Факультатив по программированию на языке C

Занятие 4
Память ч.2

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
5)Организация памяти
6)Хранение процесса в памяти

Что мы пройдем сегодня?

1)Хранение данных в памяти
2)Посмотрим на адреса и указатели
3)Еще раз разберемся с git
4)Обсудим наш проект…

Дерево языка

Дерево языка

Напоминание

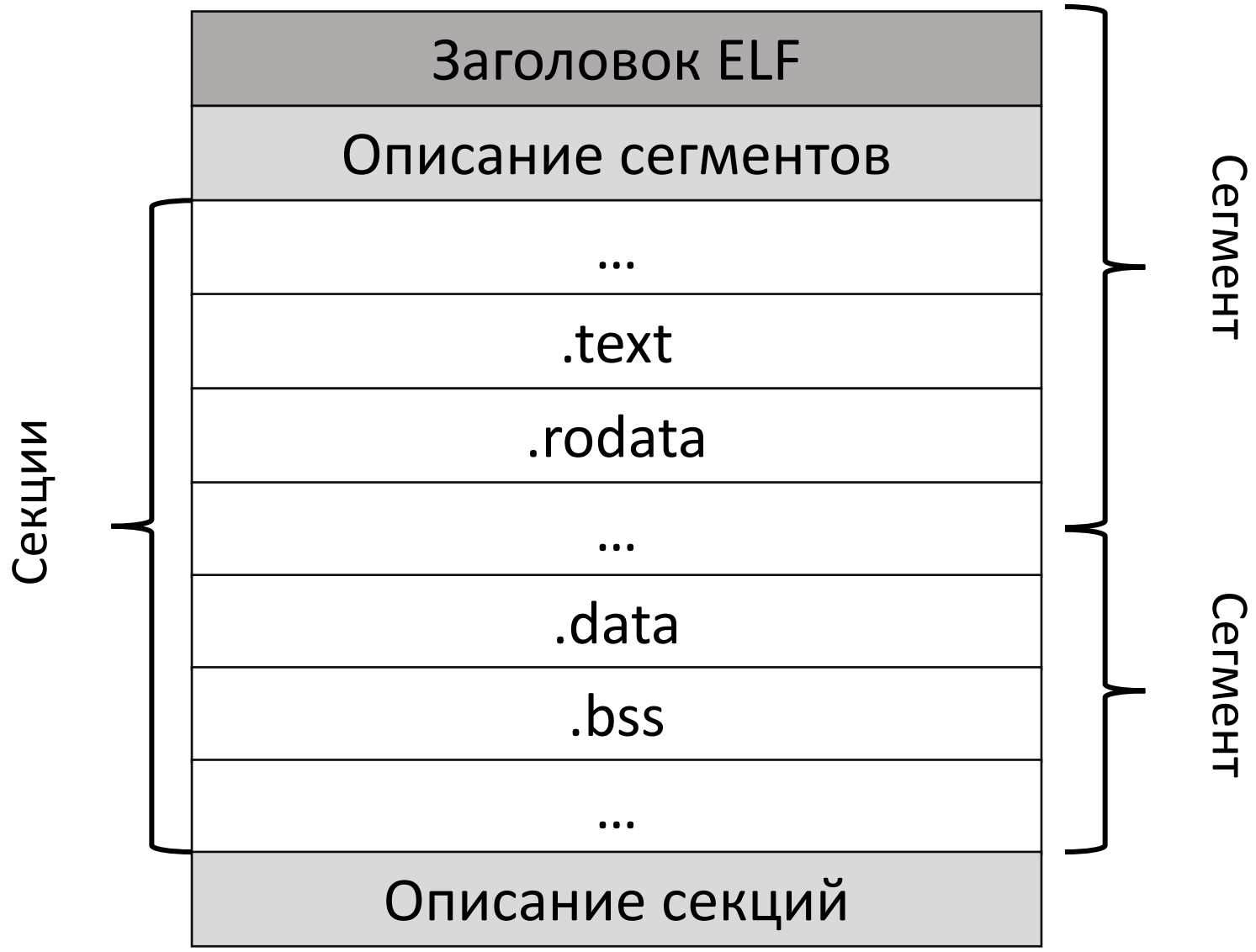
1) Память поделена на **страницы**
2) Преобразование:

Логический адрес -> Линейный адрес -> Физический адрес

Напоминание

Компоновщик

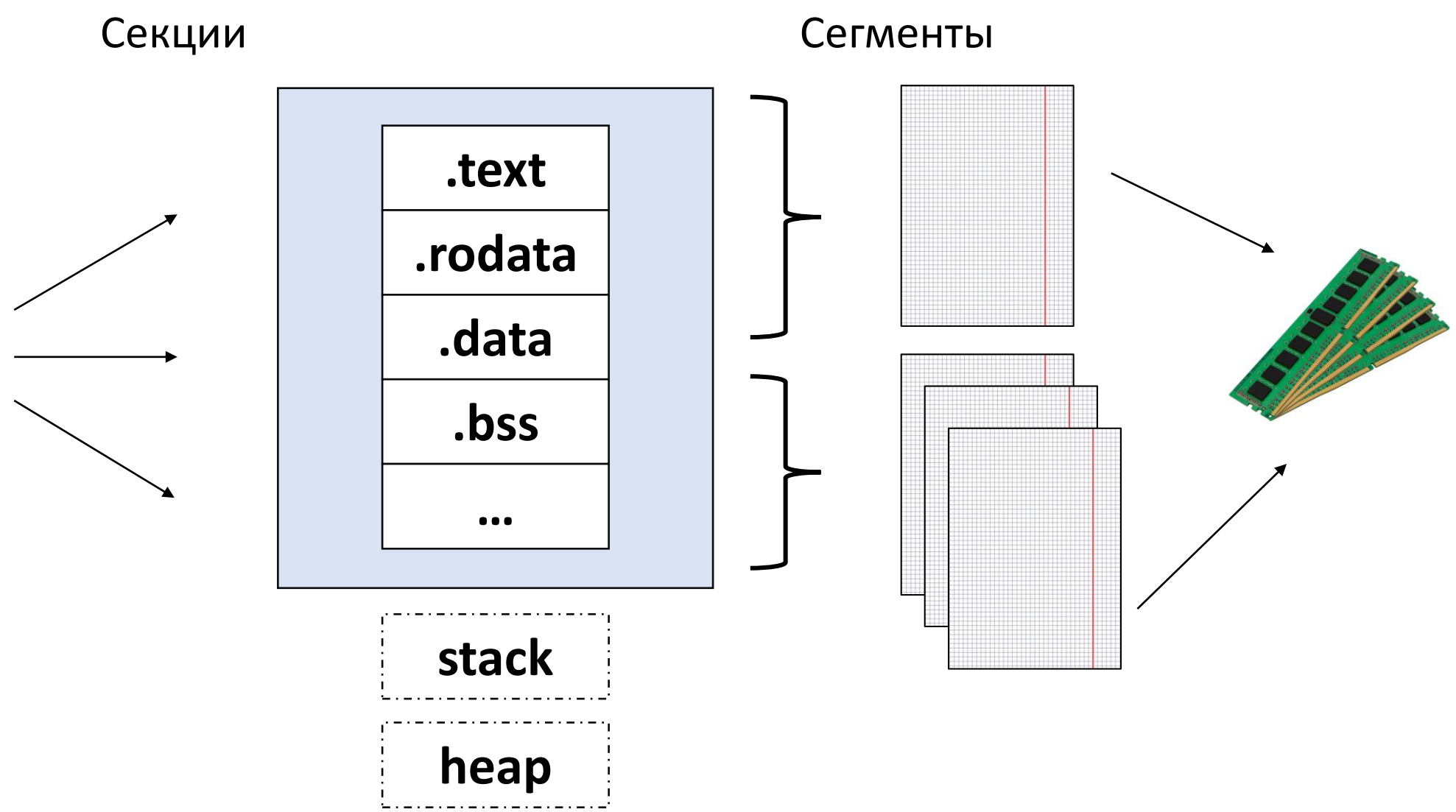
Загрузчик



LOAD

Напоминание

```
int main(void)
{
    ///
    return 0;
}
```



Секции программы

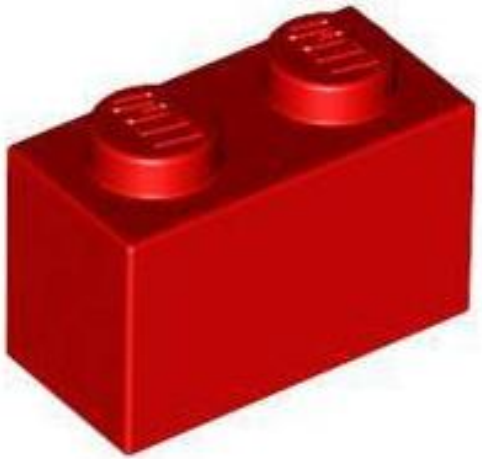
Память

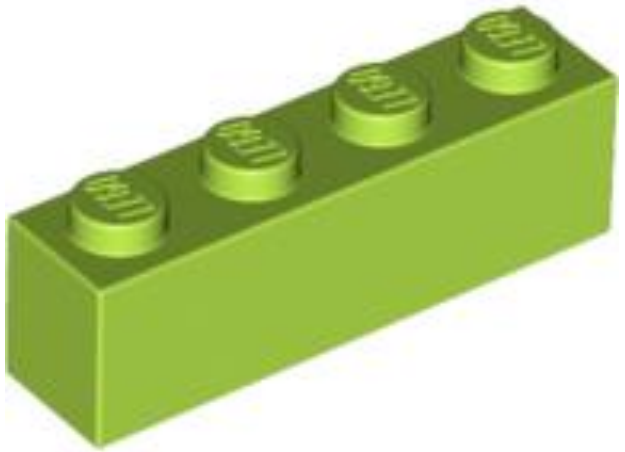

1
Noel
3
4
JOY
5
6
7
8
9
10
11
12
13
15
16
17
Believe
18
19
JOY
20
22
HoHoHo
23
24
25
MERRY CHRISTMAS

Типы данных

Type	Size (Bits)	Range
bit, _Bit	1	0 , 1
bool, _Bool	8	0 , 1
char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127
int	16	-32768 to 32767
short int	16	-32768 to 32767
unsigned int	16	0 to 65535
signed int	16	-32768 to 32767
long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
signed long int	32	-2147483648 to 2147483647
float	32	±1.175e-38 to ±3.402e38
double	32	±1.175e-38 to ±3.402e38

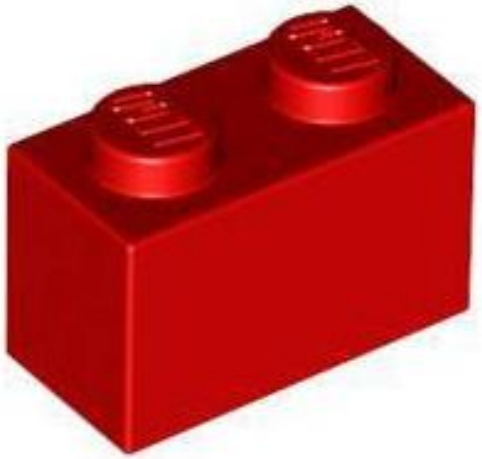
Память



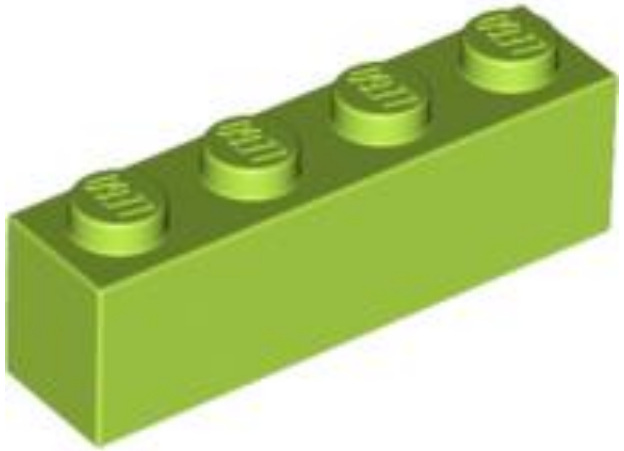


Память

Char



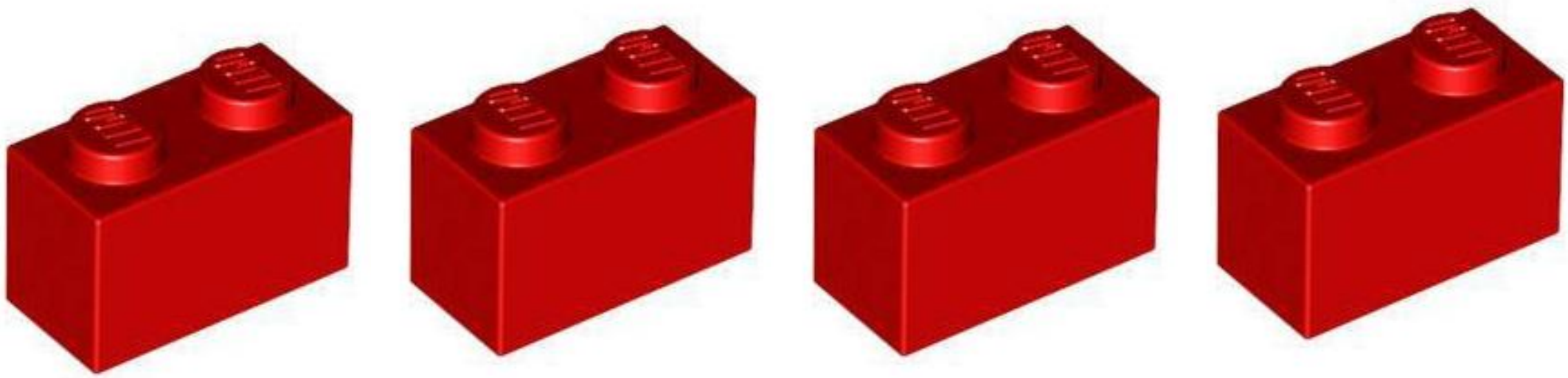
Int



Double

Память

```
int x[4];
```



Тест

Какие из предложенных вариантов обращения к элементу массива верные?
int x[3] = {1,2,3};

- x[1]
- *(&x[1])
- *(x+1)
- 1[x]

Тест

Какие из предложенных вариантов обращения к элементу массива верные?
int x[3] = {1,2,3};

- x[1]
- *(&x[1])
- *(x+1)
- 1[x]

Указатели

```
#include <stdio.h>
int main() {
  int x = 10;
  int *y = &x;
  printf("%d\n", x);
  *y = *y + 1;
  printf("%d\n", x);
  return 0;
}
```

Что напечатает данная программа?

The Ksplice Pointer Challenge

Пусть x лежит по адресу **0x7fffffffdec0**

```
#include <stdio.h>
int main() {
  int x[5];
  printf("%p\n", x);
  printf("%p\n", x+1);
  printf("%p\n", &x);
  printf("%p\n", &x+1);
  return 0;
}
```

Что напечатает данная программа?

The Ksplice Pointer Challenge

Пусть x лежит по адресу **0x7fffffffdec0**

```
#include <stdio.h>
int main() {
  int x[5];
  printf("%p\n", x);
  printf("%p\n", x+1);
  printf("%p\n", &x);
  printf("%p\n", &x+1);
  return 0;
}
```

Запустим пример 2

Что напечатает данная программа?

The Ksplice Pointer Challenge

Пусть x лежит по адресу **0x7fffffffdec0**

```
#include <stdio.h>
int main() {
  int x[5];
  printf("%p\n", x);
  printf("%p\n", x+1);
  printf("%p\n", &x);
  printf("%p\n", &x+1);
  return 0;
}
```

x + sizeof(int)*0

x + sizeof(int)*1

&x[0]

x + sizeof(x)*1

Что напечатает данная программа?

The Ksplice Pointer Challenge

Пусть x лежит по адресу **0x7fffffffdec0**

```
#include <stdio.h>
int main() {
  int x[5];
  printf("%p\n", x);
  printf("%p\n", x+1);
  printf("%p\n", &x);
  printf("%p\n", &x+1);
  return 0;
}
```

```
0x7fffffffdec0
0x7fffffffdec4
0x7fffffffdec0
0x7fffffffded4
```

Пример

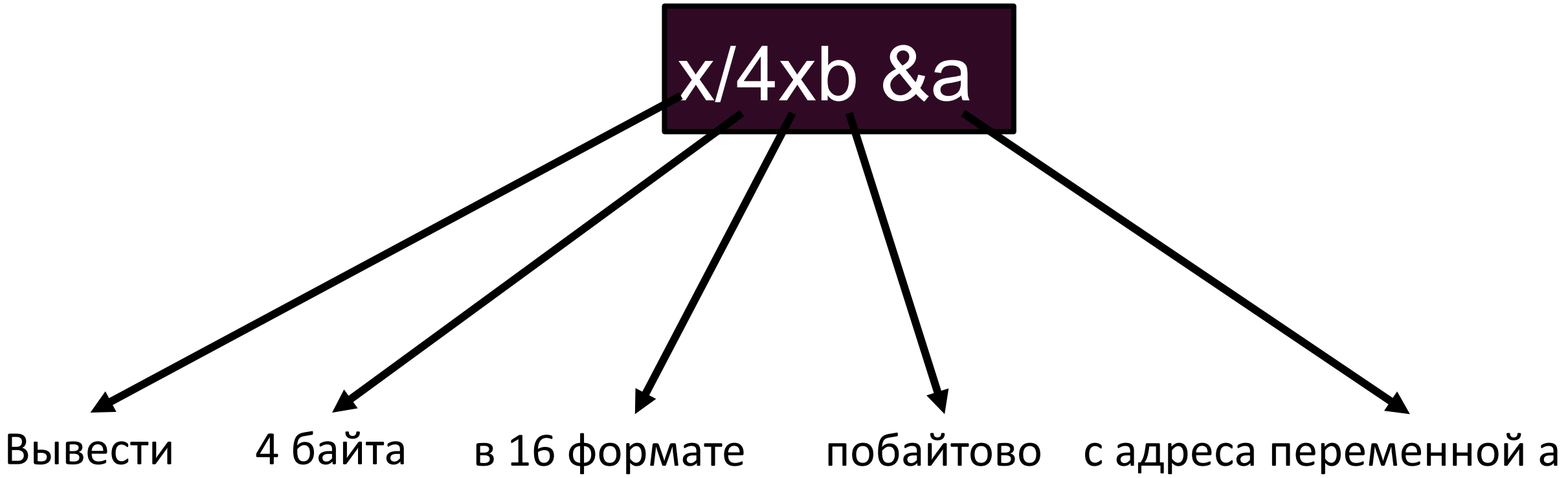
Elective -> lesson4

```
gcc -g program.c -o prog
gdb prog
```

GDB

```
list
break 32
run
info registers
info locals
ptype a
print a
print &a
print sizeof(a)
set var a =  512
print a
```

GDB


x/4xb &a
Вывести
4 байта
в 16 формате
побайтово
с адреса переменной a

GDB

```
int main()
{
    int a = 1024;
    char b = 'b';
    int c[4] = {1,2,3,4};
    int *d = &a;
    int **d1 = &d;
    double e = 3.14;
    char g[4] = {1,2,3,4};
    struct str f;
    f.a = 1;
    f.b = '2';
    f.c = 3;
    union code h;
    h.a = 1;
    h.b = '2';
    h.c = 3;
    return 0;
}
```

x/72xb &b

0x								
0x								
0x								
0x								
0x								
0x								
0x								
0x								

Ответьте на следующие вопросы

Какой тип у переменной c
Какой тип у переменной &c
Какой тип у переменной c[1]
Выполните команды: print sizeof(d) и print sizeof(*d)
Какой размер у структуры? Совпадает ли он с суммой размеров всех переменных?
Какой размер у объединения?
Положите в объединение значение 97 в ячейку h.a и выведите h.b

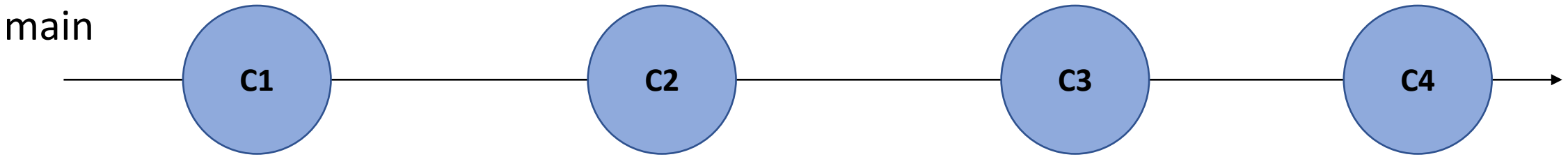
Git

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

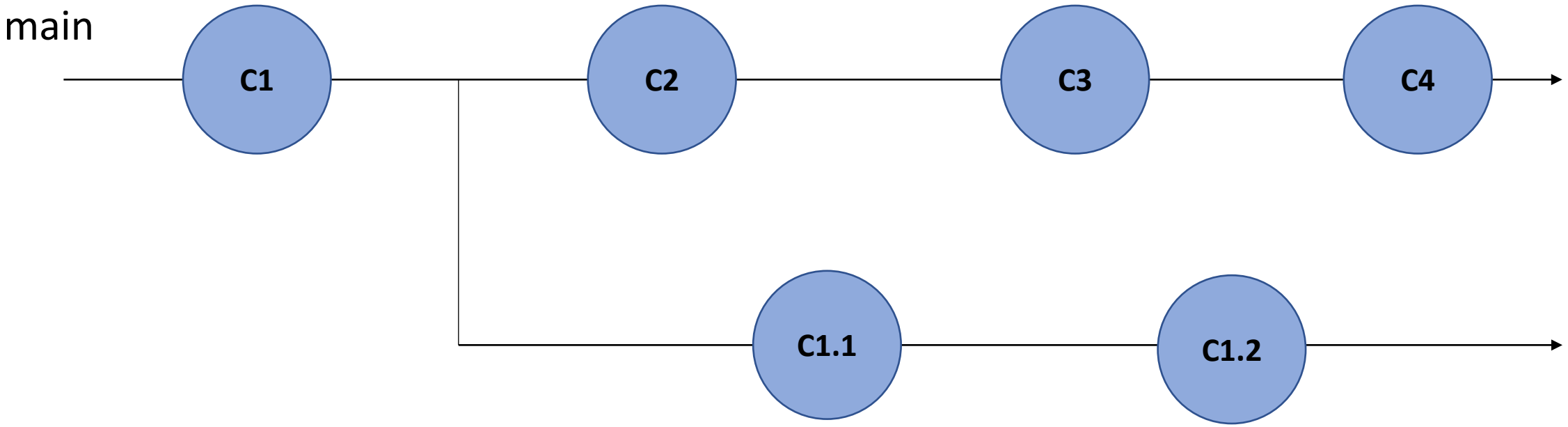
С помощью Git-а вы можете откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий

Репозитории возможно хранить в интернете

Git



Git

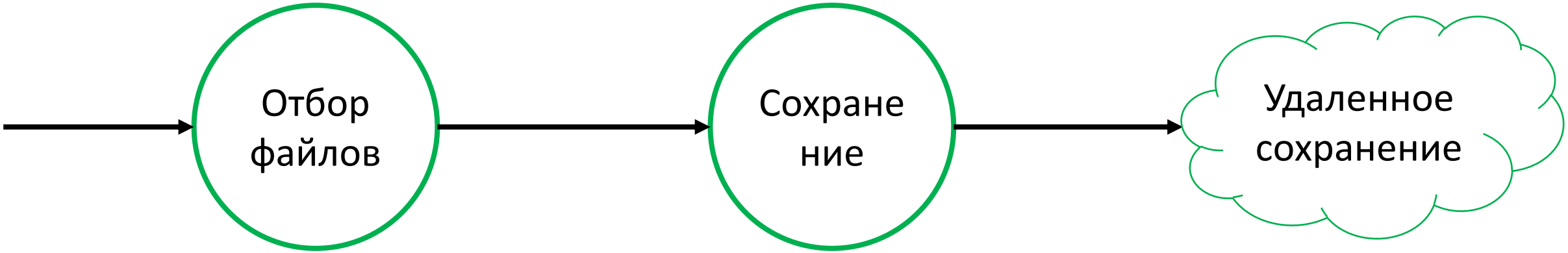


Git

Настройка

```
git init
git config --global user.name "My Name"
git config --global user.email Email@example.com
```

Git




```
git add .
```

```
git commit -m "first commit"
```

```
git push -u origin master
```

Для проверки состояния:

```
git status
```

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
```

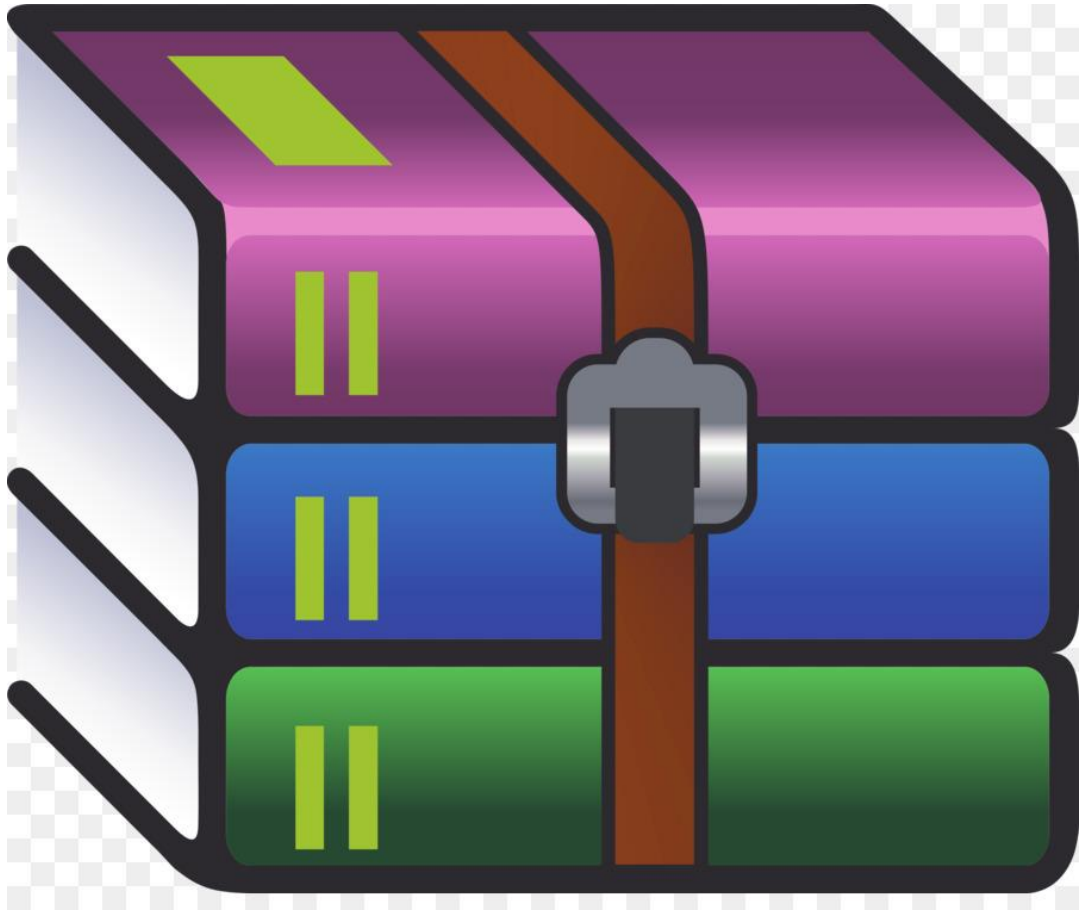
Git

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

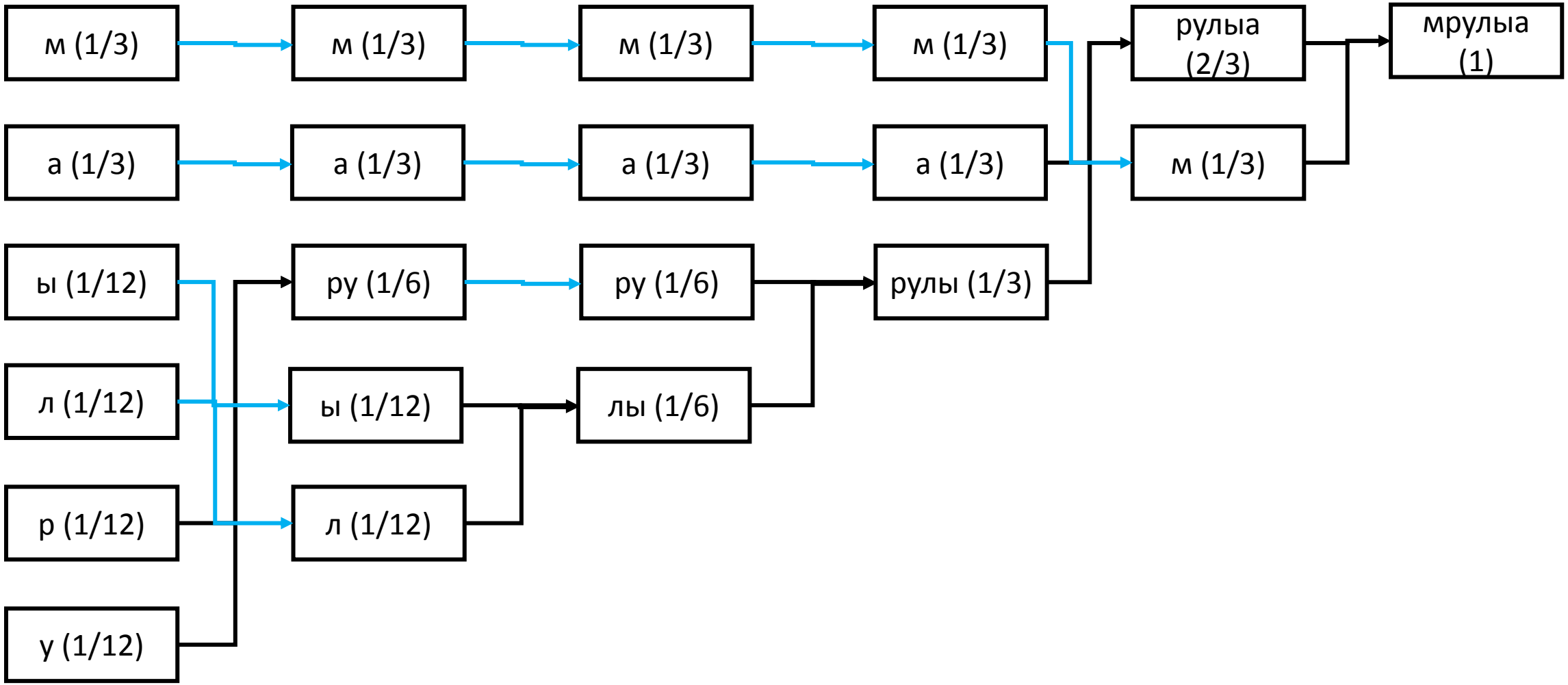
Давайте заменим стандартный равномерный ASCII код на неравномерный так, чтобы часто встречающимся символам соответствовали более короткие кодовые последовательности. Если средние битовые затраты будут меньше, чем 8 бит, то сжатие удалось!

Алгоритм Хаффмана

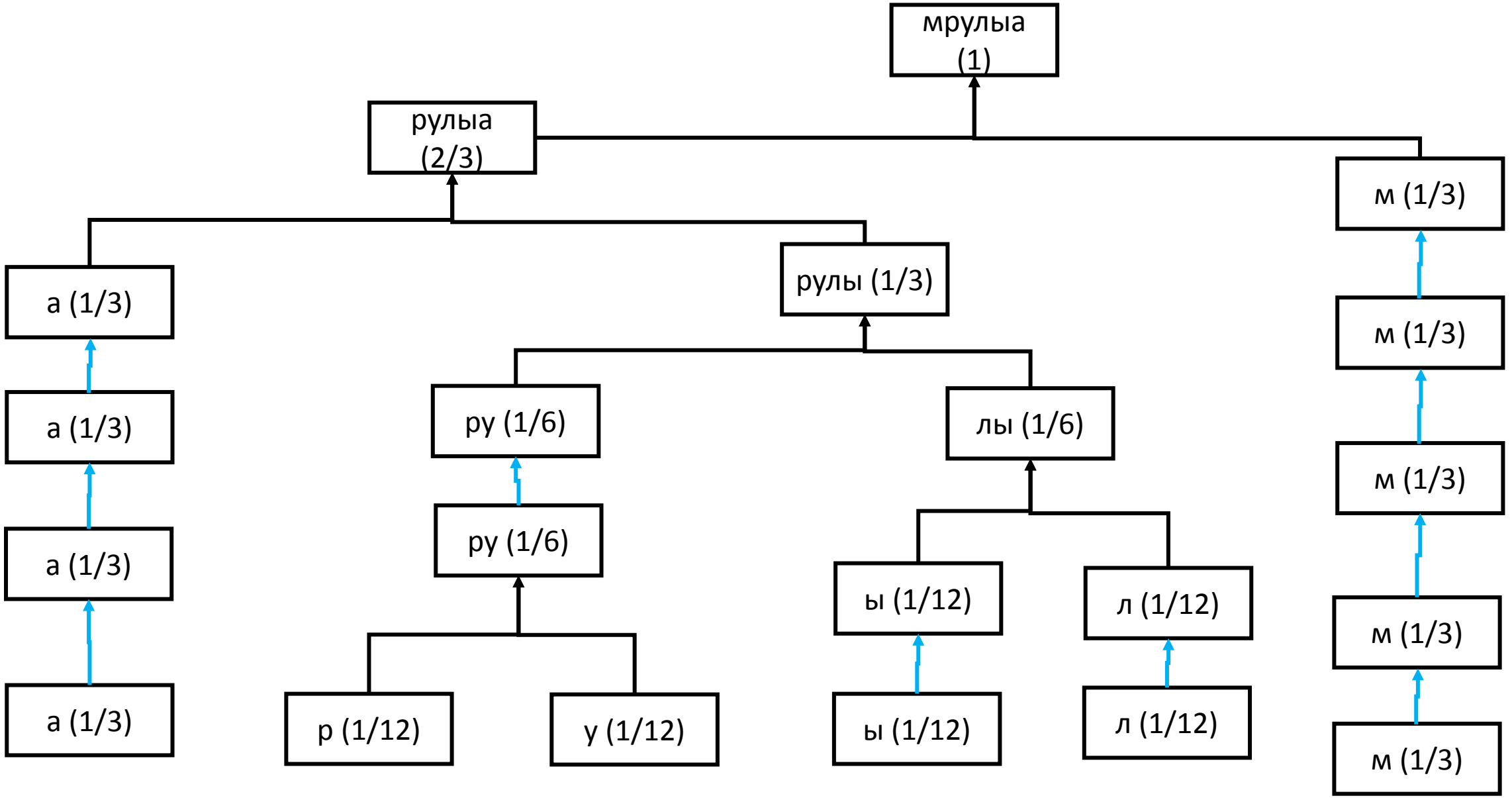
На вход алгоритма подается таблица символов

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

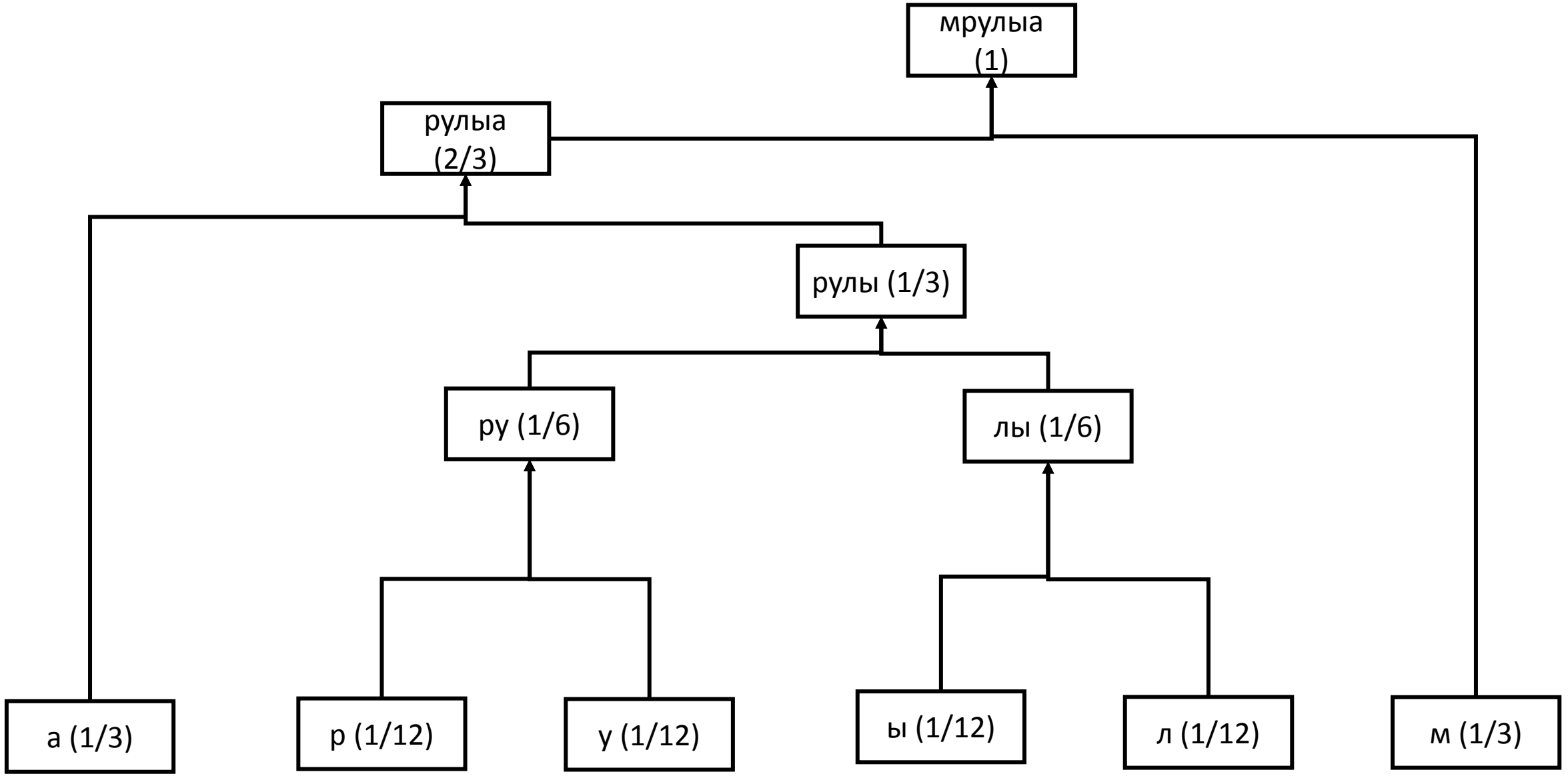
Алгоритм Хаффмана



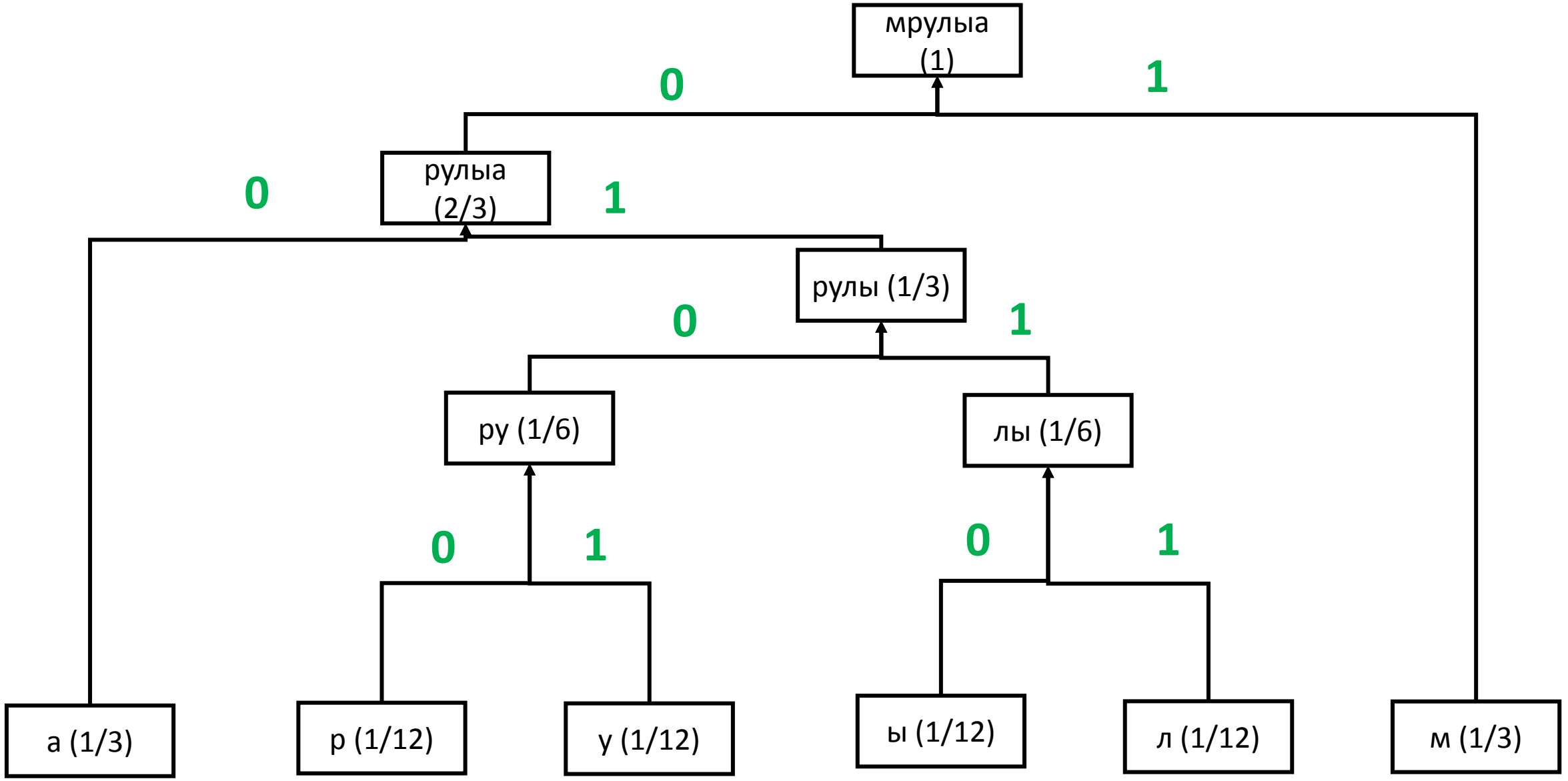
Алгоритм Хаффмана



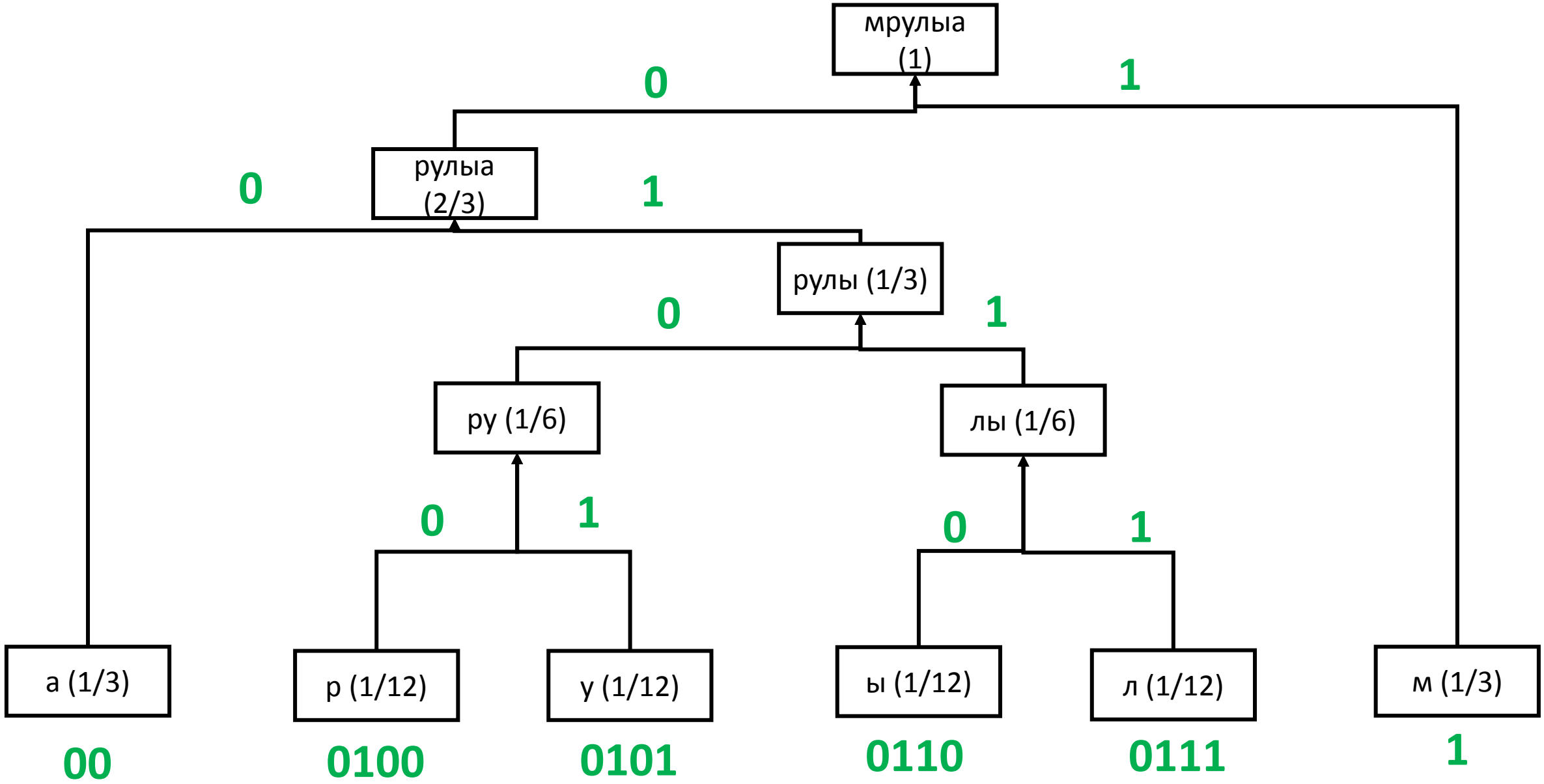
Алгоритм Хаффмана



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!

GDB – возможный вариант решения

```
int main()
{
    int a = 1024;
    char b = 'b';
    int c[4] = {1,2,3,4};
    int *d = &a;
    int **d1 = &d;
    double e = 3.14;
    char g[4] = {1,2,3,4};
    struct str f;
    f.a = 1;
    f.b = '2';
    f.c = 3;
    union code h;
    h.a = 1;
    h.b = '2';
    h.c = 3;
    return 0;
}
```

0x7fffffffde3b:	0x62	0x00	0x04	0x00	0x00	0x3c	0xde	0xff
0x7fffffffde43:	0xff	0xff	0x7f	0x00	0x00	0x00	0x00	0x00
0x7fffffffde4b:	0x00	0x00	0x00	0x08	0x40	0x40	0xde	0xff
0x7fffffffde53:	0xff	0xff	0x7f	0x00	0x00	0x1f	0x85	0xeb
0x7fffffffde5b:	0x51	0xb8	0x1e	0x09	0x40	0x01	0x00	0x00
0x7fffffffde63:	0x00	0x32	0x7f	0x00	0x00	0x00	0x00	0x00
0x7fffffffde6b:	0x00	0x00	0x00	0x08	0x40	0x01	0x00	0x00
0x7fffffffde73:	0x00	0x02	0x00	0x00	0x00	0x03	0x00	0x00
0x7fffffffde7b:	0x00	0x04	0x00	0x00	0x00	0x80	0xdf	0xff