

정보보호

- HW02 : Caesar, Vigenere Cipher-

제 출 일	2018년 09월 15일
담당교수	류재철
학 과	컴퓨터공학과
학 번	201302423
이 름	신종욱

1. 시저 암호(Caesar Cipher)

■ 과제해결 과정

이번과제는 키 값이 양수이고 정해진 HelloSecurityATZ만 잘 변환시키면 되기 때문에 키 값이 음수,문자열 공백같은 예외에 대해선 생각하지않아도 돼서 모든 문자열을 그냥 받아서 키 값을 더한다고 하면 구현하기 쉬웠다.



```
void Encrypt(char *buff,int key){
    key = key%26; //알파벳은 26개이기때문에 0~25로만 판단
    int flag=0; //대문자인지 소문자인지 판단용 상수
    if(*buff>=65&&*buff<=90) flag=1; //대문자일경우 flag=1
    *buff = *buff+key; //문자열에 키값만큼 더한다
    if((*buff>90&&flag==1)||*buff>122) *buff= *buff-26;
    //만약 대문자였고 z를 넘어 갔을 경우에는 다시 A로 와야하고
    //소문자였고 z를 넘었을 경우에는 다시 a로와야해서 -26을 한다
}

void Decrypt(char *buff,int key){
    key = key%26;
    int flag=0;
    if(*buff>=97&&*buff<=122) flag=1; //이번에는 소문자일경우
    flag=1이다 위와 다른이유는 복호화의 경우에는 값을 빼기를 하기때문에 소문자
    일경우에 확인을 해야한다.
    *buff = *buff-key;
    if((*buff<97&&flag==1)||*buff<65) *buff=*buff+26;
} //만약 소문자이고 a보다 작거나 대문자이고 A보다 작을 경우 알파벳 한바퀴를
    넘었기때문에 +26을 해준다
```

암호화,복호화 과정에서 키 값이 26보다 클가능성을 생각해서 %26으로 26보다 작게 만들고 암호화의 경우 문자와 큰 키 값이 더해져서 Z를 넘을 경우를 예외처리하여 -26을 하여 다시 알파벳으로 돌아오도록 하였고 복호화의 경우에는 반대로 많이 빼기를 하여서 a보다 작아질 경우 +26을 하여서 다시 알파벳으로 돌아오도록 하였다.

구현후 생각해보니

위와 같이 구현할 경우 공백이나 특수문자,개행문자도 다 바꾸기 때문에 알파벳일 경우만 복호화 암호화하도록 하는것도 괜찮을 것 같다.

하지만 테스트는 알파벳만 하니 그대로 사용하였다.

위와 같이 구현 할 경우 오류가 하나 발생 하였는데
바로 보이지는 않지만 HelloSecurityATZ 뒤에 있는 보이지않는 문자열이고
종료표시인 \0 때문에 암호화 복호화할때마다 하여서 에러가 생기는걸 발견하였다.
(디버깅을 하여서 \0를 찾아내었다.)


```
int fileSize = GetFileSize(input_FD); //파일 사이즈 계산
char buff;
int i;
for(i=0; i<fileSize-1;i++){
    fread(&buff,sizeof(char),1,input_FD);

    if(mode==0){
        Encrypt(&buff,key);
    }
    else if(mode==1){
        Decrypt(&buff,key);
    }
    fwrite (&buff, sizeof(char),1,output_FD);
} //모드에 따라 적절한 함수를 실행한후 결과문자열을 작성한다
//마지막 문자열까지 복호,암호화 된다면 오류날수가 있어서 제외하였
다.

fwrite ("\n", sizeof(char),1,output_FD); //단어 맨끝엔 문자열
변경을 넣어서 출력을 깔끔하게해주었다.
fclose(output_FD);
fclose(input_FD);
return 0;
}
```

그래서 fileSize-1로 바꾸어 마지막 \0 암호화 과정을 없애고 맨 끝에 \n를 한번
적는걸로 완성하였다

■ 시저 암호 결과



```
shin@shin: ~/Desktop/homework/1
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
shin@shin:~/Desktop/homework/1$ cat plain.txt
HelloSecurityATZ
shin@shin:~/Desktop/homework/1$ ./caesar
>> Input File Name : plain.txt
>> Input Mode [ 0 : Encrypt, 1 : Decrypt ] 0
>> Input Key : 3
FILE Size = 17
shin@shin:~/Desktop/homework/1$ cat encrypt.txt
KhoorVhfxulwbDWC
shin@shin:~/Desktop/homework/1$ ./caesar
>> Input File Name : encrypt.txt
>> Input Mode [ 0 : Encrypt, 1 : Decrypt ] 1
>> Input Key : 3
FILE Size = 17
shin@shin:~/Desktop/homework/1$ cat decrypt.txt
HelloSecurityATZ
shin@shin:~/Desktop/homework/1$
```

2. 비제네르 암호(Vigenere Cipher)

■ 과제해결 과정

비제네르의 경우에는 입력받은 문자열 키를 숫자로 변환시키고 그것을 반복해서 키로 사용해야한다.

이것을 구현하기 위해서 문자열 배열의 크기만큼의 정수 배열을 만들어서 문자열 한 개당 숫자로 변환시켜 1:1로 저장하여서 구현하기로 생각하였다.

```
for(i=0; i<strlen(key);i++){
    if(key[i]>=97&&key[i]<=122){
        intkey[i]=key[i]-97;
    }
    else if(key[i]>=65&&key[i]<=90){
        intkey[i]=key[i]-65;
    }
    else intkey[i]=key[i]%26;
}
```

//문자열값들을 숫자값으로 변환하는 과정이다.

//대문자와 소문자일 경우에 따라 나눠서 각각에 a값만큼 빼준다

문자열을 1:1로 숫자로 변환시키는 과정이다 A->0,B->1 으로 변해야하기 때문에 각각의 알파벳에 대소문자별 A를 빼주면 되는 과정이다



```
vigenere.c
~/Desktop/homework/1
저장(S)

for(i=0; i<fileSize-1;i++){
    fread(&buff,sizeof(char),1,input_FD);

    if(mode==0){
        Encrypt(&buff,intkey[i%strlen(key)]%26);
    }
    else if(mode==1){
        Decrypt(&buff,intkey[i%strlen(key)]%26);
    }//키값을 줄때 영단어보다 키값이 짧을경우 반복이 될수있
게 처리를 하여야해서 %이용하여서 나머지를 이용하였다.

    fwrite ( &buff, sizeof(char),1,output_FD);
}
fwrite("\n",sizeof(char),1,output_FD);
fclose(output_FD);
fclose(input_FD);
return 0;
}
```

암호화 복호화할 때 키 값을 넘겨줄 때 키 문자열이 짧아 반복 사용 되어야 할때를 구현하기 위해서 `i%strlen(key)`를 이용해 반복되도록 구현하였다.

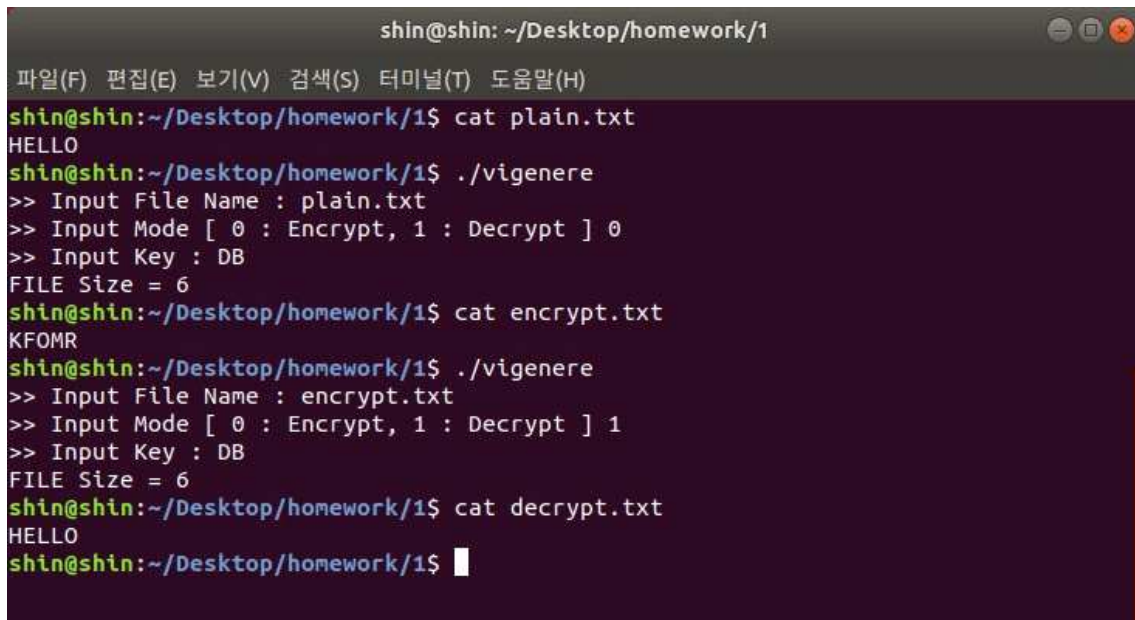


```
void Encrypt(char *buff, int intkey){
    int flag=0; //대문자인지 소문자인지 판단용 상수
    if(*buff>=65&&*buff<=90) flag=1;
    *buff = *buff+intkey;
    if((*buff>90&&flag==1)||*buff>122) *buff= *buff-26;
} //만약 대문자였고 z를 넘어 갔을 경우에는 다시 A로 와야하고
//소문자였고 z를 넘었을 경우에는 다시 a로와야해서 -26을 한다

void Decrypt(char *buff, int intkey){
    int flag=0; //대문자인지 소문자인지 판단용 상수
    if(*buff>=97&&*buff<=122) flag=1;
    *buff = *buff-intkey;
    if((*buff<97&&flag==1)||*buff<65) *buff=*buff+26;
} //이번에는 소문자일 경우 flag=1이다 위와 다른 이유는 복호화의 경우에는 값을 빼기를 하기때문에 소문자일 경우에 확인을 해야한다.
```

암호화 복호화 코드는 시저암호와 같은데 이 경우에는 key값을 이미 들어오기전에 %26을 해서 함수안에서 할 필요가 없다.

■ 비제네르 암호 결과



느낀점 : 특수문자 공백 다른언어의 문자열 처리는 어떻게 하는지도 알아가면서 구현하면 재밌을거 같다.