Факультатив по программированию на языке C

Занятие 3
Память

План занятий

№	Тема	Описание
1	Введение в курс	Основы работы с Linux. Написание и компиляция простейших программ с использованием gcc. Правила написания кода. Разбиение программы на отдельные файлы. Make файлы
2	Ввод данных. Библиотеки	Работа со вводом/выводом. Статические и динамические библиотеки. Компиляция.
3	Хранение данных. Память	Хранение процесса в памяти компьютера. Виртуальная память, сегментация. Секции программы.
4	Хранение данных.	Стек, куча. Типы данных. Преобразования типов. Gdb и отладка Хранение различных типов данных. Указатели, ссылки. Передача аргументов в функцию по ссылке/указателю.
5	Обработка данных	Переполнение данных. Правильное преобразование типов и пример ошибок, связанных с этим. Битовые операции – сдвиги, логические операции. Битовые поля.
6	Программирование под встраиваемые ОС	Перенос проекта под микрокомпьютер Raspberry Pi

Что мы уже прошли?

1)Компиляция программ
2)Make сборка
3)Создание библиотек
4)Работа с вводом/выводом

Что мы пройдем сегодня?

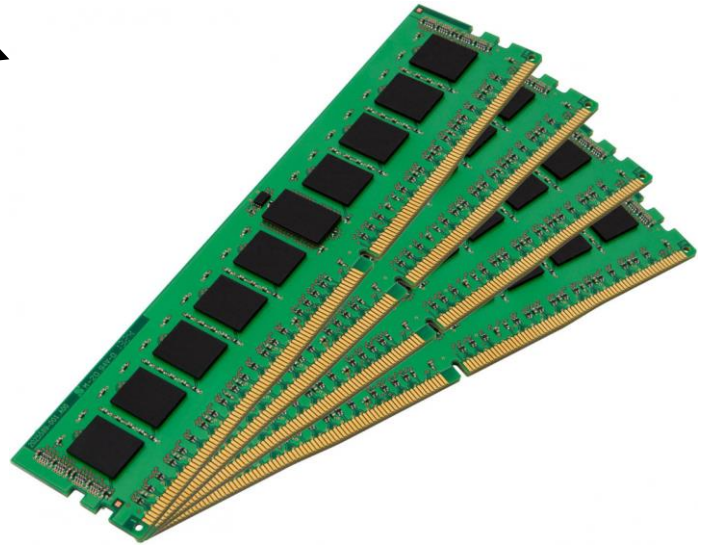
1)Организация памяти
2)Хранение процесса в памяти
3)Работа с git
4)Наконец начнем наш проект…

Дерево языка

Дерево языка

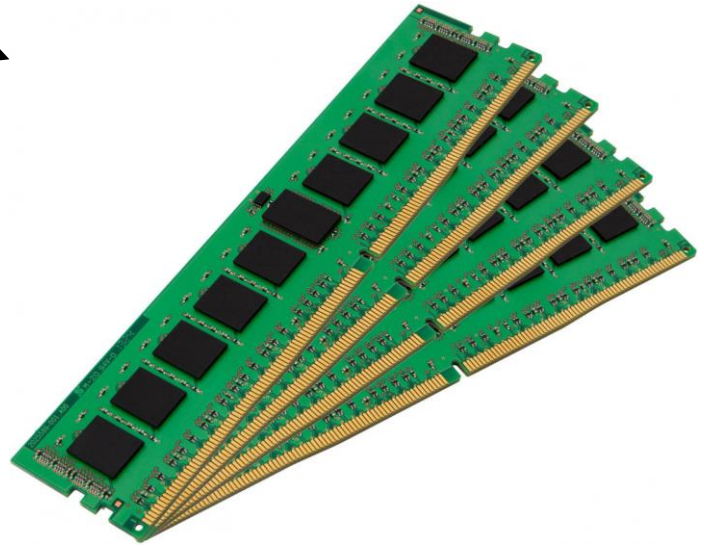
Где?

0x00004edc



Где?

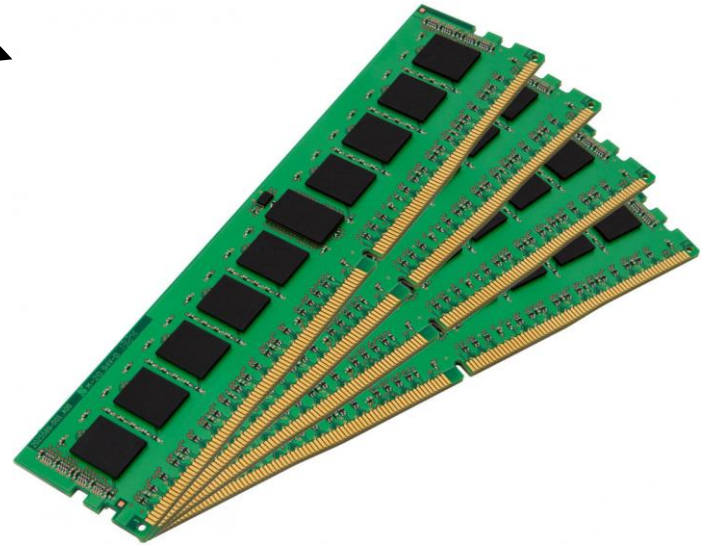
0x00004edc



Если это реальные адреса жесткого диска, то тогда зачем нужна оперативная память?

Где?

0x00004edc



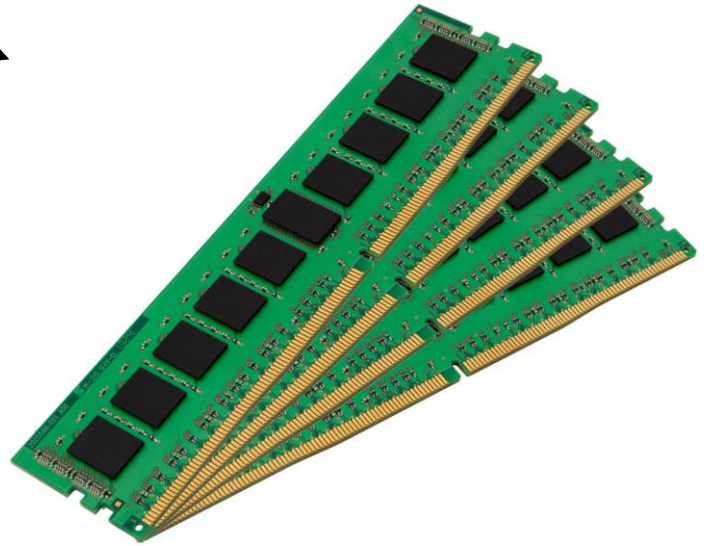
Если это реальные адреса жесткого диска, то тогда зачем нужна оперативная память?

Если это реальные адреса в оперативной памяти, то как тогда организовать одновременную работу двух процессов?

Где?

0x00004edc

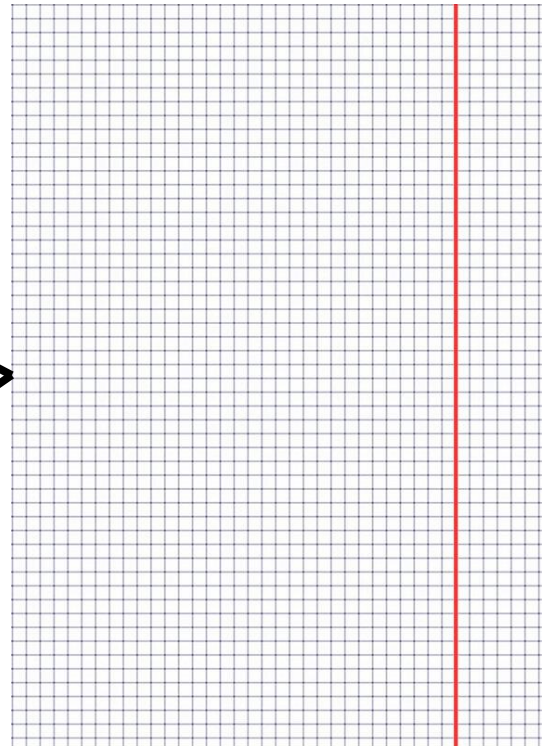
?



Пример - библиотека

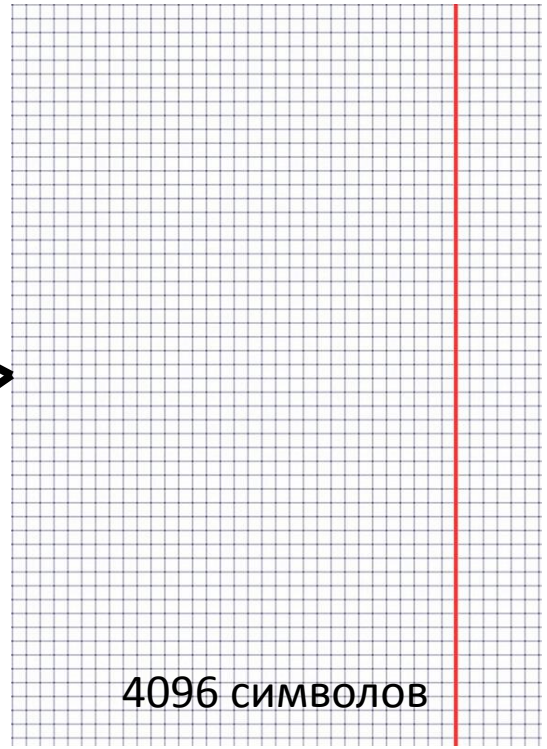
Пример - библиотека

С точки зрения **библиотекаря**



Пример

~ 2 млн страниц

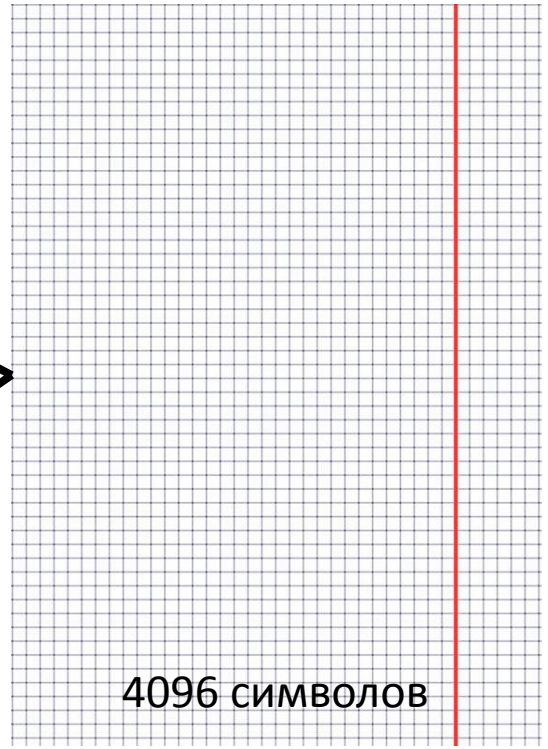


~ 2 млрд страниц

Пример

~ 2 млн страниц

~ 2 млрд страниц

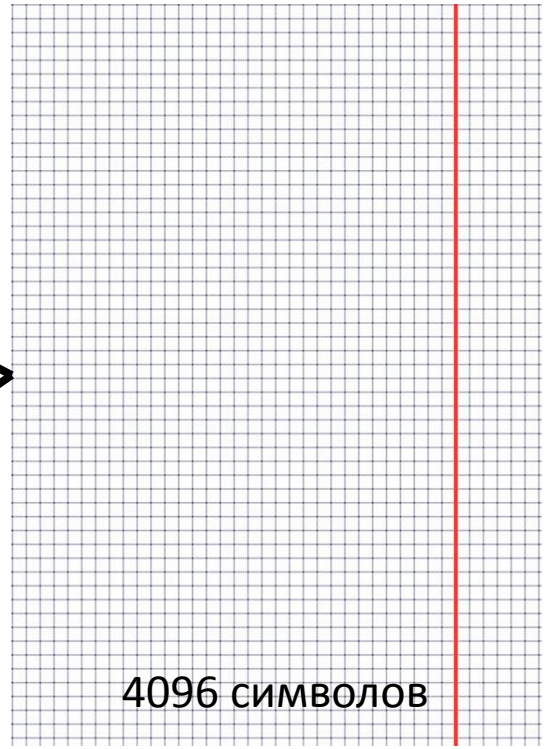


Что будет, если нужно записать 4097 символов?

Пример

~ 2 млн страниц

~ 2 млрд страниц



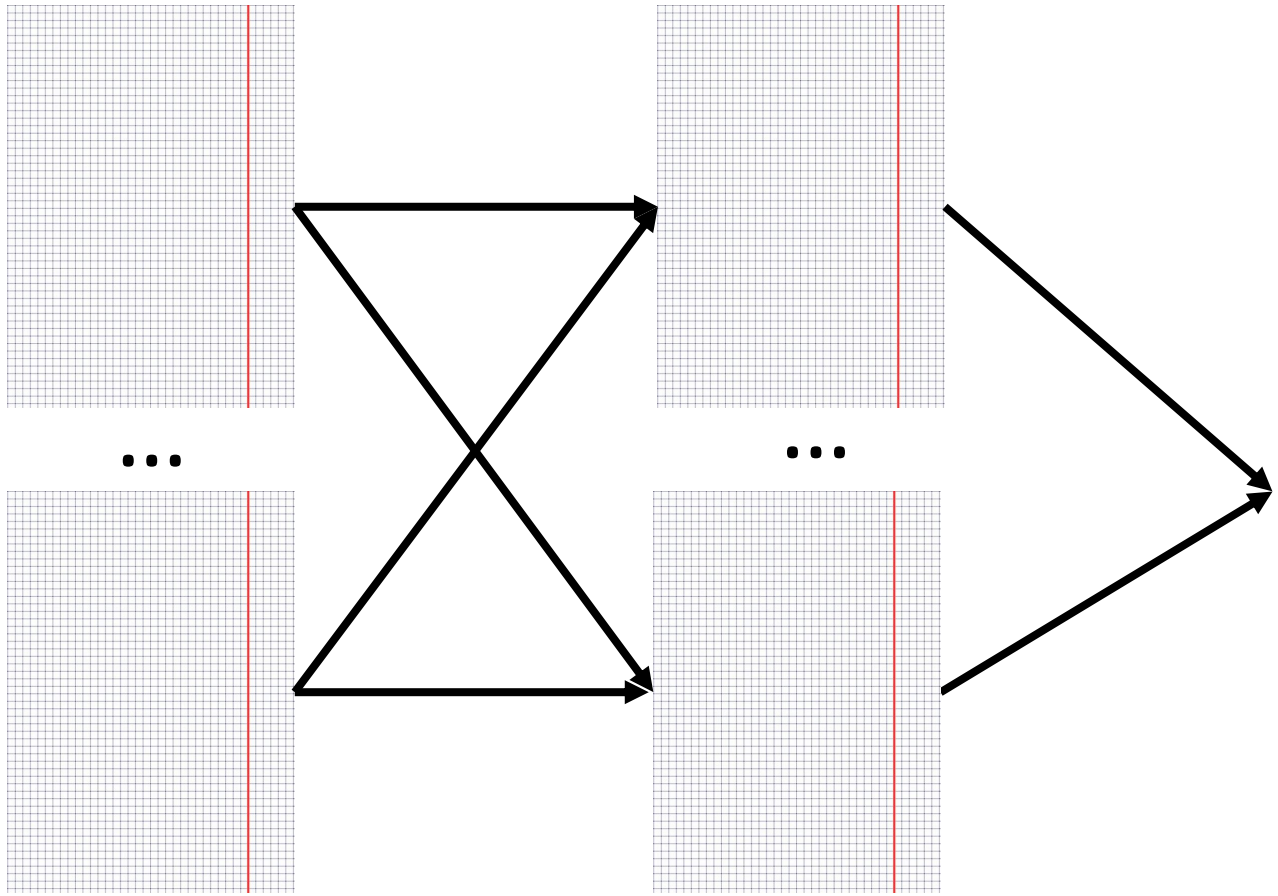
Что будет, если нужно записать 4097 символов?

Как не запутаться?

Пример

Каталог оглавлений

Оглавление

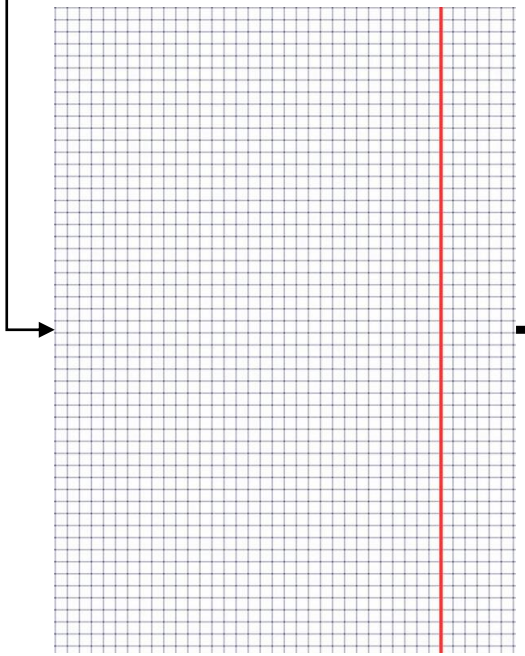


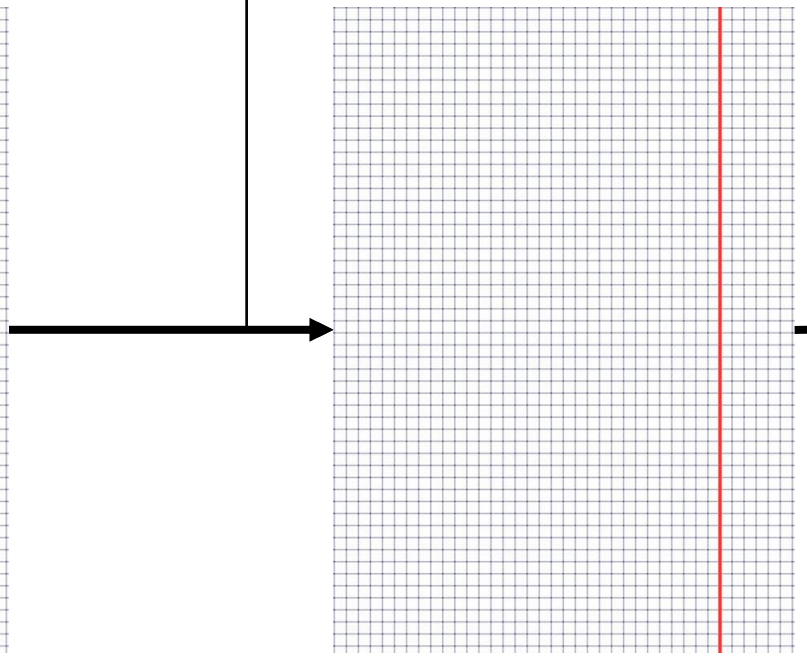
Пример

Каталог	Таблица	Смещение

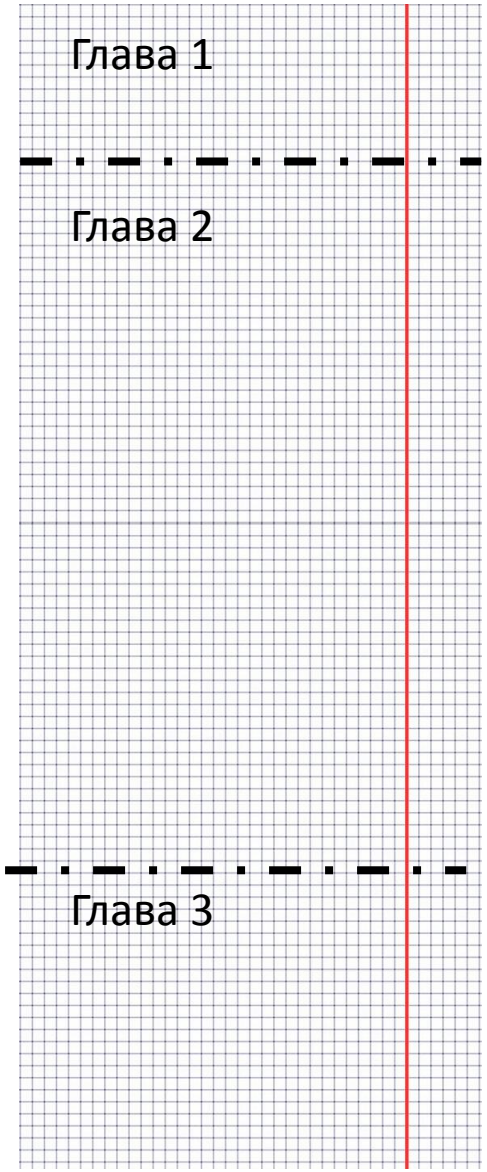
Каталог оглавлений

Оглавление



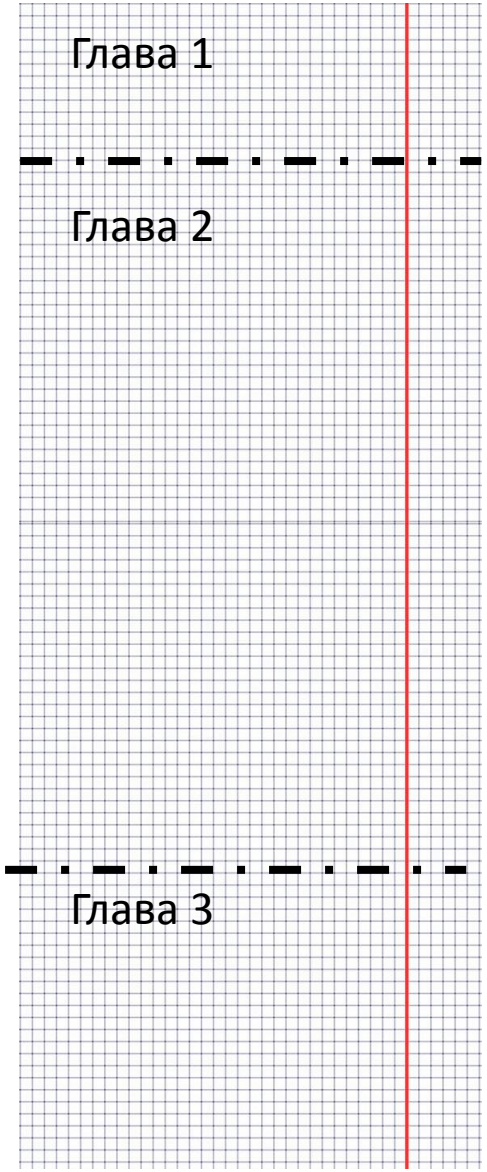


Пример



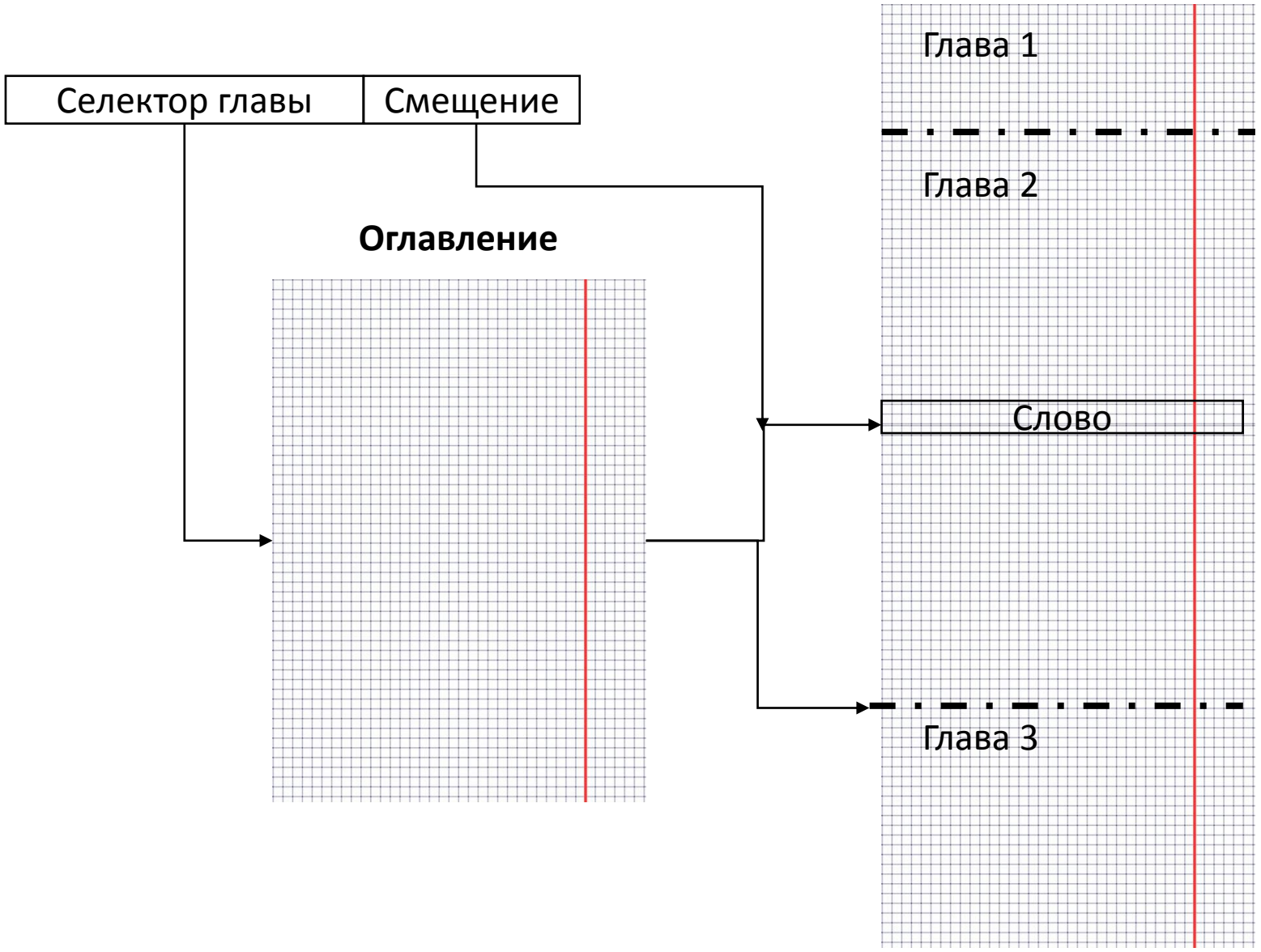
С точки зрения **писателя**

Пример

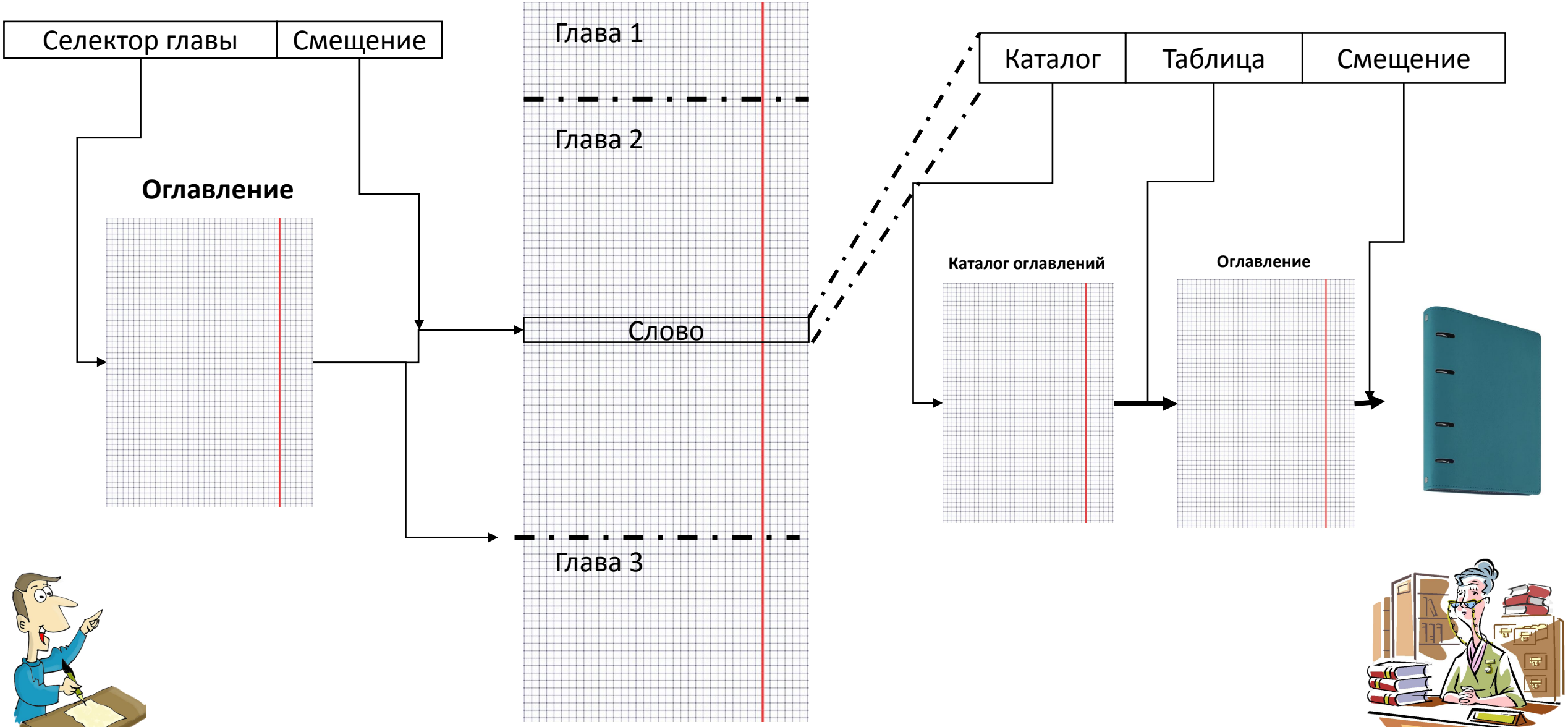


Размеры глав различаются!!!

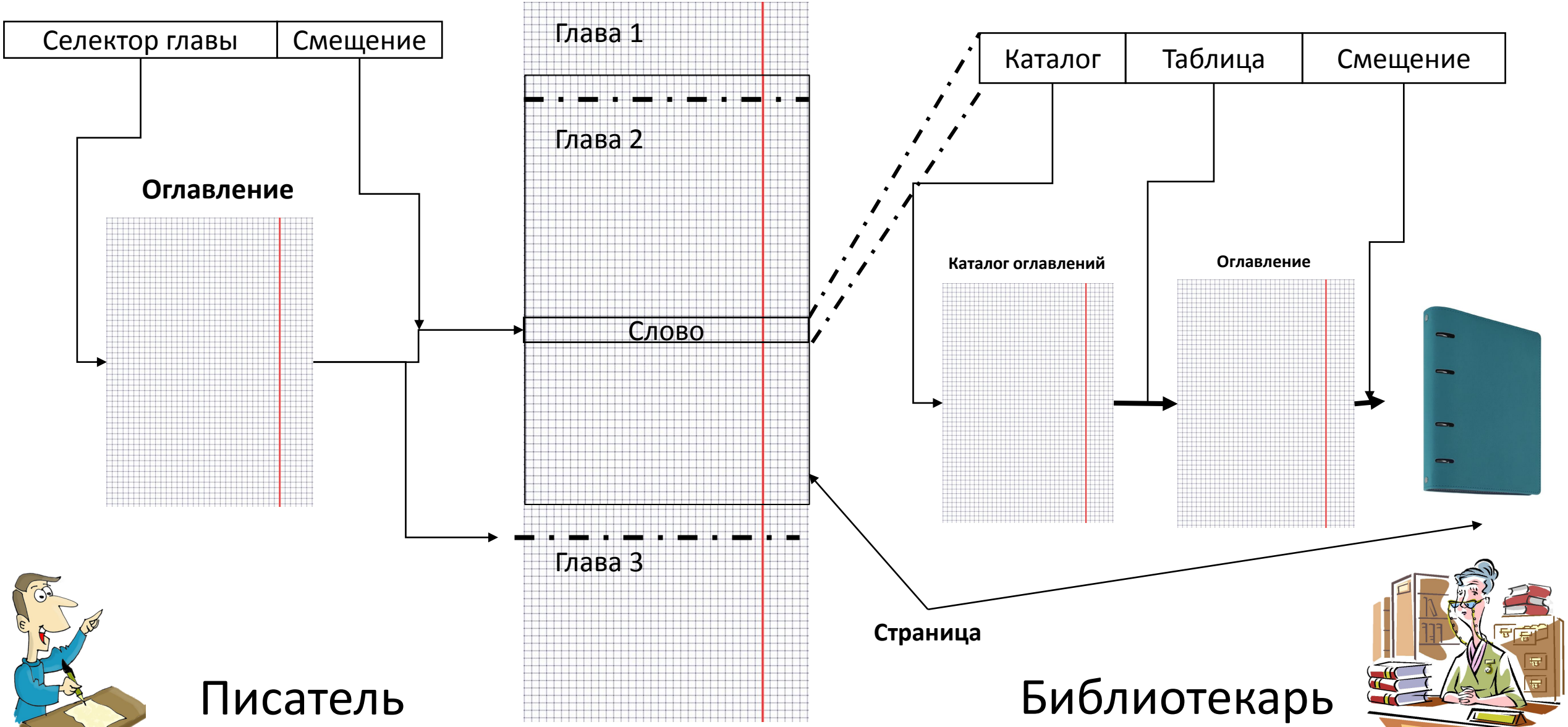
Пример


Селектор главы
Смещение
Оглавление
Глава 1
Глава 2
Слово
Глава 3

Пример


Селектор главы
Смещение
Оглавление
Глава 1
Глава 2
Слово
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

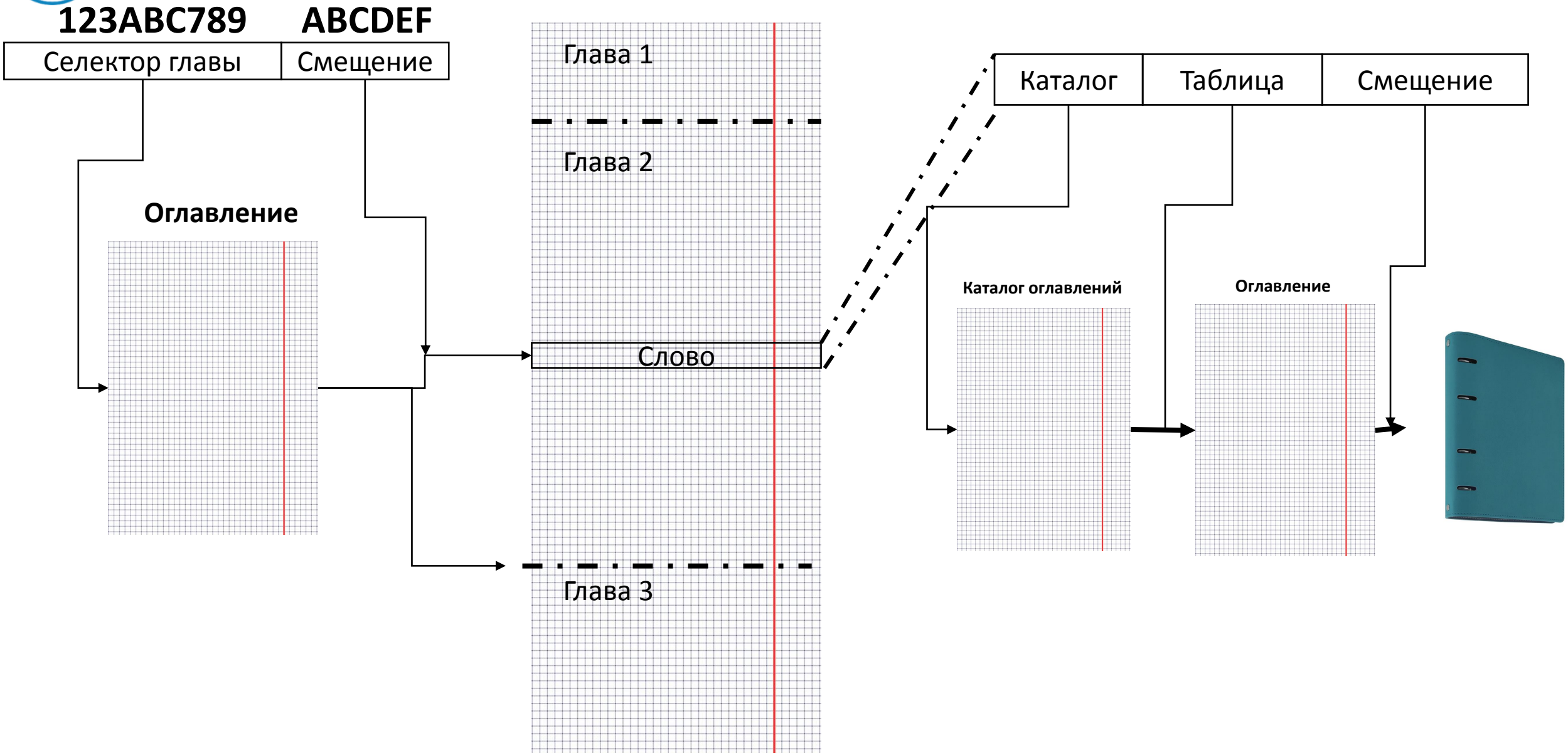
Пример



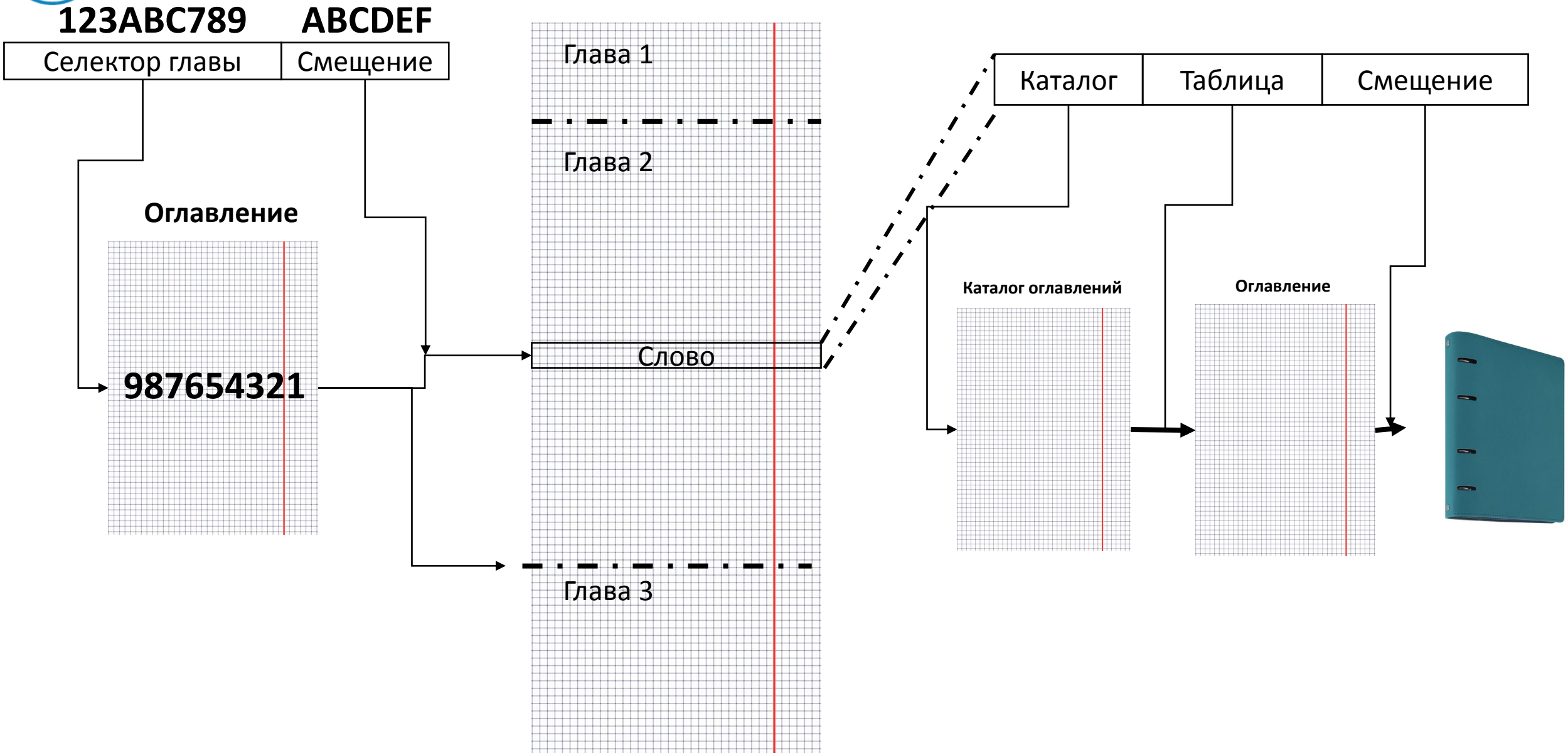
Писатель

Библиотекарь

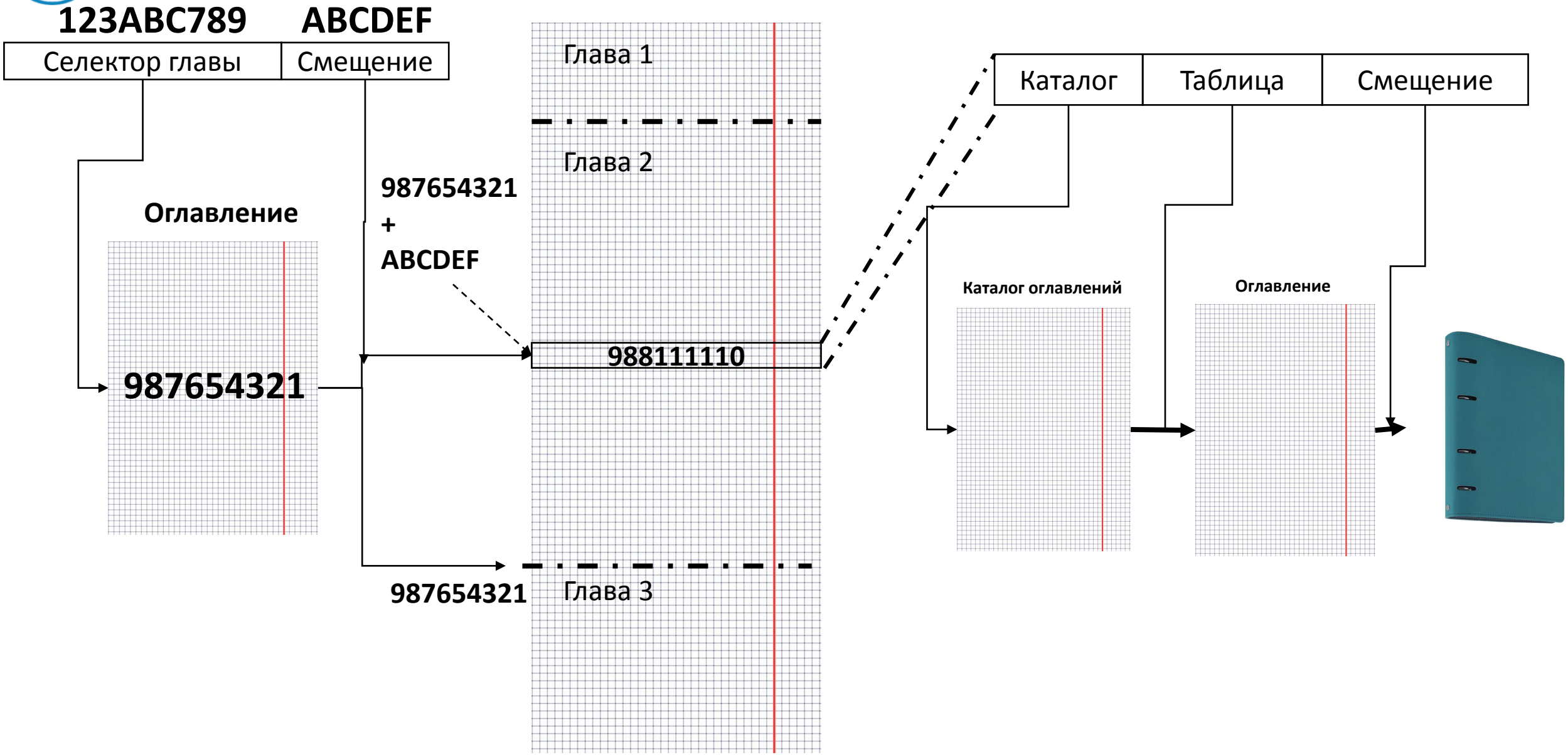
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
Глава 1
Глава 2
Слово
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

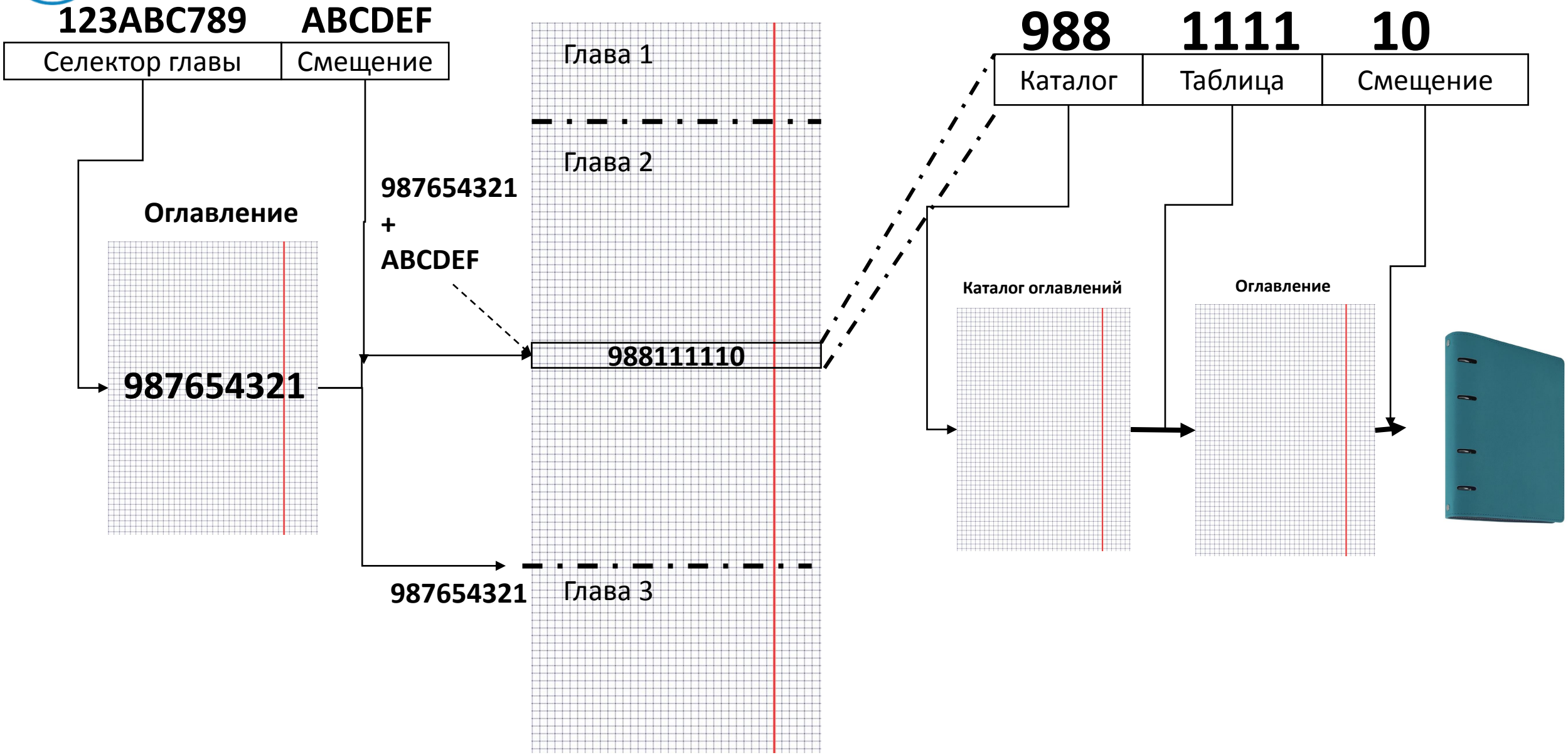
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
987654321
Глава 1
Глава 2
Слово
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

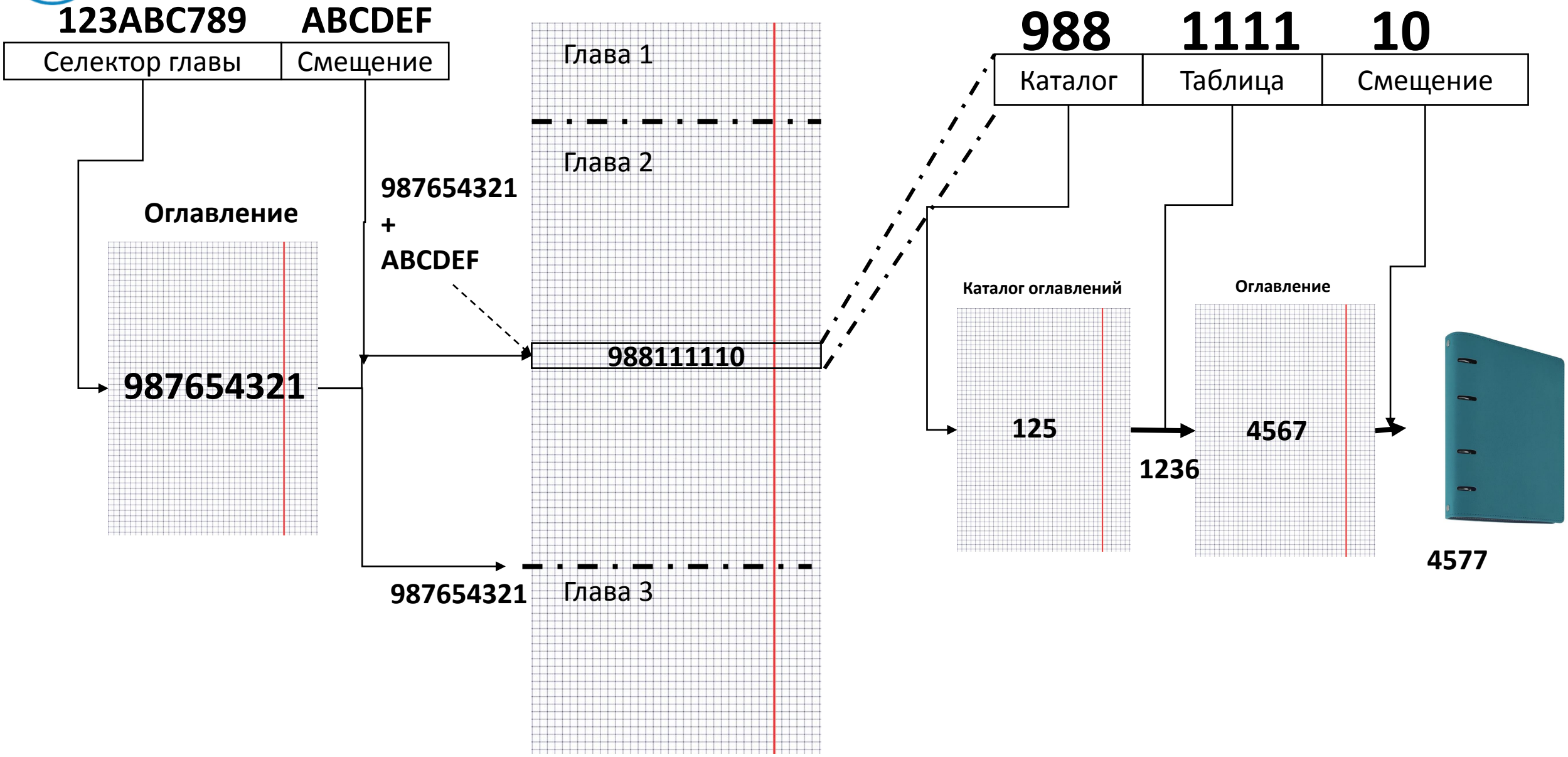
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
987654321
987654321
+
ABCDEF
Глава 1
Глава 2
988111110
987654321
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

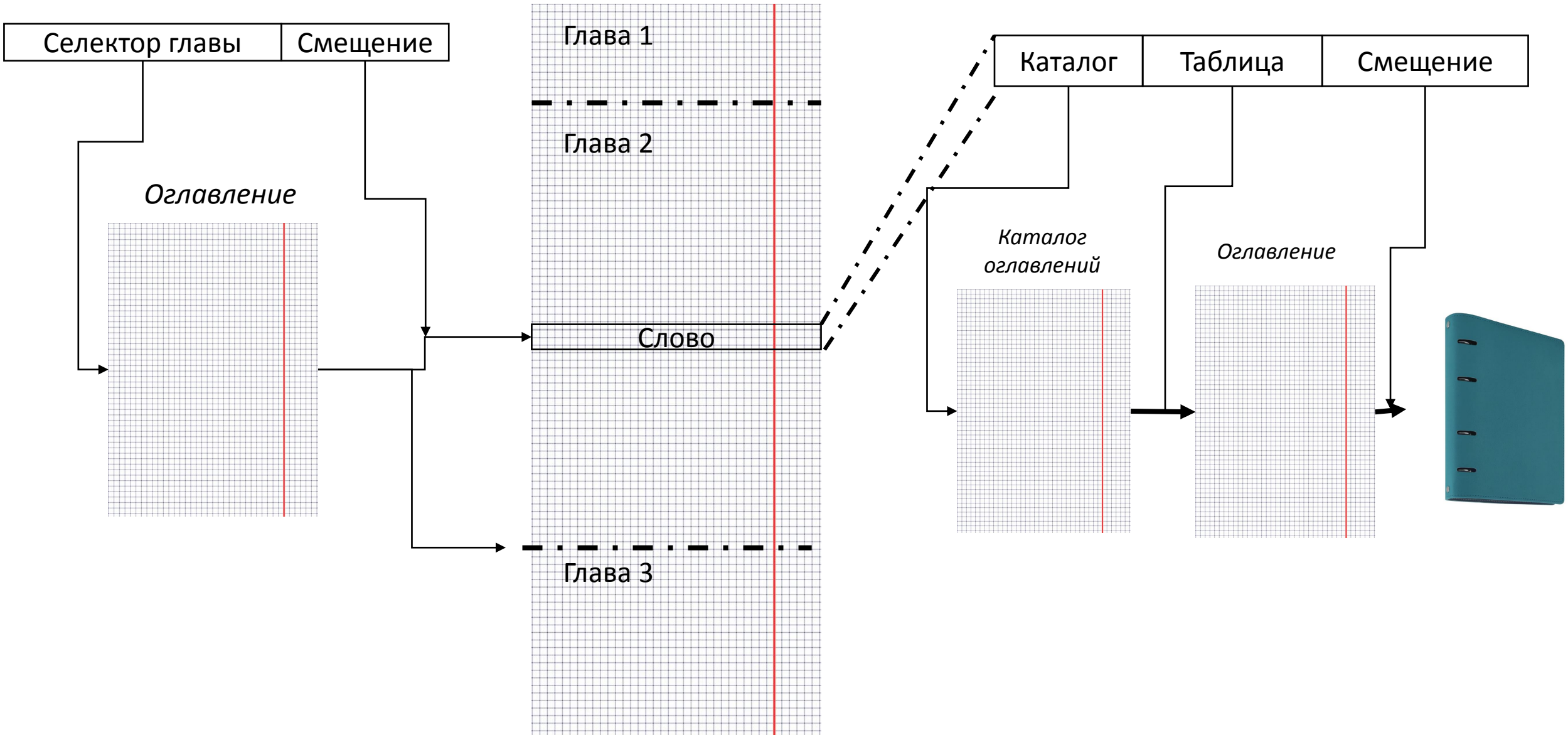
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
987654321
987654321
+
ABCDEF
Глава 1
Глава 2
988111110
987654321
Глава 3
988
1111
10
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

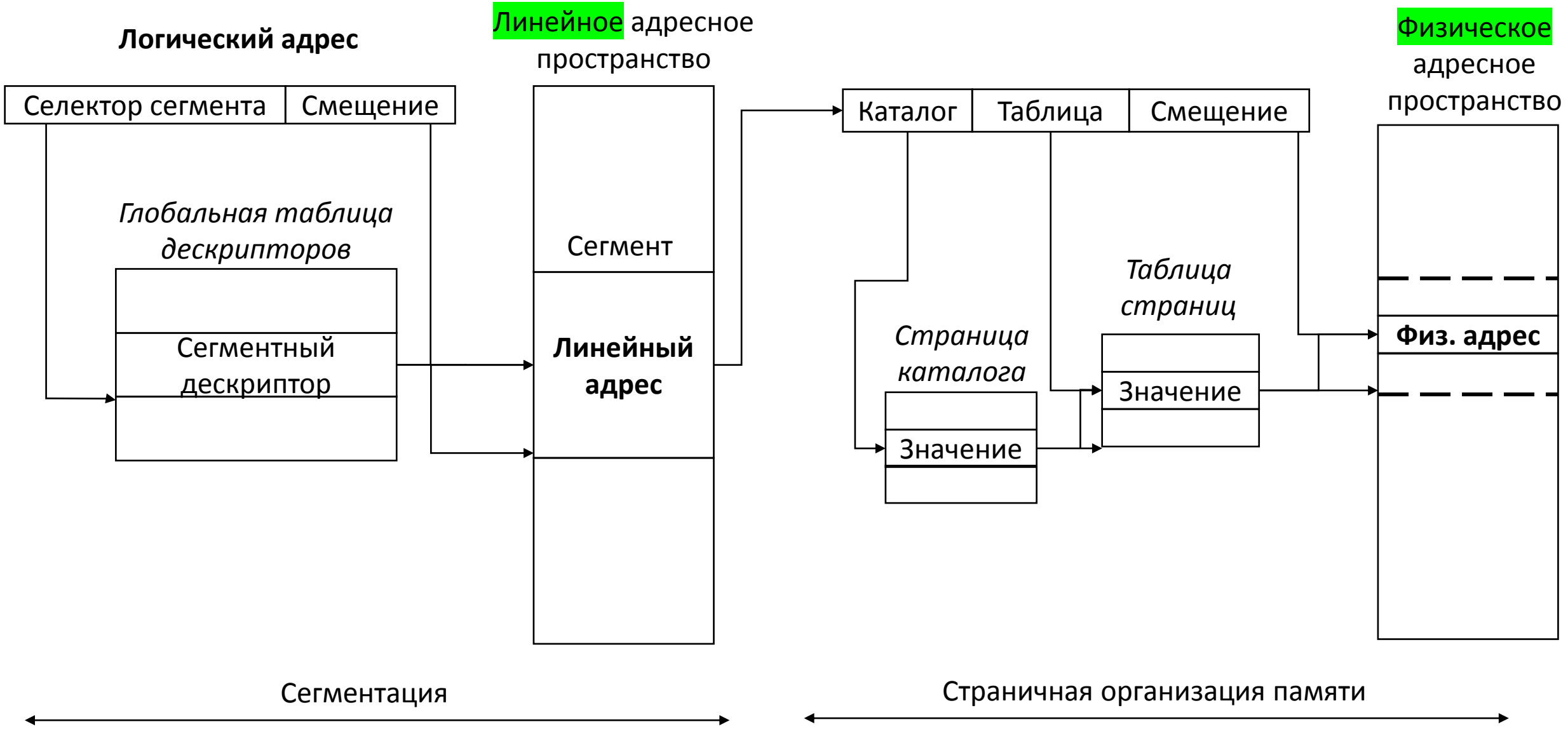
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
987654321
987654321
+
ABCDEF
Глава 1
Глава 2
988111110
987654321
Глава 3
988
1111
10
Каталог
Таблица
Смещение
Каталог оглавлений
125
1236
Оглавление
4567
4577

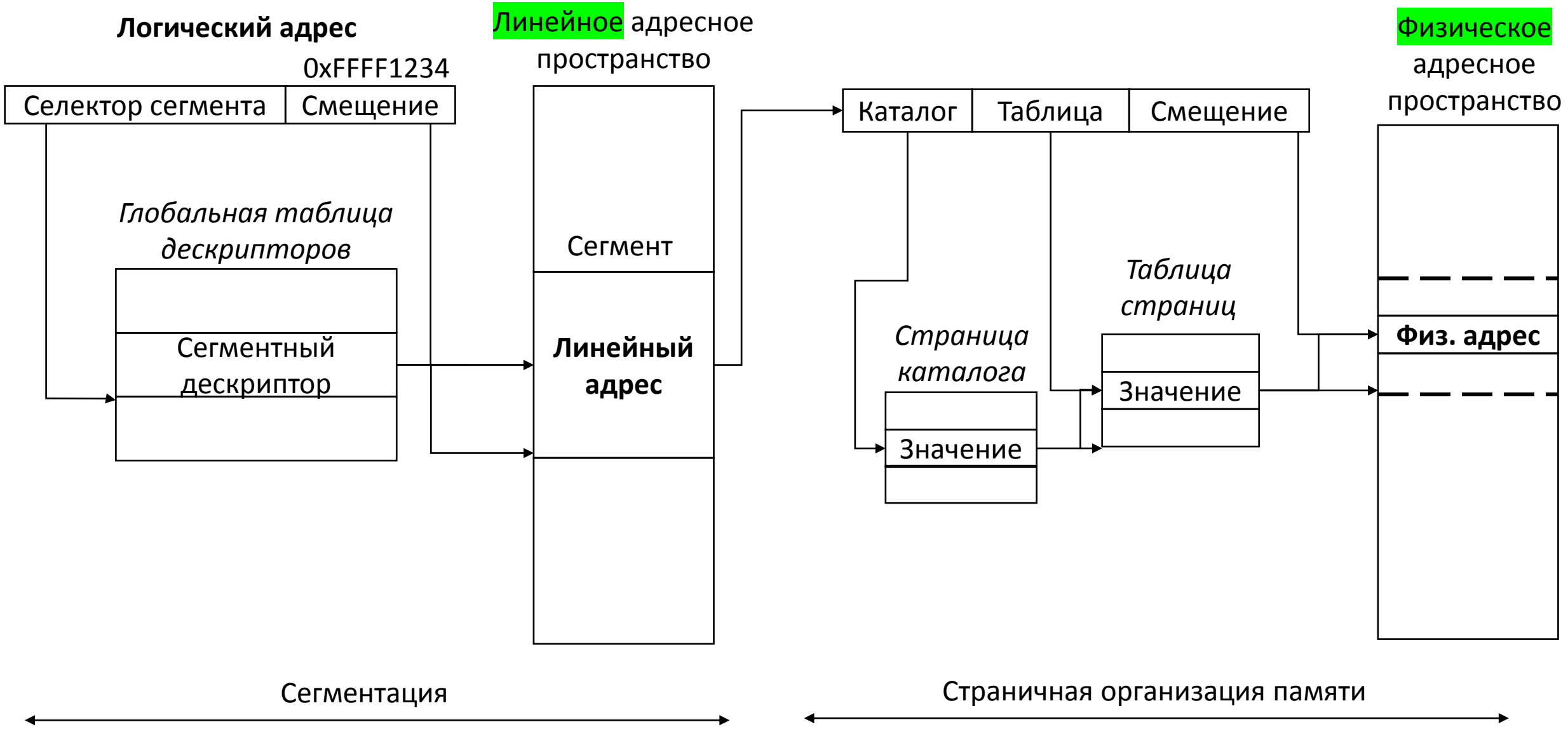
Пример


Селектор главы
Смещение
Оглавление
Глава 1
Глава 2
Слово
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

Устройство памяти (x86, 32 bit)



Устройство памяти (x86, 32 bit)



Промежуточные выводы!

1) Память поделена на страницы
2) Преобразование:

Логический адрес -> Линейный адрес -> Физический адрес

Что?

Модульное программирование

Работа с портом

- Настройка порта
- …
- Чтение данных
- …
- Отправка сигнала завершения
- …

Работа с буфером

- Запись данных в буфер
- …
- Чтение данных из буфера
- …

Работа с файлом

- Открытие файла
- …
- Запись данных
- …
- Закрытие файла
- …

main

Анализ данных

…

Удобно работать с программой

Модульное программирование

Работа с портом

- Настройка порта
- …
- Чтение данных
- …
- Отправка сигнала завершения
- …

Работа с буфером

- Запись данных в буфер
- …
- Чтение данных из буфера
- …

Работа с файлом

- Открытие файла
- …
- Запись данных
- …
- Закрытие файла
- …

main

Анализ данных

…

Как бы вы предложили разбить текст программы для удобства работы загрузчика?

Удобно работать с программой

Секции программы

```
#include <stdio.h>
#include <stdlib.h>

static int bss[1024];
static int data[1024] = {1024};
static const int rodata[8192] = {"rodata"};

int main(void)
{
	int stack;
	int *heap = (int*)malloc(1024*sizeof(int));
	free(heap);
	return 0;
}
```

Секции программы

К
О
Д

```
#include <stdio.h>
#include <stdlib.h>

static int bss[1024];
static int data[1024] = {1024};
static const int rodata[8192] = {"rodata"};

int main(void)
{
    int stack;
    int *heap = (int*)malloc(1024*sizeof(int));
    free(heap);
    return 0;
}
```

Неинициализированная глобальная переменная

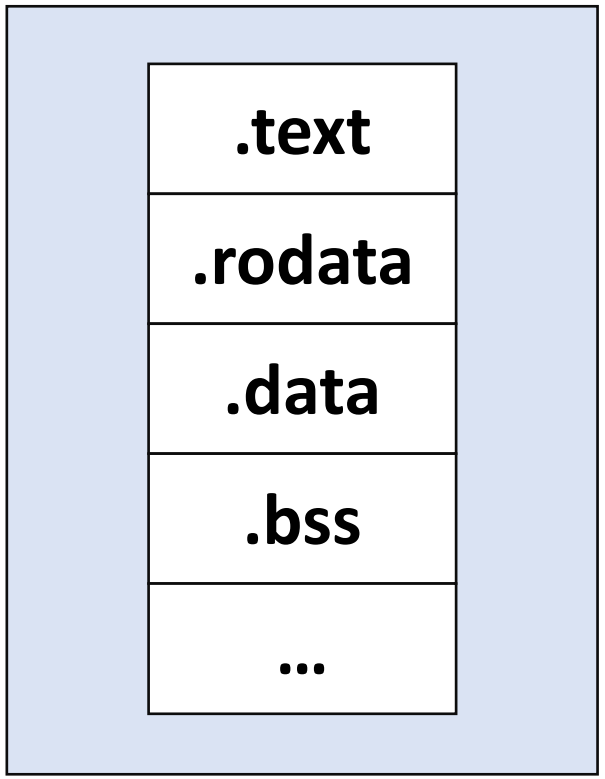
Инициализированная глобальная переменная

Инициализированная глобальная константа

Статическая переменная

Динамический массив

Секции программы



Код программы

Глобальная константы

Инициализированная глобальные переменные

Неинициализированная глобальные переменные

stack Стек

heap Куча

Что?

ELF (Executable and Linkable Format) — формат двоичных файлов, определяющий структуру бинарных файлов, библиотек, и файлов ядра

Процесс – программа во время исполнения и все её элементы: адресное пространство, глобальные переменные , регистры, стек, счетчик команд, состояние, открытые файлы, дочерние процессы и т. д

Устройство ELF файлов

Основные компоненты:

1) Заголовок
2) Описание секций
3) Описание сегментов

Устройство ELF файлов

Основные компоненты:

1)Заголовок

2)Описание секций

3)Описание сегментов

Заголовок файла

```
typedef struct
{
        unsigned char e_ident[16];
        uint16_t e_type;
        uint16_t e_machine;
        uint32_t e_version;
        uint32_t e_entry;
        uint32_t e_phoff;
        uint32_t e_shoff;
        uint32_t e_flags;
        uint16_t e_ehsize;
        uint16_t e_phentsize;
        uint16_t e_phnum;
        uint16_t e_shentsize;
        uint16_t e_shnum;
        uint16_t e_shstrndx;
} Elf32_Ehdr;
```

Пример

```
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x1070
  Start of program headers:          52 (bytes into file)
  Start of section headers:          18528 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         12
  Size of section headers:           40 (bytes)
  Number of section headers:         31
  Section header string table index: 30
```

Устройство ELF файлов

Основные компоненты:

1)Заголовок

2)Описание секций

3)Описание сегментов

Информация, хранящаяся в ELF-файле, организована в секции. Каждая секция имеет свое уникальное имя. Некоторые секции хранят служебную информацию ELF-файла (например, таблицы строк), другие секции хранят отладочную информацию, третьи секции хранят код или данные программы.

Таблица заголовков секций

```
typedef struct
{
        uint32_t sh_name;
        uint32_t sh_type;
        uint32_t sh_flags;
        uint32_t sh_addr;
        uint32_t sh_offset;
        uint32_t sh_size;
        uint32_t sh_link;
        uint32_t sh_info;
        uint32_t sh_addralign;
        uint32_t sh_entsize;
 } Elf32_Shdr;
```

Пример

Section Headers:

[Nr] Name	Type	Addr	Off	Size	ES	Flg	Lk	Inf	Al
[0]	NULL	00000000	000000	000000	00		0	0	0
[1] .interp	PROGBITS	000001b4	0001b4	000013	00	A	0	0	1
…									
[12] .init	PROGBITS	00001000	001000	000024	00	AX	0	0	4
…									
[16] .text	PROGBITS	00001070	001070	0001e9	00	AX	0	0	16
….									
[18] .rodata	PROGBITS	00002000	002000	000e1c	00	A	0	0	32
….									
[25] .data	PROGBITS	00005000	004000	000010	00	WA	0	0	4
[26] .bss	NOBITS	00005010	004010	000008	00	WA	0	0	4
….									

Устройство ELF файлов

Основные компоненты:

1) Заголовок
2) Описание секций
3) Описание сегментов

Таблица заголовков программы содержит информацию, необходимую для загрузки программы на выполнение.

Таблица заголовков программы

```
typedef struct
{
        uint32_t p_type;
        Elf32_Off p_offset;
        Elf32_Addr p_vaddr;
        Elf32_Addr p_paddr;
        uint32_t p_filesz;
        uint32_t p_memsz;
        uint32_t p_flags;
        uint32_t p_align;
} Elf32_Phdr;
```

Пример

Program Headers:

	Type	Offset	VirtAddr	PhysAddr	FileSiz	MemSiz	Flg	Align
0	PHDR	0x000034	0x00000034	0x00000034	0x00180	0x00180	R	0x4
1	INTERP	0x0001b4	0x000001b4	0x000001b4	0x00013	0x00013	R	0x1
2	LOAD	0x000000	0x00000000	0x00000000	0x003d4	0x003d4	R	0x1000
3	LOAD	0x001000	0x00001000	0x00001000	0x00274	0x00274	R E	0x1000
4	LOAD	0x002000	0x00002000	0x00002000	0x00f94	0x00f94	R	0x1000
5	LOAD	0x003edc	0x00004edc	0x00004edc	0x00134	0x0013c	RW	0x1000
6	DYNAMIC	0x003ee4	0x00004ee4	0x00004ee4	0x000f8	0x000f8	RW	0x4

…

Пример

Program Headers:

	Type	Offset	VirtAddr	PhysAddr	FileSiz	MemSiz	Flg	Align
0	PHDR	0x000034	0x00000034	0x00000034	0x00180	0x00180	R	0x4
1	INTERP	0x0001b4	0x000001b4	0x000001b4	0x00013	0x00013	R	0x1
2	LOAD	0x000000	0x00000000	0x00000000	0x003d4	0x003d4	R	0x1000
3	LOAD	0x001000	0x00001000	0x00001000	0x00274	0x00274	R E	0x1000
4	LOAD	0x002000	0x00002000	0x00002000	0x00f94	0x00f94	R	0x1000
5	LOAD	0x003edc	0x00004edc	0x00004edc	0x00134	0x0013c	RW	0x1000
6	DYNAMIC	0x003ee4	0x00004ee4	0x00004ee4	0x000f8	0x000f8	RW	0x4

…

Segment Sections...

00

01 .interp

02 .interp .note.gnu.build-id .note.gnu.property .note.ABI-tag .gnu.hash .dynsym .dynstr .gnu.version

03 .init .plt .plt.got .plt.sec **.text** .fini

04 **.rodata** .eh_frame_hdr .eh_frame

05 .init_array .fini_array .dynamic .got **.data .bss**

06 .dynamic

…

Пример со своим ELF

```
git clone https://github.com/SergeyBalabaev/Elective
```

Elective -> lesson3 -> ELF

```
chmod +x make.sh
./make.sh
./Prog
```

Пример со своим ELF

```
./make.sh
./Prog
```

Измените программу так, чтобы она выводила ваши имя и фамилию

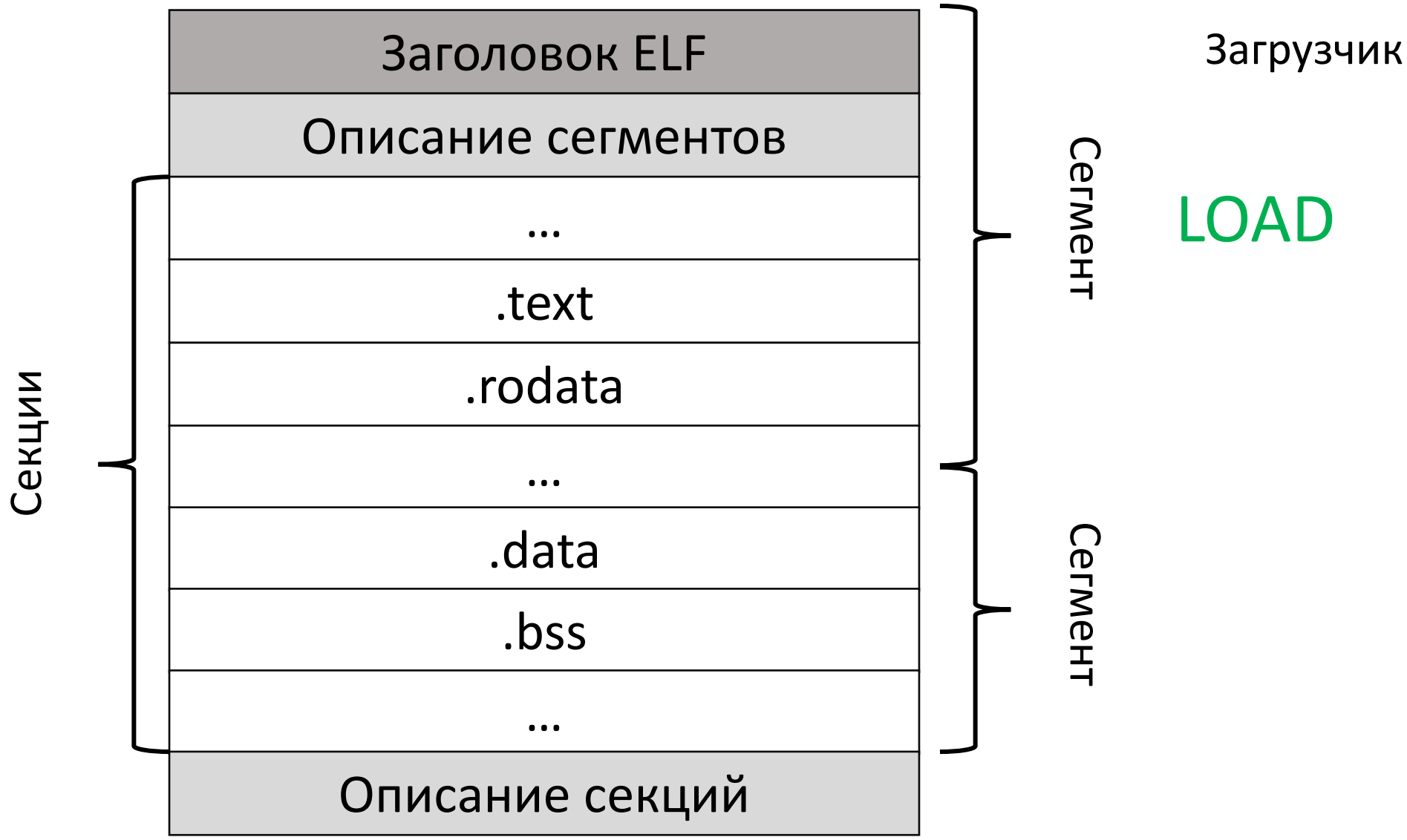
ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

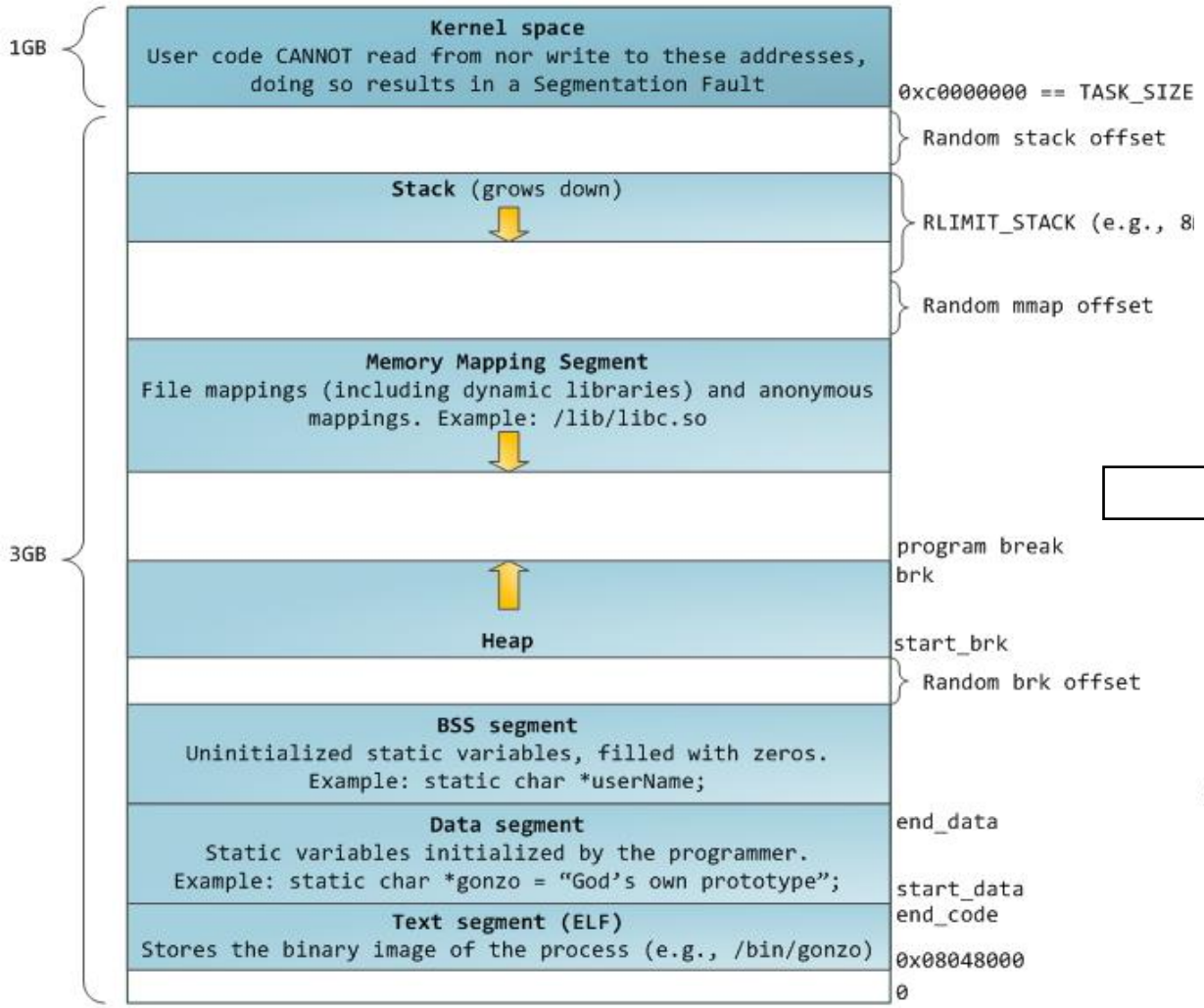
Секции и сегменты программы

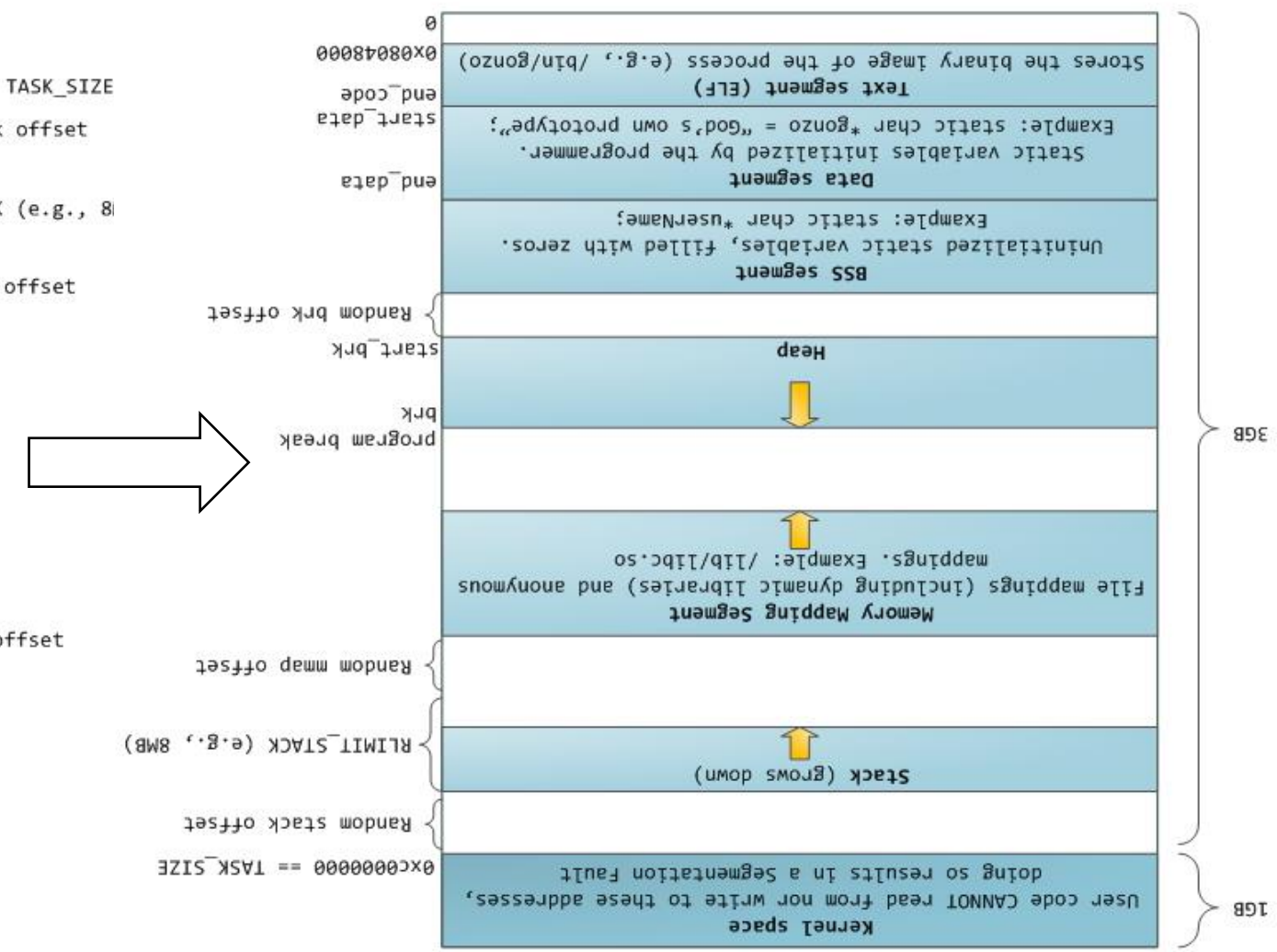
Компоновщик

Загрузчик



Секции программы





Еще один пример

Elective -> lesson3 -> Memory

```
gcc memory.c -o memory -m32 –nostartfiles
./memory
```

Должно возникнуть предупреждение – проигнорируйте его

Git

Git - это консольная утилита, для отслеживания и ведения истории изменения файлов, в вашем проекте

С помощью Git-а вы можете откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий

Репозитории возможно хранить в интернете

Git

```
sudo apt install git
git init
git add File.txt
git commit -m "first commit“
git branch -M main
git remote add origin
https://github.com/you_repository/you_project
git push -u origin master
```

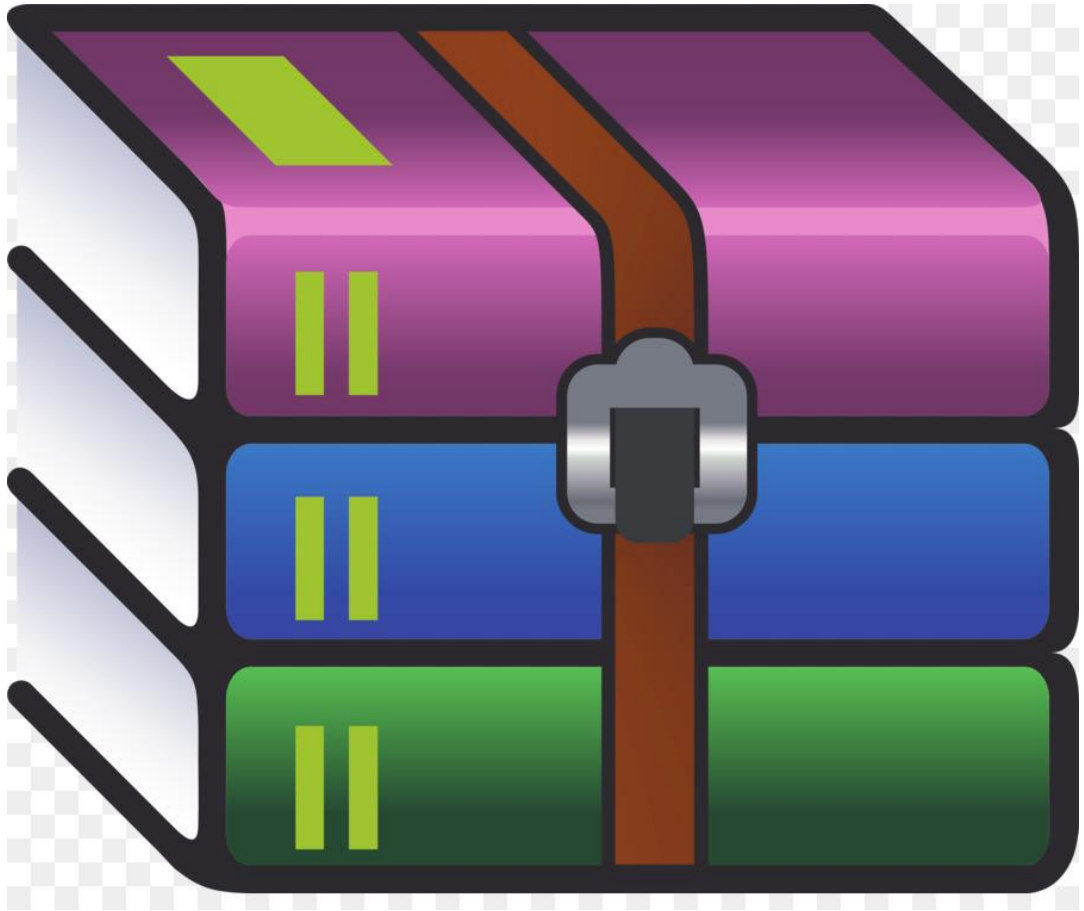
Git

Для работы
git add .
git status
git commit -m "1 commit"
git diff

Для отката
git log --oneline
git checkout 4beac58 .

Для залива на удаленный репозиторий
git remote add origin
https://github.com/you_repository/you_project
git remote -v
git push -u origin master

Практическая часть



Задание

Реализовать программу, кодирующая и сжимающая файлы по алгоритму Хаффмана
Реализовать программу для декодирования файлов

```
git clone https://github.com/SergeyBalabaev/Archiver
```

Основы теории информации

Информация (Information) — содержание сообщения или сигнала; сведения, рассматриваемые в процессе их передачи или восприятия, позволяющие расширить знания об интересующем объекте

Информация – первоначально — сведения, передаваемые одними людьми другим людям устным, письменным или каким-нибудь другим способом

Информация - как коммуникацию, связь, в процессе которой устраняется неопределенность. (К. Шеннон)

Мера информации

Пусть X – источник дискретных сообщений. Число различных состояний источника – N.
Переходы из одного состояния в другое не зависят от предыдущих состояний, а вероятности перехода в эти состояния $p_j=P\{X = x_j\}$

Тогда за меру количества информации примем следующую величину:

$$H(X) = -\sum_{k=1}^{N} p_k \log(p_k)$$

Эта величина называется **энтропией**

Энтропия

Свойства энтропии

1) Энтропия неотрицательна
2) Максимально возможное значение энтропии равно log(N)
3) Энтропия нескольких независимых файлов равна сумме энтропий каждого из них

Битовые затраты – среднее число бит приходящееся на один символ сообщения

$$R=\sum_{k=1}^{N} p_k R_k$$

R_k - число бит в коде символа x_k

Пример

Задача:

Пусть пришло следующее сообщение: «мамамылараму»

Рассчитаем энтропию сообщения и битовые затраты. Будем считать, что один символ кодируется 1 байтом.

1) Рассчитаем вероятности появления символов

мамамылараму

Символ	**м**	**а**	**ы**	**л**	**р**	**у**	**Σ**
Количество	4	4	1	1	1	1	**12**
Вероятность	1/3	1/3	1/12	1/12	1/12	1/12	**1**

Пример

2) Рассчитаем энтропию и битовые затраты

$$H(X) = -\sum_{k=1}^{N} p_k \log(p_k)$$

$$H(X) = -\sum_{k=1}^{6} p_k \log(p_k) = -\left(\frac{1}{3}\log\frac{1}{3} + \frac{1}{3}\log\frac{1}{3} + \frac{1}{12}\log\frac{1}{12} + \frac{1}{12}\log\frac{1}{12} + \frac{1}{12}\log\frac{1}{12} + \frac{1}{12}\log\frac{1}{12}\right) \sim \mathbf{2,25}$$

$$R = \sum_{k=1}^{N} p_k R_k$$

$$H(X) = \sum_{k=1}^{6} p_k R_k = \left(\frac{1}{3} * 8 + \frac{1}{3} * 8 + \frac{1}{12} * 8 + \frac{1}{12} * 8 + \frac{1}{12} * 8 + \frac{1}{12} * 8\right) = \mathbf{8}$$

Идея сжатия

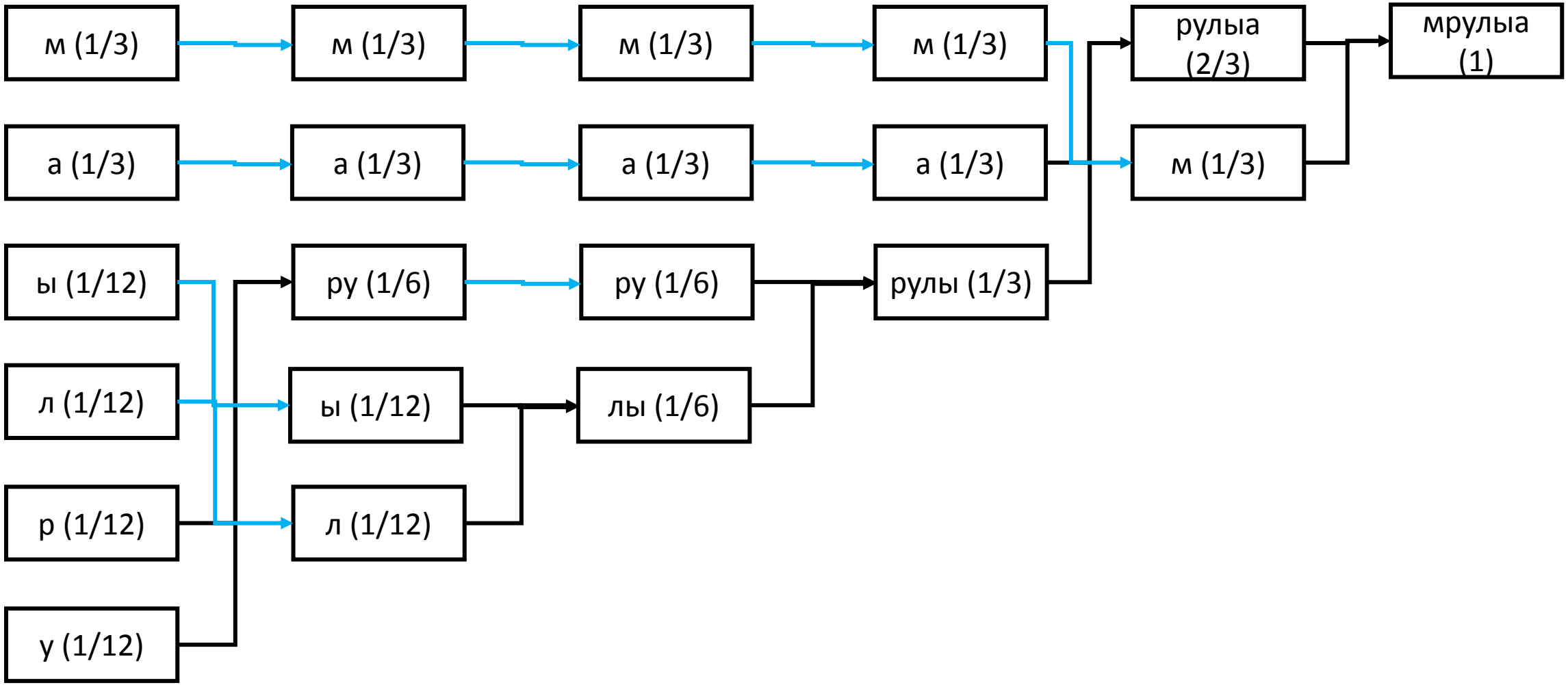
Давайте заменим стандартный равномерный ASCII код на неравномерный так, чтобы часто встречающимся символам соответствовали более короткие кодовые последовательности. Если средние битовые затраты будут меньше, чем 8 бит, то сжатие удалось!

Алгоритм Хаффмана

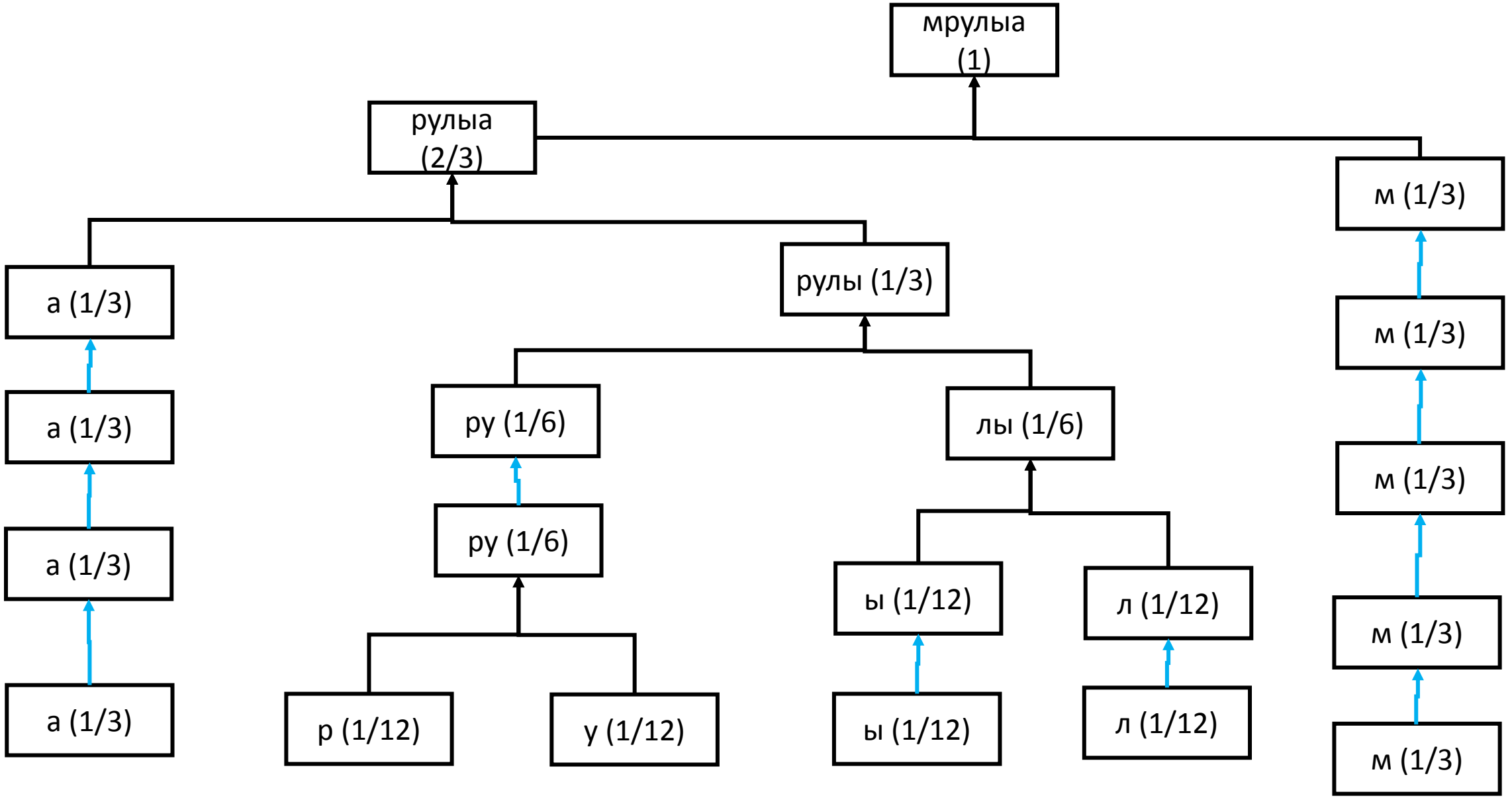
На вход алгоритма подается таблица символов

1. Построение дерева Хаффмана
   1.1 Упорядочиваем таблицу символов в порядке убывания вероятностей
   1.2 Два последних символа, имеющих наименьшие вероятности появления объединяются в новый символ
   Если есть еще символы, то возвращаемся на 1.1
2. Построение битового кода
   Для каждого узла дерева строим по два ребра, приписываем одному из них 1, другому 0

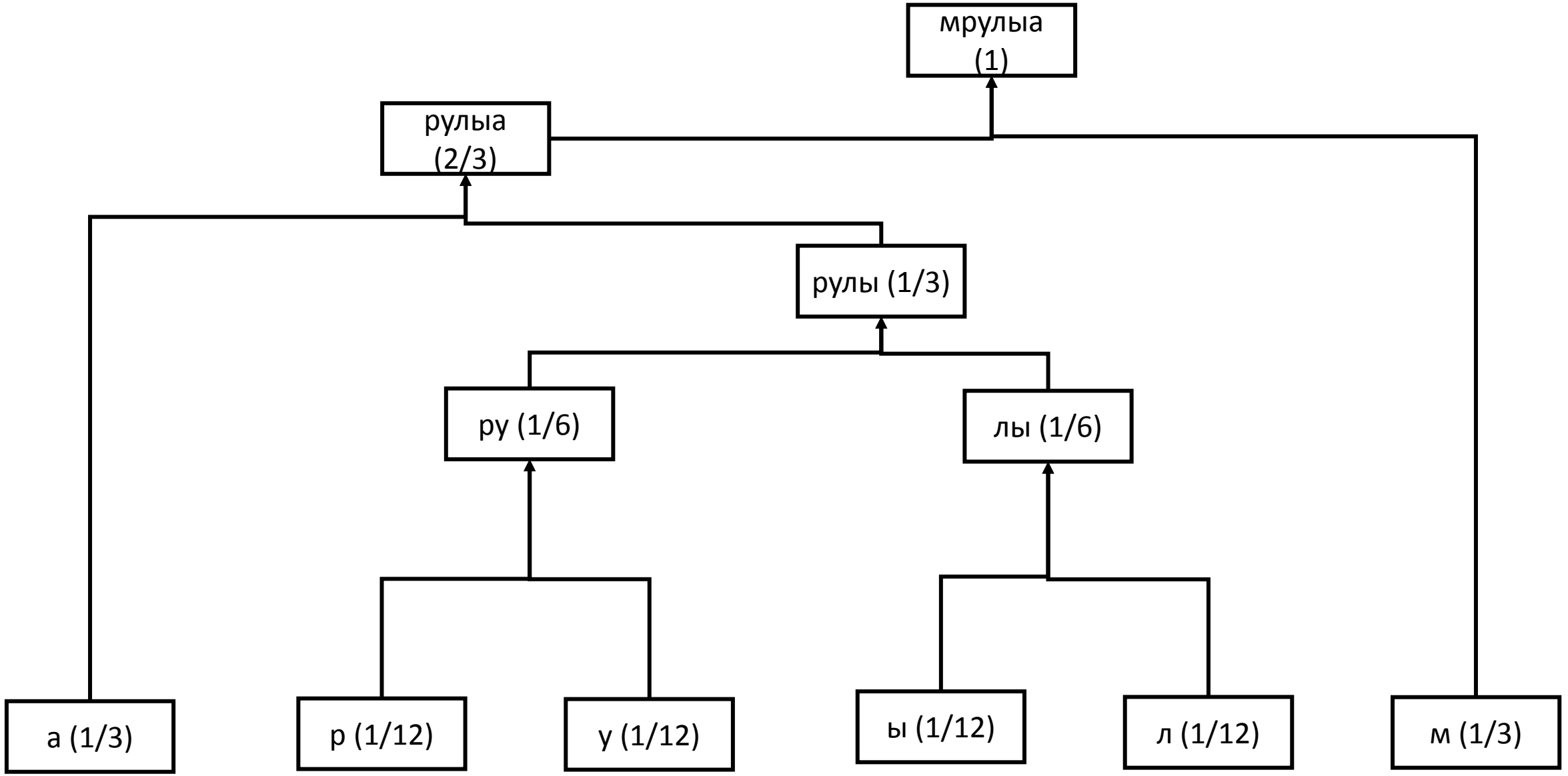
Алгоритм Хаффмана



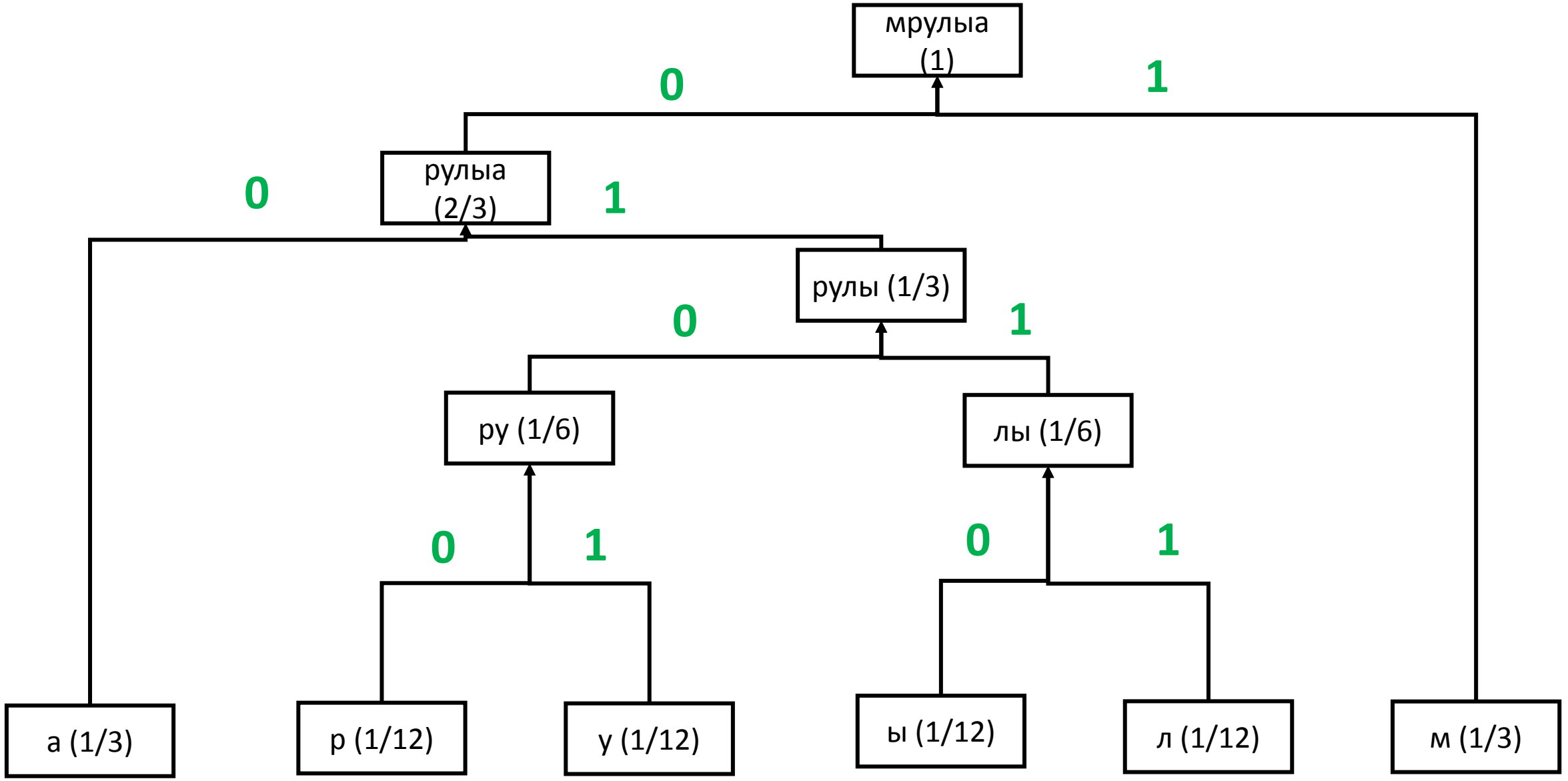
Алгоритм Хаффмана



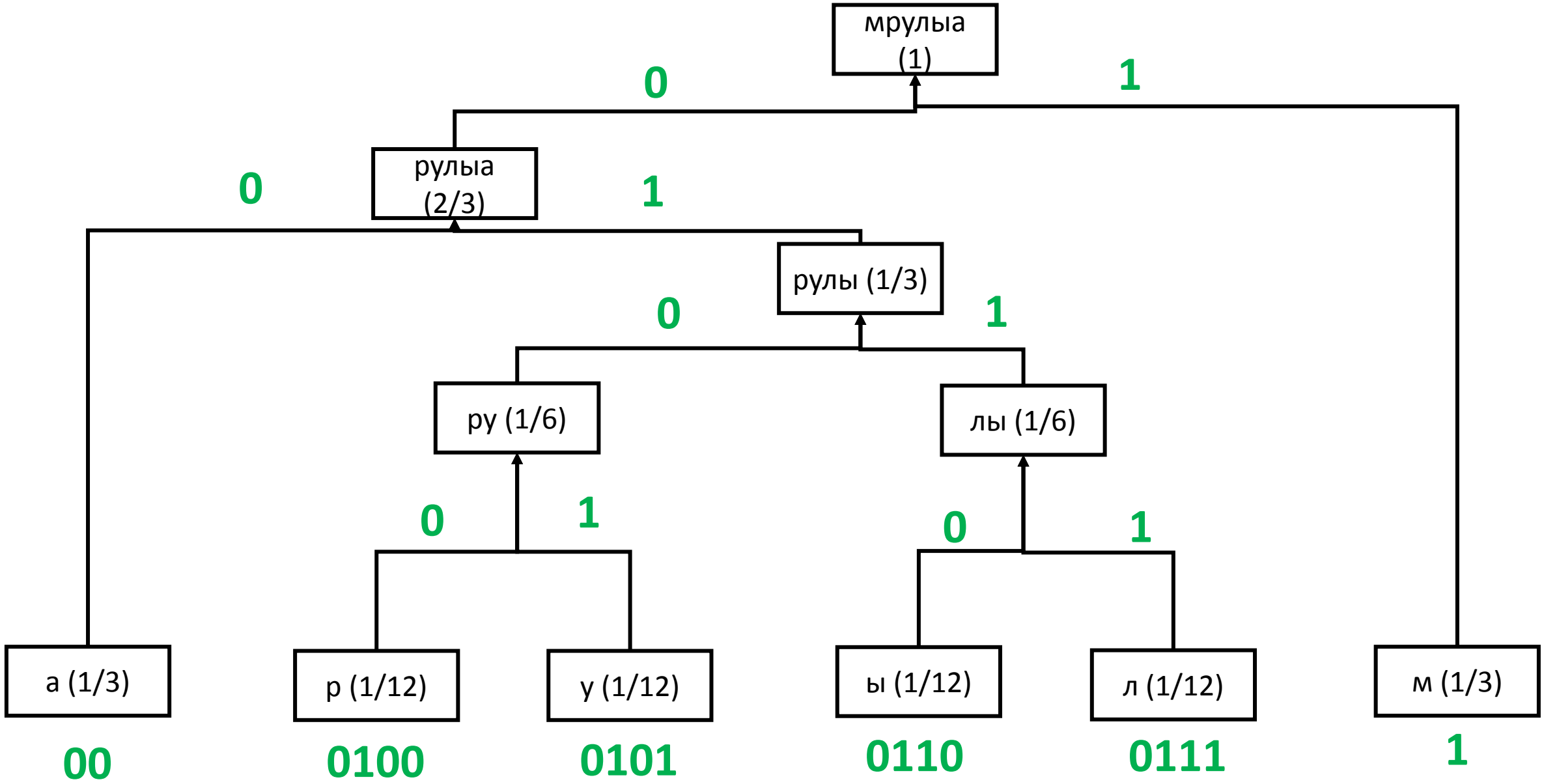
Алгоритм Хаффмана



Алгоритм Хаффмана



Алгоритм Хаффмана



Расчет битовых затрат

Без кодирования	R = 8
Кодирование «в лоб»	R ~ 2,42
Метод Хаффмана	R ~ 2,33
Энтропия	H ~ 2,25

Программная реализация

Программа должна быть разделена на 6 модулей (см. следующий слайд).
Порядок работы:

1) Открытие и чтение файла
2) Расчёт частоты встречаемости символов
3) Сортировка массива символов по частоте по убыванию
4) Создание дерева Хаффмана
5) Создание кодов Хаффмана
6) Создание промежуточного файла, состоящего из 0 и 1 – закодированный полученным кодом первый файл
7) Запись полученной последовательности в архивный файл

Программная реализация

math_func.c	types.c	arch_logic.c	information.c	file_In_out.c
Описание **математических функций** (расчет энтропии, битовых затрат и т.п.)	Описание **глобальных** типов	Описание **логики работы** архиватора	Вывод информации о работе программы	Работа со вводом и выводом

main.c

Задача на сегодня

На основе примера работы с текстовыми файлами написать программу, выполняющую следующие действия:

1) Открытие файла в бинарном виде и посимвольное чтение потока байтов
2) Расчет гистограммы появлений символов
3) Функция расчета энтропии по полученной в п. 2 гистограмме
4) Залить получившийся код на github
5) Сбросить мне ссылку на свой репозиторий

Спасибо за внимание!