Министерство образования и науки Российской Федерации

Федеральное Государственное Бюджетное Образовательное Учреждение

Высшего Профессионального Образования

Тамбовский Государственный Технический Университет

Кафедра

Лабораторная работа №2

дисциплина «Защита информации»

Вариант 17

Выполнил студент гр. –41

Д. Ю.

Проверила: И.Л.

Тамбов

**Задание:**

*Реализовать алгоритм шифрования DES.*

**Решение:**

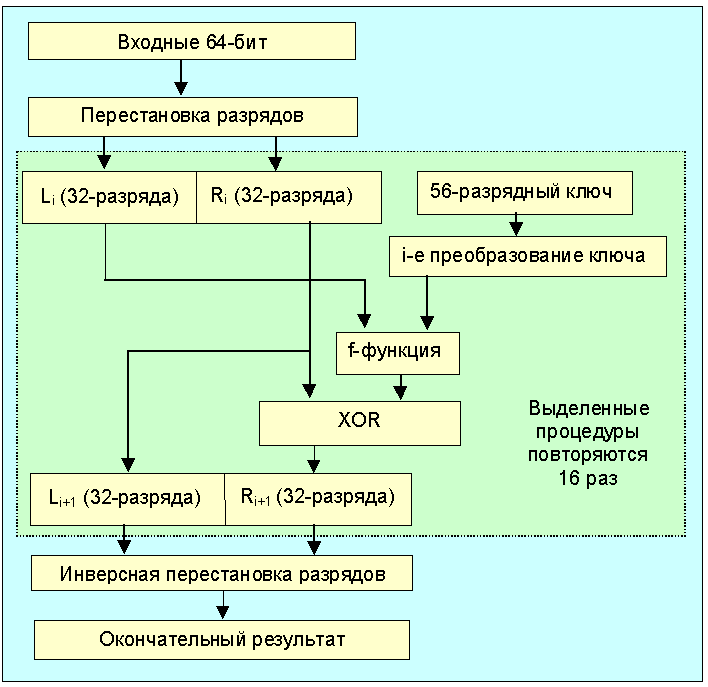
Вводится 64 битный ключ, у которого задействованы 56 разрядов, а оставшиеся выполняют роль контроля целостности.

Рисунок Структурная схема реализации алгоритма

Преобразование ключа выполняется по следующей схеме

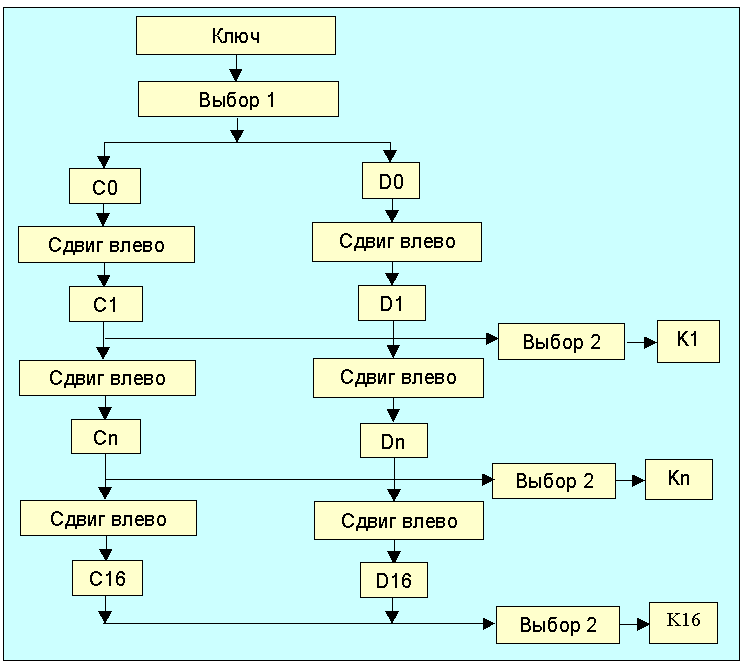
Метод EncryptionStart(string text, string key, bool IsTextBinary) задает начальные значения половинок ключа

Рисунок Алгоритм вычисления последовательности ключей

string C0 **=** ""**,** D0 **=** ""**;**

//Задать начальные значения половинок

C0 **=** **this.**SetLeftHalvesKey**(**key\_plus**);**

D0 **=** **this.**SetRightHalvesKey**(**key\_plus**);**

//Алгоритм вычисления последовательности ключей

Keys keys **=** **this.**SetAllKeys**(**C0**,** D0**);**

Следующим этапом выполняется сдвиг влево

//выполнить 16 раундов

**for** **(**int i **=** 1**;** i **<** keys**.**Cn**.**Length**;** i**++)**

**{**

keys**.**Cn**[**i**]** **=** **this.**LeftShift**(**keys**.**Cn**[**i **-** 1**],** clDESData**.**nrOfShifts**[**i**]);**

keys**.**Dn**[**i**]** **=** **this.**LeftShift**(**keys**.**Dn**[**i **-** 1**],** clDESData**.**nrOfShifts**[**i**]);**

keys**.**Kn**[**i **-** 1**]** **=** **this.**DoPermutation**(**keys**.**Cn**[**i**]** **+** keys**.**Dn**[**i**],** clDESData**.**pc\_2**);**

**}**

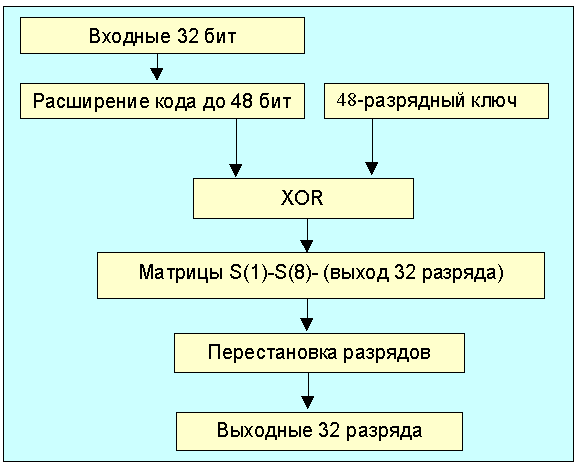
Вводится функция f, которая работает с 32-разрядными словами исходного текста (А) и использует в качестве параметра 48-разрядный ключ (J). Схеме работы функции f показана на рисунке 3.

Рисунок Алгоритм работы функции f

**public** string f**(**string Rn\_1**,** string Kn**)**

**{**

**//Расширение до 48**

string E\_Rn\_1 **=** **this.**E\_Selection**(**Rn\_1**);**

//XOR операция

string XOR\_Rn\_1\_Kn **=** **this.**XOR**(**E\_Rn\_1**,** Kn**);**

//Преобразование матриц

string sBoxedText **=** **this.**sBox\_Transform**(**XOR\_Rn\_1\_Kn**);**

string P\_sBoxedText **=** **this.**P**(**sBoxedText**);**

**return** P\_sBoxedText**;**

**}**

Инверсная перестановка выполняется последней

**public** string FinalEncription**(**string L0**,** string R0**,** Keys keys**,** bool IsReverse**)**

**{**

string Ln **=** ""**,** Rn **=** ""**,** Ln\_1 **=** L0**,** Rn\_1 **=** R0**;**

int i **=** 0**;**

**if** **(**IsReverse **==** **true)**

**{**

i **=** 15**;**

**}**

**while** **(this.**IsEnough**(**i**,** IsReverse**))**

**{**

Ln **=** Rn\_1**;**

Rn **=** **this.**XOR**(**Ln\_1**,** **this.**f**(**Rn\_1**,** keys**.**Kn**[**i**]));**

//Next Step of L1, R1 is L2 = R1, R2 = L1 + f(R1, K2), hence, value of Step1's Ln, Rn is Rn\_1, Ln\_1 in Step2.

Ln\_1 **=** Ln**;**

Rn\_1 **=** Rn**;**

**if** **(**IsReverse **==** **false)**

**{**

i **+=** 1**;**

**}**

**else**

**{**

i **-=** 1**;**

**}**

**}**

string R16L16 **=** Rn **+** Ln**;**

//выполнить перестановку

string Encripted\_Text **=** **this.**DoPermutation**(**R16L16**,** clDESData**.**ip\_1**);**

**return** Encripted\_Text**;**

**}**

Все публичные блоки вынесены в отдельный класс

/// <summary>

/// Описание таблиц констант

/// </summary>

class clDESData

**{**

**public** static **readonly** int**[]** pc\_1 **={** 57**,** 49**,** 41**,** 33**,** 25**,** 17**,** 9**,** 1**,** 58**,** 50**,**

42**,** 34**,** 26**,** 18**,** 10**,** 2**,** 59**,** 51**,** 43**,** 35**,**

27**,** 19**,** 11**,** 3**,** 60**,** 52**,** 44**,** 36**,** 63**,** 55**,**

47**,** 39**,** 31**,** 23**,** 15**,** 7**,** 62**,** 54**,** 46**,** 38**,**

30**,** 22**,** 14**,** 6**,** 61**,** 53**,** 45**,** 37**,** 29**,** 21**,**

13**,** 5**,** 28**,** 20**,** 12**,** 4 **};**

**public** static **readonly** int**[]** pc\_2 **={** 14**,** 17**,** 11**,** 24**,** 1**,** 5**,** 3**,** 28**,** 15**,** 6**,** 21**,**

10**,** 23**,** 19**,** 12**,** 4**,** 26**,** 8**,** 16**,** 7**,** 27**,**

20**,** 13**,** 2**,** 41**,** 52**,** 31**,** 37**,** 47**,** 55**,** 30**,**

40**,** 51**,** 45**,** 33**,** 48**,** 44**,** 49**,** 39**,** 56**,** 34**,**

53**,** 46**,** 42**,** 50**,** 36**,** 29**,** 32 **};**

**public** static **readonly** int**[]** ip **={** 58**,** 50**,** 42**,** 34**,** 26**,** 18**,** 10**,** 2**,** 60**,** 52**,**

44**,** 36**,** 28**,** 20**,** 12**,** 4**,** 62**,** 54**,** 46**,** 38**,**

30**,** 22**,** 14**,** 6**,** 64**,** 56**,** 48**,** 40**,** 32**,** 24**,**

16**,** 8**,** 57**,** 49**,** 41**,** 33**,** 25**,** 17**,** 9**,** 1**,** 59**,**

51**,** 43**,** 35**,** 27**,** 19**,** 11**,** 3**,** 61**,** 53**,** 45**,**

37**,** 29**,** 21**,** 13**,** 5**,** 63**,** 55**,** 47**,** 39**,** 31**,**

23**,** 15**,** 7 **};**

**public** static **readonly** int**[]** ip\_1 **={** 40**,** 8**,** 48**,** 16**,** 56**,** 24**,** 64**,** 32**,** 39**,** 7**,**

47**,** 15**,** 55**,** 23**,** 63**,** 31**,** 38**,** 6**,** 46**,** 14**,**

54**,** 22**,** 62**,** 30**,** 37**,** 5**,** 45**,** 13**,** 53**,** 21**,**

61**,** 29**,** 36**,** 4**,** 44**,** 12**,** 52**,** 20**,** 60**,** 28**,**

35**,** 3**,** 43**,** 11**,** 51**,** 19**,** 59**,** 27**,** 34**,** 2**,**

42**,** 10**,** 50**,** 18**,** 58**,** 26**,** 33**,** 1**,** 41**,** 9**,**

49**,** 17**,** 57**,** 25 **};**

**public** static **readonly** int**[]** pc\_e **={** 32**,** 1**,** 2**,** 3**,** 4**,** 5**,** 4**,** 5**,** 6**,** 7**,** 8**,** 9**,** 8**,**

9**,** 10**,** 11**,** 12**,** 13**,** 12**,** 13**,** 14**,** 15**,** 16**,**

17**,** 16**,** 17**,** 18**,** 19**,** 20**,** 21**,** 20**,** 21**,** 22**,**

23**,** 24**,** 25**,** 24**,** 25**,** 26**,** 27**,** 28**,** 29**,** 28**,**

29**,** 30**,** 31**,** 32**,** 1 **};**

**public** static **readonly** int**[]** pc\_p **={** 16**,** 7**,** 20**,** 21**,** 29**,** 12**,** 28**,** 17**,** 1**,** 15**,** 23**,**

26**,** 5**,** 18**,** 31**,** 10**,** 2**,** 8**,** 24**,** 14**,** 32**,**

27**,** 3**,** 9**,** 19**,** 13**,** 30**,** 6**,** 22**,** 11**,** 4**,** 25 **};**

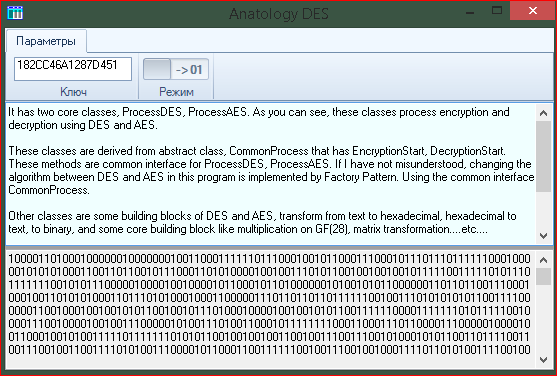
Для расшифровывания используется та же схема, что при шифровании с прежней функцией f, единственное отличие в том, что ключи используются в обратном порядке.

Рисунок Пример шифрования текста

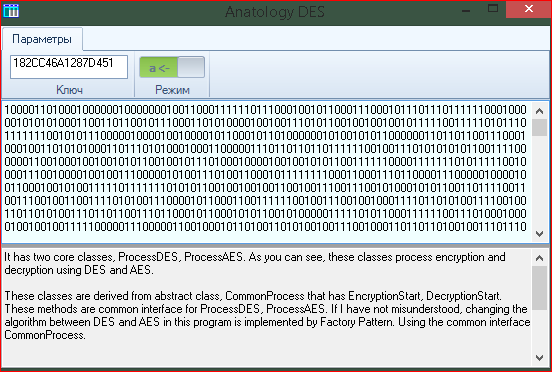


Рисунок Дешифрование

Исходные коды

**using** System**;**

**using** System**.**Text**;**

**namespace** DESanatology

**{**

class clProcessDES **:** clCommonProcess

**{**

#region Encryption Process

/// <summary>

/// Процесс шифрования

/// </summary>

/// <param name="text"></param>

/// <param name="key"></param>

/// <param name="IsTextBinary"></param>

/// <returns></returns>

**public** **override**/\*static\*/ string EncryptionStart**(**string text**,** string key**,** bool IsTextBinary**)**

**{**

#region Get 16 sub-keys using key

//ключ в hex

string hex\_key **=** **this.**FromTextToHex**(**key**);**

string binary\_key **=** **this.**FromHexToBinary**(**hex\_key**);**

string key\_plus **=** **this.**DoPermutation**(**binary\_key**,** clDESData**.**pc\_1**);**

string C0 **=** ""**,** D0 **=** ""**;**

//Задать начальные значения половинок

C0 **=** **this.**SetLeftHalvesKey**(**key\_plus**);**

D0 **=** **this.**SetRightHalvesKey**(**key\_plus**);**

//Алгоритм вычисления последовательности ключей

Keys keys **=** **this.**SetAllKeys**(**C0**,** D0**);**

#endregion

#region Encrypt process

//string hex\_text = this.FromTextToHex(text);

string binaryText **=** ""**;**

**if** **(**IsTextBinary **==** **false)**

**{**

binaryText **=** **this.**FromTextToBinary**(**text**);**

**}**

**else**

**{**

binaryText **=** text**;**

**}**

binaryText **=** **this.**setTextMutipleOf64Bits**(**binaryText**);**

#region Initialize Progress Bar

OnInitProgress**(new** clProgressInitArgs**(**binaryText**.**Length**));**

#endregion

//выходная строка

StringBuilder EncryptedTextBuilder **=** **new** StringBuilder**(**binaryText**.**Length**);**

**for** **(**int i **=** 0**;** i **<** **(**binaryText**.**Length **/** 64**);** i**++)**

**{**

string PermutatedText **=** **this.**DoPermutation**(**binaryText**.**Substring**(**i **\*** 64**,** 64**),** clDESData**.**ip**);**

string L0 **=** ""**,** R0 **=** ""**;**

L0 **=** **this.**SetLeftHalvesKey**(**PermutatedText**);**

R0 **=** **this.**SetRightHalvesKey**(**PermutatedText**);**

string FinalText **=** **this.**FinalEncription**(**L0**,** R0**,** keys**,** **false);**

EncryptedTextBuilder**.**Append**(**FinalText**);**

#region Increase Progress Bar

OnIncrementProgress**(new** ProgressEventArgs**(**FinalText**.**Length**));**

#endregion

**}**

**return** EncryptedTextBuilder**.**ToString**();**

#endregion

**}**

#endregion

#region Decryption Process

/// <summary>

/// Процесс расшифровки

/// </summary>

/// <param name="text"></param>

/// <param name="key"></param>

/// <param name="IsTextBinary"></param>

/// <returns></returns>

**public** **override**/\*static\*/ string DecryptionStart**(**string text**,** string key**,** bool IsTextBinary**)**

**{**

#region Get 16 sub-keys using key

string hex\_key **=** **this.**FromTextToHex**(**key**);**

string binary\_key **=** **this.**FromHexToBinary**(**hex\_key**);**

string key\_plus **=** **this.**DoPermutation**(**binary\_key**,** clDESData**.**pc\_1**);**

string C0 **=** ""**,** D0 **=** ""**;**

C0 **=** **this.**SetLeftHalvesKey**(**key\_plus**);**

D0 **=** **this.**SetRightHalvesKey**(**key\_plus**);**

Keys keys **=** **this.**SetAllKeys**(**C0**,** D0**);**

#endregion

#region Decrypt process

string binaryText **=** ""**;**

**if** **(**IsTextBinary **==** **false)**

**{**

binaryText **=** **this.**FromTextToBinary**(**text**);**

**}**

**else**

**{**

binaryText **=** text**;**

**}**

//выровнять блок до 64

binaryText **=** **this.**setTextMutipleOf64Bits**(**binaryText**);**

#region Initialize Progress Bar

OnInitProgress**(new** clProgressInitArgs**(**binaryText**.**Length**));**

#endregion

StringBuilder DecryptedTextBuilder **=** **new** StringBuilder**(**binaryText**.**Length**);**

**for** **(**int i **=** 0**;** i **<** **(**binaryText**.**Length **/** 64**);** i**++)**

**{**

string PermutatedText **=** **this.**DoPermutation**(**binaryText**.**Substring**(**i **\*** 64**,** 64**),** clDESData**.**ip**);**

string L0 **=** ""**,** R0 **=** ""**;**

L0 **=** **this.**SetLeftHalvesKey**(**PermutatedText**);**

R0 **=** **this.**SetRightHalvesKey**(**PermutatedText**);**

string FinalText **=** **this.**FinalEncription**(**L0**,** R0**,** keys**,** **true);**

#region It's for correct subtracted '0' that have added for set text multiple of 64bit

**if** **((**i **\*** 64 **+** 64**)** **==** binaryText**.**Length**)**

**{**

StringBuilder last\_text **=** **new** StringBuilder**(**FinalText**.**TrimEnd**(**'0'**));**

int count **=** FinalText**.**Length **-** last\_text**.**Length**;**

**if** **((**count **%** 8**)** **!=** 0**)**

**{**

count **=** 8 **-** **(**count **%** 8**);**

**}**

string append\_text **=** ""**;**

**for** **(**int k **=** 0**;** k **<** count**;** k**++)**

**{**

append\_text **+=** "0"**;**

**}**

DecryptedTextBuilder**.**Append**(**last\_text**.**ToString**()** **+** append\_text**);**

**}**

#endregion

**else**

**{**

DecryptedTextBuilder**.**Append**(**FinalText**);**

**}**

//DecryptedTextBuilder.Append(FinalText);

#region Increase Progress Bar

OnIncrementProgress**(new** ProgressEventArgs**(**FinalText**.**Length**));**

#endregion

**}**

**return** DecryptedTextBuilder**.**ToString**();**//.TrimEnd('0');

#endregion

**}**

#endregion

#region Set a length of text to multiple of 64 bits

/// <summary>

/// Выравниваем длину текста до 64 бит

/// </summary>

/// <param name="text"></param>

/// <returns></returns>

**public** string setTextMutipleOf64Bits**(**string text**)**

**{**

**if** **((**text**.**Length **%** 64**)** **!=** 0**)**

**{**

int maxLength **=** 0**;**

maxLength **=** **((**text**.**Length **/** 64**)** **+** 1**)** **\*** 64**;**

text **=** text**.**PadRight**(**maxLength**,** '0'**);**

**}**

**return** text**;**

**}**

#endregion

#region Permutation

/// <summary>

/// Перестановки

/// </summary>

/// <param name="text"></param>

/// <param name="order"></param>

/// <returns></returns>

**public** string DoPermutation**(**string text**,** int**[]** order**)**

**{**

StringBuilder PermutatedText **=** **new** StringBuilder**(**order**.**Length**);**

**for** **(**int i **=** 0**;** i **<** order**.**Length**;** i**++)**

**{**

PermutatedText**.**Append**(**text**[**order**[**i**]** **-** 1**]);**

**}**

**return** PermutatedText**.**ToString**();**

**}**

/// <summary>

/// For SBoxes Transformation

/// Перестановки

/// </summary>

/// <param name="text"></param>

/// <param name="order"></param>

/// <returns></returns>

**public** string DoPermutation**(**string text**,** int**[,]** order**)**

**{**

string PermutatedText **=** ""**;**

int rowIndex **=** Convert**.**ToInt32**(**text**[**0**].**ToString**()** **+** text**[**text**.**Length **-** 1**].**ToString**(),** 2**);**

int colIndex **=** Convert**.**ToInt32**(**text**.**Substring**(**1**,** 4**),** 2**);**

PermutatedText **=** clProcessDES**.**FromDeciamlToBinary**(**order**[**rowIndex**,** colIndex**]);**

**return** PermutatedText**;**

**}**

#endregion

#region Divide a text to left and right halves

**public** string SetLeftHalvesKey**(**string text**)**

**{**

**return** **this.**SetHalvesKey**(true,** text**);**

**}**

**public** string SetRightHalvesKey**(**string text**)**

**{**

**return** **this.**SetHalvesKey**(false,** text**);**

**}**

/// <summary>

/// Определить половинки C0 и D0

/// </summary>

/// <param name="IsLeft"></param>

/// <param name="text"></param>

/// <returns></returns>

**public** string SetHalvesKey**(**bool IsLeft**,** string text**)**

**{**

**if** **((**text**.**Length **%** 8**)** **!=** 0**)**

**{**

Console**.**WriteLine**(**"The key is not multiple of 8bit."**);**

**return** **null;**

**}**

int midindex **=** **(**text**.**Length **/** 2**)** **-** 1**;**

string result **=** ""**;**

**if** **(**IsLeft**)**

**{**//Для левой половинки ключа C0

result **=** text**.**Substring**(**0**,** midindex **+** 1**);**

**}**

**else**

**{**//Для правой половинки ключа D0

result **=** text**.**Substring**(**midindex **+** 1**);**

**}**

**return** result**;**

**}**

#endregion

#region Do Leftshift

**public** string LeftShift**(**string text**)**

**{**

**return** **this.**LeftShift**(**text**,** 1**);**

**}**

/// <summary>

/// Сдвиг влево

/// </summary>

/// <param name="text"></param>

/// <param name="count"></param>

/// <returns></returns>

**public** string LeftShift**(**string text**,** int count**)**

**{**

**if** **(**count **<** 1**)**

**{**

Console**.**WriteLine**(**"The count of leftshift is must more than 1 time."**);**

**return** **null;**

**}**

string temp **=** text**.**Substring**(**0**,** count**);**

StringBuilder shifted **=** **new** StringBuilder**(**text**.**Length**);**

shifted**.**Append**(**text**.**Substring**(**count**)** **+** temp**);**

**return** shifted**.**ToString**();**

**}**

#endregion

#region KeyµйА» ±ёЗПґВ ёЮј­µе - Get all of keys

**public** Keys SetAllKeys**(**string C0**,** string D0**)**

**{**

Keys keys **=** **new** Keys**();**

keys**.**Cn**[**0**]** **=** C0**;**

keys**.**Dn**[**0**]** **=** D0**;**

//выполнить 16 раундов

**for** **(**int i **=** 1**;** i **<** keys**.**Cn**.**Length**;** i**++)**

**{**

keys**.**Cn**[**i**]** **=** **this.**LeftShift**(**keys**.**Cn**[**i **-** 1**],** clDESData**.**nrOfShifts**[**i**]);**

keys**.**Dn**[**i**]** **=** **this.**LeftShift**(**keys**.**Dn**[**i **-** 1**],** clDESData**.**nrOfShifts**[**i**]);**

keys**.**Kn**[**i **-** 1**]** **=** **this.**DoPermutation**(**keys**.**Cn**[**i**]** **+** keys**.**Dn**[**i**],** clDESData**.**pc\_2**);**

**}**

**return** keys**;**

**}**

#endregion

#region Do Encryption

/// <summary>

/// Инверсная перестановка

/// </summary>

/// <param name="L0"></param>

/// <param name="R0"></param>

/// <param name="keys"></param>

/// <param name="IsReverse"></param>

/// <returns></returns>

**public** string FinalEncription**(**string L0**,** string R0**,** Keys keys**,** bool IsReverse**)**

**{**

string Ln **=** ""**,** Rn **=** ""**,** Ln\_1 **=** L0**,** Rn\_1 **=** R0**;**

int i **=** 0**;**

**if** **(**IsReverse **==** **true)**

**{**

i **=** 15**;**

**}**

**while** **(this.**IsEnough**(**i**,** IsReverse**))**

**{**

Ln **=** Rn\_1**;**

Rn **=** **this.**XOR**(**Ln\_1**,** **this.**f**(**Rn\_1**,** keys**.**Kn**[**i**]));**

//Next Step of L1, R1 is L2 = R1, R2 = L1 + f(R1, K2), hence, value of Step1's Ln, Rn is Rn\_1, Ln\_1 in Step2.

Ln\_1 **=** Ln**;**

Rn\_1 **=** Rn**;**

**if** **(**IsReverse **==** **false)**

**{**

i **+=** 1**;**

**}**

**else**

**{**

i **-=** 1**;**

**}**

**}**

string R16L16 **=** Rn **+** Ln**;**

//выполнить перестановку

string Encripted\_Text **=** **this.**DoPermutation**(**R16L16**,** clDESData**.**ip\_1**);**

**return** Encripted\_Text**;**

**}**

**public** /\*static\*/ bool IsEnough**(**int i**,** bool IsReverse**)**

**{**

**return** **(**IsReverse **==** **false)** **?** i **<** 16 **:** i **>=** 0**;**

**}**

#endregion

#region The function f

/// <summary>

/// Функция F()

/// </summary>

/// <param name="Rn\_1"></param>

/// <param name="Kn"></param>

/// <returns></returns>

**public** string f**(**string Rn\_1**,** string Kn**)**

**{**

string E\_Rn\_1 **=** **this.**E\_Selection**(**Rn\_1**);**

string XOR\_Rn\_1\_Kn **=** **this.**XOR**(**E\_Rn\_1**,** Kn**);**

string sBoxedText **=** **this.**sBox\_Transform**(**XOR\_Rn\_1\_Kn**);**

string P\_sBoxedText **=** **this.**P**(**sBoxedText**);**

**return** P\_sBoxedText**;**

**}**

#endregion

#region The function P

**public** string P**(**string text**)**

**{**

string PermutatedText **=** ""**;**

PermutatedText **=** **this.**DoPermutation**(**text**,** clDESData**.**pc\_p**);**

**return** PermutatedText**;**

**}**

#endregion

#region SBoxes transformation

**public** string sBox\_Transform**(**string text**)**

**{**

StringBuilder TransformedText **=** **new** StringBuilder**(**32**);**

**for** **(**int i **=** 0**;** i **<** 8**;** i**++)**

**{**

string temp **=** text**.**Substring**(**i **\*** 6**,** 6**);**

TransformedText**.**Append**(this.**DoPermutation**(**temp**,** clDESData**.**sBoxes**[**i**]));**

**}**

**return** TransformedText**.**ToString**();**

**}**

#endregion

#region E Selection

**public** string E\_Selection**(**string Rn\_1**)**

**{**

string ExpandedText **=** **this.**DoPermutation**(**Rn\_1**,** clDESData**.**pc\_e**);**

**return** ExpandedText**;**

**}**

#endregion

#region XOR операции

**public** string XOR**(**string text1**,** string text2**)**

**{**

**if** **(**text1**.**Length **!=** text2**.**Length**)**

**{**

Console**.**WriteLine**(**"Two data blocks for XOR are must get same size."**);**

**return** **null;**

**}**

StringBuilder XORed\_Text **=** **new** StringBuilder**(**text1**.**Length**);**

**for** **(**int i **=** 0**;** i **<** text1**.**Length**;** i**++)**

**{**

**if** **(**text1**[**i**]** **!=** text2**[**i**])**

**{**

XORed\_Text**.**Append**(**"1"**);**

**}**

**else**

**{**

XORed\_Text**.**Append**(**"0"**);**

**}**

**}**

**return** XORed\_Text**.**ToString**();**

**}**

#endregion

**}**

**}**