1. Char pointer:

#include<string.h>

#include<stdio.h>

void main() {

char str[]="Hello World";

char str1[11];

int i,len;

len=strlen(str);

for(i=0;i<len;i++) {

str1[i]=str[i]^0;

printf("%c",str1[i]);

}

printf("\n");

}

1. ANDXOR.C

#include <stdio.h>

#include<stdlib.h>

void main() {

char str[]="Hello World";

char str1[11];

char str2[11];

int i,len;

len = strlen(str);

for(i=0;i<len;i++) {

str1[i] = (str[i]&127) | (str[i]^127);

printf("%d",str1[i]);

}

printf("\n");

for(i=0;i<len;i++) {

str2[i] = (str[i]&127) & (str[i]^127);

printf("%d",str2[i]);

}

1. CAESAR CIPHER.JAVA:

package q2404;

import java.util.\*;

class CaesarCipher {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

int shift, i, n;

String str;

String str1 = "";

String str2 = "";

System.out.print("Enter any String: ");

str = sc.nextLine();

str = str.toLowerCase();

n = str.length();

char ch1[] = str.toCharArray();

char ch3, ch4;

System.out.print("Enter the key: ");

shift = sc.nextInt();

System.out.print("Encrypted String is: ");

for (i = 0; i < n; i++) {

if (Character.isLetter(ch1[i])) {

ch3 = (char) (((int)ch1[i] + shift - 97) % 26 + 97);

str1 = str1 + ch3;

}

else if (ch1[i] == ' ') {

str1 = str1 + ch1[i];

}

}

System.out.print(str1);

System.out.println();

System.out.print("Decrypted String is: ");

char ch2[] = str1.toCharArray();

for (i = 0; i < str1.length(); i++) {

if (Character.isLetter(ch2[i])) {

if (((int)ch2[i] - shift) < 97) {

ch4 = (char) (((int)ch2[i] - shift - 97 + 26) % 26 + 97);

}

else {

ch4 = (char) (((int)ch2[i] - shift - 97) % 26 + 97);

}

str2 = str2 + ch4;

}

else if (ch2[i] == ' ') {

str2 = str2 + ch2[i];

}

}

System.out.print(str2);

1. SUBSTITUTIONCIPHER.JAVA

package q2405;

import java.io.\*;

import java.util.\*;

public class SubstitutionCipher {

static Scanner sc = new Scanner(System.in);

static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

public static void main(String[] args) throws IOException {

String a = "abcdefghijklmnopqrstuvwxyz";

String b = "zyxwvutsrqponmlkjihgfedcba";

System.out.print("Enter any string: ");

String str = br.readLine();

String decrypt = "";

char c;

for(int i = 0; i < str.length(); i++) {

c = str.charAt(i);

int j = a.indexOf(c);

decrypt = decrypt+b.charAt(j);

}

System.out.println("The encrypted data is: " +decrypt);

}

}

1. DES.JAVA

package q2407;

import java.util.\*;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.security.spec.KeySpec;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESedeKeySpec;

import java.util.Base64.Decoder;

import java.util.Base64.Encoder;

public class DES {

private static final String UNICODE\_FORMAT = "UTF8";

public static final String DESEDE\_ENCRYPTION\_SCHEME = "DESede";

private KeySpec myKeySpec;

private SecretKeyFactory mySecretKeyFactory;

private Cipher cipher;

byte[] keyAsBytes;

private String myEncryptionKey;

private String myEncryptionScheme;

SecretKey key;

static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

public DES() throws Exception {

myEncryptionKey = "ThisIsSecretEncryptionKey";

myEncryptionScheme = DESEDE\_ENCRYPTION\_SCHEME;

keyAsBytes = myEncryptionKey.getBytes(UNICODE\_FORMAT);

myKeySpec = new DESedeKeySpec(keyAsBytes);

mySecretKeyFactory = SecretKeyFactory.getInstance(myEncryptionScheme);

cipher = Cipher.getInstance(myEncryptionScheme);

key = mySecretKeyFactory.generateSecret(myKeySpec);

}

public String encrypt(String unencryptedString) {

String encryptedString = null;

try {

cipher.init(Cipher.ENCRYPT\_MODE, key);

byte[] plainText = unencryptedString.getBytes(UNICODE\_FORMAT);

byte[] encryptedText = cipher.doFinal(plainText);

Encoder base64encoder = Base64.getEncoder();

encryptedString = base64encoder.encodeToString(encryptedText);

}

catch (Exception e) {

e.printStackTrace();

}

return encryptedString;

}

public String decrypt(String encryptedString) {

String decryptedText=null;

try {

cipher.init(Cipher.DECRYPT\_MODE, key);

Decoder base64decoder = Base64.getDecoder();

byte[] encryptedText = base64decoder.decode(encryptedString);

byte[] plainText = cipher.doFinal(encryptedText);

decryptedText= bytes2String(plainText);

}

catch (Exception e) {

e.printStackTrace();

}

return decryptedText;

}

private static String bytes2String(byte[] bytes) {

StringBuffer stringBuffer = new StringBuffer();

for (int i = 0; i <bytes.length; i++) {

stringBuffer.append((char) bytes[i]);

}

return stringBuffer.toString();

}

public static void main(String args []) throws Exception {

DES myEncryptor= new DES();

String stringToEncrypt = "cse";

String encrypted = myEncryptor.encrypt(stringToEncrypt);

String decrypted = myEncryptor.decrypt(encrypted);

System.out.println("String To Encrypt: " +stringToEncrypt);

System.out.println("Encrypted Value : " +encrypted);

System.out.println("Decrypted Value : " +decrypted);

}

1. AES.JAVA

package q2409;

import java.security.\*;

import javax.crypto.\*;

import javax.crypto.spec.\*;

import java.util.Base64;

public class AES {

public static byte[] encryptStringUsingAes(String plainText, byte[] key) throws Exception {

// write code here that uses AES with the given key to encrypt plainText and returns the resultant encrypted bytes

SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.ENCRYPT\_MODE, skeySpec);

return cipher.doFinal(plainText.getBytes());

}

public static String decryptUsingAes(byte[] encryptedTextBytes, byte[] key) throws Exception {

// write code here that uses AES with the given key to decrypt encryptedTextBytes and returns the resultant plain text

SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.DECRYPT\_MODE, skeySpec);

byte[] original = cipher.doFinal(encryptedTextBytes);

return new String(original);

}

public static void main(String[] args) throws Exception {

String plainText = args[0];

byte[] key = args[1].getBytes();

System.out.println("Plain text: " + plainText);

byte[] encrypted = encryptStringUsingAes(plainText, key);

System.out.println("Encrypted bytes in base64: " + Base64.getEncoder().encodeToString(encrypted));

String decrypted = decryptUsingAes(encrypted, key);

System.out.println("Decrypted text: " + decrypted);

}

}