Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Волгоградский государственный технический университет»

Факультет                                 электроники и вычислительной техники

(наименование факультета)

Кафедра                     Программное обеспечение автоматизированных систем

(наименование кафедры)

**ОТЧЕТ**

**О     учебной      практике на                кафедре ПОАС**

 вид практики наименование кафедры или научной

лаборатории университета

Руководитель практики от

университета,

  доцент кафедры ПОАС    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Литовкин Д.В.

должность подпись

Студент гр.      ПрИн-266       \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бирюков Н. А.

подпись

Студент гр.      ПрИн-266       \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Островский Д. С.

подпись

Отчет защищен с оценкой:

Бирюков Н. А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Островский Д. С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Волгоград 2016 г

**Содержание**

[Введение 3](#_Toc457564181)

[Формулировка задания 4](#_Toc457564182)

[Содержание практики 5](#_Toc457564183)

[Заключение 6](#_Toc457564184)

[Приложение А. Способ шифрования-дешифрования данных 7](#_Toc457564185)

[Приложение Б. Внешняя спецификация программы 8](#_Toc457564186)

[Приложение В. Проект программы 11](#_Toc457564187)

[Приложение Г. Модульные тесты 14](#_Toc457564188)

[Приложение Д. Разделение обязанностей между членами команды 33](#_Toc457564189)

[Приложение Е. Порядок конструирования программы 3](#_Toc457564190)4

[Приложение Ж. Код программы 3](#_Toc457564191)5

[Приложение З. Оценка полученных результатов 4](#_Toc457564192)1

# Введение

Целью учебной практики является закрепление навыков, связанных с процедурно-ориентированным программированием, а именно:

* составление внешней спецификации программы;
* декомпозиция программы на подпрограммы;
* создание модульных тестов;
* программирование с псевдокодом;
* создание собственных структур данных;
* реализация типовых алгоритмических структур на языке программирования;
* форматирование кода в соответствии с единым стилем кодирования;
* проектирование программы «сверху-вниз» и ее отладка – «снизу-вверх»;
* ведение репозитория кода.

Так же учебная практика дает навыки разработки программы в команде.

# Формулировка задания

 Дан файл, содержащий исходный код программы. Сначала необходимо зашифровать его, получив зашифрованный (новый) файл. А затем дешифровать файл, получив (новый) исходный файл.

Шифратор должен работать в консольном режиме, получая параметры через командную строку. Помимо исходных и зашифрованных файлов шифратор должен создавать файл с логом, содержащим отчет о процессе шифрования и дешифрования.

Минимальный формат командной строки в режиме шифрования:

<имя шифратора>.exe -e <путь к файлу для шифрования> <путь к новому зашифрованному файлу>

Минимальный формат командной строки в режиме шифрования:

<имя шифратора >.exe -d <путь к зашифрованному файлу> <путь к новому дешифрованному файлу>

Командная строка может содержать дополнительные параметры, в зависимости от способа шифрования.

Способ упаковки-распаковки данных представлен в Приложении А.

# Содержание практики

В ходе прохождения учебной практики выполнены следующие виды работ:

* составлена внешняя спецификация программы – см. Приложение Б;
* программа декомпозирована на подпрограммы – см. Приложение В;
* созданы собственные типы данных – см. Приложение В;
* созданы модульные тесты к подпрограммам – см. Приложение Г;
* написан псевдокод для подпрограмм – см. Приложение Ж;
* закодирован псевдокод - см. Приложение Ж;
* код отформатирован в соответствии с единым стилем кодирования - см. Приложение Ж.

# Заключение

В ходе прохождения учебной практики закреплены все навыки, необходимые для создания процедурно-ориентированных программ, и как результат создана программа, которая шифрует/дешифрует данные в соответствии со способом, представленном в приложении А. Порядок конструирования программы представлен в Приложении Е. Объем работ, выполненный каждым членом команды, представлен в Приложении Д. Оценка полученных результатов представлена в Приложении З.

# Приложение А. Способ шифрования-дешифрования данных

Зафиксируем натуральное число k и перестановку чисел 1,...,k. При шифровке в исходном тексте к каждой из последовательных групп по k символов применяется зафиксированная перестановка. Затем генерируется следующая (еще не применявшаяся) перестановка чисел. Пусть k = 4 и перестановка есть 3, 2, 4, 1. Тогда группа символов s1, s2, s3, s4 заменяется на s3, s2, s4, s1. Если в последней группе меньше четырех символов, то к ней добавляются пробелы.

Режим кодирования:

Входные данные: исходный текст, k — длина перестановки символов

Выходные данные: зашифрованный текст

Режим декодирования:

Входные данные: зашифрованный текст, k — длина перестановки символов

Выходные данные: исходный текст

# Приложение Б. Внешняя спецификация программы

1. **Наимен**о**вание программы**

Программа шифрования и дешифрования текста(в дальнейшем – шифратор(coder)).

1. **Назначение программы**

Программа предназначена для зашифровки/дешифровки файлов. Программой могут пользоваться любые пользователи.

1. **Требования к режиму шифрования**
	1. **Состав и структура входных данных**

На вход подается путь к текстовому файлу для шифрования, число k(длина группы символов), и подается путь к зашифрованному текстовому файлу.

Пример командной строки:

coder.exe –e C:\text.txt 5 C:\entext.txt

* 1. **Состав и структура выходных данных**

На выходе получаем зашифрованный текстовый файл.

* 1. **Ограничения на выполнение**

Число k(длина группы символов) должно быть больше 2, и меньше 50.

1. **Требования к режиму дешифрования**
	1. **Состав и структура входных данных**

На вход подается путь к зашифрованному текстовому файлу, число k(длина группы символов), и путь к новому дешифрованному текстовому файлу.

Пример командной строки:

coder.exe –d C:\entext.txt 5 C:\text.txt

* 1. **Состав и структура выходных данных**

На выходе получаем дешифрованный текстовый файл.

* 1. **Ограничения на выполнение**

Число k(длина группы символов) должно быть больше 2, и меньше 50.

**Приложение Б.А. Структура лог-файла**

Файл с логом программы будет иметь имя: LogFile.txt

Формат выводимых сообщений:

чч:мм:сс [тип сообщения] [выводимое сообщение]

**Приложение Б.Б.** **Сообщения об ошибках командной строки**

Таблица Б.1. Сообщения об ошибках командной строки

|  |  |
| --- | --- |
| **Тип ошибки** | **Выводимое сообщение** |
| Указаны не все параметры командной строки | «Ошибка выполнения программы! Неверное количество аргументов командной строки.» |
| Нарушен диапазон длины группы символов(K) | «Ошибка выполнения программы! Длина группы символов K не входит в диапазон от 2 до 20» |
| Имя выходного файла совпадает с именем входного файла, что по логике быть не может | «Ошибка выполнения программы! Имя выходного файла совпадает с именем входного файла» |
| Входной файл не существует  | «Ошибка выполнения программы! Не найден входной файл» |
| На вход подали директорию | Ошибка выполнения программы! Не указан сам файл» |

# Приложение В. Проект программы

1. **Декомпозиция программы на подпрограммы**



Рисунок В.1 – Иерархия вызова функций

1. **Собственные структуры данных**

//Максимальный размер группы символов

MAX\_SIZE\_GROUP 50

//Максимальный размер буфера

MAX\_SIZE\_BUFFER 10000

//Данные файла

struct FileData

{

 //Дескриптор файла

 int handleFile;

//Буфер

 char buf[MAX\_SIZE\_BUFFER+1];

};

//Тип сообщения

enum TypeMessage

{

 //Информация

 info,

//Ошибка

 error

};

1. **Перечень подпрограмм**

/\*! Функция зашифровки/дешифровки группы

\*\param[in|out] textGroup – группа

\*\param[in] permutation – перестановка

\*\param[in] K - длина группы символов

\*\param[in] encrypt - ключ, если true, то шифровать, иначе если false, то дешифровать

\*/

void enDeCryptTextGroup(char textGroup[MAX\_SIZE\_GROUP+1], int permutation[MAX\_SIZE\_GROUP], int K, bool encrypt);

/\*! Функция генерации следующей группы

\*\param[in|out] Permutation – новая перестановка

\*\param[in] K – длина группы символов

\*/

void nextSet(int Permutation[MAX\_SIZE\_GROUP], int K);

/\*! Функция перестановки элементов строки

\*\param[in|out] group – перестановка

\*\param[in] firstIndex – индекс первого числа

\*\param[in] secondIndex - индекс второго числа

\*/

void swap(int group[MAX\_SIZE\_GROUP],int firstIndex, int secondIndex);

/\*! Функция ведения лога файла

\*\param[in] handle - дескриптор файла

\*\param[in] type – тип выводимого сообщения

\*\param[in] message – сообщение

\*/

void logInfo(int handle, TypeMessage type, char\* message);

/\*! Функция получении имени файла из директории

\*\param[in] direct - директория с указанием файла

\*\param[in] nameFile - имя файла

\*/

void getNameFileFromDirect(char\* direct, char\* nameFile);

/\*! Функция проверки командной строки

\*\param[in] countArguments - количество аргументов командной строки

\*\param[in] arguments - аргументы командной строки

\*/

void checkComandLine(int countArguments, char\* arguments[]) throw (char\*);

/\*! Функция обработки командной строки

\*\param[in] arguments - аргументы командной строки

\*\param[out] sizeGroup - размер группы символов

\*\param[out] fileNameIn - имя исходного файла

\*\param[out] fileNameOut - имя выходного файла

\*\return true - происходит шифрация, иначе дешифрация

\*/

bool getDataFromComandLine(char\* arguments[], int& sizeGroup, char\* fileNameIn, char\* fileNameOut);

/\*! Функция получения группы символов

\*\param[in|out] input – данные файла: буфер и дескриптор файла

\*\param[out] group – группа

\*\param[in] K – длина группы символов

\*\ return возможность получения следующей группы

\*/

bool readBytes(FileData \*input, char group[MAX\_SIZE\_GROUP+1], int K);

/\*! Функция изъятия первой группы символов из буфера

\*\param[in|out] buf - буфер данных

\*\param[out] group – группа

\*\param[in] K - длина группы символов

\*/

void takeFirstGroupFromBuffer(char buf[10001], char group[MAX\_SIZE\_GROUP+1], int K);

/\*! Функция добавления пробелов в конец буфера для кратности числу K

\*\param[in|out] buf - буфер данных

\*\param[in] K - длина группы символов

\*/

void addSpacesToBuffer(char buf[10001], int K);

/\*! Функция записи группы символов в выходной файл

\*\param[in|out] output – данные файла: буфер и дескриптор файла

\*\param[in] group – группа

\*\param[in] K – длина группы символов

\*\param[in] haveGroup - имеется ли еще группа или нет

\*/

void writeBytes(FileData \*output, char group[MAX\_SIZE\_GROUP+1], int K, bool haveGroup);

# Приложение Г. Модульные тесты

**Модульные тесты для подпрограммы addSpacesToBuffer**

//не хватает одного пробела

 void notEnoughSingleSpace()

 {

 //arrange

 int K=4;

 char buf[MAX\_SIZE\_BUFFER+1]={"1234567"};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={"1234567 "};

 //act

 addSpacesToBuffer(buf,K);

 //compare

 bool result=strcmp(buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

 //не хватает максимальное количество пробелов

 void lacksTheMaximumNumberOfSpaces()

 {

 //arrange

 int K=4;

 char buf[MAX\_SIZE\_BUFFER+1]={"1"};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={"1 "};

 //act

 addSpacesToBuffer(buf,K);

 //compare

 bool result=strcmp(buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

 //буфер пустой

 void bufferIsEmpty()

 {

 //arrange

 int K=4;

 char buf[MAX\_SIZE\_BUFFER+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={""};

 //act

 addSpacesToBuffer(buf,K);

 //compare

 bool result=strcmp(buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

**Модульные тесты для подпрограммы checkComandLine**

//количество аргументов некорректно

 void countArgumentsError()

 {

 int countArguments=4;

 bool result;

 char test[200]={"Ошибка выполнения программы! Неверное количество аргументов командной строки."};

 char\* arguments[20]={"coder.exe","-e","5","C:\\entext.txt"};

 try

 {

 checkComandLine(countArguments,arguments);

 }

 catch(char\* str)

 {

 result=strcmp(test,str);

 }

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

 //значение К выходит за минимальную границу диапазона

 void diapazonKMin()

 {

 int countArguments=5;

 bool result;

 char test[200]={"Ошибка выполнения программы! Длина группы символов K не входит в диапазон от 2 до 50"};

 char\* arguments[20]={"coder.exe","-e","C:\\in.txt","1","C:\\entext.txt"};

 try

 {

 checkComandLine(countArguments,arguments);

 }

 catch(char\* str)

 {

 result=strcmp(test,str);

 }

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

 //значение К выходит за максимальную границу диапазона

 void diapazonKMax()

 {

 int countArguments=5;

 bool result;

 char test[200]={"Ошибка выполнения программы! Длина группы символов K не входит в диапазон от 2 до 50"};

 char\* arguments[20]={"coder.exe","-e","C:\\in.txt","51","C:\\entext.txt"};

 try

 {

 checkComandLine(countArguments,arguments);

 }

 catch(char\* str)

 {

 result=strcmp(test,str);

 }

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

 //одинаковые имена входного и выходного файлов

 void equalNameFile()

 {

 int countArguments=5;

 bool result;

 char test[200]={"Ошибка выполнения программы! Имя выходного файла совпадает с именем входного файла"};

 char\* arguments[20]={"coder.exe","-e","C:\\in.txt","5","C:\\in.txt"};

 try

 {

 checkComandLine(countArguments,arguments);

 }

 catch(char\* str)

 {

 result=strcmp(test,str);

 }

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

 //не указано имя входного файла

 void directIn()

 {

 int countArguments=5;

 bool result;

 char test[200]={"Ошибка выполнения программы! Не указан сам файл"};

 char\* arguments[20]={"coder.exe","-e","C:\\file","5","C:\\in.txt"};

 try

 {

 checkComandLine(countArguments,arguments);

 }

 catch(char\* str)

 {

 result=strcmp(test,str);

 }

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

 //не указано имя выходного файла

 void directOut()

 {

 int countArguments=5;

 bool result;

 char test[200]={"Ошибка выполнения программы! Не указан сам файл"};

 char\* arguments[20]={"coder.exe","-e","C:\\in.txt","5","C:\\in"};

 try

 {

 checkComandLine(countArguments,arguments);

 }

 catch(char\* str)

 {

 result=strcmp(test,str);

 }

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

**Модульные тесты для подпрограммы enDeCryptTextGroup**

bool compareText(char textGroup[MAX\_SIZE\_GROUP+1], char expectedTextGroup[MAX\_SIZE\_GROUP+1])

 {

 bool result=true;

 if(strcmp(textGroup,expectedTextGroup)!=0)

 {

 result=false;

 }

 return result;

 }

 //минимальная длина группы, шифрование

 void minimumLengthOfGroupEn()

 {

 //arrange

 int K=2;

 char textGroup[MAX\_SIZE\_GROUP+1]={"ab"};

 int permutation[MAX\_SIZE\_GROUP]={2,1};

 bool encrypt=true;

 char expectedTextGroup[MAX\_SIZE\_GROUP+1]={"ba"};

 //act

 enDeCryptTextGroup(textGroup, permutation, K, encrypt);

 //compare

 bool result=compareText(textGroup,expectedTextGroup);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //максимальная длина группы, шифрование

 void maximumLengthOfGroupEn()

 {

 //arrange

 int K=50;

 char textGroup[MAX\_SIZE\_GROUP+1]={"abcdefghijklmnopqrstuvwxyz1234567890abcdefgtuvwxyz"};

 int permutation[MAX\_SIZE\_GROUP]={20,19,18,17,16,15,14,13,12,1,2,3,4,5,6,7,8,9,10,11,30,29,28,27,26,25,24,23,22,21,40,41,42,43,44,45,46,47,50,49,48,35,34,33,32,31,36,37,38,39};

 bool encrypt=true;

 char expectedTextGroup[MAX\_SIZE\_GROUP+1]={"tsrqponmlabcdefghijk4321zyxwvudefgtuvwzyx987650abc"};

 //act

 enDeCryptTextGroup(textGroup, permutation, K, encrypt);

 //compare

 bool result=compareText(textGroup,expectedTextGroup);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //средняя длина группы, шифрование

 void averageLengthOfGroupEn()

 {

 //arrange

 int K=20;

 char textGroup[MAX\_SIZE\_GROUP+1]={"zabcdefghijklmnopqrs"};

 int permutation[MAX\_SIZE\_GROUP]={20,19,18,17,16,15,14,13,12,1,2,3,4,5,6,7,8,9,10,11};

 bool encrypt=true;

 char expectedTextGroup[MAX\_SIZE\_GROUP+1]={"srqponmlkzabcdefghij"};

 //act

 enDeCryptTextGroup(textGroup, permutation, K, encrypt);

 //compare

 bool result=compareText(textGroup,expectedTextGroup);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //нет группы

 void noneGroup()

 {

 int K=7;

 char textGroup[MAX\_SIZE\_GROUP+1]={""};

 int permutation[MAX\_SIZE\_GROUP]={6,7,1,2,3,4,5};

 bool encrypt=true;

 char expectedTextGroup[MAX\_SIZE\_GROUP+1]={""};

 //act

 enDeCryptTextGroup(textGroup, permutation, K, encrypt);

 //compare

 bool result=compareText(textGroup,expectedTextGroup);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //минимальная длина группы, дешифрование

 void minimumLengthOfGroupDe()

 {

 //arrange

 int K=2;

 char textGroup[MAX\_SIZE\_GROUP+1]={"ba"};

 int permutation[MAX\_SIZE\_GROUP]={2,1};

 bool encrypt=false;

 char expectedTextGroup[MAX\_SIZE\_GROUP+1]={"ab"};

 //act

 enDeCryptTextGroup(textGroup, permutation, K, encrypt);

 //compare

 bool result=compareText(textGroup,expectedTextGroup);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //максимальная длина группы, дешифрование

 void maximumLengthOfGroupDe()

 {

 //arrange

 int K=50;

 char textGroup[MAX\_SIZE\_GROUP+1]={"tsrqponmlabcdefghijk4321zyxwvudefgtuvwzyx987650abc"};

 int permutation[MAX\_SIZE\_GROUP]={20,19,18,17,16,15,14,13,12,1,2,3,4,5,6,7,8,9,10,11,30,29,28,27,26,25,24,23,22,21,40,41,42,43,44,45,46,47,50,49,48,35,34,33,32,31,36,37,38,39};

 bool encrypt=false;

 char expectedTextGroup[MAX\_SIZE\_GROUP+1]={"abcdefghijklmnopqrstuvwxyz1234567890abcdefgtuvwxyz"};

 //act

 enDeCryptTextGroup(textGroup, permutation, K, encrypt);

 //compare

 bool result=compareText(textGroup,expectedTextGroup);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //средняя длина группы, дешифрование

 void averageLengthOfGroupDe()

 {

 //arrange

 int K=20;

 char textGroup[MAX\_SIZE\_GROUP+1]={"srqponmlkzabcdefghij"};

 int permutation[MAX\_SIZE\_GROUP]={20,19,18,17,16,15,14,13,12,1,2,3,4,5,6,7,8,9,10,11};

 bool encrypt=false;

 char expectedTextGroup[MAX\_SIZE\_GROUP+1]={"zabcdefghijklmnopqrs"};

 //act

 enDeCryptTextGroup(textGroup, permutation, K, encrypt);

 //compare

 bool result=compareText(textGroup,expectedTextGroup);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

**Модульные тесты для подпрограммы getDataFromComandLine**

//копирование выходного файла

 void OutNameCopy()

 {

 char\* arguments[20]={"Coder.exe","-e","C:\\in.txt","4","out.txt"};

 char In[20]={""};

 char Out[20]={""};

 char OutTest[20]="out.txt";

 int K;

 bool key=getDataFromComandLine(arguments,K,In,Out);

 bool res=strcmp(Out,OutTest);

 CFIXCC\_ASSERT\_EQUALS(false,res);

 }

 //копирование абсолютного пути входного файла

 void InAbsNameCopy()

 {

 char\* arguments[20]={"Coder.exe","-e","C:\\in.txt","4","out.txt"};

 char In[20]={""};

 char Out[20]={""};

 char InTest[20]="C:\\in.txt";

 int K;

 bool key=getDataFromComandLine(arguments,K,In,Out);

 bool res=strcmp(In,InTest);

 CFIXCC\_ASSERT\_EQUALS(false,res);

 }

 //копирование относительного пути входного файла

 void InNotAbsNameCopy()

 {

 char\* arguments[20]={"Coder.exe","-e","in/in.txt","4","out.txt"};

 char In[200]={""};

 char Out[200]={""};

 char InTest[200]={""};

 \_getcwd(InTest,1000);

 strcat(InTest,"\\in\\in.txt");

 int K;

 bool key=getDataFromComandLine(arguments,K,In,Out);

 bool res=strcmp(In,InTest);

 CFIXCC\_ASSERT\_EQUALS(false,res);

 }

 //копирование значения K

 void KCopy()

 {

 char\* arguments[20]={"Coder.exe","-e","C:\\in.txt","4","out.txt"};

 char In[20]={""};

 char Out[20]={""};

 int K;

 bool key=getDataFromComandLine(arguments,K,In,Out);

 CFIXCC\_ASSERT\_EQUALS(4,K);

 }

 //вывод по ключу шифрации

 void KeyEnCopy()

 {

 char\* arguments[20]={"Coder.exe","-e","C:\\in.txt","4","out.txt"};

 char In[20]={""};

 char Out[20]={""};

 int K;

 bool key=getDataFromComandLine(arguments,K,In,Out);

 CFIXCC\_ASSERT\_EQUALS(true,key);

 }

 //вывод по ключу дешифрации

 void KeyDeCopy()

 {

 char\* arguments[20]={"Coder.exe","-d","C:\\in.txt","4","out.txt"};

 char In[20]={""};

 char Out[20]={""};

 int K;

 bool key=getDataFromComandLine(arguments,K,In,Out);

 CFIXCC\_ASSERT\_EQUALS(false,key);

 }

**Модульные тесты для подпрограммы getNameFileFromDirect**

//в командную строку подано имя

 void nameIsDir()

 {

 bool result;

 char test[20]={"in.txt"};

 char dir[200]={"in.txt"};

 char name[20]={""};

 getNameFileFromDirect(dir,name);

 bool resCmp=strcmp(test,name);

 CFIXCC\_ASSERT\_EQUALS(false,resCmp);

 }

 //в командную строку подан абсолютный путь

 void slashInDir()

 {

 bool result;

 char test[20]={"in.txt"};

 char dir[200]={"C:/File/in.txt"};

 char name[20]={""};

 getNameFileFromDirect(dir,name);

 bool resCmp=strcmp(test,name);

 CFIXCC\_ASSERT\_EQUALS(false,resCmp);

 }

 //в командную строку подан относительный путь

 void backSlashInDir()

 {

 bool result;

 char test[20]={"in.txt"};

 char dir[200]={"C:\\File\\in.txt"};

 char name[20]={""};

 getNameFileFromDirect(dir,name);

 bool resCmp=strcmp(test,name);

 CFIXCC\_ASSERT\_EQUALS(false,resCmp);

 }

 **Модульные тесты для подпрограммы logInfo**

//сообщение об ошибке

 void messageAboutError()

 {

 //arrange

 int handle;

 char message[100]={"Ошибка!!!"};

 char expectedFile[100]={"[error] Ошибка!!!\n"};

 char file[100]={""};

 //act

 char filename[100]={"logTest.txt"};

 handle=\_open(filename,\_O\_TRUNC | \_O\_BINARY | \_O\_RDWR, \_S\_IREAD | \_S\_IWRITE);

 logInfo(handle,error,message);

 lseek(handle,0,SEEK\_SET);

 int realSize=read(handle,(void\*)file, sizeof(char)\*100);

 file[realSize]='\0';

 close(handle);

 //compare

 bool result=strcmp(strchr(file,'['),expectedFile);

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

 //информативное сообщение

 void messageAboutInfo()

 {

 //arrange

 int handle;

 char message[100]={"Зашифрованна группа символов"};

 char expectedFile[100]={"[info] Зашифрованна группа символов\n"};

 char file[100]={""};

 //act

 char filename[100]={"logTest.txt"};

 handle=\_open(filename,\_O\_TRUNC | \_O\_BINARY | \_O\_RDWR, \_S\_IREAD | \_S\_IWRITE);

 logInfo(handle,info,message);

 lseek(handle,0,SEEK\_SET);

 int realSize=read(handle,(void\*)file, sizeof(char)\*100);

 file[realSize]='\0';

 close(handle);

 //compare

 bool result=strcmp(strchr(file,'['),expectedFile);

 CFIXCC\_ASSERT\_EQUALS(false,result);

 }

**Модульные тесты для подпрограммы nextSet**

bool compareText(int textGroup[MAX\_SIZE\_GROUP], int expectedTextGroup[MAX\_SIZE\_GROUP], int countChar)

 {

 bool result=true;

 for(int i=0;i<countChar && result;i++)

 {

 if(textGroup[i]!=expectedTextGroup[i])

 {

 result=false;

 }

 }

 return result;

 }

 //первый набор

 void firstSet()

 {

 int Permutation[MAX\_SIZE\_GROUP];

 int test[MAX\_SIZE\_GROUP]={1,2,4,3};

 int K=4;

 nextSet(Permutation,K);

 bool resultCmp=compareText(Permutation,test,K);

 CFIXCC\_ASSERT\_EQUALS(true,resultCmp);

 }

 //первая перестановка

 void firstPermutation()

 {

 int Permutation[MAX\_SIZE\_GROUP]={1,2,3,4};

 int test[MAX\_SIZE\_GROUP]={1,2,4,3};

 int K=4;

 nextSet(Permutation,K);

 bool resultCmp=compareText(Permutation,test,K);

 CFIXCC\_ASSERT\_EQUALS(true,resultCmp);

 }

 //вторая перестановка

 void secondPermutation()

 {

 int Permutation[MAX\_SIZE\_GROUP]={1,2,4,3};

 int test[MAX\_SIZE\_GROUP]={1,3,2,4};

 int K=4;

 nextSet(Permutation,K);

 bool resultCmp=compareText(Permutation,test,K);

 CFIXCC\_ASSERT\_EQUALS(true,resultCmp);

 }

 //перестановка в середине

 void midPermutation()

 {

 int Permutation[MAX\_SIZE\_GROUP]={2,1,3,4};

 int test[MAX\_SIZE\_GROUP]={2,1,4,3};

 int K=4;

 nextSet(Permutation,K);

 bool resultCmp=compareText(Permutation,test,K);

 CFIXCC\_ASSERT\_EQUALS(true,resultCmp);

 }

 //конечная перестановка

 void endPermutation()

 {

 int Permutation[MAX\_SIZE\_GROUP]={4,3,2,1};

 int test[MAX\_SIZE\_GROUP]={1,2,3,4};

 int K=4;

 nextSet(Permutation,K);

 bool resultCmp=compareText(Permutation,test,K);

 CFIXCC\_ASSERT\_EQUALS(true,resultCmp);

 }

**Модульные тесты для подпрограммы readBytes**

bool compare(char textGroup[MAX\_SIZE\_GROUP+1], char expectedTextGroup[MAX\_SIZE\_GROUP+1], bool result, bool expectedResult, char buf[MAX\_SIZE\_BUFFER+1], char expectedBuf[MAX\_SIZE\_BUFFER+1])

 {

 bool res=true;

 if(strcmp(textGroup,expectedTextGroup)!=0 || strcmp(buf,expectedBuf)!=0 || result!=expectedResult)

 {

 res=false;

 }

 return res;

 }

 //в буфере имеются данные

 void inTheBufferHasData()

 {

 //arrange

 FileData input;

 strcpy(input.buf,"12345678");

 char group[MAX\_SIZE\_GROUP+1]={""};

 int K=4;

 input.handleFile=8;

 char test[MAX\_SIZE\_GROUP+1]={"1234"};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={"5678"};

 bool expectedResult=true;

 //act

 bool result=readBytes(&input,group,K);

 //compare

 bool res=compare(group,test,result,expectedResult,input.buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,res);

 }

 //в буфере нет данных и файл имеет не кратное K последнее кол-во символов

 void inTheBufferNoDataAndFileHaveNotMultipleNumberLast()

 {

 //arrange

 FileData input;

 strcpy(input.buf,"");

 char group[MAX\_SIZE\_GROUP+1]={""};

 int K=4;

 input.handleFile;

 char filename[100]={"Test\_readBytes\_inTheBufferNoDataAndFileHaveNotMultipleNumberLast.txt"};

 char test[MAX\_SIZE\_GROUP+1]={"12 "};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={""};

 bool expectedResult=true;

 //act

 input.handleFile=\_open(filename,\_O\_BINARY | \_O\_RDONLY, \_S\_IREAD);

 bool result=readBytes(&input,group,K);

 close(input.handleFile);

 //compare

 bool res=compare(group,test,result,expectedResult,input.buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,res);

 }

 //в буфере нет данных и файл имеет не кратное K не последнее кол-во символов

 void inTheBufferNoDataAndFileHaveNotMultipleNumberNoLast()

 {

 //arrange

 FileData input;

 strcpy(input.buf,"");

 char group[MAX\_SIZE\_GROUP+1]={""};

 int K=4;

 input.handleFile;

 char filename[100]={"Test\_readBytes\_inTheBufferNoDataAndFileHaveNotMultipleNumberNoLast.txt"};

 char test[MAX\_SIZE\_GROUP+1]={"1234"};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={"56 "};

 bool expectedResult=true;

 //act

 input.handleFile=\_open(filename,\_O\_BINARY | \_O\_RDONLY, \_S\_IREAD);

 bool result=readBytes(&input,group,K);

 close(input.handleFile);

 //compare

 bool res=compare(group,test,result,expectedResult,input.buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,res);

 }

 //в буфере нет данных но в файле есть

 void inTheBufferNoDataButFileHaveData()

 {

 //arrange

 FileData input;

 strcpy(input.buf,"");

 char group[MAX\_SIZE\_GROUP+1]={""};

 int K=4;

 input.handleFile;

 char filename[100]={"Test\_readBytes\_inTheBufferNoDataButFileHaveData.txt"};

 char test[MAX\_SIZE\_GROUP+1]={"1234"};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={"5678"};

 bool expectedResult=true;

 //act

 input.handleFile=\_open(filename,\_O\_BINARY | \_O\_RDONLY, \_S\_IREAD);

 bool result=readBytes(&input,group,K);

 close(input.handleFile);

 //compare

 bool res=compare(group,test,result,expectedResult,input.buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,res);

 }

 //ни буфер ни файл не имеют данных

 void neitherTheBufferNorTheFileHasNoData()

 {

 //arrange

 FileData input;

 strcpy(input.buf,"");

 char group[MAX\_SIZE\_GROUP+1]={""};

 int K=4;

 input.handleFile;

 char filename[100]={"Test\_readBytes\_neitherTheBufferNorTheFileHasNoData.txt"};

 char test[MAX\_SIZE\_GROUP+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={""};

 bool expectedResult=false;

 //act

 input.handleFile=\_open(filename,\_O\_BINARY | \_O\_RDONLY, \_S\_IREAD);

 bool result=readBytes(&input,group,K);

 close(input.handleFile);

 //compare

 bool res=compare(group,test,result,expectedResult,input.buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,res);

 }

**Модульные тесты для подпрограммы swap**

bool compareText(int textGroup[MAX\_SIZE\_GROUP], int expectedTextGroup[MAX\_SIZE\_GROUP], int countChar)

 {

 bool result=true;

 for(int i=0;i<countChar && result;i++)

 {

 if(textGroup[i]!=expectedTextGroup[i])

 {

 result=false;

 }

 }

 return result;

 }

 //перемешаны первый и последний элементы

 void FirstAndEnd()

 {

 int Group[MAX\_SIZE\_GROUP]={1,2,3,4};

 int Test[MAX\_SIZE\_GROUP]={4,2,3,1};

 int First=0;

 int Second=3;

 swap(Group,First,Second);

 bool res=compareText(Test,Group,4);

 CFIXCC\_ASSERT\_EQUALS(true,res);

 }

 //перемешаны средний и первый элементы

 void MidAndFirst()

 {

 int Group[MAX\_SIZE\_GROUP]={1,2,3,4};

 int Test[MAX\_SIZE\_GROUP]={3,2,1,4};

 int First=2;

 int Second=0;

 swap(Group,First,Second);

 bool res=compareText(Test,Group,4);

 CFIXCC\_ASSERT\_EQUALS(true,res);

 }

 //перемешаны средний и последний элементы

 void MidAndEnd()

 {

 int Group[MAX\_SIZE\_GROUP]={1,2,3,4};

 int Test[MAX\_SIZE\_GROUP]={1,4,3,2};

 int First=1;

 int Second=3;

 swap(Group,First,Second);

 bool res=compareText(Test,Group,4);

 CFIXCC\_ASSERT\_EQUALS(true,res);

 }

 //перемешаны последний и первый элементы

 void EndAndFirst()

 {

 int Group[MAX\_SIZE\_GROUP]={1,2,3,4};

 int Test[MAX\_SIZE\_GROUP]={4,2,3,1};

 int First=3;

 int Second=0;

 swap(Group,First,Second);

 bool res=compareText(Test,Group,4);

 CFIXCC\_ASSERT\_EQUALS(true,res);

 }

**Модульные тесты для подпрограммы takeFirstGroupFromBuffer**

bool compare(char textGroup[MAX\_SIZE\_GROUP+1], char expectedTextGroup[MAX\_SIZE\_GROUP+1], char buf[MAX\_SIZE\_BUFFER+1], char expectedBuf[MAX\_SIZE\_BUFFER+1])

 {

 bool res=true;

 if(strcmp(textGroup,expectedTextGroup)!=0 || strcmp(buf,expectedBuf)!=0)

 {

 res=false;

 }

 return res;

 }

 //длина изымаемой группы меньше чем остаток от буфера

 void withdrawingGroupsLessThanRemainderOfTheBuffer()

 {

 //arrange

 int K=4;

 char buf[MAX\_SIZE\_BUFFER+1]={"1234567890123"};

 char group[MAX\_SIZE\_GROUP+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={"567890123"};

 char expectedGroup[MAX\_SIZE\_GROUP+1]={"1234"};

 //act

 takeFirstGroupFromBuffer(buf,group,K);

 //compare

 bool result=compare(group,expectedGroup,buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //длина изымаемой группы равна остатку от буфера

 void withdrawingGroupsEqualRemainderOfTheBuffer()

 {

 //arrange

 int K=4;

 char buf[MAX\_SIZE\_BUFFER+1]={"12345678"};

 char group[MAX\_SIZE\_GROUP+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={"5678"};

 char expectedGroup[MAX\_SIZE\_GROUP+1]={"1234"};

 //act

 takeFirstGroupFromBuffer(buf,group,K);

 //compare

 bool result=compare(group,expectedGroup,buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //изымаемая группа последняя

 void withdrawingGroupsIsLast()

 {

 //arrange

 int K=4;

 char buf[MAX\_SIZE\_BUFFER+1]={"1234"};

 char group[MAX\_SIZE\_GROUP+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={""};

 char expectedGroup[MAX\_SIZE\_GROUP+1]={"1234"};

 //act

 takeFirstGroupFromBuffer(buf,group,K);

 //compare

 bool result=compare(group,expectedGroup,buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //группы нет(буфер пуст)

 void bufferIsEmpty()

 {

 //arrange

 int K=4;

 char buf[MAX\_SIZE\_BUFFER+1]={""};

 char group[MAX\_SIZE\_GROUP+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={""};

 char expectedGroup[MAX\_SIZE\_GROUP+1]={""};

 //act

 takeFirstGroupFromBuffer(buf,group,K);

 //compare

 bool result=compare(group,expectedGroup,buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

**Модульные тесты для подпрограммы writeBytes**

bool compare(char bufFile[MAX\_SIZE\_BUFFER+1], char expectedBufFile[MAX\_SIZE\_BUFFER+1], char buf[MAX\_SIZE\_BUFFER+1], char expectedBuf[MAX\_SIZE\_BUFFER+1])

 {

 bool res=true;

 if(strcmp(bufFile,expectedBufFile)!=0 || strcmp(buf,expectedBuf)!=0)

 {

 res=false;

 }

 return res;

 }

 //буфер полон

 void bufferIsFull()

 {

 //arrange

 FileData output;

 strcpy(output.buf,"");

 for(int i=0; i<2499; i++)

 strcat(output.buf,"1234");

 char group[MAX\_SIZE\_GROUP+1]={"1234"};

 int K=4;

 bool haveGroup=true;

 char bufFile[MAX\_SIZE\_BUFFER+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={""};

 char expectedBufFile[MAX\_SIZE\_BUFFER+1]={""};

 for(int i=0; i<2500; i++)

 strcat(expectedBufFile,"1234");

 //act

 char filename[100]={"Test\_writeBytes.txt"};

 output.handleFile=\_open(filename,\_O\_TRUNC | \_O\_BINARY | \_O\_RDWR, \_S\_IREAD | \_S\_IWRITE);

 writeBytes(&output,group,K,haveGroup);

 lseek(output.handleFile,0,SEEK\_SET);

 int realSize=read(output.handleFile,(void\*)bufFile, sizeof(char)\*((10000/K)\*K));

 bufFile[realSize]='\0';

 close(output.handleFile);

 //compare

 bool result=compare(bufFile,expectedBufFile,output.buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //буфер не полон, но данная группа была последней

 void bufferIsNotFullButGroupLast()

 {

 //arrange

 FileData output;

 strcpy(output.buf,"");

 for(int i=0; i<99; i++)

 strcat(output.buf,"1234");

 char group[MAX\_SIZE\_GROUP+1]={"1234"};

 int K=4;

 bool haveGroup=false;

 char bufFile[MAX\_SIZE\_BUFFER+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={""};

 char expectedBufFile[MAX\_SIZE\_BUFFER+1]={""};

 for(int i=0; i<100; i++)

 strcat(expectedBufFile,"1234");

 //act

 char filename[100]={"Test\_writeBytes.txt"};

 output.handleFile=\_open(filename,\_O\_TRUNC | \_O\_BINARY | \_O\_RDWR, \_S\_IREAD | \_S\_IWRITE);

 writeBytes(&output,group,K,haveGroup);

 lseek(output.handleFile,0,SEEK\_SET);

 int realSize=read(output.handleFile,(void\*)bufFile, sizeof(char)\*((10000/K)\*K));

 bufFile[realSize]='\0';

 close(output.handleFile);

 //compare

 bool result=compare(bufFile,expectedBufFile,output.buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //буфер полон и группа была последней

 void bufferIsFullAndGroupLast()

 {

 //arrange

 FileData output;

 strcpy(output.buf,"");

 for(int i=0; i<2499; i++)

 strcat(output.buf,"1234");

 char group[MAX\_SIZE\_GROUP+1]={"1234"};

 int K=4;

 bool haveGroup=false;

 char bufFile[MAX\_SIZE\_BUFFER+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={""};

 char expectedBufFile[MAX\_SIZE\_BUFFER+1]={""};

 for(int i=0; i<2500; i++)

 strcat(expectedBufFile,"1234");

 //act

 char filename[100]={"Test\_writeBytes.txt"};

 output.handleFile=\_open(filename,\_O\_TRUNC | \_O\_BINARY | \_O\_RDWR, \_S\_IREAD | \_S\_IWRITE);

 writeBytes(&output,group,K,haveGroup);

 lseek(output.handleFile,0,SEEK\_SET);

 int realSize=read(output.handleFile,(void\*)bufFile, sizeof(char)\*((10000/K)\*K));

 bufFile[realSize]='\0';

 close(output.handleFile);

 //compare

 bool result=compare(bufFile,expectedBufFile,output.buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

 //буфер был пуст, но группа была последней

 void bufferIsEnptyAndGroupLast()

 {

 //arrange

 FileData output;

 strcpy(output.buf,"");

 char group[MAX\_SIZE\_GROUP+1]={"1234"};

 int K=4;

 bool haveGroup=false;

 char bufFile[MAX\_SIZE\_BUFFER+1]={""};

 char expectedBuf[MAX\_SIZE\_BUFFER+1]={""};

 char expectedBufFile[MAX\_SIZE\_BUFFER+1]={"1234"};

 //act

 char filename[100]={"Test\_writeBytes.txt"};

 output.handleFile=\_open(filename,\_O\_TRUNC | \_O\_BINARY | \_O\_RDWR, \_S\_IREAD | \_S\_IWRITE);

 writeBytes(&output,group,K,haveGroup);

 lseek(output.handleFile,0,SEEK\_SET);

 int realSize=read(output.handleFile,(void\*)bufFile, sizeof(char)\*((10000/K)\*K));

 bufFile[realSize]='\0';

 close(output.handleFile);

 //compare

 bool result=compare(bufFile,expectedBufFile,output.buf,expectedBuf);

 CFIXCC\_ASSERT\_EQUALS(true,result);

 }

# Приложение Д. Разделение обязанностей между членами команды

Таблица Д.1. Реализация подпрограмм членами команд

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Подпрограмма | Заголовок | Модульные тесты | Псевдокод | Код | Тестирование и отладка |
| enDeCryptTextGroup | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. |
| nextSet | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. |
| swap | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. |
| logInfo | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. |
| getNameFileFromDirect | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. |
| checkComandLine | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. |
| getDataFromComandLine | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. | Островский Д.С. |
| readBytes | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. |
| takeFirstGroupFromBuffer | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. |
| addSpacesToBuffer | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. |
| writeBytes | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. | Бирюков Н.А. |

# Приложение Е. Порядок конструирования программы



Рисунок Е.1 – Порядок реализации подпрограмм членами команд

# Приложение Ж. Код программы

void enDeCryptTextGroup(char textGroup[MAX\_SIZE\_GROUP+1], int permutation[MAX\_SIZE\_GROUP], int K, bool encrypt)

{

 char buf[MAX\_SIZE\_GROUP+1];

 //Скопировать исходный текст группы в буферную строку

 strcpy(buf,textGroup);

 //если строка не пустая

 if(strlen(textGroup)>0)

 {

 //Для каждого символа в группе...

 for(int i=0; i<K; i++)

 {

 //Если шифруем

 if(encrypt==true)

 //Заменить текущий символ в группе на символ, чей индекс в буфере это число в перестановке минус 1.

 textGroup[i]=buf[permutation[i]-1];

 //если дешифруем

 else if(encrypt==false)

 //Заменить символ, чей индекс это номер перестановки минус 1, на текущий символ в буфере

 textGroup[permutation[i]-1]=buf[i];

 }

 }

}

void nextSet(int Permutation[MAX\_SIZE\_GROUP], int K)

{

 int compareElement = K - 1;

 int current = K - 2;

 int l,r;

 //Если массив не был заполнен до этого

 if(Permutation[0]<1 || Permutation[0]>MAX\_SIZE\_GROUP)

 //Для каждого символа от начало до K

 for(int indexChar=0; indexChar<K;indexChar++)

 //Заполнить символ перестановки следующей цифрой

 Permutation[indexChar]=indexChar+1;

 //Пока не перебраны все элементы(начиная с конца) и элемент не меньше следующего

 while (current!= -1 && Permutation[current] >= Permutation[current + 1])

 //Перейти к элементу спереди

 current--;

 //Если все элементы перебраны

 if(current==-1)

 {

 //Записать перестановку в начальном виде

 for(int indexChar=0; indexChar<K; indexChar++)

 {

 Permutation[indexChar]=indexChar+1;

 }

 }

 //Иначе

 else

 {

 //Найти элемент, который меньше текущего (начиная с конца)

 while (Permutation[current] >= Permutation[compareElement])

 compareElement--;

 //Поменять найденный элемент с текущим

 swap(Permutation, current, compareElement);

 //Отсортировать остальную часть последовательности

 l = current + 1, r = K - 1;

 while (l<r)

 swap(Permutation, l++, r--);

 }

}

void logInfo(int handle, TypeMessage type, char\* message)

{

 char summary[200]={""};

 char currentTime[10]={""};

 //Узнать текущее время и добавить его в конец итогового сообщения

 \_strtime\_s(currentTime,9);

 strcpy(summary,currentTime);

 //Если тип сообщения-ошибка...

 if(type==error)

 //Добавить в итоговое сообщение тип "[error]"

 strcat(summary," [error] ");

 //Иначе если тип сообщения-информация...

 else if(type==info)

 //Добавить в итоговое сообщение тип "[info]"

 strcat(summary," [info] ");

 //Добавить в конец итогового сообщения само сообщение и символ перевода строки

 char buf[1000];

 strcpy(buf,message);

 strcat(buf,"\n");

 strcat(summary,buf);

 //Записать итогового сообщение в файл

 write(handle,(void\*)summary,strlen(summary));

}

void getNameFileFromDirect(char\* direct, char\* nameFile)

{

 int indexChar;

 char \*buf=new char[strlen(direct)+1];

 //Копировать директорию в буфер

 strcpy(buf,direct);

 //Начиная с конца директории для каждого символа и пока не встречен '\','/' или конец

 for(indexChar=strlen(buf); indexChar!=-1 && buf[indexChar]!='\\' && buf[indexChar]!='/';)

 //Искать первый символ имени файла

 indexChar--;

 //Копировать имя файла с найденного символа

 strcpy(nameFile,buf+indexChar+1);

 delete [] buf;

}

void checkComandLine(int countArguments, char\* arguments[]) throw (char\*)

{

 char nameFileIn[100];

 char nameFileOut[100];

 //Если количество аргументов меньше 5

 if(countArguments<5)

 //Выкинуть ошибку о недостатке аргументов

 throw "Ошибка выполнения программы! Неверное количество аргументов командной строки.";

 //Если аргумент с указанием количества символов в группе не входит в заданный ему диапазон

 if(atoi(arguments[3])<2 || atoi(arguments[3])>MAX\_SIZE\_GROUP)

 //Выкинуть ошибку о неправильном диапазоне количества символов в группе

 throw "Ошибка выполнения программы! Длина группы символов K не входит в диапазон от 2 до 50";

 //Выделить имя файла из первого пути

 getNameFileFromDirect(arguments[2],nameFileIn);

 //Выделить имя файла из второго пути

 getNameFileFromDirect(arguments[4],nameFileOut);

 //Если имена файлов одинаковые

 if(!strcmp(nameFileIn,nameFileOut))

 //Вывести ошибку о одинаковы х именах файлов

 throw "Ошибка выполнения программы! Имя выходного файла совпадает с именем входного файла";

 //Если аргументы с указанием входного и выходного файла не имеют символа точки

 if(strchr(arguments[2],'.')==NULL || strchr(arguments[4],'.')==NULL)

 //Выкинуть ошибку о том, что файл не указан

 throw "Ошибка выполнения программы! Не указан сам файл";

}

bool getDataFromComandLine(char\* arguments[], int &sizeGroup, char\* fileNameIn, char\* fileNameOut)

{

 bool key;

 //Записать размер группы символов в переменную

 sizeGroup=atoi(arguments[3]);

 //Если в строке с указанием входного файла есть '/'

 if(strchr(arguments[2],'/')!=NULL)

 {

 getcwd(fileNameIn,100);

 strcat(fileNameIn,"\\");

 strcat(fileNameIn,arguments[2]);

 //Перевести '/' в '\'

 int i=0;

 while(fileNameIn[i]!='.')

 {

 if(fileNameIn[i]=='/')

 {

 fileNameIn[i]='\\';

 }

 i++;

 }

 }

 else

 {

 strcpy(fileNameIn,arguments[2]);

 }

 //Если в строке с указанием выходного файла есть '/'

 if(strchr(arguments[4],'/')!=NULL)

 {

 getcwd(fileNameOut,100);

 strcat(fileNameOut,"\\");

 strcat(fileNameOut,arguments[2]);

 //Перевести '/' в '\'

 int i=0;

 while(fileNameOut[i]!='.')

 {

 if(fileNameOut[i]=='/')

 {

 fileNameOut[i]='\\';

 }

 i++;

 }

 }

 else

 {

 strcpy(fileNameOut,arguments[4]);

 }

 //Если ключ равен "-e"

 if(!strcmp(arguments[1],"-e"))

 //Вернуть true

 key=true;

 //Иначе

 else

 //Вернуть false

 key=false;

 return key;

}

bool readBytes(FileData \*input, char group[MAX\_SIZE\_GROUP+1], int K)

{

 strcpy(group,"");

 //Установить что очередной группы нет

 bool result=false;//результат

 int lengthBuf=strlen((\*input).buf);//длина буфера

 int realRead=0;//реально прочитанная длина из файла

 //Если буфер пуст...

 if(lengthBuf==0)

 {

 //Прочитать из файла кратное число байт( (10000/K)\*K ) для записи в буфер

 realRead=read((\*input).handleFile,(void\*)(\*input).buf,sizeof(char)\*((MAX\_SIZE\_BUFFER/K)\*K));

 (\*input).buf[realRead]='\0';

 lengthBuf=strlen((\*input).buf);

 //Если в буфере не кратное число байт...

 if(lengthBuf%K!=0)

 //Дополнить буфер до кратности пробелами

 addSpacesToBuffer((\*input).buf,K);

 }

 //Если буфер не пуст...

 if(lengthBuf>0)

 {

 //Изъять первые K символов из буфера и поместить их в выходную группу

 takeFirstGroupFromBuffer((\*input).buf,group,K);

 //Установить что имеется очередная группа

 result=true;

 }

 //Вернуть результат

 return result;

}

void takeFirstGroupFromBuffer(char buf[MAX\_SIZE\_BUFFER+1], char group[MAX\_SIZE\_GROUP+1], int K)

{

 //Определить длину буфера

 int length=strlen(buf);

 //Если длина равна K...

 if(length==K)

 {

 //Скопировать группу из буфера в массив с группой, и поставить 0 символ в буфере

 strcpy(group,buf);

 buf[0]='\0';

 }

 //Иначе если длина больше 0...

 else if(length>0)

 {

 //Скопировать в массив с группой первые K символов и поставить 0 символ в конце

 strncpy(group,buf,K);

 group[K]='\0';

 //Для каждого символа начиная с K-го...

 for(int i=K; i<length; i++)

 //Сдвинуть текущий символ на K позиций влево

 buf[i-K]=buf[i];

 //Поставить 0 символ в группе и в буфере

 buf[length-K]='\0';

 }

}

void addSpacesToBuffer(char buf[MAX\_SIZE\_BUFFER+1], int K)

{

 char space[2]={" "};

 int length=strlen(buf);

 //Если буфер не пустой...

 if(length>0)

 //Дополнить необходимым количеством пробелов буфер

 for(int i=0; i<K-(length%K); i++)

 strcat(buf,space);

}

void writeBytes(FileData \*output, char group[MAX\_SIZE\_GROUP+1], int K, bool haveGroup)

{

 //Добавить данную группу в конец буфера

 strcat((\*output).buf,group);

 //Если буфер полон или данная группа была последней...

 if(strlen((\*output).buf)==((MAX\_SIZE\_BUFFER/K)\*K) || haveGroup==false)

 {

 //Записать в файл весь буфер

 int realSize=write((\*output).handleFile,(void\*)(\*output).buf,sizeof(char)\*strlen((\*output).buf));

 //Очистить буфер

 strcpy((\*output).buf,"");

 }

}

# Приложение З. Оценка полученных результатов

Таблица З.1. Оценка Бирюковым Н. полученных результатов

|  |  |  |
| --- | --- | --- |
| Критерий оценки | Достоинства | Недостатки |
| Работоспособность + функциональность программы | Программа выполняет все функции, даже на больших файлах все прекрасно работает | Плохо работает на картинках |
| Внешняя спецификация программы |  | 80% спецификации делал я |
| Декомпозиция программы на подпрограммы | Все реализовалось как мы и думали изначально | После подведения итогов практики были придуманы несколько простые решения некоторых функций, чем мы сделали |
| Декомпозиция программы на модули | Все отлично | Нет |
| Псевдокод | Все хорошо | Возможно, не все описано корректно, особенно в алгоритме шифрации/дешифрации |
| Код (эффективность структур данных, реализация типовых алгоритмических структур, их комбинация) | Все хорошо | Нет |
| Форматирование кода | Пришли к единому стилю | Были разногласия в каком стиле писать код, т.к. у каждого участника выработался свой стиль и он его придерживается |
| Модульные тесты | Покрыты все ситуации | В функциях по работе с буфером не везде тестировали переполнение и новое считывание из файла, но как показало тестирование всей программы, функция работает верно |
| Ведение репозитория кода | Каждый участник придерживался атомарности коммитов | Порой находили опечатки, что засоряло репозиторий. У каждого участника команды свой стиль записи коммита. Нет единого стиля |
| Порядок конструирования программы + распределение подпрограмм между членами команды | Все хорошо, споров не возникало | Нет |

Таблица З.2. Оценка Островским Д. полученных результатов

|  |  |  |
| --- | --- | --- |
| Критерий оценки | Достоинства | Недостатки |
| Работоспособность + функциональность программы | Программа выполняет все функции | Анимации и картинки не обрабатываются |
| Внешняя спецификация программы |  | Я сделал только небольшую часть |
| Декомпозиция программы на подпрограммы | Шифрование реализовалось по тому алгоритму, что я предлагал изначально. Не пришлось менять впоследствии. | Многие функции в процессе создания всей программы оказались не нужны. Тратилось много времени.  |
| Декомпозиция программы на модули | Хорошо | Нет |
| Псевдокод | Хорошо | Некоторые действия описаны в общем виде |
| Код (эффективность структур данных, реализация типовых алгоритмических структур, их комбинация) | Хорошо | Нет |
| Форматирование кода | Единый стиль | Пришлось писать комментарии и код в стиле напарника |
| Модульные тесты | Покрыто большинство ситуаций | Множество функций не нуждаются, по моему мнению, в тестах. Трата времени. |
| Ведение репозитория кода | Коммиты атомарны и понятны | Нет |
| Порядок конструирования программы + распределение подпрограмм между членами команды | Хорошо | Нет |