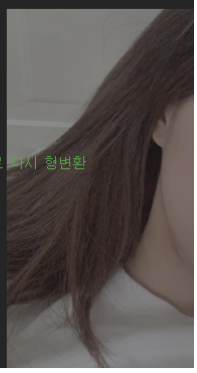
REPORT

암호 프로그래밍

과목명 :	컴퓨터보안
담당교수 :	정준호 교수님
제출일 :	2021 년 03 월 31 일
공과 대학	컴퓨터공학 과
학번 : 2016112154	이름: 정동구

1. 소스코드 및 리뷰

```
class Caeser//시저 암호
{
public:
    void encrypt(int key)//암호화
    {
        key = key % 26;//26 이상의 숫자를 입력 받을 경우 대비
        for (int i = 0; i < plain_text.length(); i++)//입력받은 평문 길이만큼
        {
            if (plain_text[i] == ' ')//공백은 그대로 암호문에 입력
            {
                encrypt_str += plain_text[i];
            }
            else
            {
                if((int)plain_text[i]+key >122)
                    encrypt_str += (char)((int)plain_text[i] + key-26);//z인 122를 넘어갈 경우 -26을 하여 앞으로 돌린다.
                else
                    encrypt_str += (char)((int)plain_text[i] + key);//ascii code이용. int로 명시적 형변환 후 숫자를 더해서 시저 암호화 후 char로 다시 형변환
            }
        }
    }
    void decrypt(int key)//복호화
    {
        key = key % 26;
        for (int i = 0; i < encrypt_str.length(); i++)
        {
            if (plain_text[i] == ' ')//공백일시 그대로 입력
            {
                decrypt_str += encrypt_str[i];
            }
            else
            {
                if ((int)encrypt_str[i] - key < 97)
                    decrypt_str += (char)((int)plain_text[i] - key + 26);//a인 97보다 작을 경우 +26을 하여 z로 넘어가게 한다.
                else
                    decrypt_str += (char)((int)plain_text[i] -key);//ascii code이용. int로 명시적 형변환 후 숫자를 더해서 시저 암호화 후 char로 다시 형변환
            }
        }
    }
    void getPlainText(std::string input_string)//평문을 입력받는다.
    {
        plain_text = input_string;
    }
    std::string plain_text;//입력받을 평문
    std::string encrypt_str;//암호문
    std::string decrypt_str;//복호문
};
```



시저 암호문 암호화 복호화를 하는 class

시저 암호는 문자를 평행이동 시켜서 생성하는 암호이다. 예를 들어 a 를 +1 의 key 로 암호화 한다 하면 a 는 우측으로 1 만큼 평행이동하여 b 가 된다. 동일한 원리로 입력받은 key 값에 따라 평문의 문자 하나 하나를 한쪽 방향으로 평행이동 시켰다.

평문의 문자 하나를 int 형으로 명시적 형변환을 하면 ascii 코드로 변하기 때문에 해당 ascii 코드 값에 정수를 더하고 빼는 형식으로 암호화 복호화를 진행하였다. 이 때 소문자 알파벳의 ascii 코드 범위인 97-122 를 벗어날 경우 특수문자가 출력 되기 때문에 조건문을 이용하여 범위를 초과할 경우 26 을 더하거나 빼는 형식으로 환형 큐처럼 앞 또는 뒤로 돌아가게 만들었다.

암호화는 입력받은 key 만큼 평행이동 하여 실행하였고, 복호화는 반대방향으로 동일한 수치만큼 평행이동 하여 실행하였다.

이 시저 암호가 가지는 문제점은 쉽게 해독이 된다는 것이다. 위의 경우처럼 알파벳 소문자만 이용할 경우 key 를 -26~+26 까지 한번씩 넣어보면서 결과를 보면 빠르게 해독이 가능하다.

```
class SimpleSubstitution//단일 치환 암호
{
public:
    void encrypt()//암호화
    {
        for (int i = 0; i < plain_text.length(); i++)
        {
            if (plain_text[i] == ' ')//공백일 경우 바로 문자열에 추가
            {
                encrypt_str += plain_text[i];
            }
            else
            {
                for (int j = 0; j < 27; j++)//평문 테이블과 비교하여 일치 할 때 해당 index를 가지는 암호 테이블 문자를 문자열에 입력
                {
                    if (plain_text[i] == plain_table[j])
                        encrypt_str += encrypt_table[j];
                }
            }
        }
    }
    void decrypt()//복호화
    {
        for (int i = 0; i < encrypt_str.length(); i++)
        {
            if (encrypt_str[i] == ' ')
            {
                decrypt_str += encrypt_str[i];
            }
            else
            {
                for (int j = 0; j < 27; j++)//암호 테이블과 비교하여 일치 할 때 해당 index를 가지는 암호 테이블 문자를 문자열에 입력
                {
                    if (encrypt_str[i] == encrypt_table[j])
                        decrypt_str += plain_table[j];
                }
            }
        }
    }
    void getPlainText(std::string input_string)
    {
        plain_text = input_string;
    }
    std::string plain_text;//평문
    std::string encrypt_str;//암호문
    std::string decrypt_str;//복호문
private:
    char plain_table[27] = "abcdefghijklmnopqrstuvwxyz"; // 평문 테이블
    char encrypt_table[27] = "WYHFXUMTJVSGENBRDZLQAPCOKI";// 암호 테이블
```

단일 치환 암호로 암호화 하고 복호화 하는 class

단일 치환 암호는 기존 문자를 다른 문자에 각각 일대일 대응시켜 암호화하는 방식이다. 알파벳을 예로 들면 a-z 까지의 알파벳의 순서를 섞어 암호문으로 이용할 테이블을 생성하고 해당 문자에 일대일 대응해 변경한다.

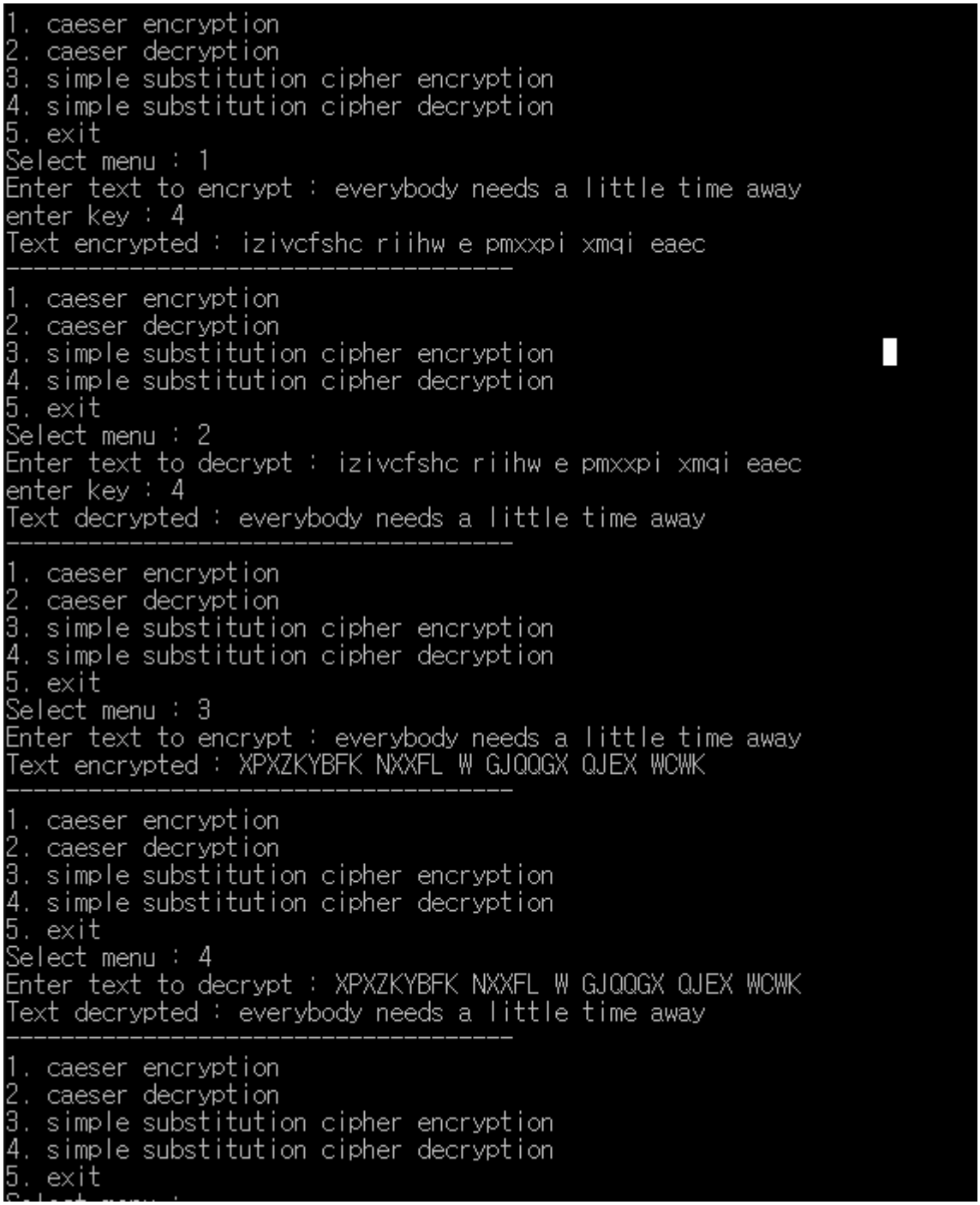
위의 소스코드에서는 소문자 알파벳 26 글자와 순서를 섞은 대문자 알파벳을 일대일 대응시켜서 암호화와 복호화를 하였다. 평문에서의 문자와 일치하는 평문테이블의 문자가 가지는 index 와 동일한 index 를 가지는 암호테이블의 문자로 대입하는 방식으로 암호문을 형성하였다. 복호화의 경우 정확히 반대로 실행하였다. 암호문의 문자와 일치하는 암호테이블의 index 를 이용하여 평문 테이블에 있는 문자로 변경하여 복호화를 하였다.

단일치환 암호는 bruteforce 로는 해독이 불가능하다. 하지만 단일치환 암호도 해독이 불가능 한 것은 아니다. 일대일 대응되기 때문에 빈도분석의 방식으로 해독이 가능하다. 영어에서 알파벳 e 가 가장 많이 쓰이는 것이 익히 알려 진 것 처럼 많이 등장하는 문자를 기준으로 빈도를 분석 하면 해독이 가능하다. 암호문의 길이가 길어질수록 복호화가 될 가능성이 높아지는 단점이 있다.

```cpp
int main()
{
    int cmd;
    std::string input;
    int key;
    Caeser a;
    SimpleSubstitution b;
    while (1)
    {
        std::cout << "1. caeser encryption₩n";
        std::cout << "2. caeser decryption₩n";
        std::cout << "3. simple substitution cipher encryption₩n";
        std::cout << "4. simple substitution cipher decryption₩n";
        std::cout << "5. exit₩n";
        std::cout << "Select menu : ";
        std::cin >> cmd;
        std::cin.ignore();
        switch (cmd)
        {
        case 1:
            std::cout << "Enter text to encrypt : ";
            std::getline(std::cin, input);
            std::cout << "enter key : ";
            std::cin >> key;
            a.getPlainText(input);
            a.encrypt(key);
            std::cout << "Text encrypted : " << a.encrypt_str << "₩n";
            break;
        case 2:
            std::cout << "Enter text to decrypt : ";
            std::getline(std::cin, input);
            std::cout << "enter key : ";
            std::cin >> key;
            a.getPlainText(input);
            a.decrypt(key);
            std::cout << "Text decrypted : " << a.decrypt_str << "₩n";
            break;
        case 3:
            std::cout << "Enter text to encrypt : ";
            std::getline(std::cin, input);
            b.getPlainText(input);
            b.encrypt();
            std::cout << "Text encrypted : " << b.encrypt_str << "₩n";
            break;
        case 4:
            std::cout << "Enter text to decrypt : ";
            std::getline(std::cin, input);
            b.getPlainText(input);
            b.decrypt();
            std::cout << "Text decrypted : " << b.decrypt_str << "₩n";
            break;
        case 5:
            std::cout << "exit program₩n";
            exit(0);
            break;
        }
        std::cout << "--------------------------------₩n";
    }
}
```

Main 함수.

2. 결과



everybody needs a little time away 라는 평문으로 각각 암호화와 복호화를 진행하였다. 시저 암호와 단일치환 암호 모두 정상적으로 암호화 복호화가 됨을 확인하였다.

시저 암호의 경우 ascii코드를 이용하여 이동시켰다. 이때 key에 의해 ascii코드 범위를 벗어 나는 경우 특수문자가 출력 되기 때문에 이부분을 코드에서 +-26 을 해주는 형태로 보정하였다. 위 문장에서 key 가 4 인데 y 가 c 로 정상 출력됨을 알 수 있다.

단일 치환 암호의 경우 테이블은 하나만 형성 하였다. 테이블이 하나이기 때문에 복호화도 동일한 테이블로 하였다. 시저 암호처럼 범위를 벗어날 일이 없기 때문에 특별히 신경 쓴 부분은 없다.

3. 소감

시저 암호와 단일 치환 암호의 경우 구현은 많이 어렵지 않았다. 둘 다 재미있게 구현 할 수 있었다. 이 중 시저 암호는 대학에 처음 입학한 나를 떠올리게 하였다. 지금은 계시지 않지만 과거 홍정모 교수님의 기초프로그래밍 수업을 수강했었는데, 이 때 첫 실습 과제가 시저 암호를 구현 하는 것 이었다. 당시에는 c 언어를 제대로 다루지도 못했기 때문에 굉장히 힘들어 했었던 기억이 있다. 그 때를 생각하면 지금의 내 모습이 조금 부끄럽다. 당시에는 너무 재미있어서 정말 열심히 공부하고 코딩했던 기억이 난다. 항상 주어진 과제보다도 더 심화해서 생각하고 조금이라도 더 해가려 했었던 것 같다. 반면에 지금의 나는 과제 어떻게든 마무리해서 제출하기에 급급한 모습을 자주 보이고는 한다. 내 인생 첫 프로그램이었던 시저 암호화 프로그램을 다시 구현하면서 그 때 처럼 열정적으로 프로그래밍을 해야겠다는 생각이 다시 들었다.

과제를 늦게 시작해 시간이 부족해서 다중치환 암호화 프로그램을 작성하지는 못했지만 과제 외적으로도 따로 내가 시간을 내어서 구현을 해보아야겠다는 생각이 많이 든다. 이번 기회를 통해 다시 초심으로 돌아가야겠다.