Домашнее задание №2 "Архитектура вычислительных систем"

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

Условие 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 отдельны□ 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.

Сборка программы

```
cmake -B <build_dir> -DCMAKE_BUILD_TYPE=Release
cmake --build <build_dir> --target main
```

Тестирование

Данные для тестов по каждому типу шифрования находятся в каталоге tests. Файл с результатами тестов также находятся в каталоге tests с пометкой .out

Время работы

Время работы программы на разных входных данных:

Количество	Время	Память
5	<0.001 сек	~2500 KB
100	< 0.01 сек	~2700 KB
1000	[0.015 сек	~3650 KB
10000	[0.3 сек	~4300
100000	~1 сек	~4700

Если мы сравним исполнение процедурного подхода с объектно-ориентированным то, у объектно-ориентированной реализации код потребляет меньше памяти и времени выполнения (хоть и не критично, примерно в 1.2-1.3 раза). Следовательно, объектно-ориентированный подход лучше и выгоднее использовать.

№ Метрики, определяющие характеристики программы:

Метрика	Характеристика

Метрика	Характеристика
Число интерфейсных модулей	6
Число модулей с реализацией	6
Общий размер исходных текстов программы	15.4 KB

Описание структуры вычислительной системы:

Размеры фундаментальных типов

Тип	Размер
double	8 bytes
int	4 bytes
char	1 byte
bool	1 byte

Базовые Классы

Container

Container	80004 bytes
Поля	Static: - Local: int length [4 bytes] EncryptionText *cont[8 * 10000 bytes] (т.к. int 4 байта и указатель 4 байта)

Функции	Static: -
	
	Local:
	void In(FILE *file);
	void InRnd(int size);
	void Out(FILE *file);
	void StraightInsertion();

EncryptionText

EncryptionText	0 byte
Поля	Static: - Local: -

Функции	Static: EncryptionText *StaticIn(FILE *file)
	EncryptionText *StaticInRnd();
	
	Local:
	~EncryptionText();
	void In(FILE *file);
	void InRnd();
	void Out(FILE *file);
	double Quotinent();

Производные классы

ReplaceByShift

ReplaceByShift : EncryptionText	**9 bytes**
Поля	Static: - Local: const char *encrypted[5 bytes]; int shift[4 bytes];

Функции	Static: -
	
	Local:
	~ReplaceByShift();
	void In(FILE *file);
	void InRnd(int size);
	void Out(FILE *file);
	double Quotient();

ReplaceWithDigit

ReplaceWithDigit : EncryptionText	**14 bytes**
Поля	Static: - Local: const char *key[5 bytes]; char *temp[5 bytes]; int digit[4 bytes];

Функции	Static: -
	
	Local:
	~ReplaceWithDigit();
	void In(FILE *file);
	void InRnd(int size);
	void Out(FILE *file);
	double Quotient();

ReplaceWithSymbols

ReplaceWithSymbols : EncryptionText	**10 bytes**
Поля	Static: - Local: const char *key[5 bytes]; char *key[5 bytes];

Функции	Static: -
	
	Local:
	~ReplaceWithSymbols();
	void In(FILE *file);
	void InRnd(int size);
	void Out(FILE *file);
	double Quotient();

Глобальная память

Глобальная память	Базовый класс	Производный класс
Поля	Все статические поля	Все статические поля

Функции	void In(Container *self, FILE *file);
	void InRnd(Container *self, int size);
	void Out(Container *self, FILE *file);
	void StraightInsertion(Container *self);
		~EncryptionText();
		void In(FILE *file);
		void InRnd();
		void Out(FILE *file);

Виртуальные функции

Глобальная память	Базовый класс	Производный класс

virtual	~EncryptionText(EncryptionText *self);
	void In(FILE *file);
	void InRnd(EncryptionText *self);
	void Out(EncryptionText *self, FILE *file);
		void In(EncryptionText *self, FILE *file);
		void InRnd(EncryptionText *self);
		void Out(EncryptionText *self, FILE *file);