야 B 컵에 있는 사이다를 담을 수 있기 때문에 A 컵에 있는 콜라를 C 컵으로 옮겼다.

두 번째 이동해서 B 컵에 있는 사이다를 a 컵에 옮겼다.

세 번째 이동해서 C 컵에 있는 사이다를 B 컵에 옮겨서 우리가 원하던 a 컵의 사이다 B 컵에 콜라가 담겨 있도록 만들었다. a=5, b=6 일때 임시 저장 공간 변수 c를 하나 선언해서 c=a,a=b,b=c 와 같은 방법으로 b=6,a=5로 만들 수 있다.

문제 1) a, b 의 수를 입력 받아 두 수를 교환한 다음에 두 수를 출력하는 순서도와 프로그램을 작성해 보자.

문제 2) 3개의 수를 입력 받아 가장 작은 수를 a, 다음 작은수를 b, 나머지 수를 c에 넣어 작은수부터 출력해보자.

다음을 만족 시키자. 변수 a,b,c에 무작위로 각각 숫자를 입력받는다. a b c를 교환하여 a b c를 출력한 결과가 아래와 같이 오름차 순으로 정렬 되도록 만든다.

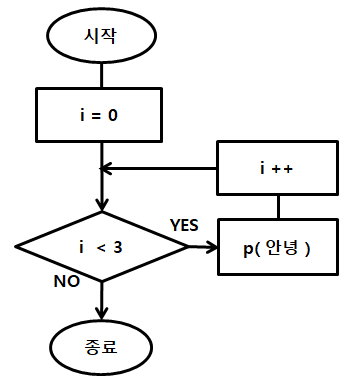
입력값 : a, b, c 출력값: a, b, c

만약에 입력값이 5,1,2이면 출력값은 1,2,5가 된다.

만약에 입력값이 3,7,4이면 출력값은 3,4,7가 된다.

만약에 입력값이 3,5,6이면 출력값은 3,5,6가 된다.

반복문을 사용해서 실제 반복횟수를 조작하는 방법을 확인해 보자. 아래 반복문 순서도와 의사코드를 확인해 보자.

i=0

while(i<3){

p(안녕)

i++

}

for문 사용법

for(i=0;i<3;i++){

p(안녕)

}

맨 처음 i가 0으로 초기화 된 상태에서 반복문의 조건문에 도착하면 i 값은 0 이므로 조건식 i<3과 비교후 Yes 부분으로 이동하여 p(안녕),i++ 가 실행이 되어 화면에 ‘안녕’이 출력되고 i 값은 1 증가하여 1이 된다. 그 다음 다시 반복문의 조건 i<3으로 이동하고 i값이 1이므로 여전히 참이다. 따라서 Yes 부분으로 이동 동하여 p(안녕),i++ 가 실행이 되어 화면에 ‘안녕’이 출력되고 i 값은 1 증가하여 2이 된다. 그 다음 다시 반복문의 조건 i<3으로 이동하고 i값이 2이므로 여전히 참이다. 따라서 Yes 부분으로 이동 동하여 p(안녕),i++ 가 실행이 되어 화면에 ‘안녕’이 출력되고 i 값은 1 증가하여 3이 된다. 그 다음 다시 반복문의 조건 i<3으로 이동하고 i값이 3이므로 여전히 참이다.

결국 프로그램이 종료되면 화면에 ‘안녕’이 3번 출력 되어 있고 i 값은 3이 된다.

반복문의 비교값이 참인 동안 중괄호 안의 코드 블록을 반복한다. 상위 순서도에서는 위에 설명한 while문 의사코드만으로도 충분히 구현 가능 하지만 프로그램 언어에서 while문을 사용하려면 아래 처럼 좀더 세분화하여 표현할 필요가 있다.

|  |
| --- |
| 초기값;  while(비교값){  //반복할 코드블록  변환값;  } |

초기값은? 반복할 코드 블록이 실행되기전에 조건식과 관련된 값을 처음 결정하는 부분이다. 초기값은 while문의 반복을 결정하는 비교값과 관련이 있어서 값을 잘못 결정하면 문제가 발생하여 무한루프가 발생 할 수 도 있어서 조심히 사용하여야 한다.

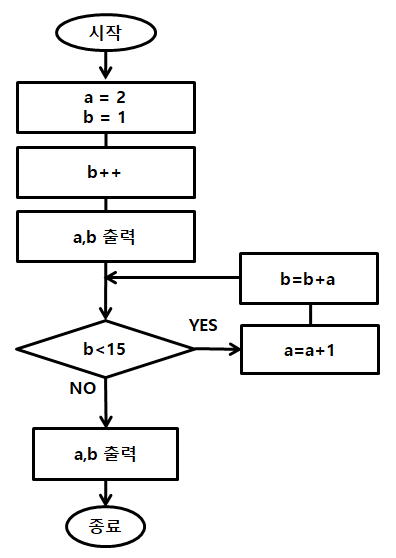
비교값은? 다시 반복할 것인지를 결정할때 초기값과 비교 대상이 되는 값이다. 비교 값이 적절하지 않으면 프로그램이 멈추거나 무한루프가 발생 할 수 있다.

변환값은? 초기값이 변하지 않으면 while문에서 비교값과 비교하였을때 결과가 바뀌지 않아 무한루프를 돌수 있다. 일정 횟수 반복 하다가 while문을 빠져나갈 수 있도록 적절한 변환값을 만들어 주어야한다.

반복문을 잘못 기술하게 되면 반복할 코드블록이 계속 반복하게되는 무한루프가 되거나 한번도 실행하지 않고 반복문을 빠져나가게 된다. 이런 문제를 해결하지 않게 하기 위해서는 반드시 초기값, 비교값, 변환값이 있어야 한다.이후 나올 순서도에 반복문이 있다면 잘못 구현하지 않도록 재대로 구현하여야 한다.

상위 순서도에서 결과값과 초기값, 비교값, 반환 값을 찾아보자.

다음 순서도와 프로그램 언어를 확인해 보고 화면에 출력되는 결과값과 초기값, 비교값, 반환 값을 찾아보자.

프로그램언어

a=2

b=1

b++

p(a)

p(b)

while(b<15){

a=a+1

b=b+a

}

p(a)

p(b)

프로그램이 종료될 때 출력되는 값들을 기술해보자.

|  |
| --- |
| 초기값;  while(비교값){  //반복할 코드블록  변환값;  } |

while문을 줄여서 for문으로 바꿔서 사용할 수 있다. 변경 방법은 다음과 같다.

|  |
| --- |
| for(초기값;비교값;변환값){  //반복할 코드블록  } |

이전 내용들을 참고 하여 다음 문제의 순서도와 프로그램을 만들어 보자.

문제 1) 1~3를 출력해보자. 이전 순서도에서 ‘안녕’를 출력하는 부분에 i를 넣어서 어떻게 출력되는지 고민해 보고 순서도와 의사 코드를 만들어 보자

문제 2) 1을 3번 더하는 순서도와 의사 코드를 만들어 보자.

문제 3) 3부터 10을 출력하는 순서도와 의사 코드를 만들어 보자.

문제 4) 1~10까지의 사이 숫자를 더한 총합을 sum에 넣어 출력하는 순서도와 의사 코드를 만들어 보자.

문제 5) 15~30까지의 사이숫자를 더하기 더한 총합을 sum에 넣어 출력하는 순서도와 의사 코드를 만들어 보자.

문제 6) 1~50까지의 짝수를 출력하는 순서도와 의사 코드를 만들어 보자.

문제 7) 1~100사이의 10의 배수을 출력하는 순서도와 의사 코드를 만들어 보자.어떤 숫자가 배수 인지 아닌지 알고 싶으면 7로 나눈 나머지가 0이면 7의 배수이고 아니면 7의 배수가 아니다.

문제 8) 30~300까지의 6의 배수의 합을 출력하는 순서도와 의사 코드를 만들어 보자.

문제 9) 숫자를 하나 입력 받아 1부터 입력한 수까지 순서대로 화면에 출력 되도록 순서도와 프로그램을 만들어 보자.

문제 10) 사용자에게 두 수를 입력받아 두 수의 사이에 있는 모든 수를 오름차 순으로 출력하는 순서도와 프로그램을 만들어 보자. 예)5 9 입력시 6 7 8 더한 결과를 얻음

문제 11) 두수를 입력 받아 사이에 있는 짝수를 화면에 오름차 순으로 출력 되도록 순서도와 프로그램을 만들어 보자.

문제 12) 1-2+3-4+5-6……+99-100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

문제 13) 1/2+2/3+3/4+4/5+ 5/6+6/7+....+99/100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

문제 14) 피보나치 수열을 10개를 순서대로 출력하는 순서도와 프로그램을 작성해 보자. 다음 순서도는 반복해서 사용자가 원하는 메뉴를 입력받아서 보여주는 순서도와 프로그램이다. 피보나치 수열이 무엇인지는 웹사이트를 검색해서 스스로 알아보자.

문제 15) 원하는 색의 전구와 밝기를 입력 받아 처리하는 순서도와 프로그램을 만들어 보자. 사용자가 원하는 밝기와 색상을 입력받아 원하는 결과를 출력받자. 아래 설명한 내용대로 운영되도록 기술 하자.

초기 변수값 : color=”빨강” brightness =50

값의 범위 : color=빨강,노랑,파랑 brightness는 0~100

제안사항 : brightness의 숫자 변경은 1씩 가능하다. brightness값이 10이라면 다음 brightness값은 11 이나 9 만 가능하다. 10을 50으로 변경하려면 반복문을 사용해야 한다.

사용자입력변수 : colorInput, brightnessInput

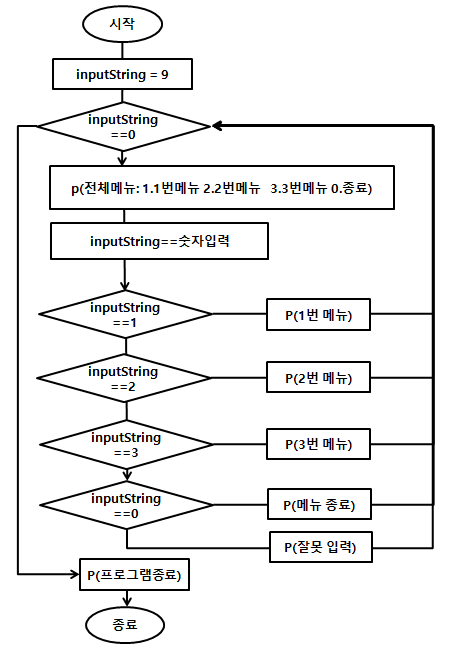
최종결과값출력 : p(“현재 색상은”+color+”밝기는”+brightness+”이다”);

문제 16) 1-2+3-4+5-6……+99-100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

문제 17) 1/2+2/3+3/4+4/5+ 5/6+6/7+....+99/100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

프로그램에서 메뉴에 대해서 생각해보자.

요즘 음식점에 가면 pos 기계라 부르는 사용자가 메뉴을 직접 선정해서 결제하도록 되어 있는 기계를 볼수 있는데 이 기계의 메뉴는 사용자가 원하는 메뉴를 선택하면 해당 메뉴의 작업이 실행되고 이 작업이 마무리되면 다음 사용자가 다시 천체메뉴를 사용할 수 있도록 지속적으로 반복하는 형태로 되어 있다.

처음에 화면에 ‘전체메뉴: 1.1번메뉴 2.2번메뉴 3.3번메뉴 0.종료’ 다음과 같은 선택 메뉴가 나오고 다음으로 사용자가 원하는 메뉴를 입력 받으면 해당 메뉴 작업이 진행되고 다시 천체메뉴로 돌아오는 순서도를 만들어 보았다 

다음 왼쪽을 확인해 보자. ‘전체메뉴: 1.1번메뉴 2.2번메뉴 3.3번메뉴 0.종료’ 다음과 같은 문자열이 화면에 나오고 사용자가 숫자를 입력 받아 사용자가 선택한 번호의 메뉴가 화면에 나온다. 1번을 선택하면 ‘1번 메뉴입니다.’ 출력하고 2번을 선택하면 ‘2번 메뉴입니다.’가 출력 되고 0를 입력하면 프로그램이 종료 된고 1~3번의 경우는 해당 번호의 메뉴를 출력하고 다시 전체메뉴가 출력되어 사용자 입력을 기다린다. inputString=9에서 9는 프로그램에 영향을 주지 않는 임의의 숫자를 넣은 것이다. p(n번메뉴)와 같은 부분에서 실제 n번 메뉴를 실행하는 내용을 복잡하게 만들수 있는데 여기서는 생략 하였다.

다음은 의사코드로 만든 것이니 확인해보자.

inputString = 9

while (inputString==0) {

p(전체메뉴: 1.1번메뉴 2.2번메뉴 3.3번메뉴 0.종료)

inputString = 숫자입력

if(inputString==1) {

p("1번 메뉴")

}else {

if(inputString==2) {

p("2번 메뉴")

}else {

if(inputString==3) {

p("3번 메뉴")

}else {

if(inputString==0) {

p("메뉴 종료")

}else {

p("잘못 입력")

}

}

}

}

}

p("프로그램종료");

상위 if문을 줄여서 다음과 같이 else if문으로 바꿀 수 있다. 다음을 확인하자.

if(inputString==1) {

p("1번 메뉴")

}else if(inputString==2) {

p("2번 메뉴")

}else if(inputString==3) {

p("3번 메뉴")

}else if(inputString==0) {

p("메뉴 종료")

}else {

p("잘못 입력")

}

문제 1) 은행 프로그램을 순서도와 프로그래밍 언어로 만들어보자.

은행 프로그램의 메뉴는 다음과 같다. ‘전체메뉴: 1.입금 2.출금 3.조회 0.종료’

account 변수에 처음에 0으로 세팅하고 입금 출금을 통에 원하는 액수를 더하거나 빼준다. 결국 account 변수에 있는 숫자가 은행계좌에 남은 돈이 된다.

입금 메뉴를 통해 입금 액을 받아 account 변수에 추가 할 수 있다.

출금 메뉴를 통해 출금 액을 account 변수에서 뺄 수 있다.

조회 메뉴를 통해 account에 입금 액이 얼마나 남아 있나 확인 할 수 있다.

조심해야 할 부분은 운영중 account의 값이 유지될수 있도록 맨상단에 선언하여 운영중에 값에 손실이 생기지 않는지 확인하자.

문제 2) 금액을 입력받아 천원, 오백원, 백원, 오십원, 십원 짜리 잔돈으로 거슬러주는 프로그램과 순서도을 구현해 보자. 예제 금액을 1000으로 나눈 몫은 돈을 거슬러 주었을때 천원짜리 개수 이고 1000으로 나눈 나머지는 천원짜리로 환산하고 남은 잔돈이다. 10원 이하는 입력받지 않는다. 최종 결과물은 다음과 같이 될 것이다. 5820원을 잔돈으로 바꾸면 천원짜리 5개, 오백원짜리 1개, 백원짜리 3개, 오십원짜리 0개,십원짜리 2개

선언할 변수:change1000,change500,change100,change50,change10,moneyInput

출력값:p(“천원짜리 “+change1000+”개, 오백원짜리 “+change500+”개, 백원짜리”+ change100+”개, 오십원짜리 “+change50+”개, 십원짜리 “+change10개)

문제 3) 실존하는 자판기와 동일하게 만들어 보자.

메뉴:

남은돈 0원.

1.사이다 700 2.콜라 500 3.환타 350 4.투입 5.반환

메뉴를 보여주고 사용자가 선택을 하면 선택후 항상 다시 메뉴를 보여 준다.

사용자 입력 변수 :

inputString, moneyInput, moneyTotal

시나리오 :

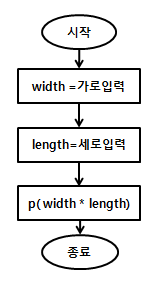
천원짜리 세장 넣고 사이다1개 콜라2개 환타1개 를 구매한다.

출력값:

사이다 콜라 콜라 환타 천원짜리 0개 오백원짜리 1개 백원짜리 4개 오십원 1개

문제5. 엘리베이터 순서도를 만들어 보자.

|  |
| --- |
| > 06.순서도 살펴보기 |



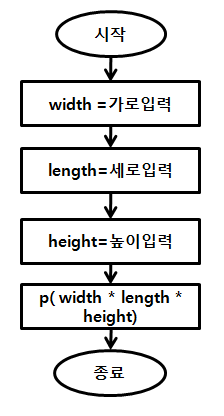
1. 세로와 가로를 입력 받아 사각형의 넓이를 만드는 순서도와 프로그램을 만들어 보자.

프로그램언어

width=가로입력

length=세로입력

p(width\*length)



2. 세로 가로 높이를 입력 받아 사각기둥의 부피를 만드는 순서도와 프로그램을 만들어 보자.

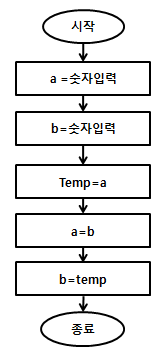
프로그램언어

width=가로입력

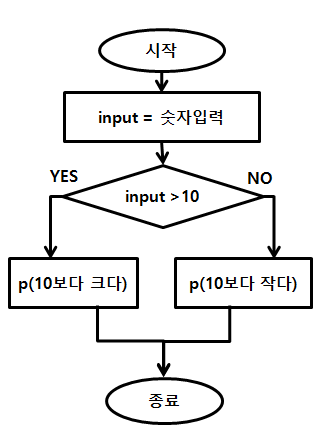
length=세로입력

height=높이입력

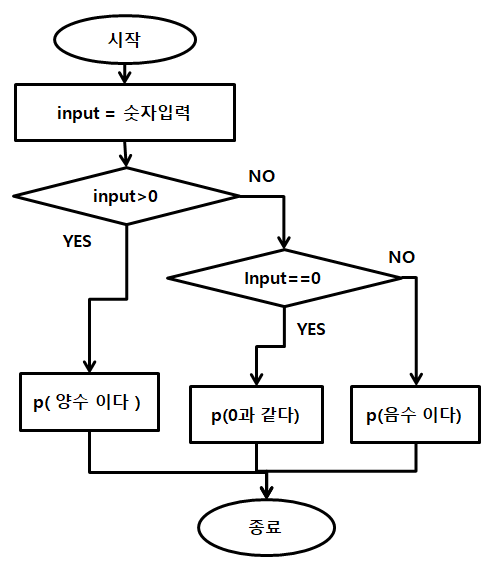
p(width\*length\*height)



3. a, b 의 수를 입력 받아 두 수를 교환한 다음에 두 수를 출력하는 순서도와 프로그램을 작성해 보자.



4. 입력한 숫자가 10보다 큰 수인지 아닌지 출력하는 순서도와 프로그램을 만들어 보자.



5. 입력 받은 숫자가 양수인지 0인지 음수인지 판단하는 순서도와 프로그램을 만들어 보자.

input=숫자입력

if(input>0){

p(양수 이다)

}else{

if(input==0){

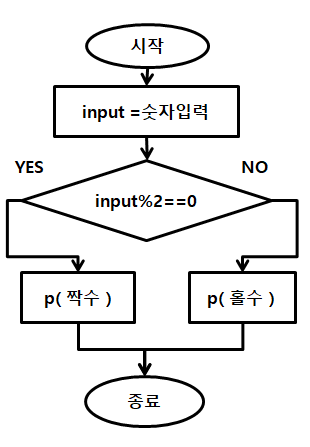
p(0과 같다)

}else{

p(음수 이다)

}

}

6. 입력 받은 숫자가 홀수인지 짝수인지 판단하는 순서도와 프로그램을 만들어 보자. 

input = 숫자입력

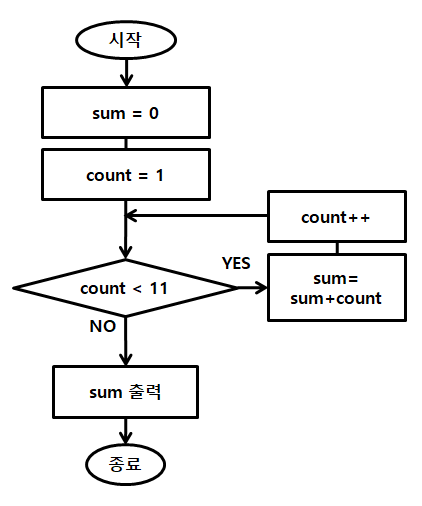
if( input%2==0){

p(짝수)

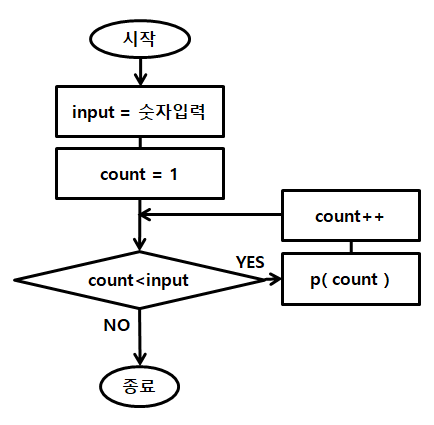
else{

p(홀수)

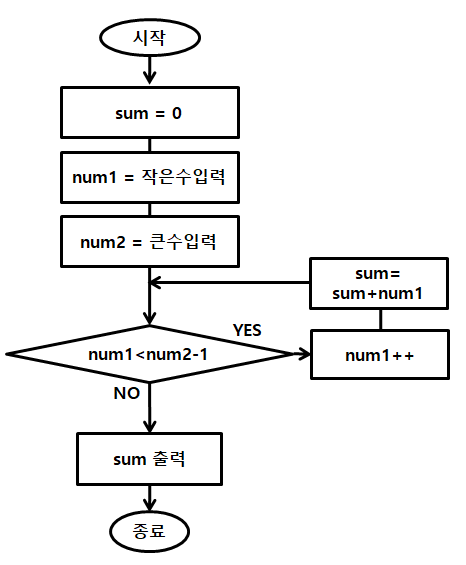
}

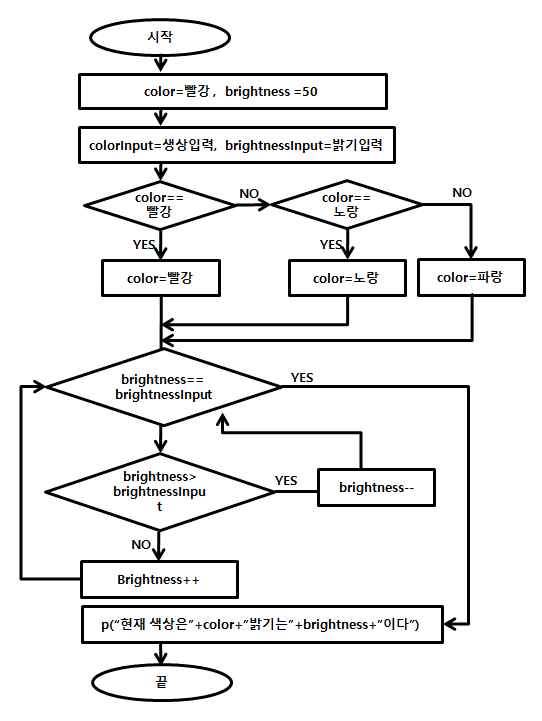


7. 1 부터 10까지 더하는 순서도와 프로그램을 만들어 보자.

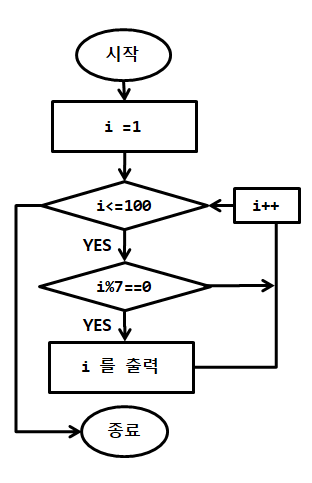


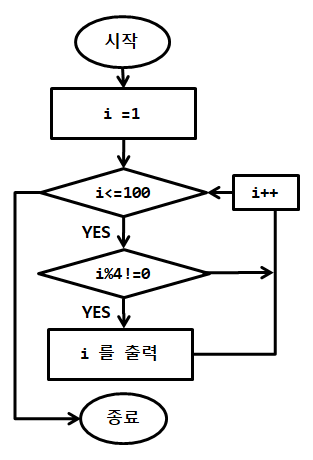
8. 숫자를 하나 입력 받아 1부터 입력한 수까지 순서대로 화면에 출력 되도록 순서도와 프로그램을 만들어 보자.

9. 사용자에게 처음에는 작은수 두 번째는 큰수를 입력 받아 두 수의 사이에 있는 모든 수를 합하는 순서도와 프로그램을 만들어 보자. 

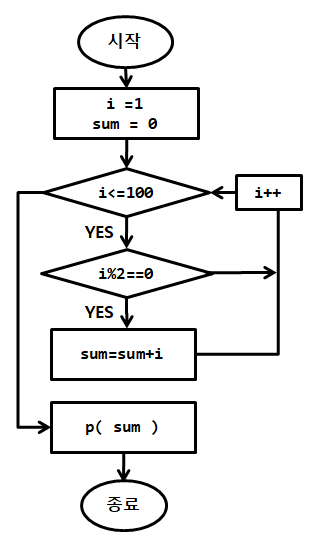


10. 원하는 색의 전구와 밝기를 입력 받아 처리하는 순서도와 프로그램을 만들어 보자. 아래 내용들이 들어가 있어야 한다.

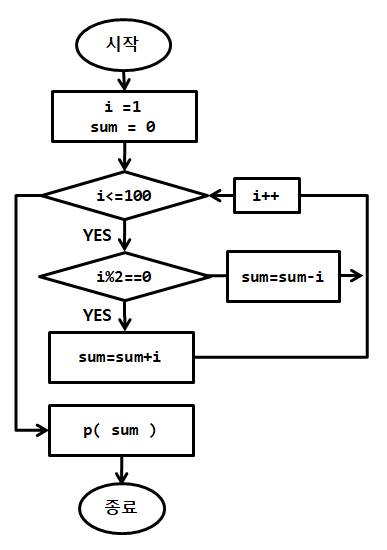
11. 1~100까지의 7의 배수를 출력하는 순서도와 프로그램을 작성해 보자. 



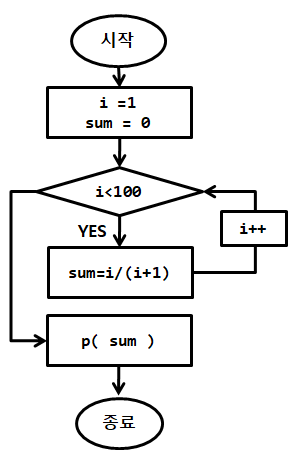
12. 1~100까지의 숫자중 4의 배수를 제외한 모든 수를 출력하는 순서도와 프로그램을 작성해 보자.

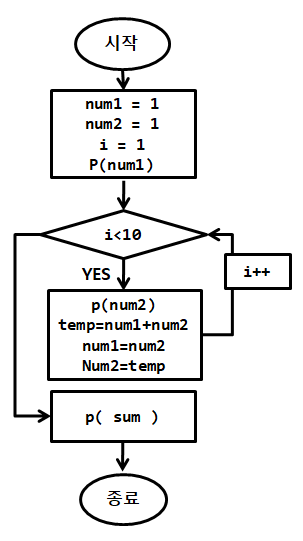


13. 1~100까지의 짝수의 합을 출력하는 순서도와 프로그램을 작성해 보자.

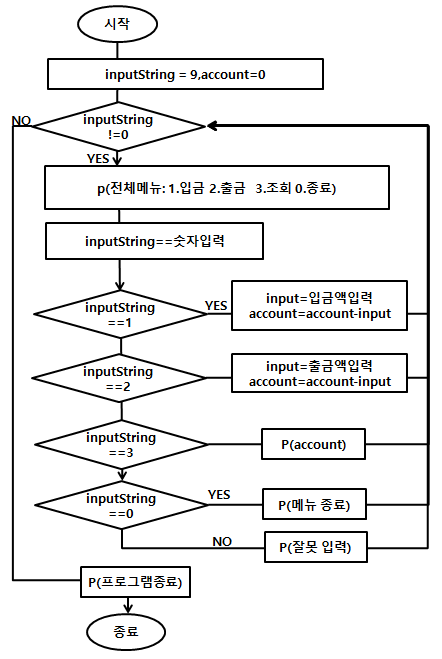


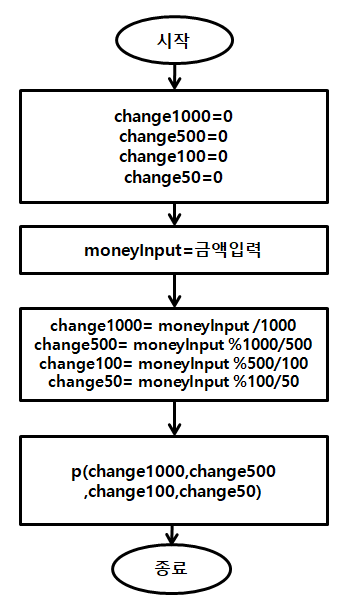
14. 1-2+3-4+5-6……+99-100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

15. 1/2+2/3+3/4+4/5+5/6+6/7+....+99/100 의 결과를 구하는 순서도와 프로그램을 작성해 보자.



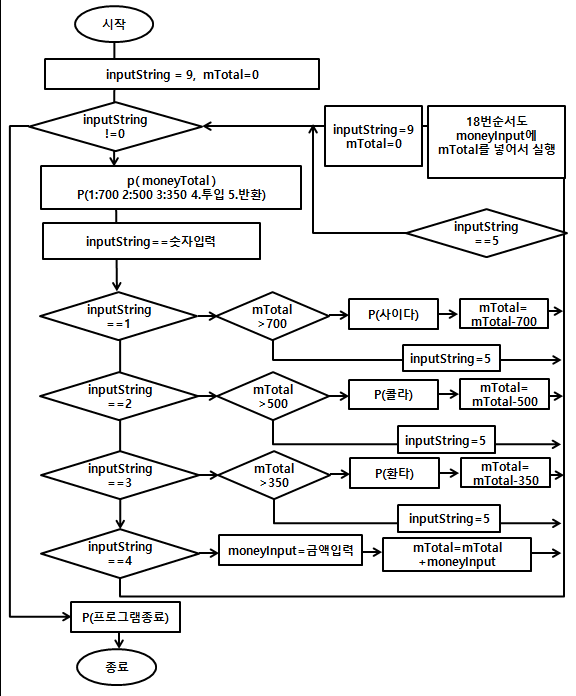
16. 피보나치 수열 10개를 순서대로 출력하는 순서도와 프로그램을 작성해 보자.

17. 은행 프로그램을 순서도와 프로그래밍 언어로 만들어보자.



18. 금액을 입력받아 천원 오백원 백원 오십원 짜리 잔돈으로 거슬러주는 프로그램과 순서도을 구현해 보자.

19. 실존하는 자판기와 동일하게 만들어 보자.



|  |
| --- |
| > 05. 배열 |

여러개의 데이터를 한번에 생성하는 것을 배열 이라고 한다. 변수는 5개의 저장 공간을 만들려면 a1=0, a2=0, a3=0, a4=0, a5=0 이런 식으로 일일이 선언 하여야 한다. 이런 번잡함을 해결하기 위해서 한번에 저장 공간을 만드는 방법이 제공되는데 이를 배열이라고 한다.

배열의 사용 방법은 배열명[인덱스]이렇게 기술한다. 배열명은 변수명이 변수들을 구분할때 사용 하듯이 배열을 구분하는데 사용한다.변수로 저장 공간 5개를 만드는 것이 a1=0, a2=0, a3=0, a4=0, a5=0와 같다면 배열로 저장 공간 5개 만드는 방법은 a[5],a[]={0,0,0,0,0}와 같다. a는 배열명이고 5는 저장공간 크기를 의미한다. 여기서 a는 변수명 처럼 본인이 원하는 이름을 넣으면 되고 5는 원하는 데이터양 만큼 선정하면 된다. 배열명[인덱스]에서 인덱스는 저장공간의 위치를 의미 한다. 인덱스는 0부터 시작한다는 것을 잊지 말자. 배열은 a[0]=0, a[1]=0, a[2]=0, a[3]=0, a[4]=0 같은 형태로 값을 넣을 수 있고 a[0], a[1], a[2], a[3], a[4] 형태로 값을 읽어 올 수 있다.

a1, a2, a3 등과 같은 경우는 저장 공간과 변수명이 1개 여서 a1에 1를 넣고 싶으면 a1=1 a2에 2를 넣고 싶으면 a2=2 이런 식으로 1:1 연결해서 접근 가능 하지만 배열 같은 경우에는 배열명이 1개 일 때도 여러 개의 저장 공간을 가진다. 같은 이름으로 여러 저장 공간에 접근하기 위해서 인덱스를 사용 하는데 a[인덱스] 즉 a[0], a[1], a[2]이런 식으로 접근 할 수 있다. 중요한 것은 인덱스는 0 부터 시작한다는 것이다. 첫번째 값에 접근하려면 a[0] 두번째 값에 접근하려면 a[1], 세번째 값에 접근하려면 a[2]하면 된다. a[12]와 같이 값을 가지고 있지 않은(할당 되지 않은) 인덱스로 값에 접근 하려 하면 에러가 발생 한다.(잘못된 결과가 발생한다.), 5개의 공간을 필요로 한다면 선언은 a[5]이고 접근할수 있는 인덱스는a[0]~a[4] 입니다.

5개 저장공간을 가진 변수와 배열에서 첫번째 값을 가져오는 방법은 변수의 경우 a1, 배열의 경우 a[0]이라고 하면 된다. 세번째 값을 가져오는 방법은 변수의 경우 a3, 배열의 경우 a[2]이라고 하면 된다. 네번째 데이터의 값을 4로 변경하고 싶을 경우 변수의 경우 a4=4, 배열의 경우 a[3]=4이라 하면 된다.

다음 상황들을 코드로 어떻게 기술하는지 확인해 보자.

a라는 배열에 5,2,3를 넣은 다음 : a[0]=5, a[1]=2, a[2]=3

변수b가 4일때 a배열의 인덱스[1]번을 b만큼 증가시킨 다음 : a[1]=a[1]+b

인덱스 0과 인덱스 1를 더해서 인덱스 2에 넣은 다음 : a[2]=a[0]+a[1]

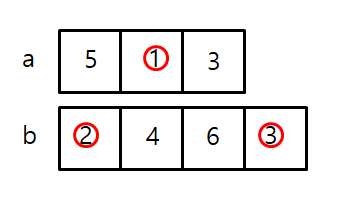
a배열의 인덱스 0,1,2를 더한 값을 a[3]에 넣은 다음 : a[3]=a[0]+a[1]+a[2]

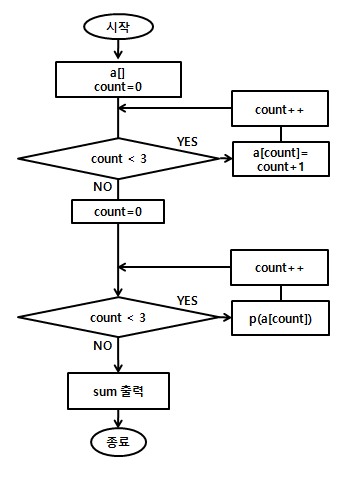
a 배열의 모든 값의 합을 변수 sum에 넣은 다음:sum=a[0]+a[1]+a[2]+a[3]

sum를 5로 나눈다음:sum=sum/5

sum를 출력해 보자. sum값은 얼마인가?

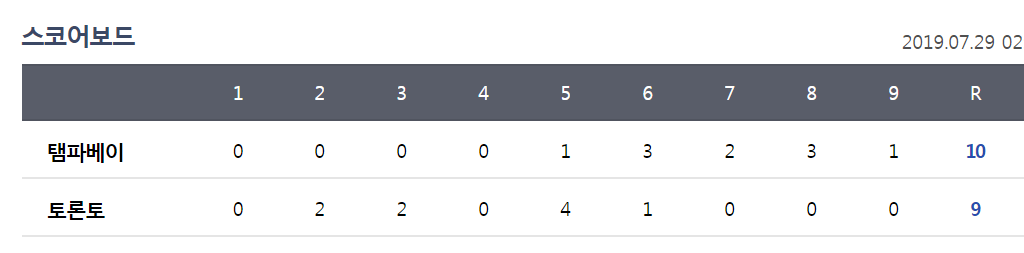
선언과 함께 배열의 값을 설정 하는 방법도 있다. a[]={1,2,3,4,5} 이렇게 하면 a[0]에 1, a[1]에 2, a[2]에 3, a[3]에 4, a[4]에 5가 들어 있는 것과 같은 결과 가 나올 것이다.

변수를 여러개 선언 하듯이 배열도 여러개 사용할 수 있다. 왼쪽이미지는 a[3],b[4]를 선언하여 총 7개의 저장 공간을 선언한 결과이다. a[0]을 이용해서 5값을 읽어 올수 있고 b[2]을 이용하여 6값을 읽어 올수 있다. 빨간동그라미 1번에 8를 넣으려면 a[1]=8이라고 하면되고 빨간 동그라미 2번에 1를 넣으려면 b[0]=1를 빨간동그라미 1,2에 들어 있는 값을 3에 넣으려면 b[3]=a[1]+b[0]하면 된다.

변수를 인덱스로 사용할 수 있다. a[]={1,2,3,4,5}, b=1, p(a[1]), p(a[b]) 와 같이 사용할 수 있고 실행 결과는 둘다 a[1]에 존재하는 2가 출력 된다.

왼쪽 순서도는 크기 3에 배열 a 1,2,3를 넣은 다음 배열에 들어 있는 내용들을 모두 화면에 찍는 순서도 이다. 한줄 한줄 따라가보면서 배열 a와 변수 count와 화면에 출력되는 내용을 확인해 보자.

아래는 야구게임 전광판이다. 여러 변수들을 이용하여 게임이 진행되는 순으로 값들을 화면에 출력해보자.0,0,0,2,0,2,0,0,.....1,0,10,9 다음과 같이 출력되면 된다. 순서도와 프로그램을 작성해 보자.



상위 그림은 야구점수 전광판 이다. 상위 정보를 표현하는데 필요한 자료들이 어떠한 것들이 있는지 기술해 보자. 일단 팀파베이, 토론토는 야구 팀이다 두팀의 1~9회까지 점수를 저장할 배열 2개와 총합을 저장할 변수 2개가 있으면 야구점수 관련된 모든 데이터를 저장할 수 있을 거 같다. 물론 배열 하나에 모든 정보를 기록 할 수 있다. 0~9까지는 탬파베이 점수 10~19까지는 토론토 점수 해도 되지만 이렇게 된다면 배열을 사용하는 것이 불편해지므로 사용하기 편한 형태로는 두팀의 회수 정보배열2개와 총합변수 2개로 하여 선언하였다. team1[], team2[], team1TotalScoure=0, team2TotalScore=0 이해하기 쉬운 형태의 변수명을 사용해서 선언해 보았다. 배열 team1에 탬파베이 점수값을 넣어보자. 코드 줄수를 줄이기 위해서 배열을 선언하면 기본 값이 0 이라고 생각하고 진행하자. 5회에 1점을 얻었다 team1 배열에 이정보를 넣으려면 team1배열의 4번째 인덱스에 1를 넣으면된어 다음과 같이 하면된다. team1[4]=1 6회에 3점을 얻었다 team1[5]=3 team1[6]=2… 이런 식으로 진행해 나가면 된다. 토론토의 점수를 배열에 넣으려면 다른 배열 team2에 넣으면 된다. team2[1]=2, team2[2]=2 이런 식으로 넣으면 된다. 두팀의 총 취득한 점수를 어떻게 구할 것인지 생각해보자. 첫번째 팀의 경우 다음과 같이 team1의 인덱스 0 부터 8까지 모두 저장하면 된다. 이 방법이 복잡하다면 초기화 방법으로 각팀의 점수를 초기화 하자. 선언과 함께 할당하는 방법을 사용하는것도 좋은 방법인거 같다.

team1TotalScoure=team1[0]+team1[1]+team1[2]+team1[3]+team1[4]+team1[5]+team1[6]+team1[7]+team1[8] 이렇게 하면 팀1의 득점한 총 점수를 출력할 수 있을 것이다. 지금 처럼 일일이 써서 넣을 수 도 있지만 다음과 같이 여러 줄로 처리할 수 있다.

team1TotalScoure=team1TotalScoure+team1[0]

team1TotalScoure=team1TotalScoure+team1[1]

team1TotalScoure=team1TotalScoure+team1[2]

team1TotalScoure=team1TotalScoure+team1[3]

team1TotalScoure=team1TotalScoure+team1[4]

team1TotalScoure=team1TotalScoure+team1[5]

team1TotalScoure=team1TotalScoure+team1[6]

team1TotalScoure=team1TotalScoure+team1[7]

team1TotalScoure=team1TotalScoure+team1[8]

team1TotalScoure=team1TotalScoure+team1[9]

여기에서 인덱스를 변수로 사용할 수 있다. 변수를 사용해서 다음과 같이 변경해 보자.

i=0

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

이렇게 변수를 사용하면 처음과 동일한 결과가 나올것이다. 이렇게 변경해서 사용하는 이유는 보면 알겠지만 다음 같은 부분이 반복되고 있어 team1TotalScoure = team1TotalScoure + team1[i] i++ 이 부분을 반복 시키면 더욱 간단한 형태로 알고리즘 순서도을 구현할 수 있다.

이제 야구 경기를 진행할 때 필요한 모든 데이터가 선언된 것 같다. 야구 경기가 진행될 때 표시 되는 전관 판 점수 처럼 순서대로 출력해 보자. 최종 출력 결과물은 0,0,0,2,0,2,0,0,.....1,0,10,9 과 같이 출력될 것이다.

문제 1) 왼쪽 순서도의 보고 의사코드를 만들어 보자.

문제 2) 배열 a에 1,2,3 을 넣은 후 배열 내의 모든 값에 2를 더한 값인 3,4,5로 변경한 다음에 배열의 내용을 화면에 출력하는 순서도와 코드를 구현하여 보자.

문제 3) 배열 a[10]에 3의 배수를 넣은 다음에 배열의 내용을 출력하는 순서도와 의사코드를 만들어 보자.

문제 4) 배열 a[100]에 1부터 100까지의 숫자를 순서대로 넣은 다음 배열의 인덱스가 짝수인 값만 출력하는 순서도와 의사코드를 만들어 보자.

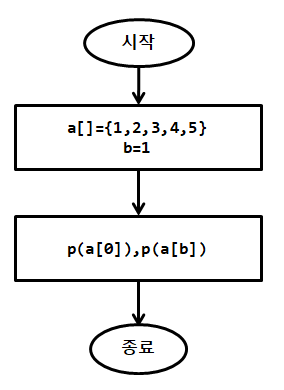
문제 5) 배열 a[]={12,1,5,3,6,8,5,3}를 만든 다음에 배열의 모든 내용을 더한 값을 sum에 저장하여 출력하는 순서도와 의사코드를 만들어 보자.

문제 6) a[]={12,1,5,3,6,8,5,3}의 a배열에서 배열 안의 숫자가 짝수인 12,6,8 의 값을 더한 결과 값을 출력하는 순서도와 프로그램을 작성해 보자.

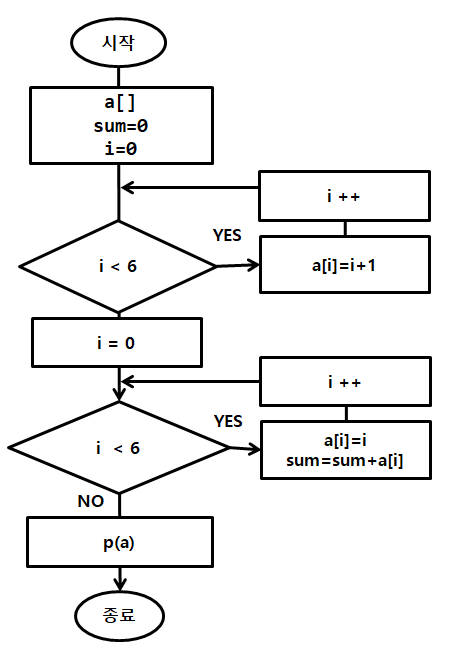
문제 7) a[]={12,1,51,3,6,8,5}의 a배열에서 가장 큰 값과 가장 작은 값을 더하는 순서도와 프로그램을 작성해 보자.

{12,1,51,3,6,8,5}에서 가장 큰수를 찾으려면 일단 가장 큰수를 저장할 변수 max에 12를 넣는다. 그다음 {12,1,51,3,6,8,5}에서 처음 부터 순서대로 비교해서 더큰 숫자가 나오면 max값에 더 큰값을 넣는다. 첫번째 값 12는 현재 max값 12와 같으므로 패스한다. 두 번째 값 1은 max값보다 작으므로 패스한다. 세 번째 값 51은 max값보다 크므로 max값에 51를 넣는다. 네 번째 값 3은 max값 51 보다 작으므로 패스한다. 다섯 번째 값 6은 max값 51 보다 작으므로 패스한다. 여섯 번째 값 8은 max값 51 보다 작으므로 패스한다. 일곱 번째 값 5는 max값 51 보다 작으므로 패스한다. 이렇게 모든 숫자를 max 와 비교하는 작업이 끝나면 제시된 모든 숫자에서 max가 가장 큰 수가 된다. 여기서는 max값 51이 모든 숫자들 중에 가장 큰 숫자이다

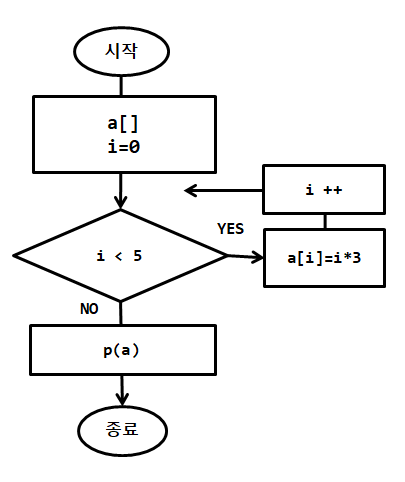
순서도로 상위 사항을 구현할 때 어떻게 해야 하는지 살펴보자. 배열의 가장 큰 값을 변수 max에 찾아서 넣는 방법은 max를 선언한 다음 배열 안의 임의의 수를 배열에 넣고(max=a[0]) 배열을 인덱스 0부터 끝까지 순회 하면서 해당 인덱스의 배열 안의 수와 max를 비교해서 (a[i]>max) 배열[인덱스]의 값이 max보다 크면(a[i]>max 이 true이면) 배열[인덱스] 값이 max 값 보다 더 크다는 이야기 이므로 max에 더 큰수 a[i]를 넣어(max=a[i]) 기존 max 값보다 더 큰 수를 max에 넣는다. 배열 안의 모든 수를 max값 보다 큰수가 나올 때마다 max에 넣으면 모든 배열의 데이터를 순회 하고 나면 max에는 배열안의 수중 가장 큰 수가 남게 된다. 그래도 이해가 가지 않으면 정답 순서도를 보고 이해해 보자.

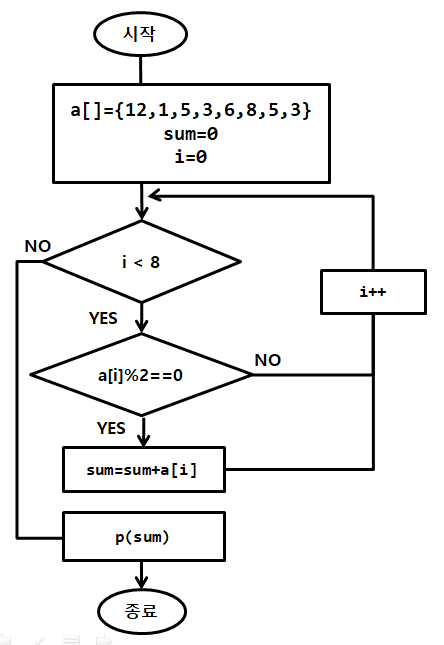
배열 순서도 답안

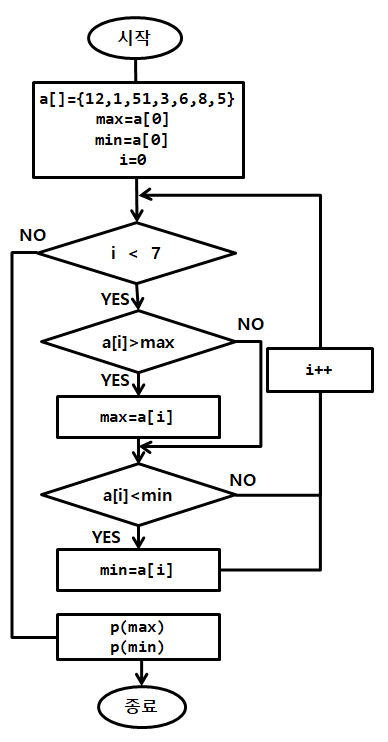
1. 인덱스 상수 숫자 대신에 변수 숫자를 넣을 수 있다. a[]={1,2,3,4,5}, b=1, p(a[0]), p(a[b]) 와 같이 사용할 수 있고 실행 결과는 1과 2가 출력 된다.

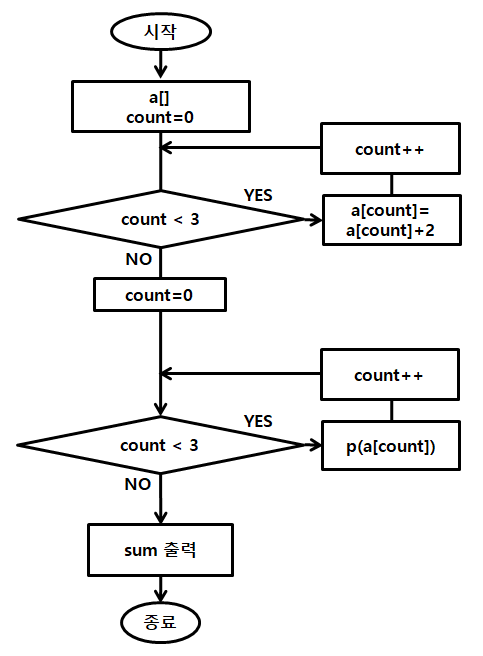


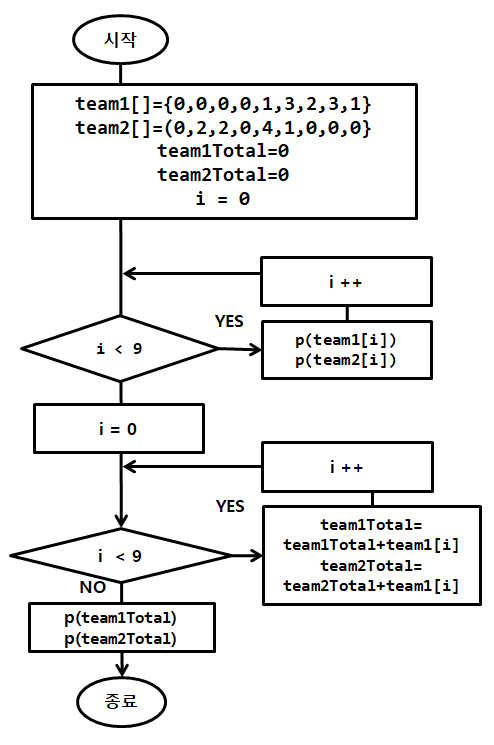
3. 다음을 순서도로 만들어 보자. 배열 a에 인덱스 0~5에 1부터 6까지 순서대로 숫자를 집어 넣은 다음에 배열의 내용을 더한 값을 sum에 저장하여 출력하는 순서도를 만들어 보자. 배열 초기화 방법을 사용하지 않고 만들어 보자.

4. 배열에 3의 배수를 넣는 순서도와 프로그램을 작성해 보자.

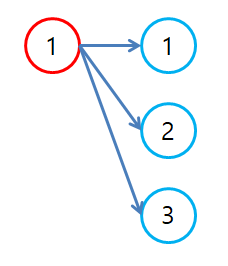
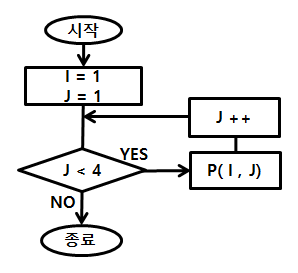
5. a[]={12,1,5,3,6,8,5,3}의 a배열에서 배열 안의 숫자가 짝수인 12,6,8 의 값을 더하는 순서도와 프로그램을 작성해 보자.

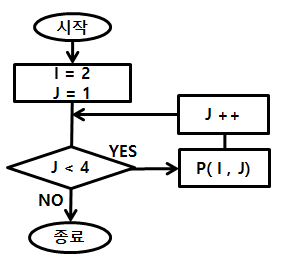
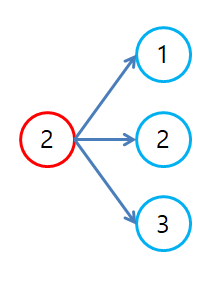
6. a[]={12,1,51,3,6,8,5}의 a배열에서 가장 큰 값과 가장 작은 값을 더하는 순서도와 프로그램을 작성해 보자.

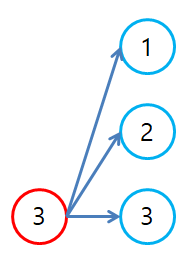
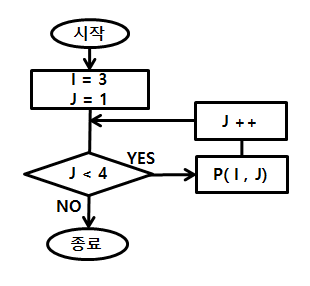


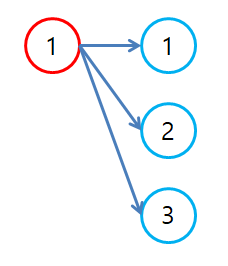
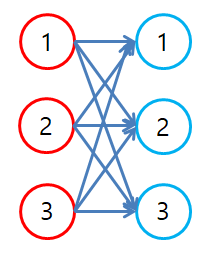
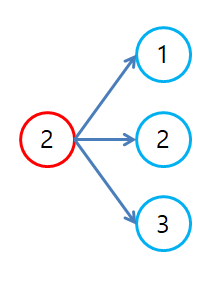
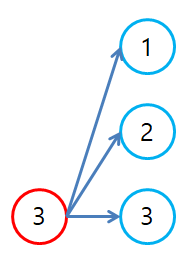
7. 아래는 야구게임 전광판이다. 여러 변수들을 이용하여 게임이 진행되는 순으로 값들을 화면에 출력해보자. 0,0,0,2,0,2,0,0, .....1,0,10,9 다음과 같이 출력되면 된다. 순서도와 프로그램을 작성해 보자.

|  |
| --- |
| > 08. 이중 반복문 |

다음 빨간색 공과 파란색공을 1개씩 서로 짝을 지어보면 다음과 같이 될 것이다. 앞이 빨간공 뒤가 파란공이라면 (1, 1),(1, 2),(1, 3) 와 같은 짝이 만들어 질것이다. 자세히 보면 빨간 공은 1인 상태로 고정하고 파란 공을 1부터 3까지 하나씩 증가 하면서 짝을 짓고 있다. 빨간색 공을 i라 하고 파란색 공을 j라 한다면 반복문을 이용해서 j값을 하나씩 증가시 키는 방법은 이미 공부 하여서 충분히 생각해 볼 수 있을 것이다. 고정된 값 i는 반복문 밖에 선언하고 반복문 안에서는 j 값을 1부터 하나씩 증가 시키며 i와 j 값을 출력하면 원하는 결과를 얻을 수 있을 것이다. 상위 순서도를 확인하고 기찻길과 의사코드를 만들어 보자. 순서도를 잘 따라가며 출력값을 확인해 보자.

왼쪽 첫 번째 이미지 처럼 빨간 공이 2라면 (2, 1),(2, 2),(2, 3)와 같은 짝이 만들어 질 것이다. 이전 상황과 잘 비교해 보면 i 값이 2로 바뀐 것 말고는 변한게 없다는 것을 알 수 있다.

왼쪽 첫 번째 이미지 처럼 빨간 공이 3이라면 (3, 1),(3, 2),(3, 3)와 같은 짝이 만들어 질 것이다. 이전 상황과 잘 비교해 보면 i 값이 3로 바뀐 것 말고는 변한게 없다는 것을 알 수 있다.

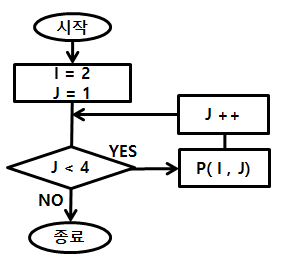
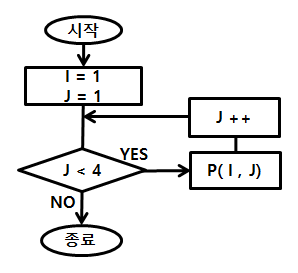
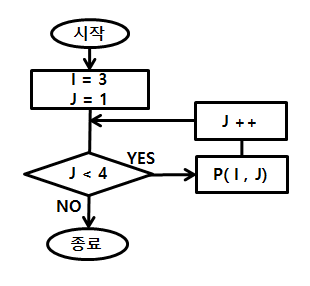


상위 맨 왼쪽 이미지 처럼 짝을 짓는다면 다음과 같은 결과가 나올 것이다.

(1, 1),(1, 2),(1, 3) (2, 1),(2, 2),(2, 3) (3, 1),(3, 2),(3, 3)

상위와 같이 출력 되려면 어떻게 순서도와 의사 코드를 구현 할 것인지 생각해 보자.

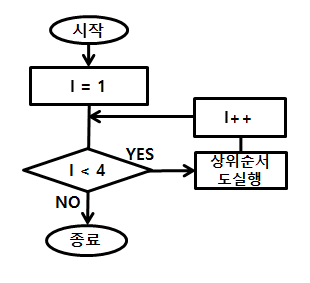
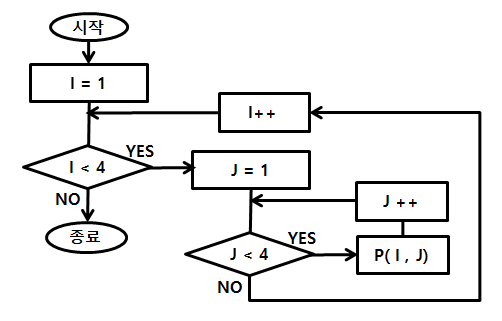
잘 살펴보면 맨 왼쪽 이미지는 나머지 세 개 이미지를 합친 것과 같다. 따라서 오른쪽 세 개 이미지를 순서도로 그리고 순서대로 실행 한다면 같은 결과가 나올 것이다.



출력 결과를 잘 살펴보면 상위 첫 번째 순서도 출력결과는 (1, 1),(1, 2),(1, 3) 이다. 두 번째 순서도 출력결과는 (2, 1),(2, 2),(2, 3)이다. 세 번째 순서도 출력결과는 (3, 1),(3, 2),(3, 3) 이다. 세 가지 순서도를 왼쪽부터 순서대로 실행한다면 우리가 원하는 결과를 얻을 수 있다.

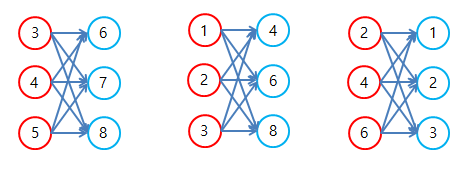
잘 생각해 보면 세 개의 순서도가 상당히 유사하다는 것을 알 수 있을 것이고 유일한 차이점을 찾는다면 첫 번째 순서도는 i는 1이고 두 번째 순서도는 i가 2이고 세 번째 순서도는 i가 3이라는 것이다. i 값이 하나씩 1, 2, 3 증가 하고 있다는 것만 빼고는 달라진 것이 하나도 없다. 결국에 i 값이 하나씩 증가 하면서 반복 되고 있다는 것이다. 숫자 하나씩 증가되면서 반복되는 것은 반복문으로 바꿀수 있다는 사실은 무수히 경험해 왔을 것이다. 반복문으로 만들어 보면 i값이 증가하는 형태의 반복문안에 상위 순서도 1개가 들어가 있으면 될 것이다.

문제 1) 사용자에게 1부터 9 사이에 숫자를 받아서 사용자가 입력한 숫자에 맞는 구구단의 해당 단을 출력해 보자. 예를 들어 사용자가 4를 입력하면 4단을 출력하면 된다.

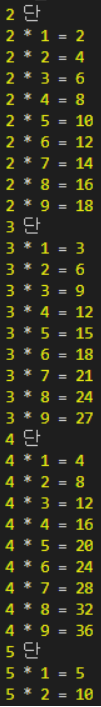
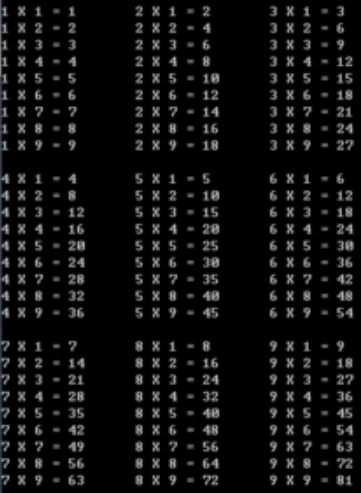
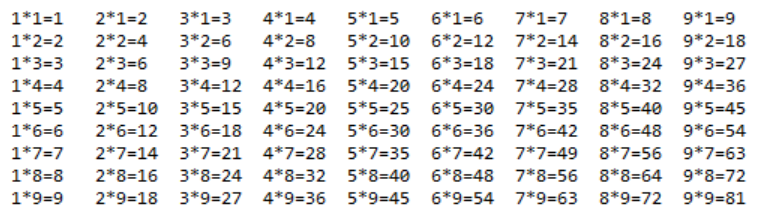


상위 왼쪽 순서도를 확인해 보면 i가 1부터 3까지 3번 반복하면서 상위순서도가 3번 실행 된다. i가 1일때 i가 1인 상태에서 상위순서도가 실행이 되고 i가 2일때 i가 2인 상태에서 상위순서도가 실행이 되고 i가 3일때 i가 3인 상태에서 상위순서도가 실행이 되면 우리가 원하던 (1, 1),(1, 2),(1, 3) (2, 1),(2, 2),(2, 3) (3, 1),(3, 2),(3, 3) 결과를 얻을 수 있다.

오른쪽 순서도는 왼쪽 순서도에서 상위 순서도 실행 부분을 실제로 붙여서 만든 순서도로 하나하나 따라 가며 출력값을 확인해 보면 원하는 결과가 출력된다는것을 알수있다. 의사코드로 바꿔보고 기찻 길도 그려보고 for문으로도 바꿔보자.

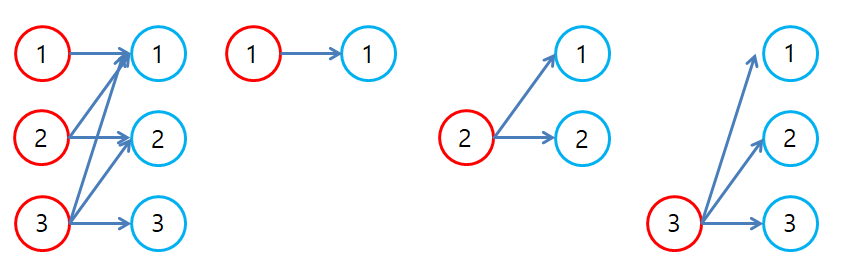


문제 1) 상위 3개의 이미지를 보고 어떤 결과가 출력되는지 기술 해 보고, 기찻길을 그려 보고,순서도를 그려보고, 의사 코드를 만들어 보자.

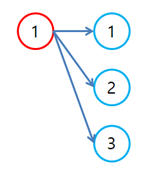
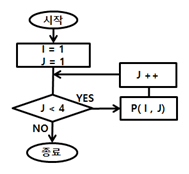
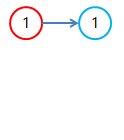


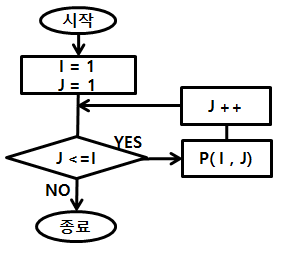
문제 2) 상위 이미지와 같은 형태의 구구단을 출력 할 수 있는 기찻길 순서도 의사 코드를 만들어 보자.

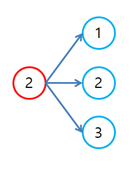
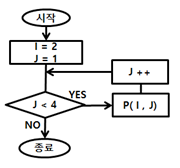
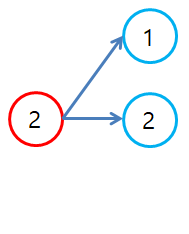
다음은 빨간색 i값이 파란색 j값에 영향을 주는 경우이다.

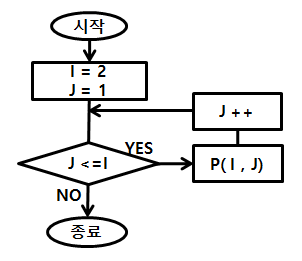


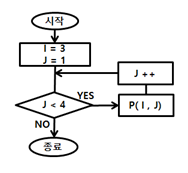
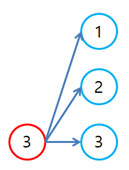
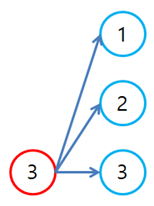
상위를 실행시키면 다음과 같은 결과가 나올 것이다. (1, 1) (2, 1),(2, 2) (3, 1),(3, 2),(3, 3) 빨간색 공을 i라 하고 파란색 공을 j라 한다면 i값은 j값을 찍는데 영향을 준다. i값이 1일때 j값은 1 한개 i값이 2일때는 j값 1,2 두 개 i값이 3일 때는 j값은 1, 2, 3 세개가 출력 된다. i값에 따라 출력되는 결과가 규칙성 있게 달라지고 있고, 이렇게 i값이 j값 반복에 영향을 주려면 어떻게 해야 하는지 생각해 보자. i값이 j값 반복에 영향을 주려면 j값 반복문 안에서 i값을 사용해야 한다.



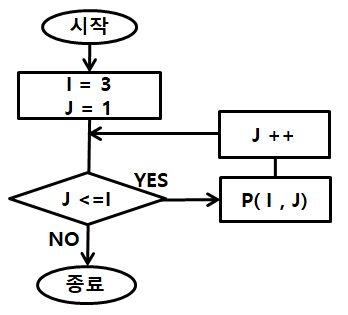
상위 첫번째 이미지와 두번째 이미지에서 반복문 밖의 i값이 반복문의 안쪽에서 출력 할때 영향을 주었다.p(i,j) 두번째 순서도 구조를 유지하면서 세번째 형태로 만들려면 i값을 가지고 반복문의 조건식에 영향을 주어야 한다. 무조건 3번 반복될 것을 i값에 따라서 반복 횟수를 줄여야 한다. 그래서 조건식을 왼쪽 처럼 J<i+1 과 같이 바꾸거나 J<=i로 바꾸어 주면 된다. 이렇게 하면 조건식을 첫번째 돌때 j값이 1이고 i값은 1이므로 1<=1 비교가 true여서 반복 부분를 반복하고 두번째 돌때 j값이 2이고 i값은 여전이 1이여서 2<=1 비교가 false여서 반복문을 빠져나와 반복문이 종료 된다. 출력 결과는 (1, 1)과 같다.



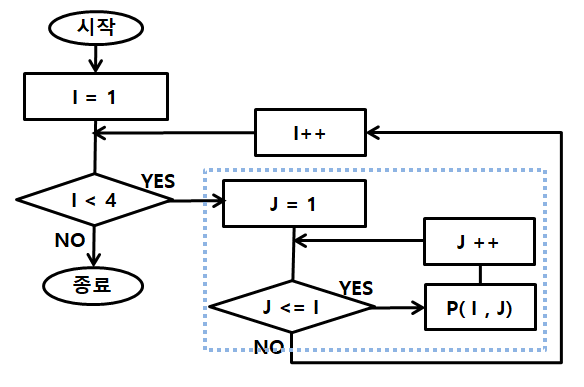
상위 첫번째 이미지와 두번째 이미지에서 반복문 밖의 i값이 반복문의 안쪽에서 출력 할때 영향을 주었다.p(i,j) 두번째 순서도 구조를 유지하면서 세번째 형태로 만들려면 i값을 가지고 반복문의 조건식에 영향을 주어야 한다. 무조건 3번 반복될 것을 i값에 따라서 반복 횟수를 줄여야 한다. 그래서 조건식을 왼쪽 처럼 J<i+1 나 J<=i로 바꾸어 주면 된다. 이렇게 하면 조건식을 첫번째 돌때 j값이 1이고 i값은 2이므로 1<=2 비교가 true여서 반복 부분을 반복하고 두번째 돌때 j값이 2이고 i값은 여전히 2이여서 2<=2 비교가 true여서 반복 부분을 반복하고 세번째 돌대 j값이 3이고 i값은 여전히 2여서 3<=2 비교가 false이므로 반복문을 빠져나와 반복문이 종료 된다.



상위 첫번째 이미지와 두번째 이미지에서 반복문 밖의 i값이 반복문의 안쪽에서 출력 할때 영향을 주었다.p(i,j) 두번째 순서도 구조를 유지하면서 세번째 형태로 만들려면 i값을 가지고 반복문의 조건식에 영향을 주어야 한다. 무조건 3번 반복될 것을 i값에 따라서 반복 횟수를 줄여야 한다. 그래서 조건식을 왼쪽 처럼 J<i+1 나 J<=i로 바꾸어 주면 된다. 이렇게 하면 조건식을 첫번째 돌때 j값이 1이고 i값은 3이므로 1<=3 비교가 true여서 반복 부분을 반복하고 두번째 돌때 j값이 2이고 i값은 여전히

3이여서 2<=3 비교가 true여서 반복 부분을 반복하고 세번째 돌때 j값이 3이고 i값은 여전히 3이여서 비교식 3<=3의 결과가 true여서 반복 부분을 반복하고 세번째 돌대 j값이 4이고 i값은 여전히 3이여서 4<=3 비교 결과가 false이므로 반복문을 빠져나와 반복문이 종료 된다.

상위 3개의 순서도는 i값이 1에서 하나씩 증가하여 3이 될때까지 반복 되는것 말고는 동일하다.

왼쪽은 i값을 1, 2, 3씩 변경 하면서 점선 부분을 반복하는 순서도이다. 

문제 3) 다음 문자열을 출력 할 수 있는 기찻길 순서도 의사 코드를 만들어 보자.

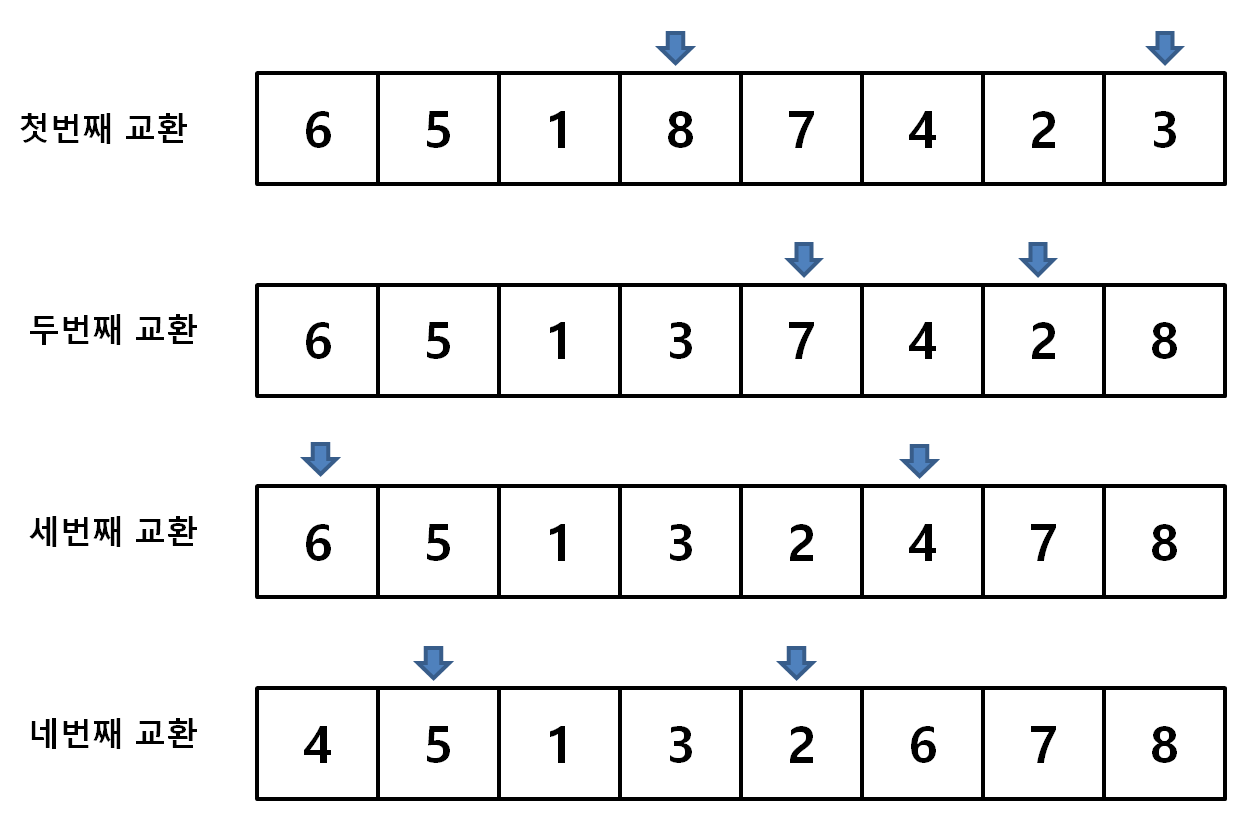
1. \*엔터\*\*엔터\*\*\*엔터\*\*\*\*엔터\*\*\*\*\*엔터\*\*\*\*\*\*엔터\*\*\*\*\*\*\*엔터
2. \*\*\*\*\*\*\*엔터\*\*\*\*\*\*엔터\*\*\*\*\*엔터\*\*\*\*엔터\*\*\*엔터\*\*엔터\*엔터
3. 공간공간공간공간공간공간\*엔터공간공간공간공간공간\*\*엔터공간공간공간공간\*\*\*엔터공간공간공간\*\*\*\*엔터공간공간\*\*\*\*\*엔터공간\*\*\*\*\*\*엔터\*\*\*\*\*\*\*엔터

공간을 진짜 공간 키보드의 스페이스 ‘ ‘로 찍고 \*은 \*로 찍고 엔터는 실제 엔터

줄바꿈으로 표시하면 아래와 같은 삼각형 모양이 나올 것이다.



다음 6,5,1,8,7,4,2,3을 값으로 가지는 배열이 아래와 같이 있을 때 배열의 내용을 1,2,3,4,5,6,7,8순으로 오름차 순으로 배열안의 내용을 바꾸어 정렬해 보자.



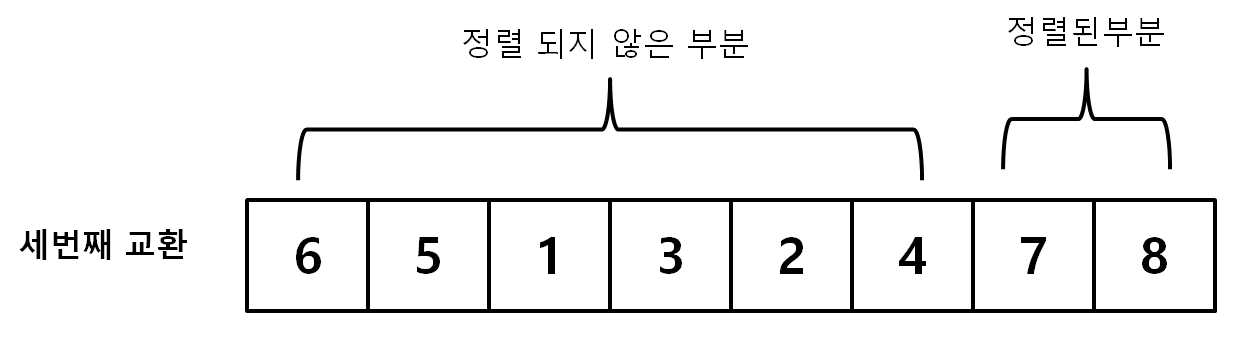
정렬 방법은 다음과 같다.

1. 배열안에서 가장 큰 수(첫번째 교환 부분의 첫 번째 화살표)를 찾고 배열의 마지막 위치(첫번째 교환 부분의 두 번째 화살표)와 두 값을 교환한다. 다음을 진행하고 나면 배열 안에 어떤 수가 들어 있더라 하더라도 가장 큰 수가 배열의 맨 마지막에 들어 간다.

2. 첫번째 교환이 이루어 지고 난 다음에는 배열의 맨마지막에 가장 큰 수가 들어가고 배열 안의 내용은 두번째 교환 부분과 같은 모양이 된다. 여기서 오름차순으로 배열 내용을 만든다고 생각해 보자. 맨 마지막 부분은 이미 정렬이 끝난 상태이므로더 이상 손을 댈 필요가 없다. 다음에 어떤 작업을 해할지 고민해보자.



3. 두번째 교환 부분에서 정렬 된 부분과 정렬되지 않은 부분을 구분한 다음 정렬된 부분은 더 이상 손대지 않으면 되고, 정렬되지 않은 부분중에서 가장 큰수 (두 번째 교환 부분의 첫 번째 화살표)를 찾아 정렬되지 않은 부분의 마지막 배열 부분(두 번째 교환 부분의 두 번째 화살표)의 값과 교환하면 세 번째 교환 부분 배열처럼 정렬이 된다.



4. 세번째 교환 부분에서 정렬 된 부분과 정렬되지 않은 부분을 구분한 다음 정렬된 부분은 더 이상 손대지 않으면 되고, 정렬되지 않은 부분중에서 가장 큰수 (세 번째 교환 부분의 첫 번째 화살표)를 찾아 정렬되지 않은 부분의 마지막 배열 부분(세 번째 교환 부분의 두 번째 화살표)의 값과 교환하면 네 번째 교환 부분 배열처럼 정렬이 된다. 이런 방법으로 계속 반복하다 보면 배열안의 모든 내용이 정렬될 것이다.

총 몇번 반복해야 정렬이 되는가?

3개라면 2개만 정렬되면 나머지 하나는 저절로 정렬이 된다. 따라서, 배열크기-1 한 만큼 반복하면 배열은 정렬이 된다. 이후 부터 배열의 크기를 배열이름.length로 표기 한다.

상위 내용을 정리해 보면 정렬하기 위해서 다음과 같은 과정을 거치면 된다.

0~7 까지 8개 중에 가장 큰수를 찾아 배열 인덱스 7에들어 있는 수와 교환해야 한다.

0~6 까지 7개 중에 가장 큰수를 찾아 배열 인덱스 6에들어 있는 수와 교환해야 한다.

0~5 까지 6개 중에 가장 큰수를 찾아 배열 인덱스 5에들어 있는 수와 교환해야 한다.

0~4 까지 5개 중에 가장 큰수를 찾아 배열 인덱스 4에들어 있는 수와 교환해야 한다.

0~3 까지 4개 중에 가장 큰수를 찾아 배열 인덱스 3에들어 있는 수와 교환해야 한다.

0~2 까지 3개 중에 가장 큰수를 찾아 배열 인덱스 2에들어 있는 수와 교환해야 한다.

0~1 까지 2개 중에 가장 큰수를 찾아 배열 인덱스 1에들어 있는 수와 교환해야 한다.

반복되는 숫자를 확인해 보면 반복문으로 바꿀수 있을 것이다.

문제 1) 상위 설명은 오름차 순으로 정렬한 것이다 내림차 순으로 정렬한 방법을 기찻길 순서도 의사 코드로 작성해 보자.

|  |
| --- |
| > 09. 블랙잭 순서도 |

어려울 수 있으니 이 챕터는 하지 않아도 된다. 재미로 보고 말자.

2명의 player가 블랙잭을 하는 프로그램을 구현하여 보자.

게임 방법은 트럼프 카드 52장을 2명의 player가 번갈아 가면서 1장씩 원하는 만큼의 카드를 받아 각자의 카드의 합이 21에 가까운 사람이 이기는 게임이다.

카드는 각자가 원하는 만큼 받을 수 있다. 각자 번갈아 가면서 카드를 받고 싶은지 확인해서 받기를 원하면 1장씩 받고 한 명이 더 이상 카드를 받길 원치 않으면 혼자서 카드를 받는다. 그리고 둘다 더 이상 카드를 받길 원하지 않으면 각자 받은 카드의 합이 21에 가까운 사람이 이긴다. 만약 카드를 받는 중 21일이 넘어가면 무조건 게임에서 패하게 되고 게임은 종료된다. 둘다 동시에 21이 넘어가게 되면 카드를 먼저 받은 사람이 지게된다.

카드별 카운트 해야할 숫자는 다르다 카드 A 의 경우에는 1과 11중 원하는 수를 카운트 한다. 2~10 카드는 카드 숫자 만큼 추가 한다. J,Q,K 카드는 10으로 계산 한다.

필요한 데이터를 확인해보자.

deck[] 카드 52장

CardShape[] {“스페이드”,”하트”,”다이아”,”나무”}

출력할 때 사용할 카드 모양 배열을 만들 었다.

CardNumber[] {“A”,”2”,”3”,”4”,”5”,”6”,”7”,”8”,”9”,”10”,”J”,”Q”,”K”}

출력할 때 사용할 카드 넘버를 배열로 만들 었다.

deckIndex 사용할 카드의 인덱스를 의미한다. 0이면 카드를 하나도

사용하지 않은 것이고 52이면 모두 사용한 것이다.

p1Deck[] 1번 플레이어가 취득한 카드

p2Deck[] 2번 플레이어가 취득한 카드

p1Flag 1번 플레이어가 카드 취득 여부

p2Flag 2번 플레이어가 카드 취득 여부

p1Index 1번 플레이어가 취득한 카드갯수

p2Index 2번 플레이어가 취득한 카드갯수

playState 게임 상태를 의미한다. “”, “p1승리”, “p2승리”, “무승부”

게임중 화면 1

p1님 카드 받겠습니까? 0=no,1=yes 1 <--사용자 입력

p2님 카드 받겠습니까? 0=no,1=yes 1 <--사용자 입력

>>카드 현황

P1:하트9, p1총점>>9

P2:스페이드A, p2총점>>11

게임중 화면 2

p1님 카드 받겠습니까? 0=no,1=yes 1<--사용자입력

p2님 카드 받겠습니까? 0=no,1=yes 1<--사용자입력

>>카드 현황

P1:하트9,클로버5, p1총점>>14

P2:스페이드A,다이아7, p2총점>>18

게임중 화면 3

p1님 카드 받겠습니까? 0=no,1=yes 0<--사용자입력

p2님 카드 받겠습니까? 0=no,1=yes 1<--사용자입력

>>카드 현황

P1:하트9,클로버5, p1총점>>14

P2:스페이드A,다이아7,스페이드8, p2총점>>16

게임중 화면 4

p2님 카드 받겠습니까? 0=no,1=yes

0

두 플레이어가 카드 수령을 중지하였습니다.

게임종료후 화면

------------------

>>카드 현황

P1:하트9,클로버5,

p1총점>>14

P2:스페이드A,다이아7,스페이드8,

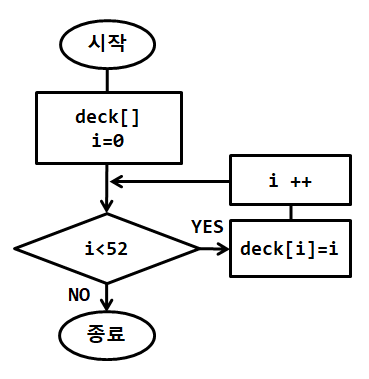
p2총점>>16

p2승리

동작순서

1. 카드 52장을 만든다.
   1. 0~51까지 고유한 번호를 넣어 카드를 구분한다.
2. 카드를 섞는다.
   1. deck배열의 내용을 섞는다.
3. 원하는 플레이어가 1장씩 번갈아 받는다.
   1. 카드를 준다는 것은 deck 배열에서 deckIndex의 위치에 카드 값을 배열 p1Deck, p2Deck에 넣어주면 된다.
   2. 카드를 주고 나면 deckIndex을 하나 증가시켜 다음 사용할 카드를 가리키고 p1Index나 p2Index을 증가하여 다음 넣을 곳을 가리킨다.
4. 카드를 받을지 여부에 플래그 p1Flag, p2Flag를 사용한다.
   1. 더이상 카드를 받지 않는 플레이어는 카드 받을 지 여부를 물어 보지 않고 카드를 주지 않는다.
5. 카드를 받을 때 마다 21이 넘으면 게임을 종료하고 게임에 진것이 된다.
6. 둘다 카드를 더이상 받지 않으면 게임을 종료하고 승패를 종료한다.
7. 카드 게임중 플레이어가 가지고 있는 카드를 주고나서 혹은 게임이 끝나고 나서 화면을 보여 준다.
   1. 게임화면 만들때 두명의 플레이어가 가진 모든 카드를 보여준다.
8. 카드 게임 종료전 게임에 승패를 화면에 출력 한다.

카드생성

순서도에서 사용하는 용어를 간단히 설명해보고 순서도를 확인해보자.

deck[]: deck배열: 전체 카드 배열: 게임을 하려면 전체 카드를 저장해 놓은 배열이 필요하다.

deck[i]: 카드의 식별자: deck배열에 들어 있는 카드 중 deck배열의 i번째 들어 있는 하나의 카드 식별자로 0~51사이의 숫자가 된다: 52장의 카드를 구분하는 숫자.

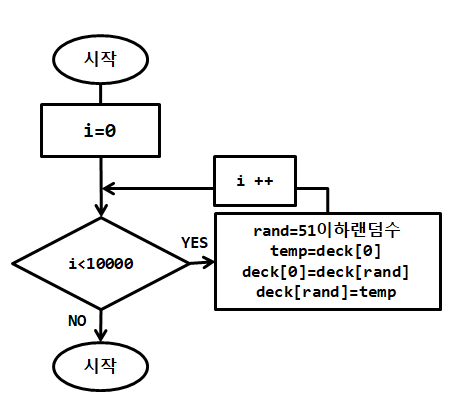
i: 카드의 위치 혹은 카드의 식별자: 카드식별자 값 혹은 deck배열에서 몇번째 위치하는 지에 대한 정보: 카드 식별자 혹은 카드를 쌓아 놓으면 층이 생기는데 결국 몇번째 층에 카드를 읽어올 것인지 결정하는 카드의 위치, 즉 deck배열의 인덱스로 사용한다.

deck[i]=i에서 앞에서 사용한 deck[i]의 i는 배열의 인덱스 카드의 위치를 나타낼때 사용한 것이고 뒤에 있는 i는 카드의 식별자 즉, 스페이드 1인지 2인지 를 구분할 수 있는 식별자이다. deck[i]에서 i는 카드의 위치를 나타내고, deck[i]는 i위치에 들어 있는 카드의 식별자를 의미한다.

프로그램에서의 카드를 생성한다는 것은 모든 카드를 구분할 수 있도록 만들면 된다. 0부터 51번까지의 번호를 넣어두고 0은 스페이드A, 1은 스페이드2, 2는 스페이드3, ... 13은 크로버A, 14는 크로버2…… 이런 식으로 0~51를 가지고 카드를 식별 할 예정 이다. 이러한 개념으로 카드1벌을 프로그램으로 만든다면 0~51까지의 숫자를 넣은 배열을 만들면 카드 1벌이 만들어진 것이고 1벌을 deck이라 한다. i를 0~51까지 반복해서 deck[0] ~ deck[51]까지 배열에 0~51까지의 식별자를 순서대로 넣으면 식별 가능한 deck[]배열 카드 52장을 넣을 수 있게 된다. deck[i]=i에서의 카드 위치인 배열인덱스 i와 카드 식별값i를 명확하게 구분해야 한다.

배열인덱스 i는 카드덱에서의 위치를 의미한다. 카드가 쌓여 있다고 생각해보자. 52장이 층을 이루고 있을 것이고 맨 상단의 카드가 배열의 인덱스 i 값 0에 해당 할 것이다. 그럼 실제로 맨위에 있는 카드가 무슨 카드인가? 그 카드에 들어 있는 카드값 (카드의 식별자)가 무엇인지 알아야 할것이다. deck[i]가 카드의 식별자이다.

상위 순서도를 차례대로 따라가 보면 카드 1벌이 만들어질 것이다.

카드섞기

deck[]: deck배열: 전체 카드 배열: 게임을 하려면 전체 카드를 저장해 놓은 배열이 필요하다.

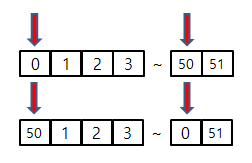
temp: 임시로 카드식별자를 저장할 공간: deck배열의 특정 위치 2개를 교환하려면 임시로 저장할 공간이 필요하다.

rand:51이하의 랜덤수: 카드의 0번 인덱스와 교환할 카드 인덱스를 구할때 사용한다.

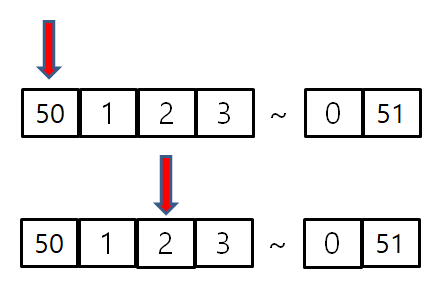
상위 순서도는 deck배열의 내용을 섞는 작업을 하는 것이다. 섞지 않으면 배열에 숫자가 들어 있는데 정렬되어 있다. 카드를 게임에 사용 하려면 카드를 섞어야 한다. 인덱스 0부터 순서대로 읽어 오면 다음 나올 카드를 누구나 알 수 있다.

프로그램에서 카드를 섞는다는 의미는 배열의 내용를 교환 한다는 의미이다.

잘 생각해보면 카드를 섞으면 해당 층의 카드 정보가 바뀔 뿐이지 카드 층 정보가 바뀌지 않는다. 카드를 쌓아 놓았 을 때 맨 위쪽을 0번 층 맨아래 층을 51번 층이라고 이야기 하였다면 카드를 섞고나서 카드를 쌓아 놓는다면 여전이 맨 위쪽층은 0층, 아래층은 51층이 된다. 결국 층정보는 변하지 않고 해당 층의 카드 정보만 변경된다.

카드 내용을 섞는 방법은 여러가지 있는데 왼쪽 이미지 처럼 인덱스 0번과 50번의 데이터를 교환 하는 것이다. 이렇게 교환이 되면 배열의 내용은 손실없이 섞이게 된다.

이때 이 50번 대신에 랜덤한 수를 구해서 0과 여러번 교환 하면 내용이 섞이게 된다. 상위 순서도는 카드를 만번 섞는 방법을 순서도로 만든 것이다.

deck에서 카드 한장 사용하기

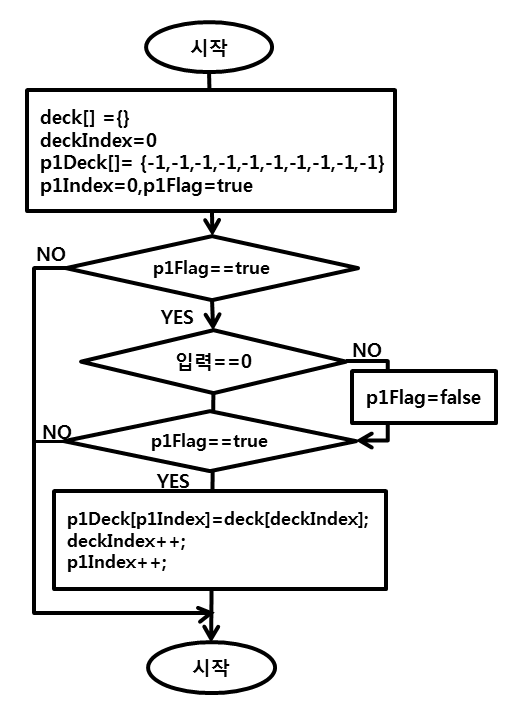
프로그램에서 카드덱에서 카드 한장을 받는다는 의미는 카드 덱 배열에서 사용한 카드 부분과 사용하지 않은 카드 부분, 다음 사용할 카드를 확인 할 수 있다는 이야기 이다. 프로그램에서 deckIndex를 가지고 확인 할 수 있다. deckIndex가 가지고 있는 수를 기준으로 deckIndex보다 작은 수의 deck배열의 인덱스는 이미 사용한 카드이고 deckIndex와 같은 위치의 deck배열에 인덱스 값은 다음에 사용할 카드가 되고, deck배열에서 deckIndex보다 큰 인덱스에 존재하는 값들은 다음에 사용할 카드들이다. 상위 이미지에서 위쪽은 게임하기전 최초 카드를 만들었을때 상황을 표현한 것인데 빨간 화살표 부분이 배열에서 다음 사용할 카드를 가르키는 deckIndex고 현상황은 deckIndex가 0 이여서 카드가 섞여진 deck배열 의 맨처음을 가리키고 있다. deckIndex가 0 이여서 deckIndex보다 작은 deck배열의 인덱스가 존재하지 않으므로 사용한 카드가 없고 deckIndex가 0이므로 deck배열의 인덱스가 0인 0번째 카드 50이 다음에 사용 할 카드 이고 1, 2, 3~, 0, 51이 아직 사용하지 않은 카드이다.

여기서 카드를 한장 사용 한다고 생각해보자. 현재 deckIndex가 0 이므로 deck 배열의 인덱스 0인 카드를 사용 할 것 이다. 카드를 사용하고 나면 다음 두가지 작업을 해야 한다. 첫번째는 deck배열의 0번 인덱스의 카드를 deckIndex보다 작게 만들어 사용한 카드로 만들어야 한다. 두번째 deckIndex의 값을 아직 사용하지 않은 deck배열의 인덱스로 바꿔주는 작업을 해야한다. 이 두가지 조건을 deckIndex++로 만족 시킬 수 있다.

카드 하나를 사용하여 deckIndex가 하나 증가하면 deckIndex는 1이 되고 이보다 작은 deck배열의 인덱스는 0이 존재하고 deckIndex보다 작은 인덱스이니 실제로 사용한 카드 이다. deckIndex가 1이되면 deck배열에서 인덱스1인 카드가 다음에 사용할 카드가 되고 deckIndex보다 큰 deck배열의 인덱스 값들은 아직 사용하지 않은 카드가 된다.

결국 카드 deck배열에서 카드 한장씩 읽어오는 방법은 deckIndex와 동일한 deck배열의 카드 식별자을 읽어온다음 deckIndex를 하나 증가시켜 주면된다.

위쪽 이미지에서 아래부분은 deck배열의 카드를 2장 사용하여 deckIndex가 2가되고 사용한 카드가 50,1 다음 사용할 카드가 2, 아직 사용하지 않은 카드가 3 ~ 0,51 인 상태를 보여주고 있는 것이다.

사용자가 카드 한장 받기

p1Deck[]: p1 플레이어의 카드 배열: 게임중 플레이어가 받은 카드를 의미한다.

p1Index: p1 플레이어가 가지고 있는 카드 수를 의미 한다.

p1Falg: p1 플레이어가 카드를 받을지 여부를 결정하는 플래그이다.

deck배열에서 카드 한장을 사용하면 사용자가 카드 한장을 받는 작업을 프로그 램으로 구현하여야 한다. 사용자가 카드 한장을 받는 다는 것은 p1Deck 배열에 카드를 넣는것을 의미한다.

여기 순서도에서는 카드 한장을 받는 것으로 구현 되어 있지만 실질적으로 사용자가 지속적으로 카드를 여러장 받아야 한다. 한장 받는 작업을 여러번 하면된다.

일단 사용자가 이전에 카드를 그만 받기로 결정 하지 않았다면(p1Flag값이 true이면 카드를 계속 받기로 한 것이다.), 카드를 그만 받을 것인지 계속 받을 것인지 물어 보고 사용자로 부터 입력을 받는다. 사용자가 1을 입력하면 계속 받을 예정이어서 p1Flag값을 변경하지 않고 true로 남겨두면 되고 0을 입력하면 더이상 카드를 받지 않을 예정이어서 p1Flag값을 false로 변경하여 더이상 카드 받는 작업을 그만 두게 한다.

사용자가 카드를 계속 받기를 원하면 deck배열에서 카드 한 장을 꺼내 p1플레이어의 카드를 보관하는 배열 p1Deck[]에 넣어 주면 된다. p1Deck[]에 -1를 10개 넣은 이유는 p1플레이어가 카드 10개를 받을 수 있도록 미리 세팅한 것이다.

deck에서 카드 한장을 p1Deck[]에 넣는 방법은 deck 배열에서 다음에 사용할 카드 인덱스가 deckIndex이고 p1Deck[] 배열에서 p1플레이어가 받은 카드를 저장할 인덱스 번호가 p1Index이므로 p1Deck[p1Index]=deck[deckIndex]과 같은 방법으로 deck배열 에서 p1Deck배열로 카드 식별자를 복사하면 된다. 카드를 복사한 다음에는 deck에서 카드 한장을 사용 하였 으므로 deckIndex++를 이용하여 다음 사용할 카드로 이동 하여야 한다. p1Deck[]배열에 카드 한장이 들어 갔으므로 p1Index++를 이용하여 다른 카드를 저장 할 수 있도록 다음 저장 공간을 가리켜야 한다.

카드출력

0~51까지의 카드 식별자로 사용자가 알아보기 쉬운 스페이드A, 1은 스페이드2, 2는 스페이드3, ... 13은 클로버A, 14는 클로버2… 형태로 출력 할 수 있어야 한다.

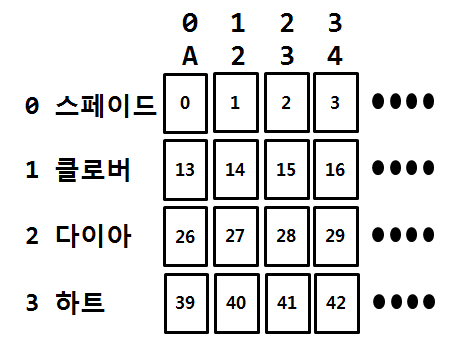
deck배열이나 p1Deck배열에는 0부터 51번까지의 카드 식별 번호가 들어 있을뿐 0은 스페이드A, 1은 스페이드2, 2는 스페이드3, ... 13은 클로버A, 14는 클로버2…… 이런 카드 정보는 들어 있지 않다. 0~51까지의 숫자를 가지고 스페이드A ~ 하트 k 까지 사용자가 알아보기 쉬운 형태로 화면에 출력 할 수 있어야 한다.

cardShape[]: 스페이드와 같은 카드 모양 종류를 문자열로 저장하는 배열이다.

cardNumber[]: A,1,2..와 같은카드 숫자 모양을 문자열로 저장하는 배열이다.

p1Deck[]:p1플레이어가 가지고 있는 카드묶음이다.

p1Index: p1플레이어가 가지고 있는 카드 개수이다.

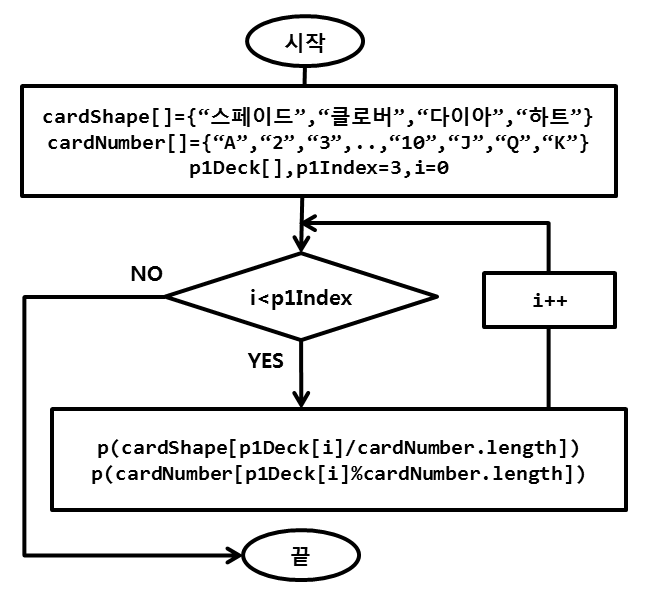
0~51번호를 가지고있는 52장의 카드를 잘 생각해 보면 4가지 모양을 13장씩 4등분 하여 가지고 있다. 0~12는 스페이드, 13~25는 크로버, 26~38는 다이아, 39~51는 하트가 된다. cardShape 배열에는 스페이드, 클로버, 다이아, 하트가 문자열로 들어가 있어 cardShape[0]를 출력하면 스페이드가 출력 된다. 카드 식별 번호가 0~12인 카드는 스페이드로 출력 해야 하고 cardShape[0]이 스페이드 문자열 이므로 0~12숫자를 0으로 만들면 문자열 스페이드로 cardShape를 이용해서 출력 할 수 있다. 0~12 사이의 숫자를 0으로 만들려면 13으로 나눈 몫을 구하면된다. 정수의 계산에서 정수/정수는 정수가 되어 소수점 부분이 없어진다. 13~25사이 숫자도 1로 만들려면 cardShape 배열을 이용하여 클로버로 출력 할 수 있고 13~25를 13로 나누면 1이된다. 26~38를 13으로 나누면 2가 되고 39~51를 13으로 나누면 3이 된다. 결론적으로 카드 식별자를 13으로 나눈 몫을 cardShpe배열의 인덱스로 사용하면 카드의 모양을 문자열로 출력할 수있다. 사용자가 가지고 있는 첫번째 카드의 카드 식별 번호는 p1Deck[0]이고 이식별 번호를 13으로 나누면 실제 카드의 모양이된다. 13은 한 모양의 카드가 표현할 수 있는 숫자 모양 개수이므로 cardNumber배열의 개수와 동일하여 13 대신에 cardNumber.length를 사용하였다. p1Deck[0]/cardNumber.length 이렇게 하면 

p1플레이어가 가지고있는 1번째 카드식별번호/13이므로 0~3까지의 숫자중 하나가 만들어져 cardShape배열의 인덱스로 사용되어 스페이스인지,클로버인지.. 화면에 출력 할 수 있다. 결국 다음을 실행하면 p(cardShape[p1Deck[0]/cardNumber.length]) 결과로 스페이드,클로버,... 처럼 해당 카드에 카드 모양이 출력된다.

다음은 카드 식별 번호로 해당 카드 넘버를 출력하는 방법을 생각해보자. 카드넘버는 A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K 13개가 있다. 상위 이미지를 살펴보면 카드 식별 번호를 13으로 나눈 나머지가 카드 번호가 된다. 0,13,26,39를 13으로 나누면 0이되고 cardNumber배열에 인덱스를 0으로 넣으면 cardNumber[0]이되고 이것은 A에 해당한다. 나머지가 1이면 2가 되고 3이면 4 가되는 것을 확인 할 수 있다. 결론은 p1플레이어가 가지고 있는 첫번째 카드의 숫자를 문자열로 출력하고 싶다면 p(cardNumber[p1Deck[0]%cardNumber.length])이라고 하면 된다. p1플레이어의 첫번째 카드모양과 카드 번호를 출력하고 싶다면 아래처럼 2줄 써주면 될것이다.

p(cardShape[p1Deck[0]/cardNumber.length]) p(cardNumber[p1Deck[0]%cardNumber.length])

아래 순서도는 플레이어가 가지고 있는 카드 3장을 모두 출력하는 순서도이다. p1Deck 배열에 들어 있는 카드 식별 번호를 이용해서 반복하여 카드별로 모양과 번호를 출력하는 순서도 이다.



점수계산

플레이어가 가지고 있는 카드를 모두 더하여 점수를 계산하는 순서도이다.

p1Index: p1플레이어가 가지고 있는 카드 개수이다.

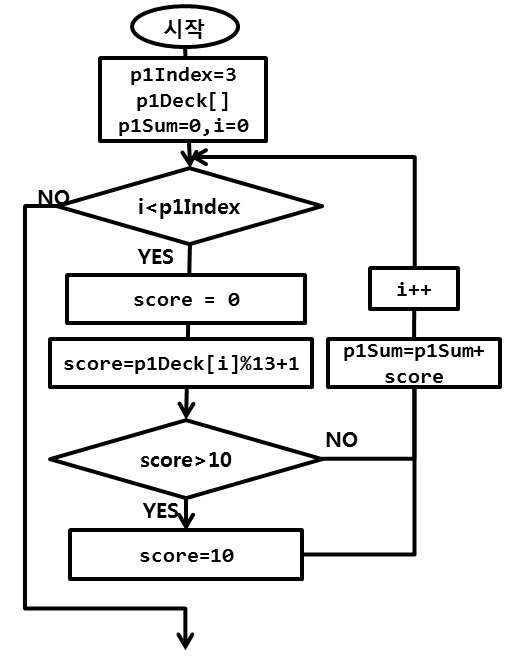
p1Deck: p1플레이어가 가지고 있는 카드 식별자 배열이다.

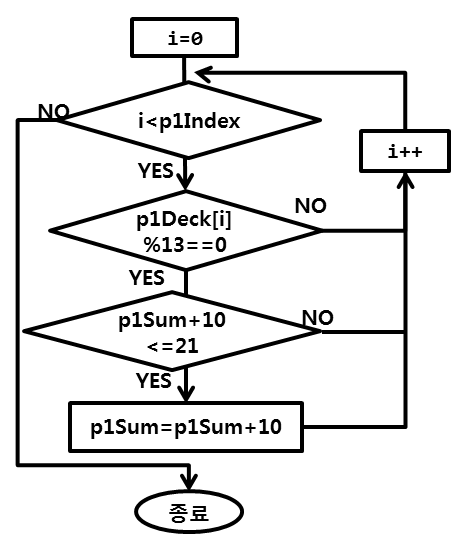
p1Sum: p1플레이어가 가지고 있는 카드들의 총합을 구하기 위해서 누적하는 변수이다.

scoure: 카드 1개의 점수이다.

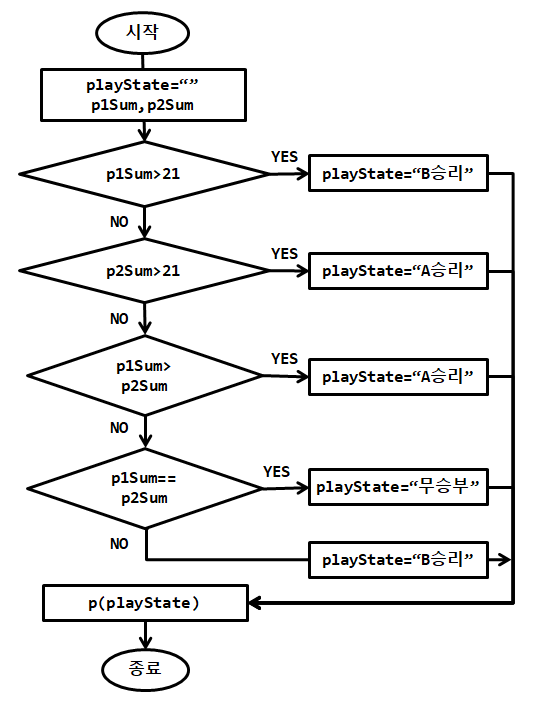
카드별 카운트 해야할 숫자는 다르다 카드 A 의 경우에는 1과 11중 원하는 수를 카운트 한다. 2~10 카드는 카드 숫자 만큼 추가 한다. J,Q,K 카드는 10으로 계산 한다.

일단 A같은 경우는 1과 11로 원하는 수를 선택해서 구할 수 있지만 1로 생각하고 계산한 다음 차후 설명할 예정이다. 카드식별자숫자%13한 결과는 0,1,2,3,4… 일때 A,2,3,4,5… 카드에 해당하고 카드점수는 카드식별자숫자%13한 결과에 +1한 것과 같아야 한다. 아래 순서도를 보면 i<p1Index로 p1플레이어가 가지고 있는 카드수 만큼 반복하고 있고 카드 한장을 score로 선언한 다음 0으로 초기화한후 사용자가 가지고 있는 p1Deck 배열에서 카드식별자를 읽어와 사용자가 가지고 있는 카드 1장의

카드 식별자 숫자를 읽어와 (p1Deck[i]) %13으로 나눈후 +1를 해주면 p1플레이어가 가지고 있는 한장의 카드 점수를 알수 있다. J,Q,K카드는 카드 식별 숫자를 %13으로 나누면 11, 12, 13이되고 이들 카드를 10 으로 계산하기로 되어 있으므로 만약 카드 점수인 scour가 10을 넘어간다면 11, 12, 13 인 경우 이므로 scour를 10으로 변경하여 천체 카드점수를 저장하는 p1Sum변수에 넣어주고 다음 카드를 합산하기 위하여 i를 하나 증가 시켜주고 i<p1Index로 p1플레이어 가 가지고 있는 카드 수만큼 반복 하여 p1플레이어가 가지고 있는 전체 카드 점수를 더한 p1Sum 값을 구할 수 있다. 총합산이 구해지면 전체 카드 점수 p1Sum값을 구한 것처럼 

보이지만 A 가 상황에 따라 1 또는 11로 변경 될 수 있다는 부분을 아직 적용하지 않았다. 왼쪽 순서도는 상황에 따라 A를 11로 변경해서 p1플레이어의 총합산 p1Sum에 적용하는 순서도이다.

카드를 처음부터 순회 하기 위해서 i를 0으로 다시 초기화 하였고 i<p1Index 를 이용하여 카드 수만큼 반복하였다. p1Deck[i]%13==0 부분은 카드 식별자 숫자가 0 인지 즉 A 인지 확인 하는 방법이다. A가 아니라면 아무런 작업을 할 필요가 없고 A라면 이미 1이 더해져 있는 상태여서 A를 11로 처리하고 싶다면 10만 더해주면 되는데 게임상 21이 넘으면 게임에 진것이 되므로 더한 후의 값이 21이 넘는 지 확인하여 21이하라면 전체합을 나타내는 p1Sum변수에 10을 더해 주면 A를 11로 변경한 것이 되고 21이 넘는다면 A를 1로 적용한 상태로 남겨 두면 된다. 이런 식으로 i를 하나 증가 시켜 p1Deck[i]에 또 다른 A가 있는지 p1Deck[i]배열을 탐색하여 A가 있을 때마다 상위와 같은 작업을 반복하면 A를 1 또는 11로 적용한 카드의 총점수인 p1Sum이 나온다.

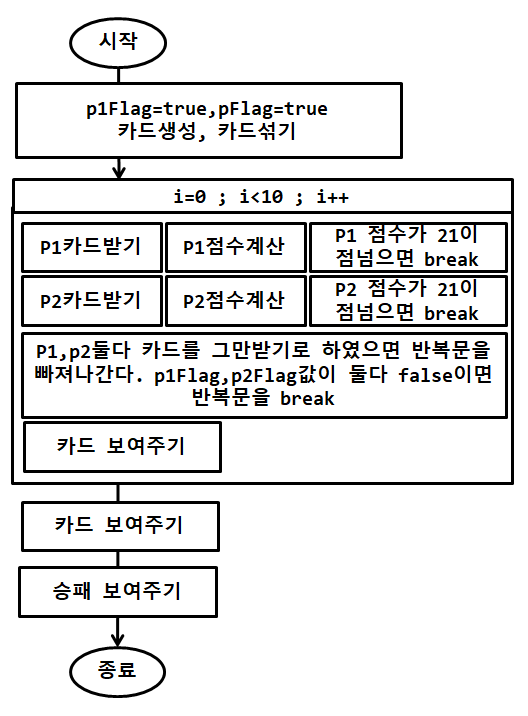
승패출력

p1Sum과 p2Sum은 p1,p2 플레이어가 가지고 있는 카드의 총합이다. playState는 두명의 플레이어중 누가 이겼는지 졌는지 문자열 로 만들어 저장 하는 변수이다. 게임 룰이 21를 넘으면 게임에서 지기 때문에 둘다 21이 넘으면 B가 승리하는 것으로 처리하였다.

게임진행

p1Flag는 p1플레이어가 카드를 계속 받을 것인지 받지 않을 것인지 결정하는 플래그 값이다. true이면 카드를 받는다.

p2Flag는 p1플레이어가 카드를 계속 받을 것인지 받지 않을 것인지 결정하는 플래그 값이다. true이면 카드를 받는다.

i=0;i<10;i++은 박스 부분을 10번 반복하라는 이야기이다. 

게임을 시작하면 카드생성과 카드섞이를 한 다음 p1플레이어가 카드 받을지를 경정하고 카드를 받으면 받은 카드의 점수를 계산한 다음 카드점수가 21이 넘으면 반복문을 빠져나가 게임을 종료하고 그렇지 않으면 p2플레이어가 카드 받는 작업을 진행한다. 이작업은 최대 10번 반복할 수 있고 중간에 둘다 카드 받는 작업을 더이상 원하지 않으면 보유한 카드를 보여주고 승패를 보여주고 게임을 종료한다.

