Домашнее задание №3 "Архитектура ВС с динамической типизацией"

Задача: Тексты, состоящие из цифр и латинских букв, зашифрованные различными способами, Вариант 135 (9, 10)

Условия задачи

Базовые альтернативы

- Шифрование заменой символов (указатель на массив пар: [текущий символ, замещающий символ]; зашифрованный текст – строка символов)
- Шифрование циклическим сдвигом кода каждого символа на n (целое число, определяющее сдвиг; зашифрованный текст – строка символов)
- Шифрование заменой символов на числа (пары: текущий символ, целое число – подстановка при шифровании кода символа в виде короткого целого; зашифрованный текст – целочисленный массив)

Общие для всех альтернатив переменные

- Открытый текст – строка символов

Общие для всех альтернатив функции

- Частное от деления суммы кодов незашифрованной строки на число символов в этой строке (действительное число)

Функционал

После размещения данных в контеи□нер необходимо осуществить их обработку в соответствии с вариантом задания. Обработанные данные после этого заносятся в отдельный□ фаи□л результатов. Упорядочить элементы контейнера по возрастанию используя сортировку с помощью прямого включения (Straight Insertion). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

Реализация

Программа написана на Python 3 в объектно-ориентированном стиле.
Точка входа: main.py
Запуск программы:
1. Ввод данных из файла `python main.py input <input file> <output file>`
2. Для рандомного заполенения `python main.py rnd <count elements> <output file>` . Основная реализация расположена в директории "src"
Тесты находятся в директории tests. Входные данные в поддиректории "input", а выходные данные в "output". В случае некорректных данных, поступающих в программу, в выходной файл поступает сообщение об ошибке. Также есть файл time.txt, в котором записано время тестов программы.

Тесты

В первой строке задается размер контейнера. Следующие строки описывают элементы контейнера: в начале каждой строки задан тип шифра, через пробел текст в двойных кавычках, параметр шифрования(определяет, например, количество сдвигов),далее идут пары вида char-char(заменяемый-заменяющий).
Пример ввода:

```
6
3 "" 3 a-1 b-2 c-3
3 "2345" 3 2-4 4-9 3-5
3 "10" 2 1-0 0-1
1 "ACS+HUmexUrk!" 3 +-  U-o x-w
2 "" 1
3 "12345" 5 1-10 2-20 3-30 4-40 5-50
```

Пример вывода:

```
 There are  6 elements in container
1. "", Hash=0; Count of replacements:  3 ; Symbols:  a <-> 1 b <-> 2 c <-> 3  ; Result:  [  ]
2. "2345", Hash=52; Count of replacements:  3 ; Symbols:  2 <-> 4 4 <-> 9 3 <-> 5  ; Result:  [ 4 5 9 5 ]
3. "10", Hash=48; Count of replacements:  2 ; Symbols:  1 <-> 0 0 <-> 1  ; Result:  [ 0 1 ]
4. "ACS+HUmexUrk!", Hash=83; Count of replacements:  3 ; Symbols:  + <->   U <-> o x <-> w  ; Result:  ACS Homework!
5. "", Hash=0; ReplaceShift: 1; Result: ""
6. "12345", Hash=51; Count of replacements:  5 ; Symbols:  1 <-> 10 2 <-> 20 3 <-> 30 4 <-> 40 5 <-> 50  ; Result:  [ 10 20 30 40 50

-----Straight Insertion-----

There are  6 elements in container
1. "", Hash=0; Count of replacements:  3 ; Symbols:  a <-> 1 b <-> 2 c <-> 3  ; Result:  [  ]
2. "", Hash=0; ReplaceShift: 1; Result: ""
3. "10", Hash=48; Count of replacements:  2 ; Symbols:  1 <-> 0 0 <-> 1  ; Result:  [ 0 1 ]
4. "12345", Hash=51; Count of replacements:  5 ; Symbols:  1 <-> 10 2 <-> 20 3 <-> 30 4 <-> 40 5 <-> 50  ; Result:  [ 10 20 30 40 50
5. "2345", Hash=52; Count of replacements:  3 ; Symbols:  2 <-> 4 4 <-> 9 3 <-> 5  ; Result:  [ 4 5 9 5 ]
6. "ACS+HUmexUrk!", Hash=83; Count of replacements:  3 ; Symbols:  + <->   U <-> o x <-> w  ; Result:  ACS Homework!
```

Характеристики программы

	HW02-OoStaticLangArch	HW03-DynamicLangArch
Число интерфейсных модулей	5	-
Число модулей реализации	5	8
Общий размер исходных текстов	14 848 байт	13 326 байт
Размер исполняемого файла	33 336 байт	15 273 байт (__pycache__)
Тест	**Время работы теста (сек.)**	
1-9	≈0	0.000664 - 0.000996
10	≈0	0.001181
20 случайных элементов	≈0	0.002091
1000 случайных элементов	0.015625	0.050444
10000 случайных элементов	0.0625	0.516384

По сравнению с предыдущей работой увеличилось число модулей реализации (из-за запрета циклических зависимостей требуется бóльшая модульность), несколько уменьшился размер исходных текстов, исчез исполняемый файл (вместо него интерпретатор генерирует во время исполнения более компактный байт-код), и в разы упала производительность. С одной стороны, Python в какой-то степени упрощает программирование и обеспечивает компактность и кроссплатформенность программ. С другой стороны, Python как интерпретируемый язык с динамической типизацией сильно проигрывает в производительности компилируемым языкам со статической типизацией.

Структура вычислительной системы

Отображение содержимого классов

Таблица классов	Таблица имен	Описания
Container	__init__	`def __init__(self, array: List[Text]) -> None`
		from_file| `def from_file(cls, input_file: TextIO)`
		random_symbols| `def random_symbols(cls, length: int)`
		__len__| `def __len__(self) -> int`

Таблица классов	Таблица имен	Описания
		insertion_sort| `def insertion_sort(self)`
		output| `def output(self, output_file: TextIO) -> None`
Text	Key	`class Key(IntEnum)`
		__init__| `def __init__(self, text: str) -> None`
		__len__| `def __len__(self) -> object`
		from_file| `def from_file(cls, text: str, input_file: TextIO)`
		hash| `def hash(self)`
		random_encryption| `def random_encryption(cls, text: str)`
		encrypted_text| `def encrypted_text(self) -> str`
		output| `def output(self, output_file: TextIO) -> None`
CharEncryption	__init__	`def __init__(self, text: str, replacements: List[Tuple[str, Union[int, str]]], to_int: bool) -> None`
		from_file| `def from_file(cls, text: str, input_file: TextIO) -> Text`
		file_to_int| `def from_file__to_int(cls, text: str, input_file: TextIO, to_int: bool) -> Text`
		random_to_int| `def random_encryption__to_int(cls, text: str, to_int: bool) -> Text`
		check_validation| `def valid_replacements(self) -> bool`
		replacement_string| `def replacement_string(self, index: int) -> str`
		encrypted_text| `def encrypted_text(self) -> str`
		random_encryption| `def random_encryption(cls, text: str) -> Text`
		output| `def output(self, output_file: TextIO) -> None`
ShiftEncryption	__init__	`def __init__(self, text: str, shift: int)`
		from_file| `def from_file(cls, text: str, input_file: TextIO) -> Text`
		encrypted_text| `def encrypted_text(self) -> str`
		random_encryption| `def random_encryption(cls, text: str) -> Text`
		output| `def output(self, output_file: TextIO) -> None`

Отображение на память методов классов

Память программы	Таблица имен	Память данных
main.py	__name__	str
src.main.main	args	Tuple[str]
		file|bool
		input_file|TextIO

Память программы	Таблица имен	Память данных
		count_random|int
		output_file|TextIO
		start|float
		container|Container
		exception|Exception
src.checkErrors.not_space	input_file	TextIO
		following|str
src.checkErrors.read_char	input_file	TextIO
		following|str
src.checkErrors.read_until_space	input_file	TextIO
		content|List[str]
		following|str
src.container.Container.__init__	array	List[Text]
src.container.Container.from_file	input_file	TextIO
		length|int
src.container.Container.random_symbols	length	int
src.container.Container.insertion_sort	key	float
		tmp|str
		i|int
		j|int
src.container.Container.output	output_file	TextIO
		index|int
src.replaceShift.ShiftEncryption.__init__	text	str
		shift|int
src.replaceShift.ShiftEncryption.from_file	text	str
		input_file|TextIO
		shift|int
src.replaceShift.ShiftEncryption.random_encryption	text	str
src.replaceShift.ShiftEncryption.output	output_file	TextIO
src.replaceShift.ShiftEncryption.encrypted_text	begin	int
		end|int
		symbol|str
src.replaceChar.CharEncryption.__init__	text	str

Память программы	Таблица имен	Память данных
		replacements|List[Tuple[str, Union[int, str]]]
		to_int|bool
src.replaceChar.CharEncryption.from_file	text	str
		input_file|TextIO
src.replaceChar.CharEncryption.file_to_int	text	str
		input_file|TextIO
		to_int|bool
		count_replaces|int
		arr|List[Tuple[str, Union[int, str]]]
		from_symbol|str
		to_symbol|Union[int, str]
		space|str
		text_object|CharEncryption
src.replaceChar.CharEncryption.random_encryption	text	str
src.replaceChar.CharEncryption.random_to_int	text	str
		to_int|bool
		rnd|int
		pos|set
		pos1|Union[List[bool], set]
		arr|List[Tuple[str, Union[int, str]]]
		from_symbol|str
		to_symbol|Union[int, str]
src.replaceChar.CharEncryption.replacement_string	index	int
src.replaceChar.CharEncryption.output	output_file	TextIO
		index|int
src.replaceChar.CharEncryption.encrypted_text	pair_map	Dict[str, Union[int, str]]
		couple|Tuple[str, Union[int, str]]
		encrypted|List[Union[int, str]]
		symbol|str
src.replaceChar.CharEncryption.check_validation	pair	Tuple[str, Union[int, str]]
src.text.Text.__init__	text	str
src.text.Text.output	output_file	TextIO
		hash_string|str

Память программы	Таблица имен	Память данных
src.text_factory.input_text	input_file	TextIO
		key|int
		arr|List[str]
		current_symbol|str
		content|str
		text|Text
src.text_factory.random_symbols	content	str
		key|int
		text|Text

Сравнение

	Статическая типизация (HW2)	Динамическая типизация (HW3)
Тесты	**Время работы теста (сек.)**	
2	<0.001 сек	0.0007089 сек
4	<0.001 сек	0.0012162 сек
10	<0.0015 сек	0.0016475 сек
50	0.0015625 сек	0.0131493 сек
100	0.00625 сек	0.0175105 сек

Очевидно, что из-за использования динамической типизации программа работает в разы медленнее. Поэтому для больших проектов лучше использовать языки с статической типизацией. Но также можно сказать, что инамическая типизация концептуально проще, чем статическая, а также приводит к созданию более компактных программ, поскольку она является более гибкой и не требует указания типов. Преимущества статической типизации более заметны для больших и сложных программ.