Пояснительная записка (Д/З №2)

Глазков Максим БПИ208 - Вариант 289 (9, 21)

1. Описание задания

Обобщенный артефакт, используемый в задании	Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив)	Общие для всех альтернатив переменные	Общие для всех альтернатив функции
Тексты, состоящие из цифр и латинских букв, зашифрованые различными способами	1. Шифрование заменой символов (указатель на массив пар: [текущий символ, замещающий символ]; зашифрованный текст – строка символов) 2. Шифрование циклическим сдвигом кода каждого символа на n (целое число, определяющее сдвиг; зашифрованный текст – строка символов) 3. Шифрование заменой символов на числа (пары: текущий символ, целое число – подстановка при шифровании кода символа в виде короткого целого; зашифрованный текст – целочисленный массив)	Открытый текст – строка символов	Частное от деления суммы кодов незашифрован ной строки на число символов в этой строке (действительн ое число)

2. Описание структуры ВС

Типы данных	
Тип	**Размер**
Int	4
Double	8
Char	1
Bool	1
FILE	4

Класс Random (8 байт)	
Поле / Метод	**Размер**
Глобальных методов / полей нет	
Локальные методы / поля:	
Int first	4
Int last	4
Get(): Нет локальных объявлений	**0**
Класс Crypter (2124 байт)	
Поле / Метод	**Размер**
Глобальные методы / поля:	
Static const int max_len = 2000	4
Static const int possible_symb = 62	4
Локальные методы / поля:	

Const char symb[possible_symb]	62
Const char crypt[possible_symb]	62
Char text[max_len]	2000
Crypter(char* text): Int i	**4** 4[0]
Crypter(): Int interval_val Int i	**8** 4[0] 4[4]
FileWrite(FILE *fp): Char code[15]	**15** 15[0]
PairCrypt(): Char new_mess[max_len] Int i Int j	**2008** 2000[0] 4[2000] 4[2004]
ShiftCrypt(): Char new_mess[max_len] Int shift Int i	**2008** 2000[0] 4[2000] 4[2004]
NumericCrypt(): Char new_mess[max_len * 3] Int new_index Int code Int i Int num	**6016** 6000[0] 4[6000] 4[6004] 4[6008] 4[6012]
GetTextCode(): Double sum Int i	**12** 8[0] 4[8]

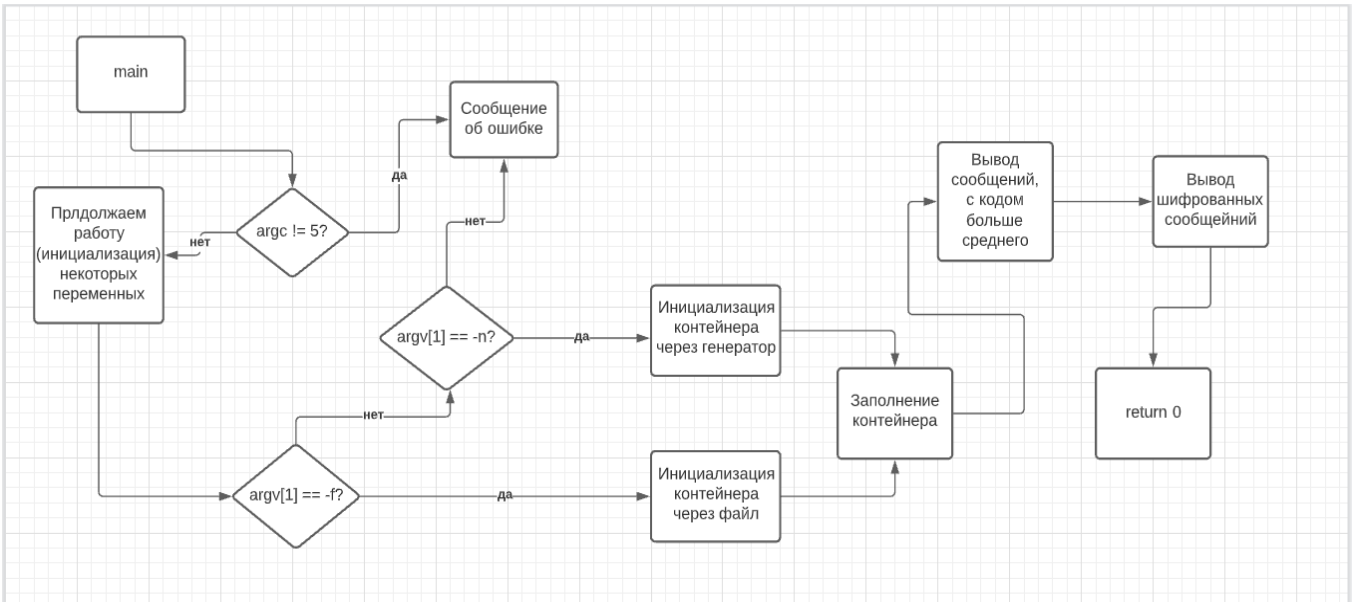
Класс Container (4248000 байт) ~ 4 Мб	
Поле / Метод	**Размер**
Глобальные методы / поля:	
Static const int max_len	4
Локальные методы / поля:	
Crypter messages[max_len]	4248000
Int curr_index	4
Container(FILE *fp): Int symbol Int symbol_index Char text[max_len]	**2008** 4[0] 4[4] 2000[8]
Container(FILE *fp): Int i	**4** 4[0]
FileWrite(FILE *fp): Int i	**4** 4[0]
AverageCode(): Double sum Int i	**12** 8[0] 4[8]
AverageRemove(): Crypter new_mess[max_len] Int new_index Int i	**4248008** 4248000[0] 4[4248000] 4[4248004]
Clear(): Нет локальных объявлений	**0**

Глобальные объявления вне классов	
Объявление	**Размер**

Random space	8
Random letter	8
Random interval	8

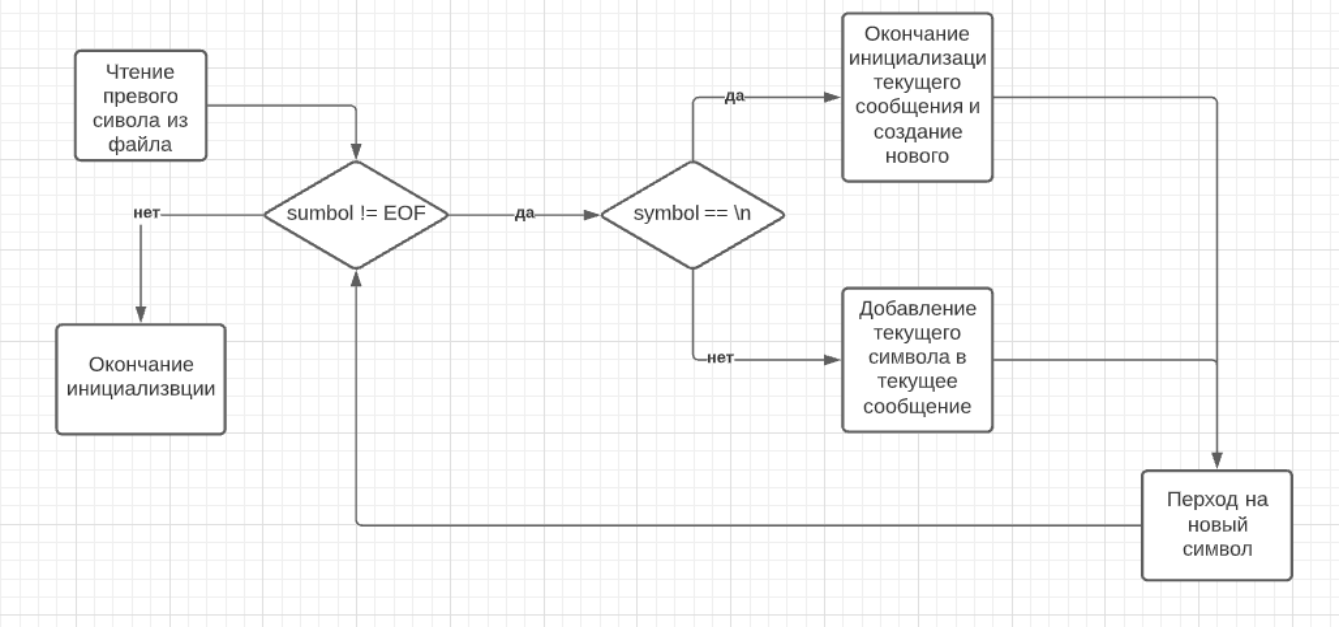
Функция main() (4248021 байт)	
Переменная	**Размер**
Container cont	4248000
Int size	4[4248000]
Bool isFile	1[4248004]
FILE out	4[4248005]
FILE aver	4[4248009]
Double search_time	8[4248013]

Отработка метода main

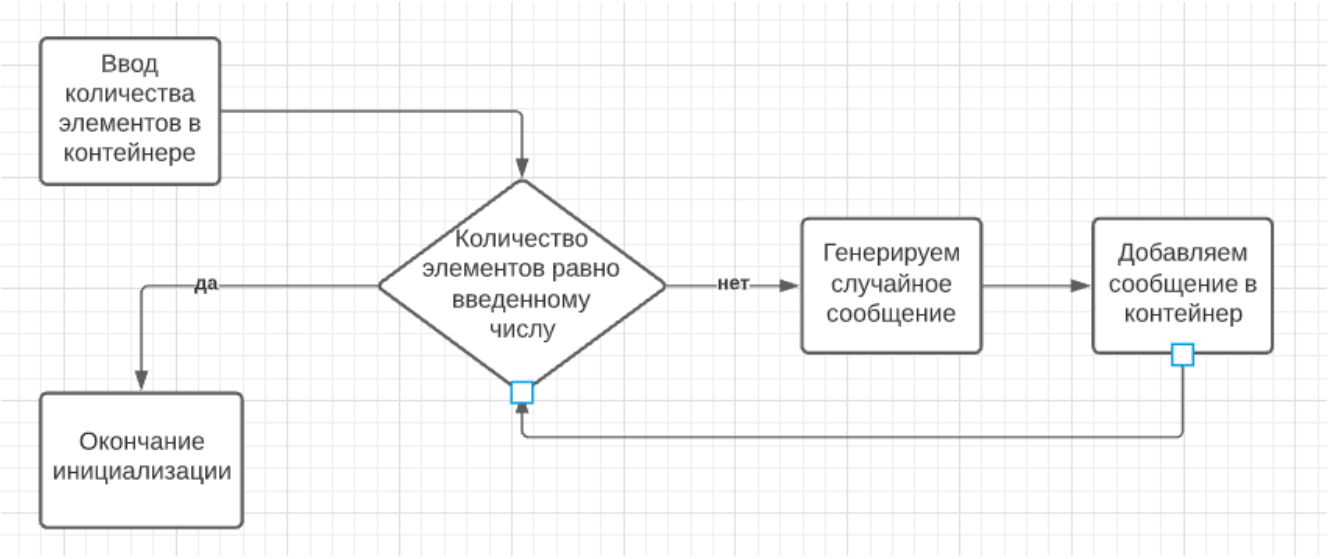


Логика вывода в файл для каждого сообщения в контейнере

Логика при инициализации контейнера через файл



Логика при инициализации контейнера случайными сообщениями



3. Входные и выходные данные

- В консоль поступает консольная команда с указанием на тестовый файл: <Программа.exe> <-f> <Тестовый_файл.txt> <Файл вывода_1.txt> <Файл вывода_2.txt>
- В консоль поступает команда без указания на тестовый файл: <Программа.exe> <-n> <Число> <Файл вывода_1.txt> <Файл вывода_2.txt>
- На вход подается только файл в формате .txt. Пример входных данных:

  Thanks for playing
  Zero function
  I love C

В файле должны быть записаны строки, в которых будут только заглавные и прописные буквы латинского алфавита или цифры от 0 до 9, сообщения должны быть разделены знаком переноса на новую строку (“\n”), иначе весь текст будет восприниматься программой как одно единое сообщение. В случае, если в сообщении присутствуют символы, не соответствующие требованиям, описанным выше, ввод будет являться некорректным

- После обработки данных программа записывает все результаты в введенный файл в следующем виде:

==========

Message text: Thanks for playing
Message code: 98
Pair crypt: jbrfCQ Wiq GYrd9fw
Shift crypt: Vjcpmu"hqt"rnc{kpi
Numeric crypt: 84149711171153212111114321121897121151113

==========

Где сначала идет текст сообщения, код сообщения, как целое число (коды сообщений могут совпадать), и три разных способа зашифровки сообщения. Во втором выходном фале записываются только те сообщения, код которых не меньше среднего кода всех сообщений в контейнере. Отображение данных о сообщениях точно такое же как и в приведенном примере. В консоль не выводятся результаты работы программы, только сообщения об ошибках или об успешном завершении работы программы

4. Дополнительные сведения

- Время работы программы будет выводится в конце работы программы в случае ее успешного завершения
- Результат работы тестов можно увидеть в подкаталоге “outs” (тесту с названием “test0<i>.txt” соответствуют выводы в файлах “out0<i>.txt” и “average_out0<i>.txt”)
- Результаты работы программы на случайно сгенерированных сообщениях можно увидеть в в файлах “rnd_out.txt” и “aver_rnd_out.txt”

5. Сравнение с процедурным подходом

- Архитектура программы, написанной при ООП подходе сложнее, чем при процедурном. При написании ООП программы пришлось нарисовать схему и сделать некоторые наброски, связанные с целостной композицией программы (из каких элементов она состоит и как они друг с другом связаны). При процедурном подходе все намного проще, можно писать программу без продуманного начального плана
- При процедурном подходе программа отрабатывает немного быстрее, чем ООП программа.
- Процедурный подход больше подходит для написания небольших задач (например, написать несколько математических функций), так как он просто быстрее, а ООП стиль для больших программ, потому что большую

процедурную программу намного сложнее читать, чем такую же, но с использованием ООП