Содержание

[Введение 4](#_Toc9702988)

[1 Постановка и анализ задачи 5](#_Toc9702989)

[2 Проектирование структуры данных 7](#_Toc9702990)

[3 Проектирование модульной структуры программы 10](#_Toc9702991)

[4 Проектирование алгоритмов 11](#_Toc9702992)

[5 Разработка тестов 14](#_Toc9702993)

[Приложение А. Список тестовых вопросов 15](#_Toc9702994)

[Приложение Б. Листинг программных модулей 16](#_Toc9702995)

Введение

# Постановка и анализ задачи

Целью курсового проекта является автоматизация процесса проверки знаний студентов по дисциплине «Основы компьютерных наук».

Задачей курсового проектирования является разработка программы модульной структуры (далее – программа), позволяющей провести тестирование студентов, рассчитать полученный студентом балл и сохранить результаты прохождения тестов по одной из тем дисциплины «Основы компьютерных наук».

Программа должна реализовывать следующие функции:

* регистрация пользователей;
* шифрование учетных данных;
* редактирование тестовых вопросов;
* проведение тестирования;
* редактирование критериев оценивания теста;
* управление пользователями;
* управление тестами,
* сохранение статистики прохождения теста;
* расчет сводной статистики по тестам;
* формирование отчетов по статистике.

В рамках курсового проекта будет реализован модуль для шифрования учетных данных пользователей (модуль «Шифрование»).

Модуль «Шифрование» должен решать следующие задачи:

* Выбор метода шифрования данных:
  + шифр сдвига;
  + шифр Цезаря;
  + афинный шифр;
  + шифр RSA;
* Шифрование имени пользователя и пароля согласно выбранному методу;
* Проверка правильности введенных пользователем учетных данных.

Реализация данного модуля не зависит от реализации других модулей. От модуля «Шифрование» будет зависеть реализация следующих функций:

* регистрация пользователей;
* управление пользователями;
* проведение тестирования.

# Проектирование структуры данных

В таблице (Таблица 2.1) представлено описание функций системы и данные, с которыми соответствующая функция будет работать.

Таблица . – Функции системы

|  |  |  |
| --- | --- | --- |
| **№** | **Функция** | **Данные** |
| 1 | шифрование методом сдвига | шифруемый символ, ключ шифрования, алфавит допустимых символов |
| 2 | поиск номера символа в алфавите | искомый символ, алфавит допустимых символов |
| 3 | шифрование методом Цезаря | шифруемый символ, ключ шифрования, алфавит допустимых символов |
| 4 | шифрование афинным методом | шифруемый символ, ключи шифрования, алфавит допустимых символов |
| 5 | возведение целого числа в степень по модулю | основание степени, показатель степени, модуль |
| 6 | шифрование методом RSA | шифруемый символ, алфавит допустимых символов |
| 7 | шифрование входной строки | строка для шифрования, метод шифрования |
| 8 | проверка ошибок | строка для шифрования, зашифрованный текст, метод шифрования |

Таким образом, модуль «Шифрование» будет работать со следующими данными:

* строка, которую необходимо зашифровать;
* символы шифруемой строки;
* метод шифрования;
* ключ шифрования;
* алфавит допустимых символов.

Входными данными для модуля «Шифрование» будут:

* учетные данные пользователя;
* выбранный метод шифрования.

Выходными данными будут:

* данные, зашифрованные согласно выбранному методу;
* признак верности введенных учетных данных при повторной авторизации.

Для хранения данных потребуются:

* целочисленные константы, являющиеся ключами шифрования;
* строковые константы, хранящие допустимые символы используемого алфавита;
* логическая переменная, являющаяся признаком правильности введенных учетных данных при повторной авторизации пользователя.

# Проектирование модульной структуры программы

Данная программа представляет собой единый модуль.

Данный модуль используется следующими модулями:

* модулем «Регистрация» для шифрования учетных данных пользователя;
* модулем «Пользователи» для проверки верности введенных пользователем учетных данных.

# Проектирование алгоритмов

Для реализации заявленных в курсовом проекте задач необходима разработке следующих алгоритмов:

* поиск символа в алфавите (Листинг 1);
* шифрование методом сдвига (Листинг 2);
* шифрование методом Цезаря (Листинг 3);
* шифрование афинным методом (Листинг 4);
* возведение числа в степень по модулю (Листинг 5);
* шифрование методом RSA (Листинг 6);
* шифрование строки по выбранному методу (Листинг 7);
* проверка введенных учетных данных (Листинг 8).

Шифрование данных реализовано с помощью нескольких различных методов:

* шифр сдвига: для каждого символа входной строки с помощью функции Find берется номер этого символа из алфавита допустимых символов, и к этому значению прибавляется значение ключа KEY;
* шифр Цезаря: для каждого символа входной строки с помощью функции Find берется номер этого символа из алфавита допустимых символов, это значение увеличивается на значение ключа KEY, и находится остаток от деления данного значения на длину алфавита;
* афинный шифр: для каждого символа входной строки с помощью функции Find берется номер этого символа из алфавита допустимых символов, это значение умножается на значение первого ключа AKEY, затем прибавляет значение второго ключа BKEY, и находится остаток от деления данного значения на длину алфавита. Ключ шифрования должен быть взаимно простым со значением длины алфавита. Значение первого ключа находится как частное от целочисленного деления значения ключа на количество символов алфавита, второго ключа — как остаток от целочисленного деления значения ключа на количество символов алфавита;
* шифр RSA: для каждого символа входной строки с помощью функции Find берется номер этого символа из алфавита допустимых символов, данное значение возводится в степень, значение которой является открытым ключом PExp, по модулю N. Открытый ключ PExp и модуль N должны быть взаимно простыми целыми числами. Модуль N является произведением двух простых чисел.

**function** Find(smbl)

//Поиск символа в алфавите

//Вход: шифруемый символ smbl

//Выход: номер символа в алфавите ALPH

i ← 1;

**while** smbl <> ALPH[i] do

i ← i + 1;

**end while**

**end function**

Листинг ­– Поиск символа в алфавите

**function** Shift(sym)

//Шифрование методом сдвига

//Вход: шифруемый символ sym

//Выход: зашифрованный символ

KEY ← 6;

Find(sym) + KEY;

**end function**

Листинг ­– Поиск символа в алфавите

**function** Caesar(sym)

//Шифрование методом Цезаря

//Вход: шифруемый символ sym

//Выход: зашифрованный символ

KEY ← 5;

(Find(sym) + KEY) mod ALPH\_LEN;

**end function**

Листинг ­– Поиск символа в алфавите

**function** Affine(sym)

//Шифрование афинным методом

//Вход: шифруемый символ sym

//Выход: зашифрованный символ

KEY ← 1359;

AKEY ← KEY div ALPH\_LEN;

BKEY ← KEY mod ALPH\_LEN;

(Find(sym) \* AKEY + BKEY) mod ALPH\_LEN;

**end function**

Листинг 4 ­– Поиск символа в алфавите

**function** ModExp(num, exp, n)

//Возведение числа в степень по модулю

//Вход: основание степени num, показатель степени exp, модуль n

//Выход: результат операции

**for** i ← 2 (1) exp do

num ← (num \* num) mod n

**end for**

**end function**

Листинг ­– Поиск символа в алфавите

**function** RSA(sym)

//Шифрование методом RSA

//Вход: шифруемый символ sym

//Выход: зашифрованный символ

PExp ← 17; // public exponent

N ← 581629; // modulus

num ← Find(sym)

ModExp(num, PExp, N)

**end function**

Листинг ­– Поиск символа в алфавите

**function** Encrypt(s, f)

//Шифрование строки по выбранному методу

//Вход: строка для шифрования s; метод шифрования f

//Выход: зашифрованная строка

**for** i ← 1 (1) length(s) do

res ← res + f(s[i])

**end for**

**end function**

Листинг ­– Поиск символа в алфавите

**function** Errata(pl, cip, f)

//Проверка введенных учетных данных

//Вход: данные, введенные пользователем pl; зашифрованные данные //cip; метод шифрования f

//Выход: True, если найдены ошибки; False, если данные введены верно

not (Encrypt(pl, f) = cip)

**end function**

Листинг ­– Поиск символа в алфавите

# Разработка тестов

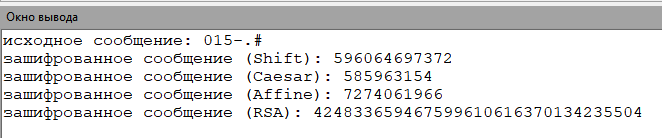


Рисунок – Пример шифрования 1

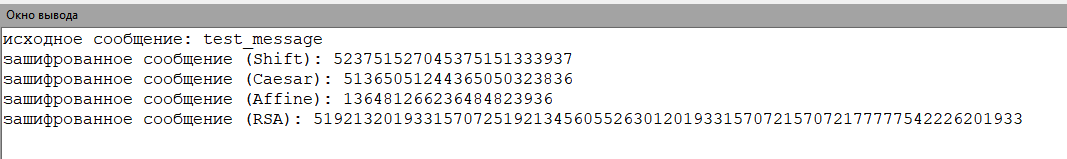


Рисунок – Пример шифрования 2

Проверка работы методов шифрования (Рисунок 2, Рисунок 1)

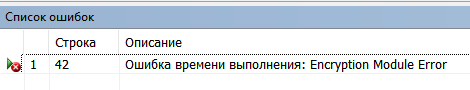


Рисунок –

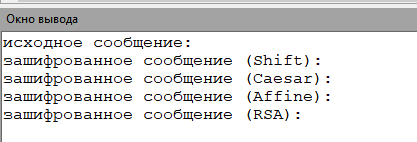


Рисунок – Пустая строка

Наличие недопустимого символа приводит к ошибке работы модуля (Рисунок 3).

Шифрование пустой строки (Рисунок 4).

Приложение А. Список тестовых вопросов

Приложение Б. Листинг программных модулей

**unit** Encryption;

//Модуль "Шифрование"

//Обеспечивает шифрование учетных данных и проверку

//верности введенных пользователем данных

**interface**

**type**

TFunc = **function** (sym: char): integer;

**function** RSA(sym: char): integer;

**function** Shift(sym: char): integer;

**function** Caesar(sym: char): integer;

**function** Affine(sym: char): integer;

**function** Encrypt(**const** s: string; f: TFunc):string;

**function** Errata(**var** pl, cip: string; f: TFunc):boolean;

**implementation**

**const**

//Алфавит допустимых символов

CAPLAT = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

SMLAT = 'abcdefghijklmnopqrstuvwxyz';

NUM = '0123456789';

SYM = '-\_\*#.';

ALPH = CAPLAT + SMLAT + NUM + SYM; // 67

ALPH\_LEN = 67;

**function** Find(smbl: char): integer;

//Поиск символа в алфавите

//Вход: шифруемый символ

//Выход: номер символа в алфавите

**var**

i: integer;

**begin**

i := 1;

**while** smbl <> ALPH[i] **do**

**begin**

i := i + 1;

**if** i > ALPH\_LEN **then**

**raise new** System.Exception('Encryption Module Error');

**end**;

Find := i;

**end**;

**function** Shift(**const** sym: char): integer;

//Шифрование методом сдвига

//Вход: шифруемый символ

//Выход: зашифрованный символ

**const**

KEY = 6;

**begin**

Shift := Find(sym) + KEY;

**end**;

**function** Caesar(sym: char): integer;

//Шифрование методом Цезаря

//Вход: шифруемый символ

//Выход: зашифрованный символ

**const**

KEY = 5;

**begin**

Caesar := (Find(sym) + KEY) **mod** ALPH\_LEN;

**end**;

**function** Affine(sym: char): integer;

//Шифрование афинным методом

//Вход: шифруемый символ

//Выход: зашифрованный символ

**const**

// gcd(key, size of symbol set) == 1

KEY = 1359;

AKEY = KEY **div** ALPH\_LEN;

BKEY = KEY **mod** ALPH\_LEN;

**begin**

Affine := (Find(sym) \* AKEY + BKEY) **mod** ALPH\_LEN;

**end**;

**function** ModExp(num, exp, n: integer): integer;

//Возведение числа в степень по модулю

//Вход: основание степень, показатель степени, модуль

//Выход: результат операции

**var**

c: Biginteger;

a: **array of** byte;

res, mul: integer;

**begin**

c := 1;

**for var** i := 1 **to** exp **do**

c := (c \* num) **mod** n;

a := c.ToByteArray();

res := 0;

mul := 1;

**for var** i := Low(a) **to** High(a) **do**

**begin**

res := res + a[i] \* mul;

mul := mul \* 256;

**end**;

ModExp := res;

**end**;

**function** RSA(**const** sym: char): integer;

//Шифрование методом RSA

//Вход: шифруемый символ

//Выход: зашифрованный символ

**const**

//p = 971;

//q = 599;

PExp = 17; // public exponent

N = 581629; // modulus

**var**

num: integer;

**begin**

num := Find(sym);

RSA := ModExp(num, PExp, N);

**end**;

**function** Encrypt(**const** s: string; f: TFunc):string;

//Шифрование строки по выбранному методу

//Вход: строка для шифрования; метод шифрования

//Выход: зашифрованная строка

**var**

res: string;

**begin**

**for var** i := 1 **to** length(s) **do**

**begin**

res := res + f(s[i]);

**end**;

Encrypt := res;

**end**;

**function** Errata(**var** pl, cip: string; f: TFunc):boolean;

//Проверка введенных учетных данных

//Вход: данные, введенные пользователем; зашифрованные данные; метод шифрования

//Выход: True, если найдены ошибки; False, если данные введены верно

**begin**

Errata := **not** (Encrypt(pl, f) = cip);

**end**;

**begin**

**end**.