14. 참조표(lookup table)

• 대학 4학년생인 남자가 졸업할 때가 되었습니다. 이 학생의 최대의 관건은 (1) 직장은 구했느냐? (2) 결혼할 여자는 있느냐? 입니다. 유무에 따라 4가지로 구분해 보면 다음과 같습니다.

(1) 직장도 있고(=1), 애인도 있다(=1): 금메달
(2) 직장은 있고(=1), 애인은 없다(=0): 은메달
(3) 직장은 없지만(=0), 애인은 있다(=1): 동메달
(4) 직장도 없고(=0), 애인도 없다(=0): 목메달

• 위와 같은 얘기를 프로그래밍 한다고 생각해 봅시다.
• 직장의 유무와 애인의 유무를 관리하기 위해 우리는 2개의 변수를 선언할 수 있습니다.

```
#include <stdio.h>

class CAlumni {
    int bWork,bLover;
  public:
    CAlumni(int bWork,int bLover) {
        CAlumni::bWork=bWork;
        CAlumni::bLover=bLover;
    }//CAlumni
    void State();
};//class CAlumni

void CAlumni::State()
{
```

```
    int i;
    if (bWork==1 && bLover==1) i=0;
    else if (bWork==1 && bLover==0) i=1;
    else if (bWork==0 && bLover==1) i=2;
    else if (bWork==0 && bLover==0) i=3;

    if (i==0) printf("금메달\n");
    else if (i==1) printf("은메달\n");
    else if (i==2) printf("동메달\n");
    else if (i==3) printf("목메달\n");
}//CAlumni::State

void main()
{
    CAlumni kim(1,0);

    kim.State();
}//main
```

• 두 번째 if 블록은 i를 인덱스로 하고, 출력할 문자열을 내용으로 하는 테이블(Table)을 이용하면 간단하게 구현됩니다. 배열이 이러한 용도로 사용되었을 때, **참조표**라고 합니다.

```
char sTable[][7]={"금메달","은메달","동메달","목메달"};//전역변수

void CAlumni::State()
{
    int i;
    i=(bWork<<1) | bLover;
    printf("%s\n",sTable[i]);
}//CAlumni::State
```

점수에 따라 등급을 출력하는 경우

- 성적 처리 프로그램을 만드는 데, 등급을 출력하기를 원한다고 합시다.

점수대	등급
90~100	수
80~89	우
70~79	미
60~69	양
0~59	가

 등급표

```
char *sTable[]={"가","가","가","가","가",
                "가","양","미","우","수","수"};
int i;

scanf("%d",&i);
printf("%s\n",sTable[i%10]);
```

기교(technique)

(1) 0과 1을 토글(Toggle)하는 경우: 프로그램에서 변수의 값을 0↔1을 토글하기 위해서는 다음의 문장을 사용합니다.

```
int i=0;
i^=1;//비트 배타합(eXclusive Or) 연산의 특징을 이용합니다.
```

- 1과 3을 토글하려면, 다음과 같이 할 수 있습니다.
- 먼저 0과 3의 Xor값을 미리 계산합니다.

0001 ^ 0011 → 0010

- 두 수의 배타합의 결과인 2와 기존 값의 배타합 연산을 적용하면 임의의 두개의 정수 값을 토글할 수 있습니다.

```
int i=1;
i^=2;//1과 3을 토글합니다.
```

(2) 1과 -1을 토글하는 경우

```
int i=1;
i=-i;
```

(3) 0,1,2,3을 토글하는 경우: 2개 보다 많은 연속된 숫자를 토글해야 하는 경우는 나머지 연산자 %를 사용할 수 있습니다.

```
int i=0;
i=(i+1)%4;
```

- 만약, 5,6,7,8을 토글하고 싶다면, 다음과 같이 합니다.

```
int j=0, i;
j=(j+1)%4;
i=j+5;
```

임의의 숫자열을 토글하려면

- 토글을 원하는 숫자열이 아래와 같이 규칙성이 없는 임의의 숫자라고 가정해 봅시다.

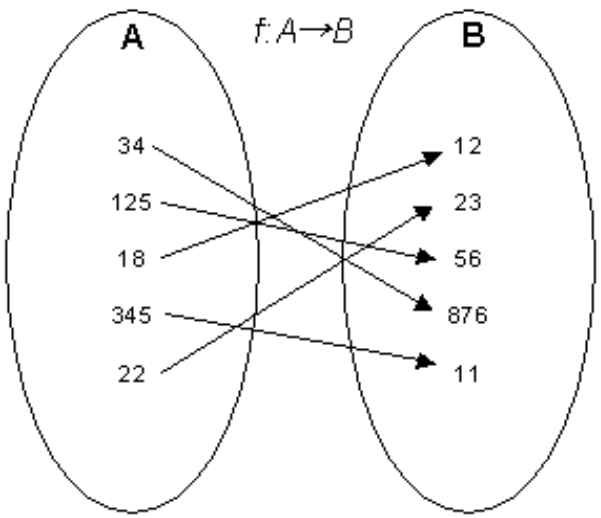
100→34→200→79

- 이러한 숫자열을 토글하는 방법은 참조표를 이용하는 것입니다.

```
const int toggle[]={100,34,200,79};
int j=0, i;
i=(i+1)%4;
j=toggle[i];//얼마나 간단한가요!
```

임의의 사상(Mapping)을 구현

• 이번에는 아래 그림과 같은 함수를 구현하는 문제를 고려해 봅시다. 사용자가 입력하는 값에 따라, 대응하는 치역(Range)의 값이 출력되어야 합니다.



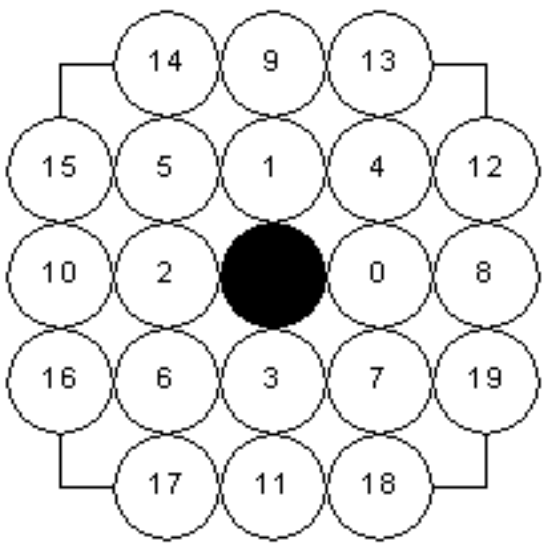
어떤 사상

```
#include <stdio.h>

int mapping[5][2]={{34,876},{125,56},{18,12},{345,11},{22,23}};

void main() {
    int i,n;
    scanf("%d",&n);
    for (i=0;i<5;++i)
        if (n==mapping[i][0]) {
            printf("%d\n",mapping[i][1]);
            break;
        }//if
}//main
```

실습문제

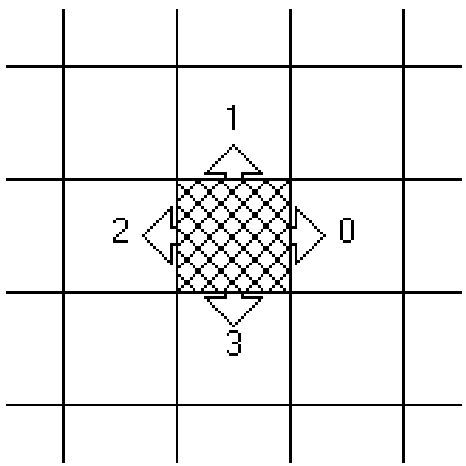
1. 512와 1024를 토글하는 문장을 작성하세요.(힌트: 512 XOR 1024)

2. 변위(Offset)가 포함된 좌표의 검사에는 참조표를 사용하여야 합니다. 아래 그림은 **컴퓨터 바둑(Computer Go)**에서 돌(Stone)들간의 관계를 고려하기 위해, 사용할 참조표입니다. 예를 들면, 중앙의 검은 돌 입장에서, 8과는 한칸, 13과는 날일자(日)의 관계가 있습니다. 이를 참조표로 구현하세요. 참조표의 인덱스가 그림처럼 정의되었다는 것에 주목하세요. 인덱스가 그림처럼 정의 되지 않으면, 어떤 불편함 점이 발생하는지 고려해 보세요.

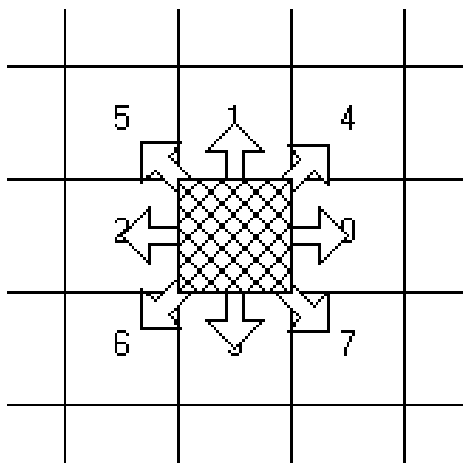


3. 한글 완성형 코드를 조합형 코드로 바꾸는 클래스를 설계하세요.

4. **연결된 요소(Connected Component)**를 발견하는 문제: 2차원 평면에서 셀(Cell)들간의 관계는 인접한 이웃(Adjacent Neighbor)으로써 표현하는데, **4-연결 이웃(4-Connected Neighbor: Von Neumann Neighbor)**과 **8-연결 이웃(8-Connected Neighbor: Moore Neighbor)**이 일반적입니다.



Von Neumann



Moore

이웃

• 위 그림에서 셀에 표시된 번호는 참조표의 인덱스를 의미하는데, 주의해서 보기 바랍니다. 위와 같이 레이블을 설정하면, 한 개의 참조표만으로 2가지 이웃을 쉽게 정의할 수 있습니다.

• 참조표를 아래와 같이 전역변수로 정의할 수 있습니다.

```
int offset[8][2]={{1,0},{0,-1},{-1,0},{0,1},{1,-1},{-1,-1},{-1,1},{1,1}};
```

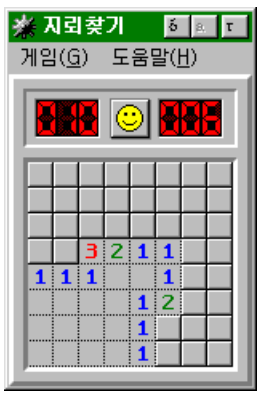
• 이제 이 참조표를 노이먼 이웃으로 이용하려면, 다음과 같이 합니다.

```
for (int i=0; i<4; ++i)
    if ((x+offset[i][0], y+offset[i][1]) is my condition) ...
```

• 무어 이웃으로 이용하려면, 다음과 같이합니다.

```
for (int i=0; i<8; ++i)
    if ((x+offset[i][0], y+offset[i][1]) is my condition) ...
```

(1) 지뢰찾기(Mine Sweeper) 게임에서 연속한 빈칸을 발견하는 클래스를 설계하세요.



왼쪽 그림에서 처럼, 사용자가 지뢰가 없는 빈 부분을 마우스로 클릭했을 때, 숫자가 있는 셀을 포함하여 빈 칸 모두를 표시해야 합니다. 이러한 셀들은 4-연결 이웃으로 구현하여야 합니다.

@