Факультатив по программированию на языке C

Занятие 2
Основы языка C

План занятий

№	Тема	Описание
1	Введение в курс	Языки программирования. Основы работы с Linux.
2	Основы языка С	Написание и компиляция простейших программ с использованием gcc. Правила написания кода. Разбиение программы на отдельные файлы. Make файлы
3	Ввод данных. Библиотеки	Работа со вводом/выводом. Статические и динамические библиотеки. Компиляция.
4	Язык ассемблера	Основы анализа программ на языке ассемблер.
5	Хранение данных. Память	Хранение процесса в памяти компьютера. Виртуальная память, сегментация. Секции программы.
6	Хранение данных.	Стек, куча. Типы данных. Преобразования типов. Gdb и отладка Хранение различных типов данных. Указатели. Передача аргументов в функцию по указателю.
7	Обработка данных	Безопасные функции. Битовые операции – сдвиги, логические операции. Битовые поля.
8	Программирование под встраиваемые ОС	Работа с микрокомпьютером Raspberry Pi

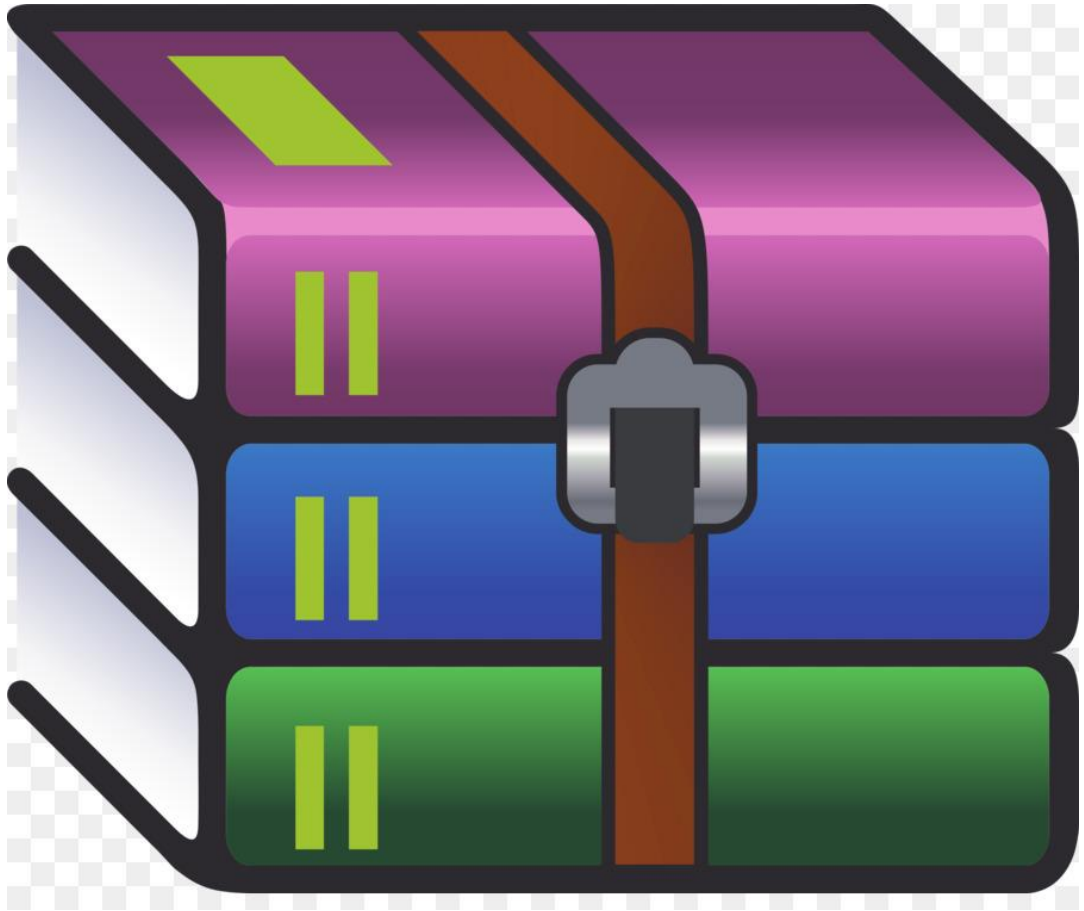
Что мы пройдем сегодня?

1)Основы работы с командной строкой
2)Компиляция программ
3)Разбиение программы на модули
4)Защита от многократного включения .h файлов
5)Чуть-чуть посмотрим ассемблер
6)Создание библиотек
7)Работа с файлами
8)Работа с git
9)Начнем наш проект…

Дерево языка

Дерево языка

Практическая часть



Задание

Реализовать программу, кодирующая и сжимающая файлы по алгоритму Хаффмана

Реализовать программу для декодирования файлов

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**

Размерность – [бит]

Энтропия

Свойства энтропии

1) Энтропия неотрицательна
2) Энтропия нескольких независимых файлов равна сумме энтропий каждого из них

Энтропия

Свойства энтропии

1) Энтропия неотрицательна
2) Энтропия нескольких независимых файлов равна сумме энтропий каждого из них

При каком условии энтропия максимальна?

Энтропия

Свойства энтропии

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

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

Пусть нам пришло следующее сообщение: «мамамылараму» Рассчитаем энтропию сообщения и битовые затраты. Будем считать, что один символ кодируется 1 байтом.

Пример

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

1) Рассчитаем энтропию и битовые затраты

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

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

Пример

1) Рассчитаем энтропию

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

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

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

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

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!

Идея сжатия

Тогда произведем замену!

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**
Код	0	1	00	01	10	11	

Идея сжатия

Тогда произведем замену!

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**
Код	0	1	00	01	10	11	

Сработает ли такой вариант?

Идея сжатия

Тогда произведем замену!

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**
Код	0	1	00	01	10	11	

Наш код: 01010000111010 11

ллыыурру

Сработает ли такой вариант?

Префиксные коды

Введем правило:
Ни один код не может быть началом другого.

Поэтому будем использовать **префиксные коды**

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**
Код	0	10	110	1110	11110	11111	

Префиксные коды

Введем правило:
Ни один код не может быть началом другого.

Поэтому будем использовать **префиксные коды**

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**
Код	0	10	110	1110	11110	11111	

Подобный вариант **избыточен**!

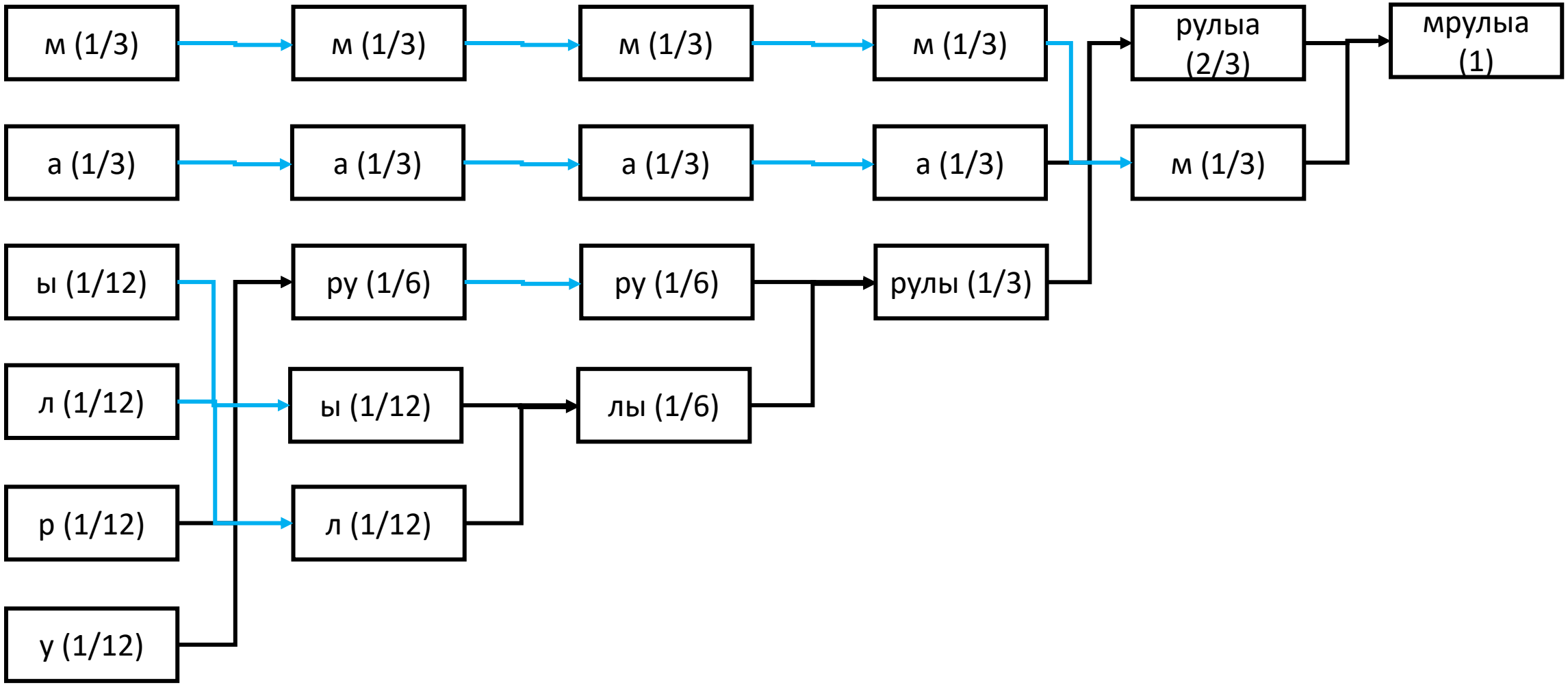
R ~ 2,42

Алгоритм Хаффмана

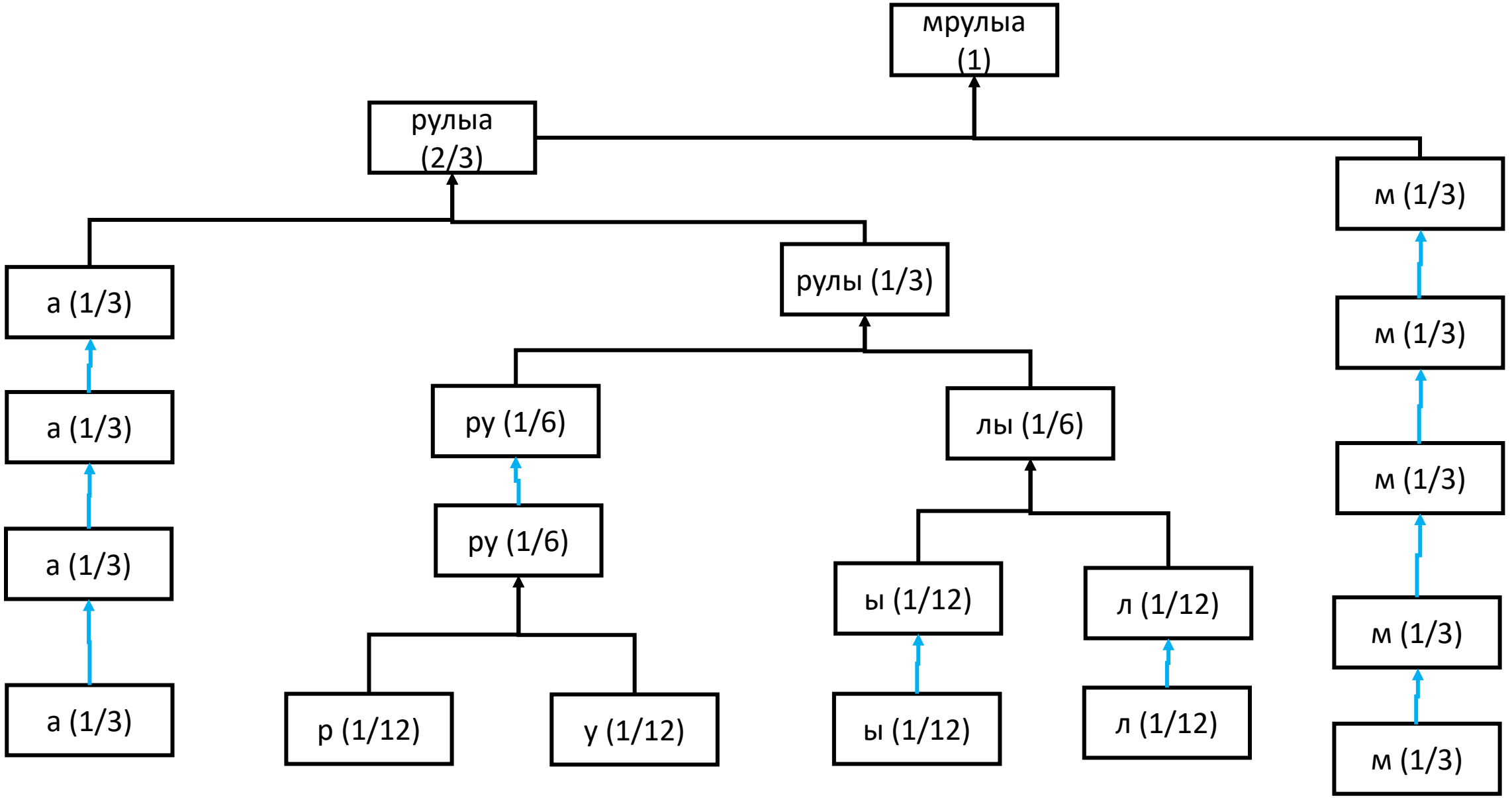
На вход алгоритма подается таблица символов

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

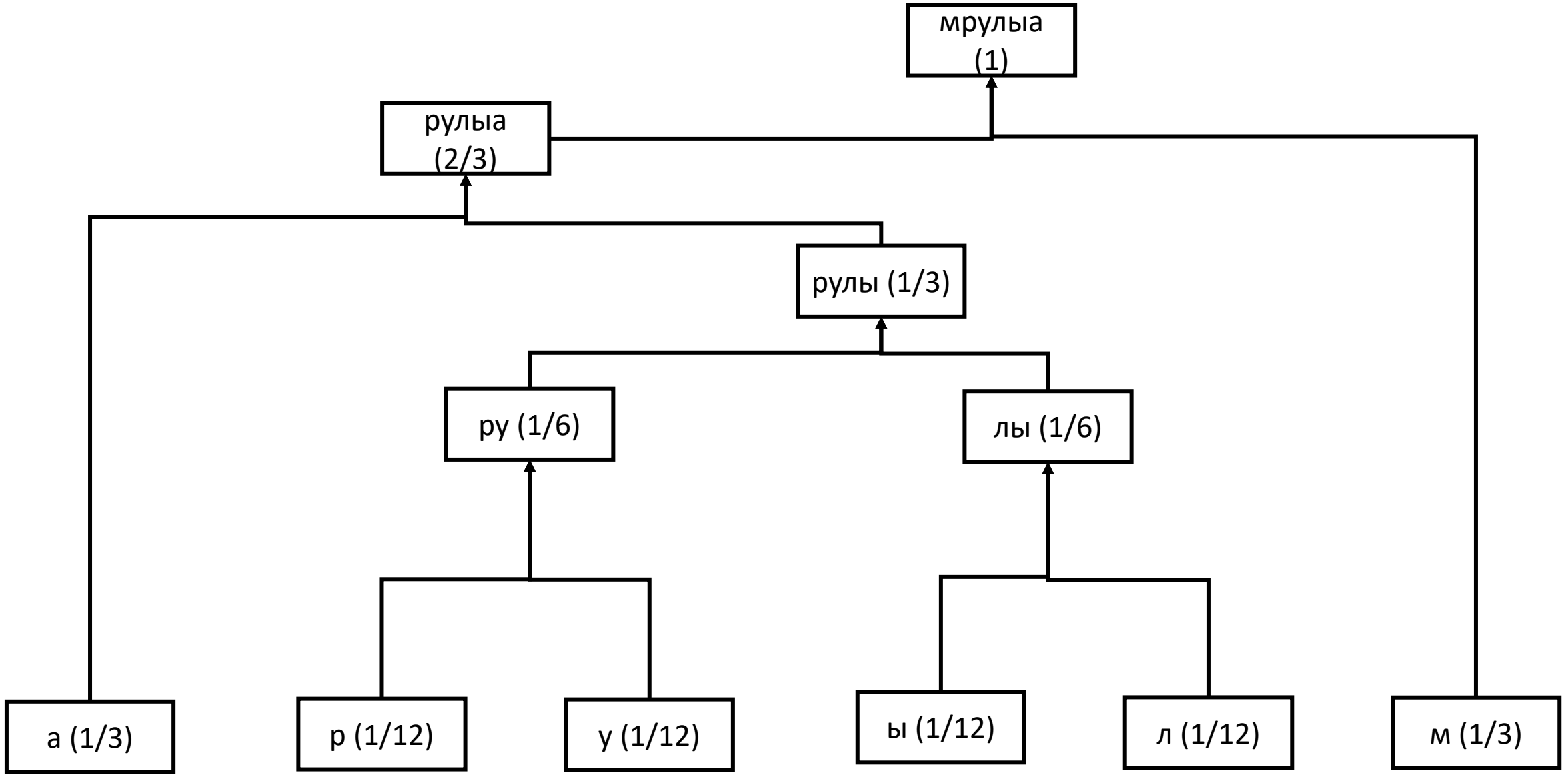
Алгоритм Хаффмана



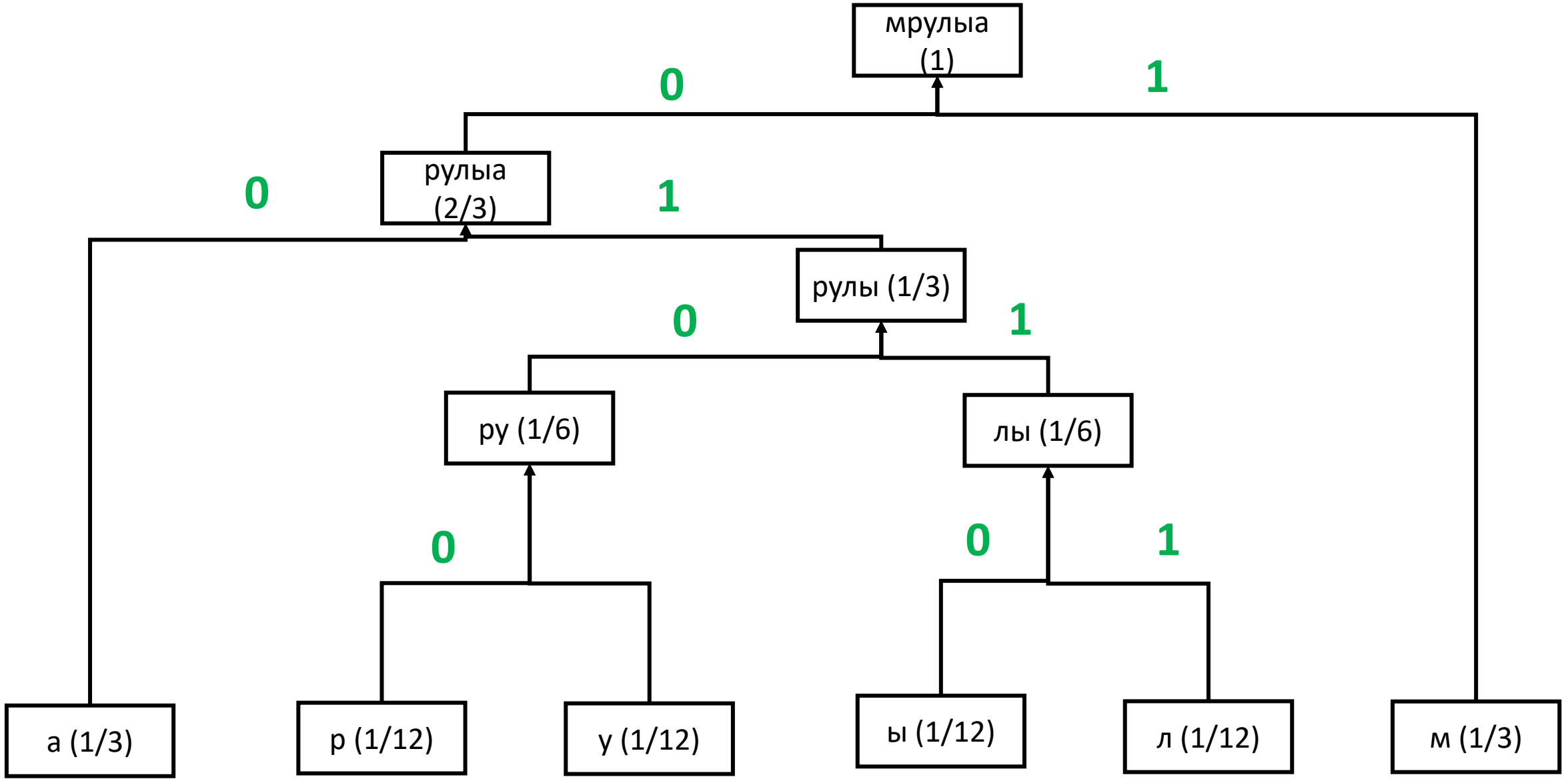
Алгоритм Хаффмана



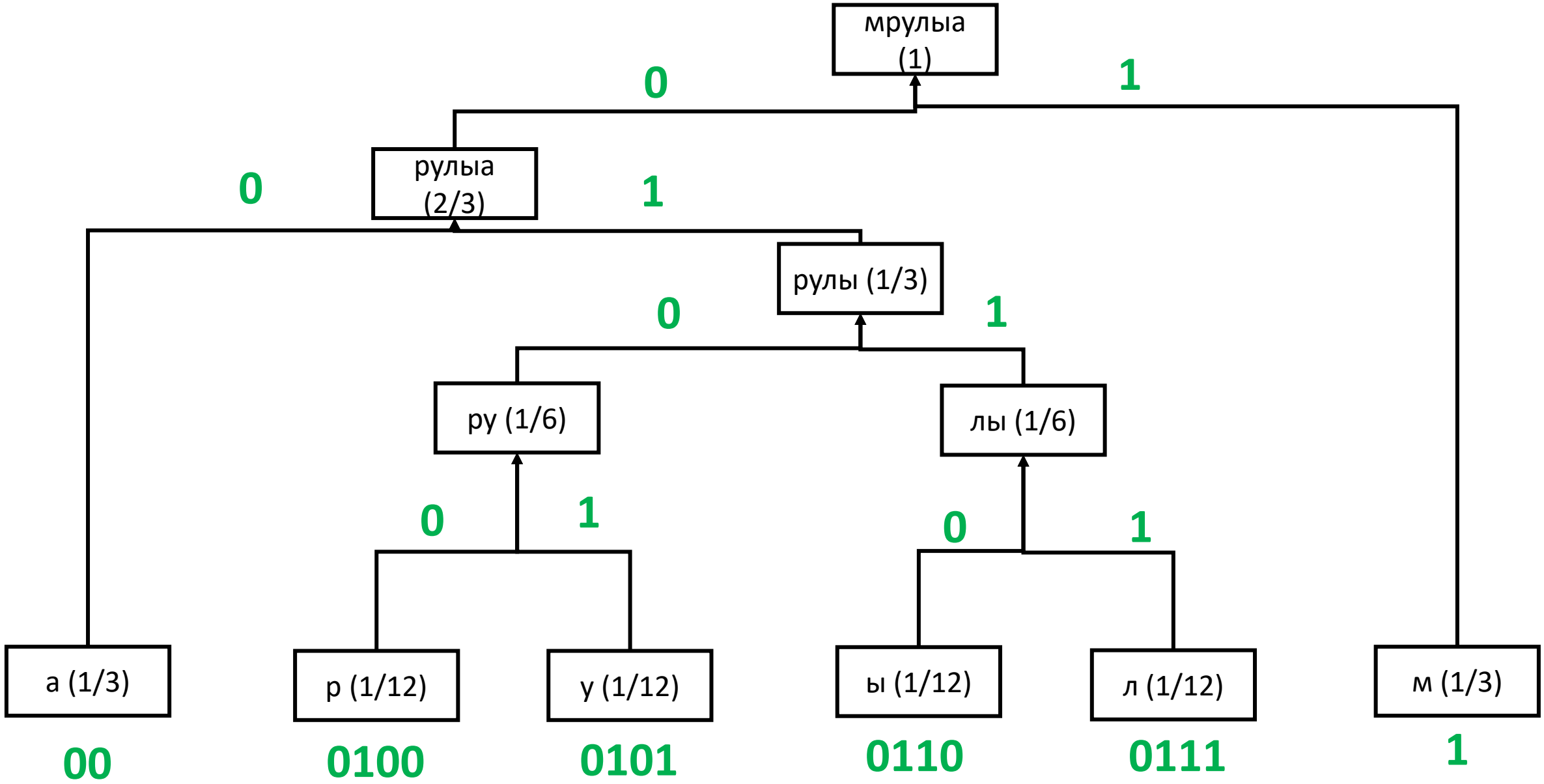
Алгоритм Хаффмана



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

Подробнее можно изучить:

Умняшкин С. В., Основы теории цифровой обработки сигналов, Техносфера, 2019, Глава 5

Работа с командной строкой

```
sab@SAB:.../Lesson1$
```

№	Команда	Описание	Пример
0	man	Описание работы команды	man ls
1	pwd	Показать текущее местонахождение	~/SAB$ pwd /home/user/SAB
2	ls	Позволяет просмотреть содержимое текущего каталога	~/SAB$ ls 1 1.txt
3	cd <путь к директории>	Перейти в другую директорию	~$ cd ~ /SAB/2 (полный путь) или ~/SAB$ cd 2 (короткий путь)
4	mkdir <название директории>	Создание директории	~/SAB$ mkdir 1
5	touch <название файла>	Создание файла	~/SAB$ touch 1.txt
6	nano <название файла>	Редактирование файла	~/SAB$ nano 1.txt
7	cp <что_копировать куда_копировать>	Копирование файла	~/SAB/1$ cp 1.txt ~/SAB/2
8	cp -r <путь_к_папке путь_к_новому_месту>	Копирование директории	~/SAB/1$ cp -r 1 ~/SAB/2
9	mv <что_переместить куда_переместить>	Переместить файл	~/SAB/1$ mv 1.txt ~/SAB/2
10	rm <название файла>	Удалить файл	~/SAB/1$ rm 1.txt
11	rm -r <название файла>	Удалить директорию	~/SAB/1$ rm -r 1

Потренируемся

1. Откройте терминал Linux
2. Создайте директорию. Назовите ее «Task1»
3. Войдите внутрь директории
4. Создайте внутри нее еще две директории – «Task1_1» и «Task1_2»
5. Войдите внутрь директории Task1_1
6. Создайте внутри ее файл с названием «File_1» и еще одну директорию «Task1_1_1»
7. Откройте созданный файл и запишите туда любую информацию. Закройте его
8. Скопируйте «File_1» и «Task1_1_1» в директорию «Task1_2»
9. Опуститесь на уровень ниже и удалите директорию Task1_1
10. Из данной директории переместите «File_1» в директорию «Task1»

Перенаправление ввода и вывода

Ввод и вывод распределяется между тремя стандартными потоками:

- **stdin** — стандартный ввод (клавиатура),- - 0
- **stdout** — стандартный вывод (экран), - 1
- **stderr** — стандартная ошибка (вывод ошибок на экран). - 2

< file — использовать файл как источник данных для стандартного потока ввода.
> file — направить стандартный поток вывода в файл (перезапись)
2> file — направить стандартный поток ошибок в файл (перезапись)
>>file — направить стандартный поток вывода в файл (добавление)
2>>file — направить стандартный поток ошибок в файл. (добавление)
&>file или **>&file** — направить с.п. вывода и с.п. ошибок в файл.

Перенаправление ввода и вывода

< file — использовать файл как источник данных для стандартного потока ввода.
> file — направить стандартный поток вывода в файл (перезапись)
2> file — направить стандартный поток ошибок в файл (перезапись)
>>file — направить стандартный поток вывода в файл (добавление)
2>>file — направить стандартный поток ошибок в файл. (добавление)
&>file или **>&file** — направить с.п. вывода и с.п. ошибок в файл.

```
sab@SAB: /$ ps > 1.txt
sab@SAB: /$ cat 1.txt
sab@SAB: /$ ps >> 1.txt
sab@SAB: /$ ps qq > 1.txt
sab@SAB: /$ ps qq 2> 1.txt
```

Pipes (Каналы)

Каналы используются для перенаправления потока из одной программы в другую.

```
sab@SAB: /$ ps | grep p
sab@SAB: /$ ps > 1.txt; ls >> 1.txt; cat 1.txt | grep a
```

Несколько полезных команд Linux

hexdump — показывает шестнадцатеричное представление данных, поступающих на стандартный поток ввода.
cat — считывает данные со стандартного потока ввода и передает их на стандартный поток вывода.
grep — возвращает только строки, содержащие (или не содержащие) заданное регулярное выражение.
sudo - запуск программы от имени других пользователей, а также от имени суперпользователя.
bc – калькулятор
python3 – среда разработки Python

```
sab@SAB: /$ echo "10*10" | bc
sab@SAB: /$ hexdump 1.txt
```

Основы работы с командной строкой Linux

```
sab@SAB:.../Lesson1$
```

nano Program.c

```c
#include <stdio.h>
int main()
{
        printf("Hello World! \n");
        return 0;
}
```

gcc Program.c –o Program

```
sab@SAB:.../Lesson1$ ./Program
Hello World!
```

Правила написания кода

Правило 1. Имена переменных

Правильно	Неправильно

int enter_range_array()
{
int range_array;
scanf("%d", &range_array);
return range_array;
}
	int vvodrazmeraMassiva()
	{
	int x;
	scanf("%d", &x);
	return x;
	}

Правила написания кода

Правило 2. Фигурные скобки

Правильно	Неправильно

int enter_range_array()
{
int range_array;
scanf("%d", &range_array);
return range_array;
}
	int enter_range_array(){
	int range_array;
	scanf("%d", &range_array);
	return range_array;}

Правила написания кода

Правило 2. Фигурные скобки

Правильно	Неправильно

if (a>0){
b++;
}
	if (a>0)
	{
	b++;
	}

Правила написания кода

Правило 3. Отступы

Правильно	Неправильно

int enter_range_array()
{
int range_array;
scanf("%d", &a);
return a;
}
	int enter_range_array()
	{
	int range_array;
	scanf("%d", &a);
	return a;
	}

if (a>0){
b++;
}
	if (a>0)
	{
	b++;
	}

Спасибо за внимание!