정보보호

HW04 : Vigenere Cipher
Useing Key Table

제 출 일	2018년 10월 01일
담당교수	류재철
학 과	컴퓨터공학과
학 번	201302423
이 름	신종욱

1. 키테이블을 이용한 비제네르 암호화

■ 과제해결 과정

일단 이전과제의 코드를 재활용하였다.
실제로 키테이블 배열을 이동시키면서 구현을 해도 되지만 그러면 많은 for을 한 개씩 이동해야 함으로 오버헤드가 커지는 단점이 있다.
그래서 더하기 빼기 연산과 나머지 연산를 암,복호화 때 적절한 사용하여 구현 하였다.

```
14    int mode;
15    char keytable[]="koegbpdwitncmfryqzulashxvj";//키테이블 값 선언
16    int key1,key2;//평행이동할 키값 저장용
17    printf(">> Input file name : ");
18    scanf("%s",inputFileName);
19    printf(">> Input mode [ 0 : ENC , 1: DEC ]");
20    scanf("%d",&mode);//암복호화 선택
21
22    printf("key table : %s\n",keytable);
23    printf("input 2 key : ");
24    scanf("%d %d",&key1,&key2);//평행이동할 키값을 받음
25    //2개를 받는데 한개의 키값으로 진행하고싶다면 중복으로 입력
```

키테이블 값은 하드코딩으로 하였다.
평행이동 할 킷값은 scanf로 2개를 받도록 하여서 구현하였다.
여기서 키 테이블은 항상 소문자임으로 복호화과정에선 대문자가 입력으로 들어오는 과정은 생각할 필요가 없다.

또 암호화의 출력은 항상 키 테이블에 의존함으로 소문자이다.
암호화일 땐 대,소문자->소문자라고 생각하고 구현했다.

```
45
46    int flag=0;//평행이동 키값 순환을 위한 변수
47    int key_modified;//평행이동된 값을 저장하기위한 변수
48
49    for(i=0;i<fileSize;i++)
50    {
51       fread(&buff,sizeof(char),1,input_FD);
52       if(buff<65||(buff>90&&buff<97)||buff>122) {//문자열이 아니라면 그대로
   넘어간다
53       }
54       else if(flag==0){//첫번째 문자열은 그대로 키값 사용
55       key_modified=0;
56       flag=1;
57       }
58       else if(flag==1){//첫번째 키값만큼 평행이동 필요
59       key_modified-=key1;//평행 이동시킨다는건 테이블이 이동했다고 생각해야>
   함으로 인덱스를 -나 +를 해야한다 둘중 아무거나를 해도 어차피 암,복호화 과정>
   이 반대라서 한개는 보정을 해야해서 어느것을 해도 상관이없다
60       flag=2;  //다음에는 두번째 키값으로 가야함으로 flag수정
61       }
62       else{//두번째 키값만큼 평행이동 필요
63       key_modified-=key2;
64       flag=1;  //다음에는 첫번째 키값으로 가야함으로 flag수정
65       }
66       if(key_modified<0){
67          key_modified+=26;
68       }//계산시 음수가 나올경우 +26을 더하여 보정함으로 배열접근을 유지
69       if(mode==0)
70       {
71          enc(&buff,key_modified,keytable);
```

이전 과제와 같이 받은 file사이즈를 이용하여 한글자씩 반복문을 실행하면서 암,복호화를 진행한다.

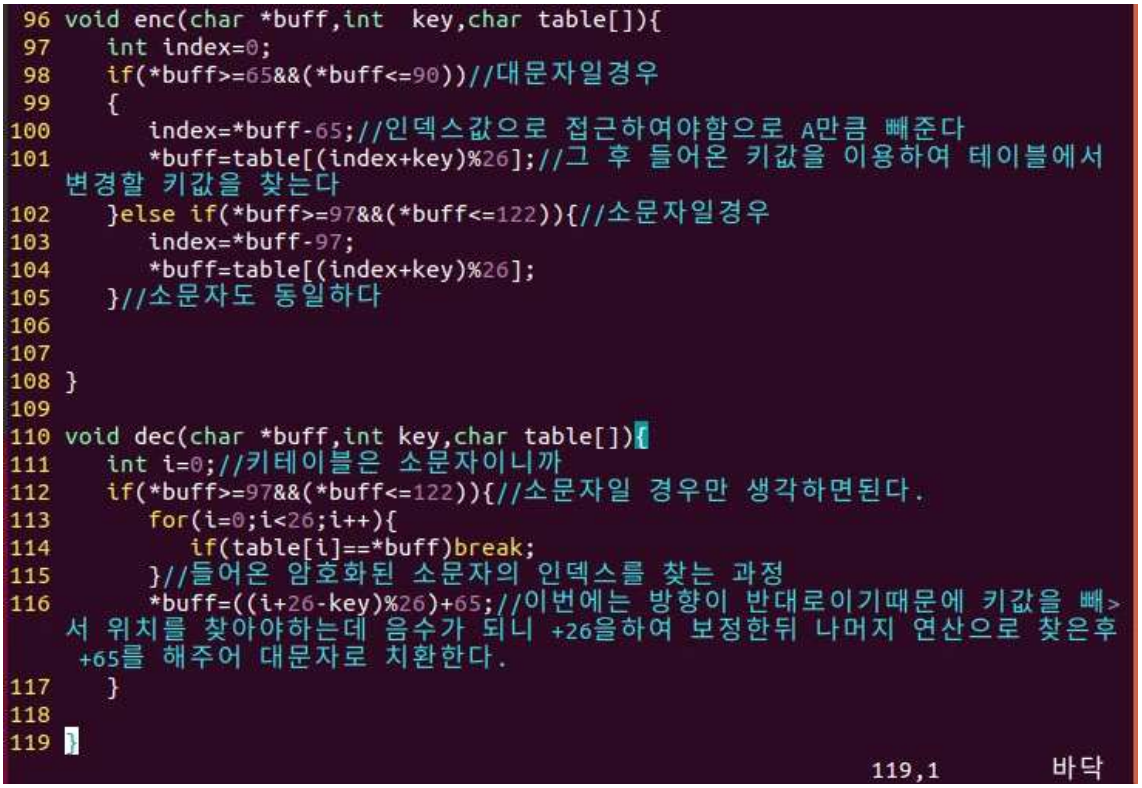
일단 문자열(알파벳)이 아닐 경우 그냥 무시하면 됨으로 바로 if(mode==0)문으로 가서 암,복호화를 진행한다 추후에 나올 암,복호화 코드에서도 문자열 판별을 하여 문자열이 아닐 경우에는 그대로 지나가도록 구현하였다.

flag 변수를 이용하여 바뀌는 평행이동 값을 구현하였다. 최초에는 평행이동이 없는 점을 착안하여 0으로 시작하고 1,2가 반복되어 서로 문자열일 때만 평행이동 하도록 구현하였다.

key_modified 변수는 암,복호화시 함수로 전달할 인자 값으로서 이때 평행이동 한 횟수를 기억하여 올바른 키테이블로 접근할 수 있도록 한다.

음수가 나올 경우에는 보정을 하여 오류를 막았다.

암,복호화시 이전함수와 다르게 키테이블이 매개변수로 추가되었다.



```
 96 void enc(char *buff,int  key,char table[]){
 97    int index=0;
 98    if(*buff>=65&&(*buff<=90))//대문자일경우
 99    {
100       index=*buff-65;//인덱스값으로 접근하여야함으로 A만큼 빼준다
101       *buff=table[(index+key)%26];//그 후 들어온 키값을 이용하여 테이블에서
    변경할 키값을 찾는다
102    }else if(*buff>=97&&(*buff<=122)){//소문자일경우
103       index=*buff-97;
104       *buff=table[(index+key)%26];
105    }//소문자도 동일하다
106
107
108 }
109
110 void dec(char *buff,int key,char table[]){
111    int i=0;//키테이블은 소문자이니까
112    if(*buff>=97&&(*buff<=122)){//소문자일 경우만 생각하면된다.
113       for(i=0;i<26;i++){
114          if(table[i]==*buff)break;
115       }//들어온 암호화된 소문자의 인덱스를 찾는 과정
116       *buff=((i+26-key)%26)+65;//이번에는 방향이 반대로이기때문에 키값을 빼>
    서 위치를 찾아야하는데 음수가 되니 +26을하여 보정한뒤 나머지 연산으로 찾은후
     +65를 해주어 대문자로 치환한다.
117    }
118
119 }
                                                    119,1         바닥
```

암호화시에는 평행이동 할 때 변경된 인덱스를 이용하여 키테이블에 접근하여 암호화 한다는 생각으로 구현했다.
입력받은 문자열을 일단 A는 0으로 B는 1로 바꾸기위해 A,a만큼 빼준후 key값을 더 한후 나머지 연산을 하면 올바른 암호화된 값을 찾을 수 있다.

복호화시에는 소문자일경우에만 작동하도록 하였고
입력받은 문자열을 키테이블에서 탐색하여서 찾을 때 break를 하여 I값을 이용해 킷값과 암호화랑은 반대로 –key 연산하여 평행이동 된 인덱스로 계산한 후 +65를 하여서 대문자로 치환하였다.

■ 실행결과 화면

1. plain.txt에 저장된 암호화 문자를 복호화 시키는 과정

```
shin@shin:~/Desktop/homework/3$ ls
bb.c  plain.txt  tablevi
shin@shin:~/Desktop/homework/3$ cat plain.txt
dppulz fa zes rmtwyyxu ihhb dw uogf qffch csw lcrta lalgsp dbwzgxnl xtuiusxue. c
gzwolg flunm pu gfw sntu lsdmzxa dzubdsk, r xywgix kkktec rirx jpzbiiahu yg fvz
erqf-dddyiz xccl. dzubdsk rkhikrc vj oltvp sexrgb, ykl 16py wpcfcbi qzqkue emzbp
fvoh aw ueh quyzbyx zkxy, jdspcbc(cf dwdqiu, lmozx). eaxmj ahcsnpy jwos unevgu s
ssbxd tk slpemb vvvwkt nluihbdwx dzn jffs zcs pp kviykrggv ud: byqeblv > yzzbpsj
 > xulpwho > vaawiue > xshinw
shin@shin:~/Desktop/homework/3$ ./tablevi
>> Input file name : plain.txt
>> Input mode [ 0 : ENC , 1: DEC ]1
key table : koegbpdwitncmfryqzulashxvj
input 2 key : 3 4
FILE Size = 430
shin@shin:~/Desktop/homework/3$ cat decrypt.txt
GIMCHI IS THE ACCEPTED WORD IN BOTH NORTH AND SOUTH KOREAN STANDARD LANGUAGES. E
ARLIER FORMS OF THE WORD INCLUDE TIMCHAI, A MIDDLE KOREAN TRAN SCRIPTION OF THE
SINO-KOREAN WORD. TIMCHAI APPEARS IN SOHAK EONHAE, THE 16TH CENTURY KOREAN RENDI
TION OF THE CHINESE BOOK, XIAOXUE(IN KOREAN, SOHAK). SOUND CHANGES FROM MIDDLE K
OREAN TO MODERN KOREAN REGARDING THE WORD CAN BE DESCRIBED AS: TIMCHAI > DIMCHAI
 > JIMCHAI > JIMCHUI > GIMCHI
```

구한 평문값 : GIMCHI IS THE ACCEPTED WORD IN BOTH NORTH AND SOUTH KOREAN STANDARD LANGUAGES. EARLIER FORMS OF THE WORD INCLUDE TIMCHAI, A MIDDLE KOREAN TRAN SCRIPTION OF THE SINO-KOREAN WORD. TIMCHAI APPEARS IN SOHAK EONHAE, THE 16TH CENTURY KOREAN RENDITION OF THE CHINESE BOOK, XIAOXUE(IN KOREAN, SOHAK). SOUND CHANGES FROM MIDDLE KOREAN TO MODERN KOREAN REGARDING THE WORD CAN BE DESCRIBED AS: TIMCHAI > DIMCHAI > JIMCHAI > JIMCHUI > GIMCHI

2. decrypt.txt에 저장된 복호화된 문자를 다시 암호화

```
shin@shin: ~/Desktop/homework/3
파일(F)  편집(E)  보기(V)  검색(S)  터미널(T)  도움말(H)
shin@shin:~/Desktop/homework/3$ ./tablevi
>> Input file name : decrypt.txt
>> Input mode [ 0 : ENC , 1: DEC ]0
key table : koegbpdwitncmfryqzulashxvj
input 2 key : 3 4
FILE Size = 430
shin@shin:~/Desktop/homework/3$ cat encrypt.txt
dppulz fa zes rmtwyyxu ihhb dw uogf qffch csw lcrta lalgsp dbwzgxnl xtuiusxue. c
gzwolg flunm pu gfw sntu lsdmzxa dzubdsk, r xywgix kkktec rirx jpzbiiahu yg fvz
erqf-dddyiz xccl. dzubdsk rkhikrc vj oltvp sexrgb, ykl 16py wpcfcbi qzqkue emzbp
fvoh aw ueh quyzbyx zkxy, jdspcbc(cf dwdqiu, lmozx). eaxmj ahcsnpy jwos unevgu s
ssbxd tk slpemb vvvwkt nluihbdwx dzn jffs zcs pp kviykrggv ud: byqeblv > yzzbpsj
 > xulpwho > vaawiue > xshinw
shin@shin:~/Desktop/homework/3$ cat plain.txt
dppulz fa zes rmtwyyxu ihhb dw uogf qffch csw lcrta lalgsp dbwzgxnl xtuiusxue. c
gzwolg flunm pu gfw sntu lsdmzxa dzubdsk, r xywgix kkktec rirx jpzbiiahu yg fvz
erqf-dddyiz xccl. dzubdsk rkhikrc vj oltvp sexrgb, ykl 16py wpcfcbi qzqkue emzbp
fvoh aw ueh quyzbyx zkxy, jdspcbc(cf dwdqiu, lmozx). eaxmj ahcsnpy jwos unevgu s
ssbxd tk slpemb vvvwkt nluihbdwx dzn jffs zcs pp kviykrggv ud: byqeblv > yzzbpsj
 > xulpwho > vaawiue > xshinw
shin@shin:~/Desktop/homework/3$
```

느낀점 : 처음에는 암,복호화가 같은 알고리즘인줄 알아서 쉬운줄 알았는데 자세히보니 복잡한 알고리즘이였다.

그래도 일단 암호화에 성공하고 그 코드를 이용하여서 복호화에 성공하여서 다행이다.