**Computer Algorithm**

**Chapter 03**

**Extended Euclid Algorithm &**

**Symmetric Key Algorithm**

**[실습] 유클리드와 대칭 암호화 알고리즘**

**실습 목표**

* **유클리드 알고리즘과 확장된 유클리드 알고리즘에 대해 학습하여 이해한다.**
* **문자열을 암호화하는 여러가지 방법에 대해 학습한다.**
* **RSA 알고리즘에 대해 학습한다.**
* **대칭 암호화 알고리즘을 이용한 프로그램을 구현한다.**

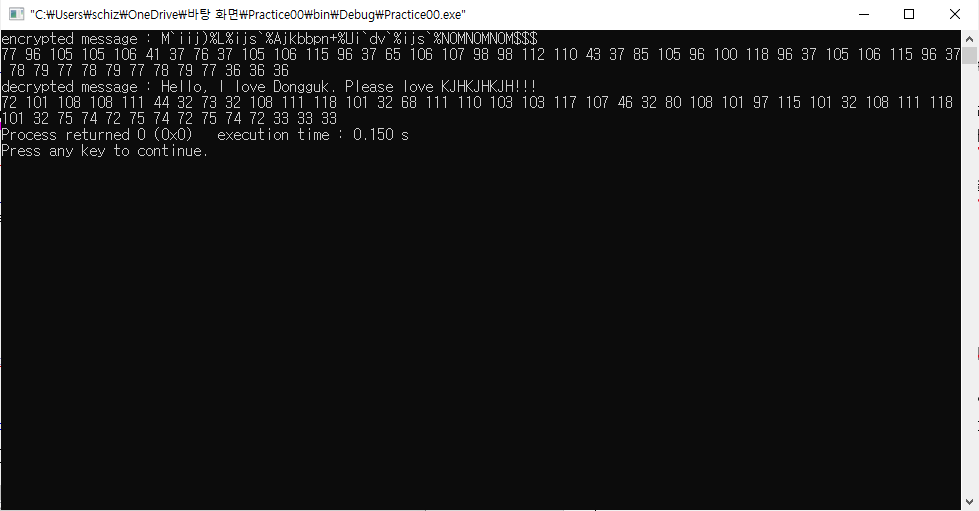
**요구사항**

* **실습과제 1) (50점) 문자열 암호화하기**

문자열 “Hello, I love Dongguk. Please love ${name}!!!”을 XOR(C 언어에서는 ^사용)을 사용한 암호화 방법으로 암호화 및 복호화 하는 프로그램을 구현한다. Key 값은 5(2진법 101)로 한다.

여기서 ${name}은 본인 영문 이름으로 작성한다. 구현 알고리즘 은 아래와 같다.

* 1. 주어진 문자열의 문자를 한자 씩 읽어, 읽은 문자 각각에 대해 아래와 같은 방식으로 정수형으로 변환한다.
     1. c = (unsigned int) ch; // 문자 ch를 정수형으로 변환한다.
     2. 변환된 문자 c를 key값과 XOR 연산을 한 후 (C언어에서의 XOR 연산자는 ^ 이다) 이를 다른 문자열 encryption에 변환된 문자를 저장한다. (문자열 마지막 문자인 '\0'은 XOR 연산없이 저장하도록 한다.)
  2. Message의 모든 문자들이 변환이 되어 encryption에 저장되었다면 암호화된 문자열 encryption의 내용을 출력한다.
  3. Encryption의 문자들을 하나씩 key값과 XOR 연산(^)을 한 후 이를 다른 문자열 decryption에 저장한다. ('\0'은 XOR 연산없이 저장한다.)
  4. 복호화된 문자열 decryption의 내용을 출력한다.



* **실습과제 2) (50점) 확장 유클리드 알고리즘 구현하기**

키보드를 통해 두 수 를 입력받아 를 만족하는 를 확장 유클리드 알고리즘을 사용하여 구하는 프로그램을 작성한다. (배경지식 참고)

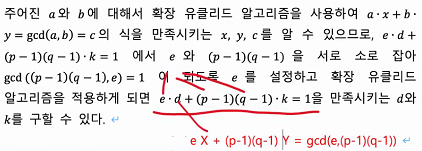
유클리드 알고리즘

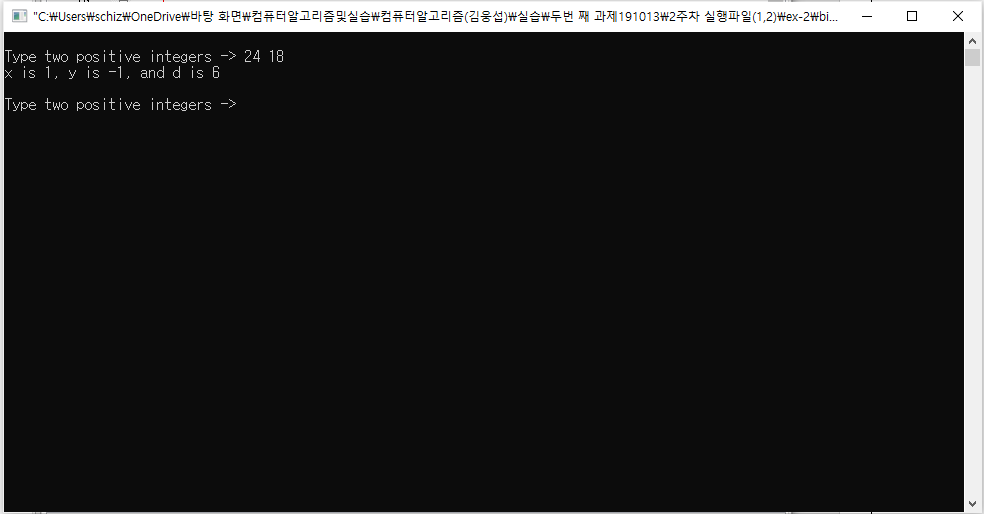
GCD(a, b) = GCD(b, a % b) , a > b

예) GCD(24, 18) = GCD(18, 6) = GCD(6, 0) = 6

A = 24, b = 18, AX + BY = 6, X= 1, Y = -1

\*참고

****



**배경지식**

유클리드 알고리즘은 유클리드 호제법이라고 불리며 2개의 자연수의 최대 공약수를 구하는 알고리즘의 하나이다. 호제법이라는 말은 두 수가 서로 상대방 수를 나누어 결국 원하는 수를 얻는 알고리즘을 의미한다. 유클리드 알고리즘은 RSA 암호화 기법에 사용된다.

대칭 암호화 알고리즘은 하나의 암호키(key)를 쌍방이 공유하며 이를 통해 메시지(m)를 암호화하는 방법으로 이 되도록 하는 함수 를 정의하고 통해 메시지를 암호화하고 복호화하는 알고리즘이다.

1)

문자열을 암호화하는 여러가지 방법 중 하나인 대칭암호화 기법은 공유하는 암호키를 활용하는 방법으로 현재까지도 널리 사용되고 있으며, 예를 들면 암호를 주고받는 대상간에 private키와 연산방식(예를 들면 XOR연산)을 공유하고 이를 사용하여 문자열을 암호화하는 방법이 있다.

Private key와 XOR를 사용한 방법은 주어진 문자열의 각 문자에 private key와 XOR연산을 통하여 문자를 변환하는 방법으로 예를 들면 아래의 문자 N은 private key가 111(2)인 경우 아래와 같이 암호화된다.

N = 0x4E = 0100 1110

key = 0x07 = 0000 0111 XOR

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

dataencrypt = 0100 1001

즉 문자 N은 private key가 111(2)인 경우 XOR연산을 통해 01001001(2)로 변환된다.

또한 이 문자를 동일한 private key와 XOR연산으로 복호화 하게 되면 원래 값 N으로 복원된다.

dataencrypt = 0x49 = 0100 1001

key = 0x07 = 0000 0111 XOR

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

datadecrypt = 0100 1110 🡨 N의 이진 표현

%% 참고

XOR연산은 두 값의 자릿수를 비교해, 값이 같으면 0 다르면 1로 계산하는 연산이다.

0101

예) XOR 0011

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

= 0110

C언어에서 XOR연산은 ^를 사용하므로 정수 a와 b의 XOR 연산을 할 때는 a^b 형식으로 한다.

2)

확장 유클리드 알고리즘은 기본적으로 유클리드 알고리즘과 거의 유사하며 차이점은 두 수 와 가 주어지는 경우 유클리드 알고리즘은 만을 구하는데 반해 를 만족하는 를 모두 구한다는 점이 다르다.

확장유클리드 알고리즘의 입력값은 두 수 와 이며 출력값은 관계를 만족시키는 이다. 이러한 관계를 만족시키는 는 다양하게 존재하며, 확장 유클리드 알고리즘은 이 중에 한 쌍을 출력한다.

다음은 확장 유클리드 알고리즘의 슈도코드이다.

**extended\_Euclid**(a,b)

입력값 : 양의 정수 (, 즉 두 입력 값중에 큰 값은 , 작은 값은 가 되도록 한다.)

출력값 : 정수쌍 (). 여기서 () 는 주어진 와 에 대해 를 만족하는 와 를 출력한다.

// 탈출지점 정의

// gcd(a, 0) 인 경우 a를 gcd(a, 0)의 값으로 리턴한다.

// 따라서 b가 0인 경우 x = 1, y = 0, a를 리턴

// a x + b y = a 🡨 x = 1, b = 0 인 경우에

**if** (b = 0)

**return** (1, 0, a)

// 재귀호출을 통해 (x, y, gcd(b, a%b) = c) 값을 얻는다.

(x, y, c) = **extended\_Euclid**(b, a % b) 🡨 bx + (a%b)y = c

// 리턴 값은 triplet이며 triplet의 각 원소들은 순서를 지켜서 구조체에 그 값을 저장한 후 구조체를 리턴한다.

**return** (y, x – \* y, c)

확장 유클리드 알고리즘의 정확성은 수학적 귀납법을 통해 증명된다.

Basis) 인 경우 가 성립된다.

Inductive step)

🡨 Euclid 정리에서 도출

🡨

이므로, 만약 Extended Euclid알고리즘이 정확한 를 리턴한다고 가정한다면,

🡨 Euclid 정리를 적용.

🡨

따라서

이 된다.

따라서 위 슈도코드는 정확하다고 판단된다.

이 알고리즘은 RSA 알고리즘에서 값과 값이 주어진 경우 값을 구할 때 사용된다.

주어진 와 에 대해서 확장 유클리드 알고리즘을 사용하여 의 식을 만족시키는 를 알 수 있으므로, 에서 와 을 서로 소로 잡아 이 되도록 를 설정하고 확장 유클리드 알고리즘을 적용하게 되면 을 만족시키는 와 를 구할 수 있다.

RSA알고리즘에서 와 , 는 미리 주어지므로 와 은 미리 구할 수 있으며 확장 유클리드 알고리즘을 적용하면 의 와 를 구할 수 있다. extended\_Euclid()함수를 적용하면 또는 가 음의 정수 값이 나오는 경우가 있는데, 이 경우 , 와 modulo 에 대하여 동치관계를 가지는 양의 정수 값을 찾아 그 값으로 대체하면 된다. 구하는 방법은 또는 에 을 더해주어 양수가 되도록 하면 된다.

위에서 설명한 extended\_Euclide() 알고리즘은 ()의 triplet을 사용하고 있으며 triplet 값을 리턴하게 된다. C언어에서 트리플을 리턴하기 위해서는 아래 예제와 같은 구조체를 사용할 수 있으며 트리플 ()의 은 각각 first, second, third로 이름짓는다.

참고용 예제코드는 다음과 같다.

// (x, y, c)를 저장하는 역할을 하는 구조체를 정의한다.

**struct** \_node {

**int** first;

**int** second;

**int** third;

};

**struct** \_node \*extended\_Euclid(**int** a, **int** b)

{

**struct** \_node \*n1, \*n2;

n1 = (**struct** \_node \*)malloc(**sizeof**(**struct** \_node));

... // 기타 필요한 변수를 선언하고 이들을 초기화한다.

**if**(b == 0) {

n1->first = 1;

n1->second = 0;

n1->third = a;

**return** n1; // n1=(1,0,a)의 형태로 리턴

}

**else** {

/\* 재귀를 사용하여 extended\_Euclid 알고리즘의 다음 부분을 여기에 구현한다.

(first, second, third) = extended\_Euclid(b, a % b)

🡨 first = x, second = y, third = c 가 된다.

return (second, first – ⌊a/b⌋ \* second, third)

🡨 ax+by=ay^'+b(x^'- ⌊a/b y'⌋) 이므로 (y^',x^'- ⌊a/b y^' ⌋,c)를 리턴한다.

\*/

}

}

**void** main(**void**)

{

**struct** \_node \*n;

... // do some more initialization.

printf("\n\n Type two positive integers -> ");

scanf("%d %d", &u, &v);

**if** (u < 0 || v < 0)

// if inputs are invalid, then do nothing.

**continue**;

**if** (u == 0 || v == 0)

// if one of inputs are 0, then terminate

**break**;

n = extended\_Euclid(u, v);

printf("x is %d, y is %d, and c is %d\n ", n->first, n->second, n->third);

}

**제출방법**

* 보고서 작성방법: 실습문제 번호별로 결과가 나온 화면의 내용을

캡쳐하여 보고서에 붙여 놓는다.

* 소스코드의 파일이름에 연습문제 번호를 붙이는 것을 잊지 않는다. 예) ex-1.c, ex-2.c
* 결과 보고서에 이름과 작성 날짜를 기입하는 것을 잊지 않는다. 예) 김웅섭\_2020\_09\_01.doc
* 실행결과를 보고서에 작성하여 소스코드와 함께 제출한다.
* 제출 마감 : e-class 제출 마감시간까지