МИНОБРНАУКИ РОССИИ
федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»
Институт дополнительного образования
Высшая инженерная школа

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА ПРОТОТИПА АРХИВАТОРА НА ОСНОВЕ АЛГОРИТМА КОДИРОВАНИЯ ХАФФМАНА НА ЯЗЫКЕ C++ С ПРИМЕНЕНИЕМ СТАНДАРТОВ C++11-14-17

по программе профессиональной переподготовки:
«Разработчик прикладного программного обеспечения (Языки C и C++)»

Выполнил:
Чернов Павел Дмитриевич

Руководитель:
ст. преподаватель ВИШ ИДО
Полубенцева Марина Игоревна

Санкт-Петербург 2020

Постановка задачи

Цель: разработка прототипа программы-архиватора на основе алгоритма кодирования Хаффмана

Задачи:

1. **Изучить предметную область;**
2. **Разработать технические требования к программе;**
3. **Спроектировать:**
   – архитектуру программы;
   – структуру создаваемого архива;
   – классы на основе технических требований;
   – графический пользовательский интерфейс.
4. **Реализовать:**
   – спроектированные классы;
   – алгоритм кодирования Хаффмана;
   – функциональность которая не была реализована в найденном аналоге;
   – спроектированный графический пользовательский интерфейс.
5. **Тестирование программы:**
   – разработать план тестирования программы;
   – реализовать тестирование программы.
6. **Провести сравнение с существующими аналогами.**

Предметная область

Алгоритм кодирования Хаффмана

Исходные данные: ABRACADABRA

Таблица вероятностей появления символов					
Символ	A	B	R	C	D
Частота	5	2	2	1	1
Вероятность	0,45	0,18	0,18	0,09	0,09

Кодовое дерево Хаффмана

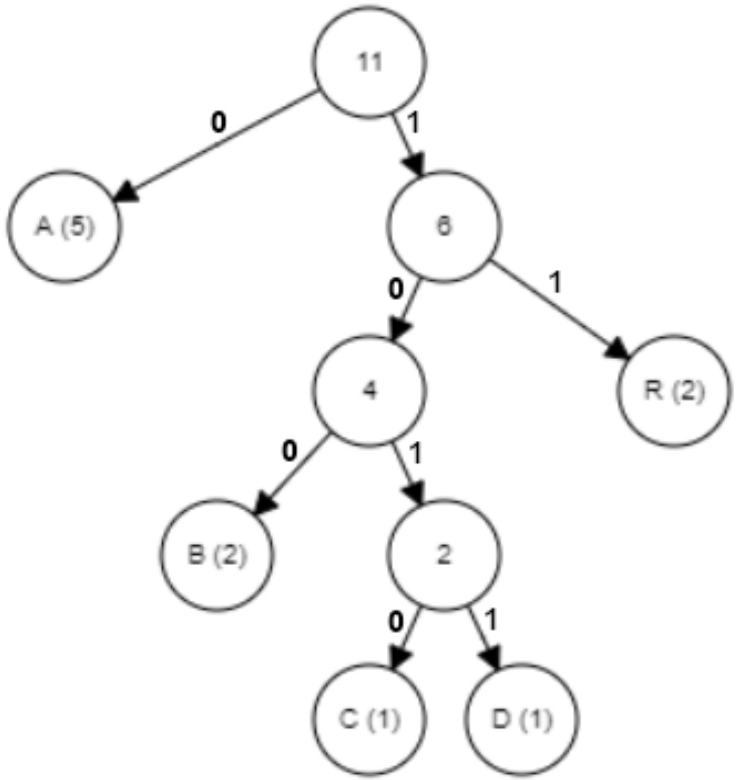


Таблица кодов	
Символ	*Код*
A	0
R	11
B	100
C	1011
D	1010

Закодированные данные:
0 100 11 0 1011 0 1010 0 100 11 0

Технические требования

1) Режимы работы:
- режим архивации данных.
- режим извлечения данных

2) Реализуемые функции:
- добавить путь к исходным данным (файлу и/или папке);
- возможность сжатия нескольких файлов и/или папок;
- предотвращение дублирования путей к исходным данным;
- редактирование (изменение/удаление) пути к исходным данным;
- ввод пути и имени выходного файла (архива);
- редактирование пути и имени выходного файла (архива);
- сжатие исходных данных в однотомный архив, используя для сжатия алгоритм Хаффмана;
- оповещение пользователя о результате выполнения процесса сжатия;
- ввод пути к архиву для распаковки;
- редактирование пути к архиву для распаковки;
- ввод пути к каталогу для распаковки архива;
- редактирование пути к каталогу для распаковки архива;
- распаковка исходных данных из архива;
- оповещение пользователя о результате процесса распаковки.

3) Графический пользовательский интерфейс.

4) Язык реализации - C++ с применением версий стандарта C++11, C++14, C++17.

5) Совместимость с работой под управлением операционной системы Windows 10 (32-bit).

Инструменты

Интегрированные среды разработки:

– *Microsoft Visual Studio 2019* - удобная отладка и профилирование

– *Qt Creator 5.14* - удобное создание оконных приложений; механизм «сигналы-слоты»

Специализированные программы:

– *Qt Linguist 5.14* – создание перевода программ, создаваемых в Qt

– *Qt Designer 5.14* – создание форм для программ, создаваемых в Qt

Информационные ресурсы:

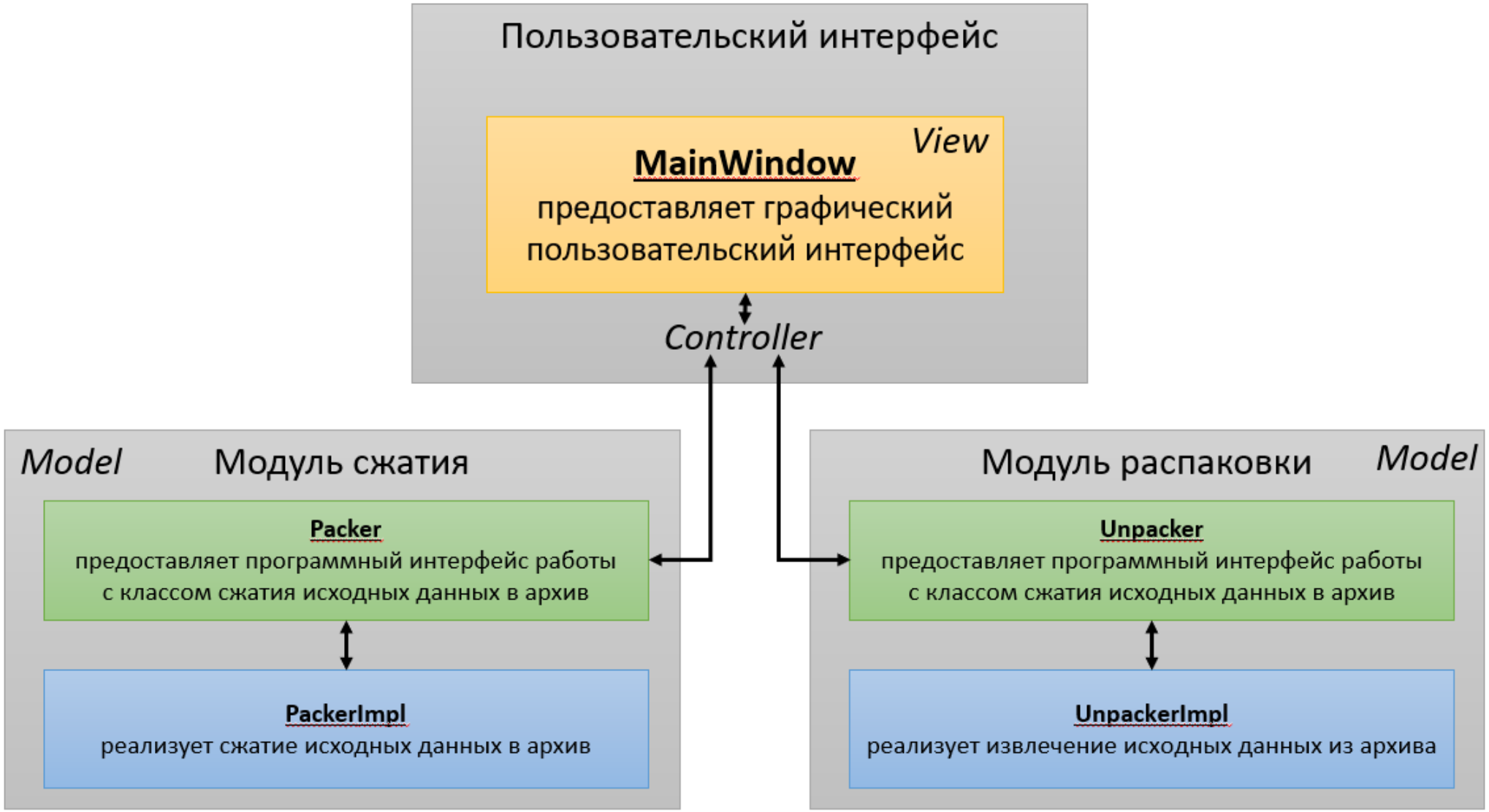
– *Cppreference* – документация по языку C++

– *Qt Assistant* – документация по Qt

– *Stack Overflow* – онлайн сообщество разработчиков ПО

– *Windows Dev Center* – платформа для разработчиков ПО для ОС Windows

Архитектура



class Packer

Псевдонимы		
Псевдоним	*Тип*	*Назначение*
Path	std::string	псевдоним типа пути к файлу/каталогу
FileName	std::string	псевдоним типа имени файла

Реализует программный интерфейс работы с классом PackerImpl

Методы	
Сигнатура	*Назначение*
public:	
Packer()	конструктор
void AddInputPath(const Path& path)	вызвать метод добавления пути к исходным данным класса PackerImpl
void ChangeInputPath(const Path& oldPath, const Path& newPath)	вызвать метод редактирования пути к исходным данным класса PackerImpl
void RemoveInputPath(const Path& path)	вызвать метод удаления пути к исходным данным класса PackerImpl
void SetOutputPath(const Path& path)	вызвать метод установки пути к каталогу размещения выходного файла класса PackerImpl
void ChangeOutputPath(const Path& path)	вызвать метод установки пути к каталогу размещения выходного файла класса PackerImpl
void SetOutputFileName(const FileName& fileName)	вызвать метод установки имени выходного файла класса PackerImpl
void ChangeOutputFileName(const FileName& fileName)	вызвать метод установки имени выходного файла класса PackerImpl
void Pack()	вызвать метод сжатия исходных данных в архив класса PackerImpl

class PackerImpl

Псевдонимы		
Псевдоним	*Тип*	*Назначение*
Data	char	псевдоним типа кодируемых данных

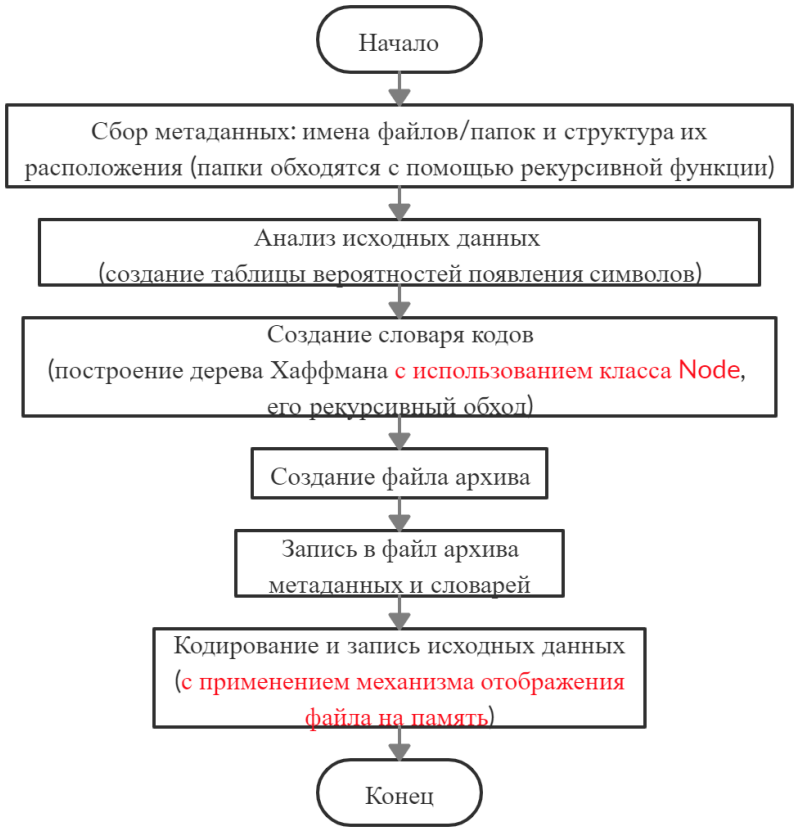
Поля вложенного вспомогательного класса Node		
Тип	*Идентификатор*	*Назначение*
private:		
Node*	_rightBranch	указатель на правый дочерний Node
Node*	_leftBranch	указатель на левый дочерний Node
double	_frequency	вероятность появления символа
Data	_data	кодируемый символ
bool	_containsData	признак Node, содержащего кодируемый символ

Методы вложенного вспомогательного класса Node	
Сигнатура	*Назначение*
public:	
Node(Node* right, Node* left, double frequency, Data data, bool flag)	конструктор
Node* GetRightBranch() const	получить указатель на правый дочерний Node
Node* GetLeftBranch() const	получить указатель на левый дочерний Node
double GetFrequency() const	получить значение вероятности появления символа
Data GetData() const	получить кодируемый символ
bool ContainsData() const	проверить содержет ли Node кодируемый символ

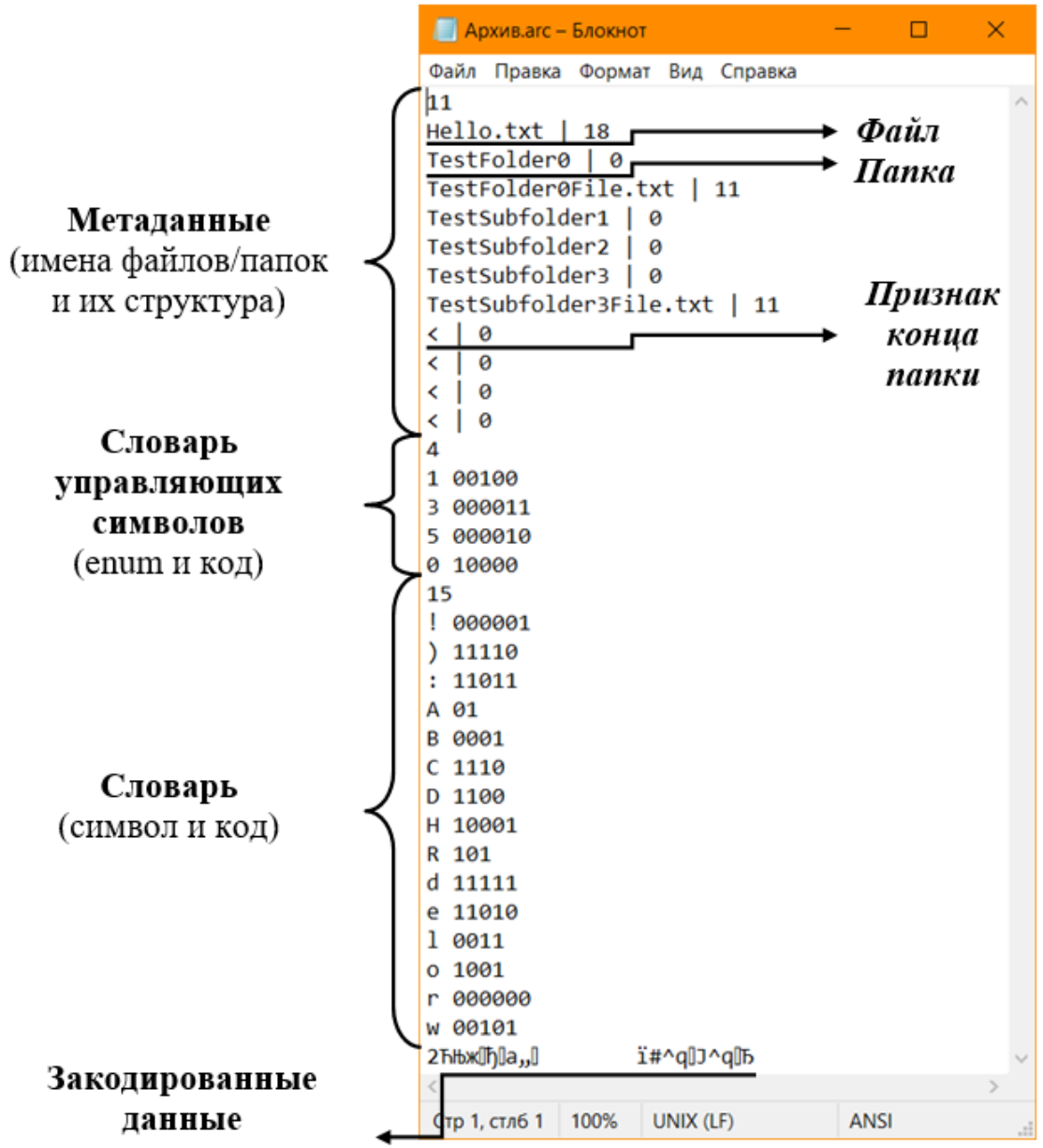
Обращение к классу возможно только через класс-интерфейс Packer.

Содержит вспомогательный класс Node, используемый при построении дерева кодирования Хаффмана.

Сжатие данных в архив



Структура архива



class Unpacker

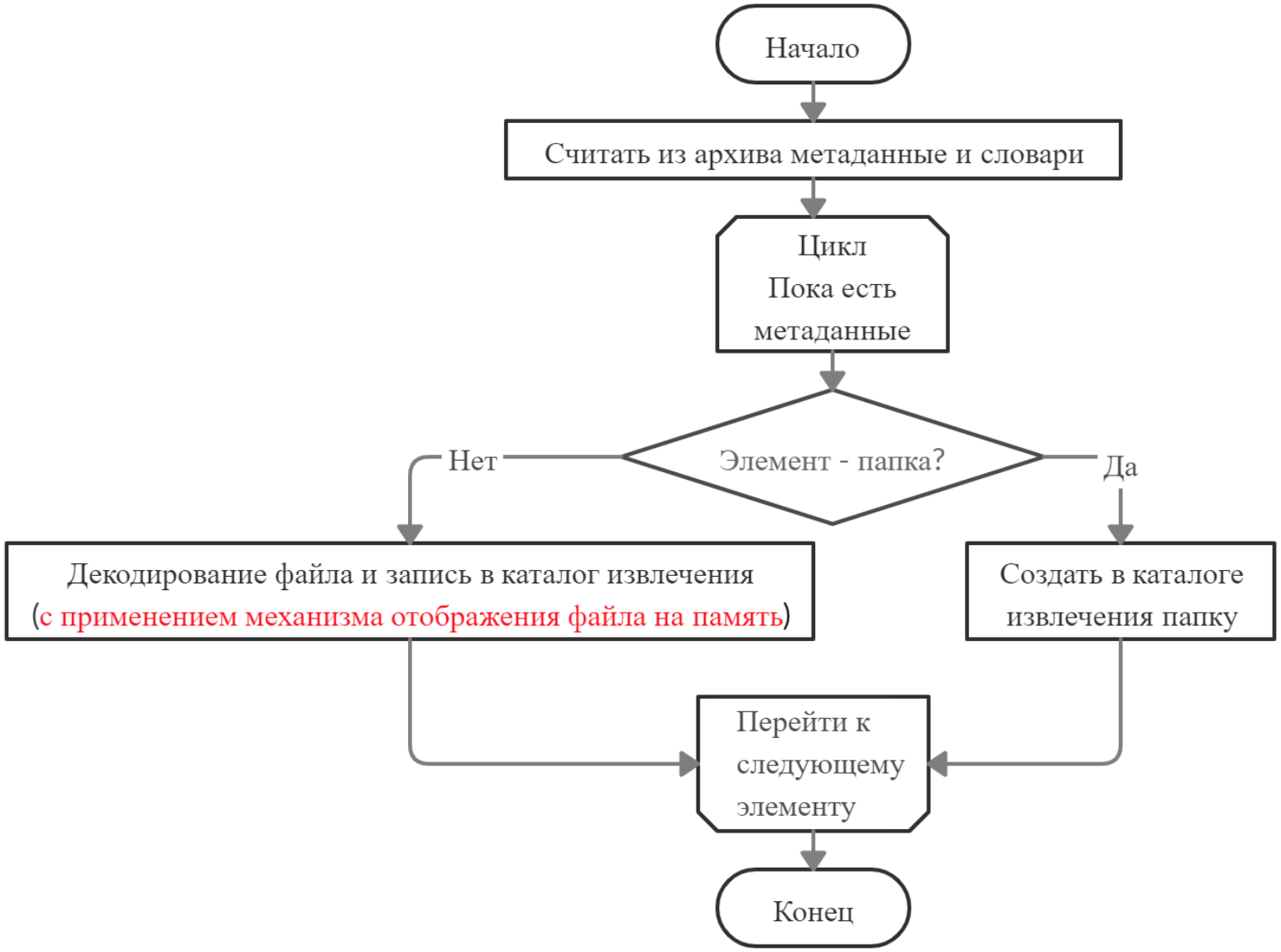
Псевдонимы		
Псевдоним	*Тип*	*Назначение*
Path	std::string	псевдоним типа пути к файлу/каталогу

Реализует программный интерфейс работы с классом UnpackerImpl

Методы	
Сигнатура	*Назначение*
public:	
Unpacker()	конструктор
void SetInputFilePath(const Path& path)	вызвать метод установки пути к архиву класса UnpackerImpl
void ChangeInputFilePath(const Path& path)	вызвать метод установки пути к архиву класса UnpackerImpl
void SetOutputPath(const Path& path)	вызвать метод установки пути к каталогу извлечения класса UnpackerImpl
void ChangeOutputPath(const Path& path)	вызвать метод установки пути к каталогу извлечения класса UnpackerImpl
void Unpack()	вызвать метод распаковки архива класса UnpackerImpl

class UnpackerImpl

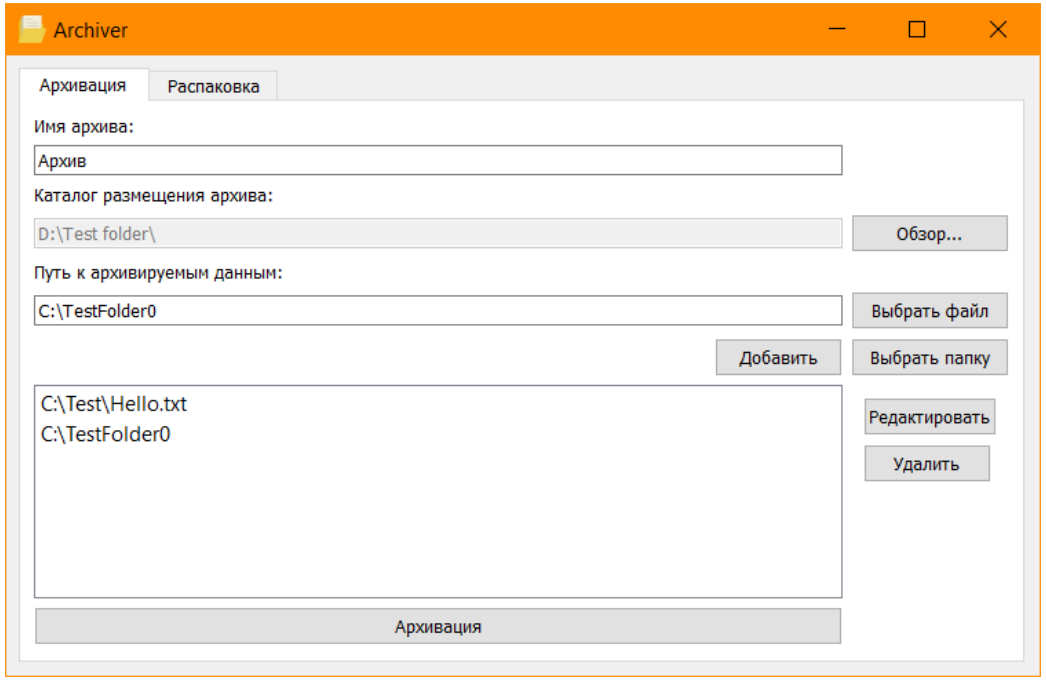
Распаковка архива



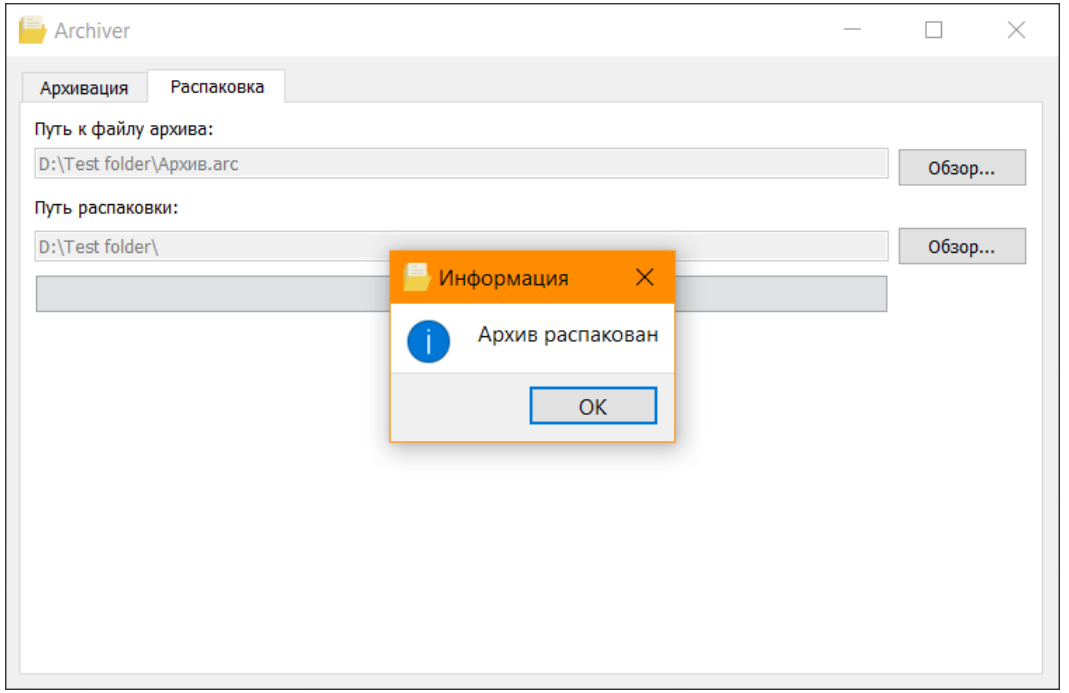
class MainWindow

Реализует управляющую логику и графический интерфейс.
Использует механизм сигналов-слотов для взаимодействия с моделью.

Интерфейс архивации



Интерфейс распаковки



Тестирование

Характеристики, подлежащие тестированию:
1) Корректность запуска и закрытия программы;
2) Пользовательский интерфейс;
3) Функциональность;
4) Обработка исключительных ситуаций.

Методы тестирования:
– позитивное (1, 2);
– негативное (4);
– динамическое (3).

Виды тестирования:
– функциональное (3);
– нефункциональное (1, 2, 4).

Метрики:
– полнота тестирования;
– успешность прохождения тестов.

Критерии завершения тестирования:
– Выполнено более 80% от запланированного объема работ

Проведенные тесты:
– Запуск и закрытие приложения (полностью, успешность 100%);
– Функциональность (полностью, успешность 100%);
– Пользовательский интерфейс (полностью, успешность 100%);
– Исключительные ситуации (полностью, успешность 100%).

Результаты:
Успешность прохождения тестов составила 100% при минимальной допустимой границе в 80%. Проведение тестирования завершено в связи с достижением критерия завершения тестирования.

Сравнение с аналогом

	Прототип архиватора	**Huffman 1.0 (© Боровский Максим)**
Форма распространения	Freeware	Freeware
Ввод имени архива	да	нет (файл переименовывается)
Ввод пути создания архива	да	нет
Сохранение метаданных об архивируемом файле	да	нет
Поддержка больших файлов	да	нет (файлы свыше 50 Mb вызывают ошибку Out of memory)
Архивация папок	да	нет
Архивация нескольких элементов	да	нет
Вывод таблицы кодов	нет	да
Визуализация построенного дерева кодирования Хаффмана	нет	да

Количественная оценка

№ п/п	Формат файла	Исходный размер (kb)	Прототип Сжатый размер (kb)	Прототип Степень сжатия (%)	Huffman Сжатый размер (kb)	Huffman Степень сжатия (%)
1	rtf	34 532	15 845	54,115	15 840	54,13
2	xls	73	38	47,945	36	50,685
3	txt	511	268	47,554	267	47,75
4	evtx	20 548	11 337	44,827	11 335	44,836
5	fb2	405	239	40,988	238	41,235
6	txt	537	329	38,734	328	38,92
7	txt	390	242	37,949	241	38,205
8	txt	3 612	2 243	37,901	2 241	37,957
9	fb2	599	379	36,728	379	36,728
10	doc	9 775	6 277	35,785	6 275	35,806
11	txt	12 426	8 216	33,881	8 214	33,897
12	csv	1 208	802	33,609	801	33,692
13	xml	6 220	4 254	31,608	4 253	31,624
14	mhtml	1 476	1 078	26,965	1 077	27,033
15	ppt	235	180	23,404	179	23,83
16	html	24	19	20,833	18	25
17	cpp	12	10	16,667	9	25
18	doc	1 233	1 070	13,22	1 069	13,301
19	bmp	655	607	7,328	605	7,634
20	pdf	5 415	5 275	2,585	5 273	2,622
21	arc	1 078	1 074	0,371	1 073	0,464
22	mp3	5 685	5 675	0,176	5 673	0,211
23	jpeg	821	820	0,122	819	0,244
24	mkv	131 078	131 026	0,04	Out of memory	
25	mp4	891 729	891 726	0,00	Out of memory	

Заключение

1. **Изучена предметная область;**
2. **Разработаны технические требования к программе;**
3. **Спроектированы:**
   - архитектура программы;
   - структура создаваемого архива;
   - классы на основе технических требований;
   - графический пользовательский интерфейс.
4. **Реализованы:**
   - спроектированные классы;
   - алгоритм кодирования Хаффмана;
   - функциональность которая не была реализована в найденном аналоге;
   - спроектированный графический пользовательский интерфейс.
5. **Проведено тестирование программы:**
   - разработан план тестирования программы;
   - реализовано тестирование программы.
6. **Проведено сравнение с существующими аналогами.**
7. **Определены дальнейшие задачи разработки программы:**
   - Модифицировать реализацию функции Unpack() класса UnpackerImpl, реализовав в ней многопоточное декодирование исходных данных;
   - Реализовать работу с популярными форматами архивов;
   - Реализовать поддержку популярных алгоритмов сжатия.