МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курганский государственный университет»

Кафедра программного обеспечения автоматизированных систем

ТЕОРИЯ ИНФОРМАЦИИ

Методические указания
к выполнению практических и контрольных работ
для студентов направлений подготовки 09.03.04, 09.03.03

Курган 2022

Кафедра: «Программное обеспечение автоматизированных систем».

Дисциплина: «Теория информации».

Направления: 09.03.04 «Программная инженерия»,
09.03.03 «Прикладная информатика».

Составил: канд. техн. наук, доцент А. М. Семахин.

Печатается в соответствии с планом издания, утверждённым методическим советом университета «16» декабря 2021 г.

Утверждены на заседании кафедры «03» февраля 2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дисциплина «Теория информации» имеет целью формирование теоретических знаний и практических навыков применения методов получения, преобразования, накопления, отображения и передачи информации.

Предмет дисциплины – технологии хранения, преобразования и передачи информации.

Задачи дисциплины – изучение методов кодирования, передачи по каналам связи, восстановления информации и приобретение практических навыков разработки программных приложений в интегрированной среде программирования Microsoft Visual Studio 2019 Community, отладке и документированию программ.

Методические указания содержат теоретическое обоснование и варианты заданий для выполнения практических работ по дисциплине «Теория информации».

Методические указания разработаны в соответствии с требованиями государственного образовательного стандарта по подготовке бакалавров по направлению 09.03.04 «Программная инженерия», 09.03.03 «Прикладная информатика».

1 КОЛИЧЕСТВО ИНФОРМАЦИИ

1.1 Основные понятия и определения

Теория информации – раздел математики, исследующий процесс хранения, преобразования и передачи информации.

Предмет теории информации – теоремы, устанавливающие предельные возможности методов обработки и передачи сообщений.

Содержание теории информации – исследование методов кодирования для представления сообщений различных источников и надёжной передачи сообщений по каналам связи с шумом.

Количество информации – это числовая характеристика информации, отражающая степень неопределенности, которая исчезает после получения информации.

На *синтаксическом уровне* для оценки количества информации используют вероятностные методы, которые принимают во внимание только вероятностные свойства информации и не учитывают другие (смысловое содержание, полезность, актуальность и т. д.). Разработанные в середине XX в. математические и, в частности, вероятностные методы позволили сформировать подход к

оценке количества информации как к мере уменьшения неопределенности знаний. Такой подход, называемый также вероятностным, постулирует принцип: если некоторое сообщение приводит к уменьшению неопределенности наших знаний, то можно утверждать, что такое сообщение содержит информацию. При этом сообщения содержат информацию о каких-либо событиях, которые могут реализоваться с различными вероятностями. Формулу для определения количества информации для событий с различными вероятностями и получаемых от дискретного источника информации предложил американский ученый К. Шеннон в 1948 г. Согласно этой формуле количество информации может быть определено следующим образом:

$$I = -\sum_{i=1}^{N} p_i \log_2 p_i , \qquad (1.1)$$

где I – количество информации;

N – количество возможных событий (сообщений);

p_i – вероятность отдельных событий (сообщений).

На *семантическом уровне* информация рассматривается с точки зрения смыслового содержания.

На *прагматическом уровне* информация рассматривается с точки зрения ее полезности (ценности) для достижения потребителем информации (человеком) поставленной практической цели. Данный подход при определении полезности информации основан на расчете приращения вероятности достижения цели до и после получения информации [1, 2]. Количество информации, определяющее ее ценность (полезность), находится по формуле:

$$I = \log_2 P_1 - \log_2 P_0 = \log_2 \frac{P_1}{P_0} , \qquad (1.2)$$

где P_0, P_1 – вероятность достижения цели соответственно до и после получения информации.

В качестве единицы измерения (меры) количества информации, определяющей ее ценность, может быть принят 1 бит (при основании логарифма, равном 2), т. е. это такое количество полученной информации, при котором отношение вероятностей достижения цели равно 2.

Рассмотрим три случая, когда количество информации, определяющее ее ценность, равно нулю и когда она принимает положительное и отрицательное значение.

Количество информации равно нулю при $P_0 = P_1$,т. е. полученная информация не увеличивает и не уменьшает вероятность достижения цели.

Значение информации является положительной величиной при $P_1 > P_0$, т. е. полученная информация уменьшает исходную неопределенность и увеличивает вероятность достижения цели.

Значение информации является отрицательной величиной при $P_1 < P_0$, т. е. полученная информация увеличивает исходную неопределенность и уменьшает вероятность достижения цели. Такую информацию называют дезинформацией.

1.2 Практическая работа № 1 «Определение количества информации»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ расчёта количества информации и определения целесообразности информации.

Используемые приемы и технологии: комбинаторные меры, формула Шеннона, целесообразность информации по Харкевичу, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: количество информации, сообщение, априорная неопределённость, апостериорная неопределённость.

1.2.1 Варианты заданий

Номер N фамилии студента по журналу.

Разработать визуальное приложение на языке Visual C++, осуществляющее расчет количества информации с использованием комбинаторных мер, формулы Шеннона и определение целесообразности информации по Харкевичу.

1.2.2 Методические указания

1 Рассчитать количество информации с использованием комбинаторных мер.

Количество информации с использованием комбинаторных мер определяется по формулам:

- сочетания из h элементов по l:

$$Q_h^l = \frac{h!}{l!*(h-l)!} = \frac{h*(h-1)*...*(h-l+1)}{1*2*3*...*l}; \quad (1.3)$$

- сочетания с повторениями:

$$Q_{h\,повт.}^l = \frac{(h+l-1)!}{l!*(h-l)!}; \quad (1.4)$$

- размещения из h элементов по l элементу:

$$Q_h^l = h*(h-1)*(h-2)*...*(h-l+1) = \frac{h!}{(h-l)!}; \quad (1.5)$$

- размещения с повторениями по l из h элементов:

$$Q = (\frac{l}{h})повт = h^l;, \quad (1.6)$$

при условии, что $h = N+2$, $l = 3$, если N – четное и $l = 2$, если N – нечетное, где N – номер фамилии студента по журналу;

- перестановки h элементов:

$$Q = 1*2*3*...*h = h!; \quad (1.7)$$

- перестановки с повторениями элементов:

$$Q = \frac{(\alpha+\beta+...+\gamma)}{\alpha!*\beta!*\gamma!}, \quad (1.8)$$

при условии, что $h = N/2$, $\alpha = 2$, $\beta = 2$, $\gamma = N$.

2 Рассчитать количество информации по Шеннону.

Количество информации для трёх неравновероятных состояний рассчитывается по формуле Шеннона:

$$H = -(p_1 * \log_2 p_1 + p_2 * \log_2 p_2 + p_3 * \log p_3), \quad (1.9)$$

где $p_3 = \frac{1}{N+1}$;

N – номер фамилии студента по журналу;

$$p_1 + p_2 + p_3 = 1. \qquad (1.10)$$

Шаг изменения вероятности Δp принять равным 0,05. Определить информационную емкость по Хартли и соответствующее значение вероятностей, при которых данная емкость достигается. Построить график изменения энтропии H при $p_3 = const$.

3 Рассчитать целесообразность информации по Харкевичу.

Мера Харкевича – мера оценки целесообразности информации, предложенная А. А. Харкевичем. Расчёт целесообразности информации по Харкевичу производится при получении двух дополнительных информаций I_1 и I_2 после передачи ее по направлениям 1–3 и 1–4. Схема передачи информации от источника 1 до цели 2 показана на рисунке 1.

Для расчета использовать следующие формулы:

$$I_{целес.1} = \log_2 P(4-3-2) - \log_2 P(1-4), \qquad (1.11)$$

$$I_{целес.2} = \log_2 P(3-2) - \log_2 P(1-3), \qquad (1.12)$$

где $P(1-4)$, $P(1-3)$ – вероятности достижения цели до получения информации;

$P(4-3-2)$, $P(3-2)$ – вероятности достижения цели после получения и использования информации.

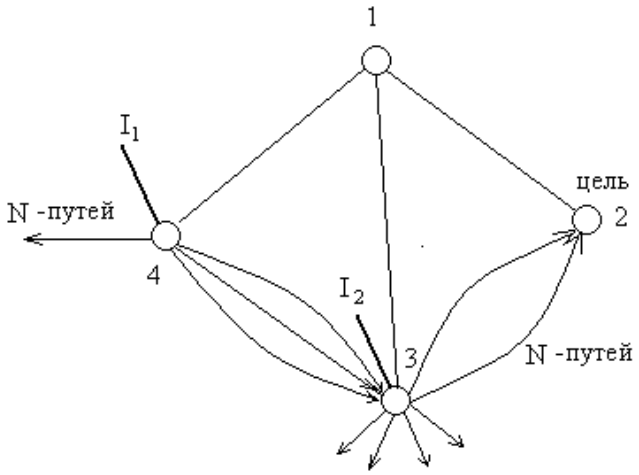

Рисунок 1 – Схема передачи информации от источника до цели

4 Оформить отчет по практической работе.

5 Ответить на контрольные вопросы.

1.2.3 Контрольные вопросы

1 Что называется количеством информации?

2 Каким образом рассчитывается количество информации при равновероятном появлении символов в сообщении?

3 Каким образом рассчитывается количество информации при разновероятном появлении символов в сообщении?

4 Каким образом рассчитывается количество информации с использованием комбинаторных мер?

5 Каким образом рассчитывается целесообразность информации по Харкевичу?

2 ВИДЫ ИНФОРМЦИИ

Количество информации, получаемое в результате эксперимента, определяется как мера уменьшения неопределённости:

$$I = H - H_0, \tag{2.1}$$

где H – неопределённость (энтропия) до проведения эксперимента;

H_0 – неопределённость после проведения эксперимента (остаточная).

2.1 Виды информации для дискретных случайных величин

Для дискретных случайных величин различают четыре вида информации.

Полная информация $I(X)$ о величине X при непосредственном её наблюдении определяется по формуле:

$$I(X) = H(X) = M[-\log_2 P(X)] \geq 0 \tag{2.2}$$

Полное количество информации $I(Y->X)$ о величине X, содержащееся в величине Y:

$$I(Y->X) = H(X) - H(X/Y); \tag{2.3}$$

$$I(Y->X) = I(X->Y) = I(Y \leftrightarrow X) \geq 0, \tag{2.4}$$

где $I(Y \leftrightarrow X)$ – полная взаимная информация, содержащаяся в X и Y;

$$I(X \leftrightarrow Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \log_2 \frac{P(x_i, y_j)}{P(x_i) P(y_j)}. \quad (2.5)$$

Частная информация $I(y_i \to X) \geq 0$ о величине X, содержащаяся в значении y_i величины Y:

$$I(y_i \to X) = \sum_{i=1}^{n} P(x_i / y_j) \log_2 \frac{P(x_i, y_j)}{P(x_i)}. \quad (2.6)$$

Частная информация $I(y_i \to x_i)$ о значении x_i величины X, содержащаяся в значении y_j величины Y определяется по формуле:

$$I(y_i \to x_i) = I(y_j \leftrightarrow x_i), \quad (2.7)$$

где $I(y_i \to x_i)$ – частная взаимная информация двух значений;

$$I(y_i \to x_i) = \log_2 \frac{P(x_i, y_j)}{P(x_i) P(y_j)} = \log_2 \frac{P(x_i / y_j)}{P(x_i)} = \log_2 \frac{P(y_j / x_i)}{P(y_j)}. \quad (2.8)$$

2.2 Виды информации для непрерывных случайных величин

Для непрерывных случайных величин различают виды информации.

Частная взаимная информация двух значений x *и* y определяется по формуле:

$$I(y \leftrightarrow x) = \log_2 \frac{P(x, y)}{P(x) P(y)}. \quad (2.9)$$

Частная информация о величине X, *содержащаяся в значении* y *величины* Y определяется по формуле:

$$I(y \to x) = \int_{-\infty}^{+\infty} p(x / y) \log_2 \frac{P(x, y)}{P(x) P(y)} dx. \quad (2.10)$$

Полная взаимная информация, содержащаяся в значении X *и* Y определяется по формуле:

$$I(X \leftrightarrow Y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p(x / y) \log_2 \frac{P(x, y)}{P(x) P(y)} dx dy. \quad (2.11)$$

2.3 Практическая работа № 2
«Оценка количества информации»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ расчёта полной информации, взаимной частной информации, условной частной информации, среднего количества информации, среднего количества взаимной информации в сообщениях.

Используемые приемы и технологии: методы расчёта видов информации, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: полная информация, взаимная частная информация, условная частная информация, среднее количество информации, среднее количество взаимной информации.

2.3.1 Варианты заданий

Варианты заданий приведены в таблице 2.1.

По двоичному симметричному каналу связи с помехами передаются два сигнала x_1 и x_2 с априорными вероятностями $p(x_1)=\frac{3}{4}$ и $p(x_2)=\frac{1}{4}$. Из-за наличия помех вероятность правильного приёма каждого из сигналов уменьшается до $p(y_1/x_1)=\frac{7}{8}$ и $p(y_2/x_2)=\frac{7}{8}$. Необходимо определить:

1) полную информацию $I(x)$ на выходе источника сигналов;

2) взаимную частную информацию $I(y_2,x_2)$ двух случайных сигналов на выходе и входе канала связи относительно друг друга (количество информации о сигнале x_2 источника в сообщении y_2 приёмника);

3) условную частную информацию $I(x_2/y_2)$, содержащуюся в сообщении x_2 источника при условии приёма сообщения y_2;

4) среднее количество информации $I(y_2,X)$ в принятом сообщении y_2 относительно всех передаваемых сообщений $X(x_1,x_3)$;

5) среднее количество взаимной информации $I(Y,X)$ в сообщениях Y приёмника о сообщениях X источника.

Таблица 2.1 – Исходные данные вариантов заданий

Вариант	$p(x_1)$	$p(x_2)$	$p(y_1 / x_1)$	$p(y_2 / x_2)$
1	3/4	1/4	7/8	7/8
2	2/3	1/3	5/7	5/7
3	4/5	1/5	8/9	8/9
4	5/6	1/6	7/8	7/8
5	6/7	1/6	8/9	8/9
6	2/3	1/3	7/8	7/8
7	4/5	1/5	8/9	8/9
8	3/4	1/3	5/7	5/7
9	5/6	1/6	8/9	8/9
10	4/5	1/5	5/7	5/7

2.3.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее оценку количества информации:

- расчёт полной информации $I(x)$ на выходе источника сигналов;
- расчёт взаимной частной информации $I(y_2, x_2)$ двух случайных сигналов на выходе и входе канала связи относительно друг друга (количество информации о сигнале x_2 источника в сообщении y_2 приёмника);
- расчёт условной частной информации $I(x_2 / y_2)$, содержащейся в сообщении x_2 источника при условии приёма сообщения y_2;
- расчёт среднего количества информации $I(y_2, X)$ в принятом сообщении x_2 относительно всех передаваемых сообщений $X(x_1, x_3)$;
- расчёт среднего количества взаимной информации $I(Y, X)$ в сообщениях Y приёмника о сообщениях X источника.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

2.3.3 Контрольные вопросы

1 Каким образом определяется количество информации, получаемое в результате эксперимента?

2 Какие виды информации для дискретной случайной величины?

3 Какие виды информации для непрерывной случайной величины?

4 Каким образом определяется полная информация $I(X)$ о величине X при непосредственном её наблюдении?

5 Каким образом определяется частная информация $I(y_i \to x_i)$ о значении x_i величины X, содержащаяся в значении y_j величины Y?

3 ВИДЫ ЭНТРОПИИ

Энтропи́я (информационная) – мера хаотичности информации, неопределённость появления какого-либо символа первичного алфавита. При отсутствии информационных потерь численно равна количеству информации на символ передаваемого сообщения.

3.1 Виды энтропии для дискретных случайных величин

Энтропия дискретной случайной величины определяется по формуле:

$$H(X) = \sum_{i=1}^{N} p(x_i) \log_2 \frac{1}{p(x_i)} = -\sum_{i=1}^{N} p(x_i) \log_2 p(x_i), \qquad (3.1)$$

где $p(x_i)$ – вероятность появления i-ого значения x_i случайной величины X;

$\log_2 \frac{1}{p(x_i)} = H_i$ – мера неопределённости i-ого значения.

Энтропия максимальна, когда значения случайной величины равновероятны:

$$p(x_1) = p(x_2) = ... = p(x_n) = \frac{1}{N} \qquad (3.2)$$

Тогда

$$H(X) = -\sum_{i=1}^{N} \frac{1}{N} \log_2 \frac{1}{N} = \log_2 N, \qquad (3.3)$$

где $\log_2 N$ – мера Хартли.

3.2 Виды энтропии для непрерывных случайных величин

Энтропия непрерывной случайной величины определяется по формуле:

$$H(X) = -\int_{-\infty}^{+\infty} p(x_i) \log_2 p(x) dx - \log_2 \Delta x, \qquad (3.4)$$

где $p(x)$ – плотность вероятности случайной величины X;

Δx – шаг квантования, определяющий точность перехода от непрерывной величины к дискретной.

3.3 Практическая работа № 3 «Оценка энтропийных характеристик»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ расчёта оценок энтропийных характеристик.

Используемые приемы и технологии: вероятностная мера неопределённости Шеннона, энтропия непрерывной случайной величины, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: энтропия, безусловная энтропия, условная энтропия, дифференциальная энтропия, условная дифференциальная энтропия.

3.3.1 Варианты заданий

Три дискретных источника информации $X(x_1, x_2, x_3)$, $Y(y_1, y_2)$, $Z(z_1, z_2)$. Вероятности появления сообщений $P(x_i)$, $P(y_i)$, $P(z_i)$ заданы векторами $P_x = (P_{x_1}, P_{x_2}, P_{x_3})^T$, $P_y = (P_{x_1}, P_{x_2} + P_{x_3})^T$, $P_z = (\frac{P_{x_2}}{P_{x_2} + P_{x_3}}, \frac{P_{x_3}}{P_{x_2} + P_{x_3}})^T$. Необходимо определить среднюю неопределённость каждого источника и установить связь между энтропиями.

Варианты заданий приведены в таблице 3.1.

Таблица 3.1 – Варианты исходных данных

Вариант	P_{x1}	P_{x2}	P_{x3}
1	3/15	4/15	8/15
2	4/16	5/16	7/16
3	5/17	3/17	9/17
4	6/18	4/18/	8/18
5	5/19	6/19	8/19
6	7/20	8/20	5/20
7	8/21	9/21	4/21
8	6/22	8/22	8/22
9	9/23	8/23	5/23
10	11/24	5/24	8/24

3.3.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее оценку энтропийных характеристик:

- расчёт энтропии для дискретной случайной величины;
- расчёт энтропии для непрерывной случайной величины.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

3.3.3 Контрольные вопросы

1 Что называется энтропией?

2 Какие виды энтропии различают для дискретной случайной величины?

3 Какие виды энтропии различают для непрерывной случайной величины?

4 Каким образом рассчитывается вероятностная мера неопределённости Шеннона?

5 Каким образом рассчитывается относительная энтропия?

4 ИНФОРМАЦИОННЫЕ ХАРАКТЕРИСТИКИ СИСТЕМ

4.1 Основные сведения

Избыточность источника оценивается коэффициентом избыточности по формуле:

$$R = 1 - \frac{H(X)}{H_{\max}} = 1 - \frac{I(X)}{I_{\max}}. \qquad (4.1)$$

Производительность источника – количество информации в сообщении за одну секунду

$$\bar{I}(X) = \frac{I(X)}{\bar{\tau}_X} = \nu_X I(X), \qquad (4.2)$$

где $\bar{\tau}_X$ – средняя длительность одного сообщения;

ν_X – средняя скорость создания сообщения источником.

Средняя скорость создания сообщения источником определяется по формуле

$$\nu_x = \frac{1}{\tau_x} \tag{4.3}$$

Скорость передачи информации по каналу – количество информации на выходе за одну секунду.

Средняя скорость передачи сообщения по каналу определяется по формуле:

$$\nu_k = \frac{1}{\tau_k}, \tag{4.4}$$

где τ_k – средняя длительность сообщения в канале связи.

Пропускная способность – максимально возможная скорость передачи информации по каналу связи. Пропускная способность дискретного канала без помех определяется по формуле:

$$C = \nu_k \log_2 N . \tag{4.5}$$

4.2 Практическая работа № 4
«Оценка информационных характеристик систем»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ расчёта оценок информационных характеристик систем.

Используемые приемы и технологии: скорость передачи информации, пропускная способность канала связи, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: производительность источника, избыточность источника, скорость передачи информации, пропускная способность канала связи.

4.2.1 Варианты заданий

По двоичному симметричному каналу связи с помехами передаются два сигнала x_1 и x_2 с априорными вероятностями $p(x_1)$ и $p(x_2)$. Из-за наличия помех вероятность правильного приёма сигнала уменьшается до $p(y_1 / x_1)$ и $p(y_2 / x_2)$. Длительность каждого сигнала $\tau = 0{,}1$ с. Необходимо определить:

1) производительность и избыточность источника;

2) скорость передачи информации и пропускную способность канала.

Исходные данные вариантов заданий приводятся в таблице 4.1.

Таблица 4.1 – Исходные данные вариантов заданий

Вариант	$p(x_1)$	$p(x_2)$	$p(y_1 / x_1)$	$p(y_2 / x_2)$
1	2/3	1/3	7/8	7/8
2	4/5	1/5	8/9	8/9
3	3/4	1/3	5/7	5/7
4	5/6	1/6	8/9	8/9
5	4/5	1/5	5/7	5/7
6	3/4	1/4	7/8	7/8
7	2/3	1/3	5/7	5/7
8	4/5	1/5	8/9	8/9
9	5/6	1/6	7/8	7/8
10	6/7	1/6	8/9	8/9

4.2.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее оценку информационных характеристик систем:

- расчёт производительности и избыточности источника;
- расчёт скорости передачи информации и пропускной способности канала связи.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

4.2.3 Контрольные вопросы

1 Назовите существуют информационные характеристики системы.

2 Что такое пропускная способность канала?

3 Как рассчитывается коэффициент избыточности?

4 Как рассчитывается производительность источника?

5 Как рассчитывается скорость передачи информации по каналу?

5 ПРЕОБРАЗОВАНИЕ СИГНАЛОВ

Существуют две формы представления информации:

– непрерывная (аналоговая);

– прерывистая (цифровая, дискретная).

Непрерывная форма характеризует процесс, не имеющий перерывов и изменяющийся в любой момент времени и на любую величину.

Цифровой сигнал изменяется в определённые моменты времени и принимает заранее обусловленные значения.

Для преобразования аналогового сигнала в цифровой необходимо провести дискретизацию непрерывного сигнала во времени, квантование по уровню и кодирование отобранных значений.

5.1 Теорема Котельникова

Теорема была сформулирована В. А. Котельниковым в 1933 году в его работе «О пропускной способности эфира и проволоки в электросвязи». Она звучит следующим образом: если аналоговый сигнал *x(t)* имеет спектр, ограниченный частотой F_{max}, то он может быть восстановлен с какой угодно точностью по своим дискретным отсчётам, взятым с частотой

$$f_{дискр} \geq 2F_{\max}, \tag{5.1}$$

где F_{max} – верхняя частота в спектре,

или (формулируя по-другому) по отсчётам, взятым с периодом:

$$T_{дискр} \leq \frac{1}{2F_{\max}}. \tag{5.2}$$

Для дискретизации аналогового сигнала без потери информации частота отсчетов должна быть как минимум в два раза выше верхней граничной частоты спектра сигнала.

Теорема Котельникова говорит о том, что непрерывный сигнал можно представить в виде следующего ряда:

$$\sum x(k\Delta t)\frac{\sin\omega(t-k\Delta t)}{\omega(t-k\Delta t)}. \tag{5.3}$$

Под интегральной суммой написана формула отсчётов фунцкии *x(t)*. Мгновенные значения этой функции есть значения дискретизированного сигнала в каждый из моментов времени.

Также имеет место следующее утверждение: если максимальная частота в сигнале превышает половину частоты прерывания, то способа восстановить сигнал из дискретного в аналоговый без искажений не существует [3, 4].

5.2 Практическая работа № 5
«Дискретное преобразование сигналов»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ в дискретном преобразовании сигналов.

Используемые приемы и технологии: теорема Котельникова, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: дискретизация, квантование, теорема Котельникова.

5.2.1 Варианты заданий

Используя теорему Котельникова, произвести дискретное преобразование и восстановление аналогового сигнала, аналитическое выражение которого имеет следующий вид:

$$u(t) = U_{m1}\cos(w_1 t + \varphi_{01}) + U_{m2}\cos(w_2 t + \varphi_{0.2}), \qquad (5.4)$$

где $u(t)$ – мгновенное значение напряжения;

t – текущее время;

U_{m1} и U_{m2} – амплитуды напряжения;

ω_1 и ω_2 – угловая частота ($\omega = 2\pi F$);

φ_{01} и φ_{02} – начальная фаза напряжения.

Значения параметров сигнала выбирается в соответствии с таблицей 5.1 или 5.2.

Таблица 5.1 – Варианты заданий с 1 по 10

	Порядковый номер фамилии обучающегося по списку									
	1	2	3	4	5	6	7	8	9	10
U_{m1}, *В*	1	2	3	4	5	6	7	8	9	10
U_{m2}, *В*	6	7	8	9	10	2	3	4	5	6
F_1,*кГц*	6	7	8	9	10	2	3	4	5	6
F_2, *кГц*	9	10	2	3	4	5	6	8	9	10
φ_{01}, *град*	10	20	30	40	50	60	70	80	10	20
φ_{02}, *град*	30	40	50	60	70	10	20	30	40	50

Таблица 5.2 – Варианты заданий с 11 по 20

	Порядковый номер фамилии обучающегося по списку									
	11	12	13	14	15	16	17	18	19	20
U_{m1}, *В*	1	2	3	4	5	6	7	8	9	10
U_{m2}, *В*	2	3	4	5	6	3	4	5	6	7
F_1,*кГц*	3	4	5	6	8	5	6	8	9	10
F_2, *кГц*	7	8	9	10	2	10	2	3	4	5
φ_{01}, *град*	70	10	20	30	40	50	40	50	60	70
φ_{02}, *град*	40	50	60	70	80	10	20	30	40	50

5.2.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее преобразование сигналов.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

5.2.3 Контрольные вопросы

1 Что называется дискретным сигналом?

2 Что называется непрерывным сигналом?

3 Что называется дискретизацией?

4 Что называется квантованием?

5 Как формулируется теорема Котельникова?

6 ЭФФЕКТИВНОЕ КОДИРОВАНИЕ

6.1 Метод Шеннона – Фано

Минимизация среднего числа двоичных символов, необходимых для выражения одной буквы сообщений, при отсутствии шума, позволяет уменьшить время передачи или емкость запоминающего устройства. Эффективное кодирование базируется на основной теореме Шеннона для каналов без шума.

К. Шеннон доказал, что сообщения, составленные из букв некоторого алфавита, можно закодировать так, что среднее число двоичных символов на букву будет сколь угодно близко к энтропии источника этих сообщений, но не менее этой величины. Теорема не указывает конкретного способа кодирования,

но из нее следует, что при выборе каждого символа кодовой комбинации необходимо стараться, чтобы он нес максимальную информацию. Каждый символ должен принимать значения 0 или 1 по возможности с равными вероятностями, и каждый выбор должен быть независим от значений предыдущих символов.

При отсутствии статистической взаимосвязи между буквами конструктивные методы построения эффективных кодов были даны впервые Шенноном и Фано. Код Шеннона-Фано строится следующим образом: буквы алфавита сообщений выписываются в таблицу в порядке убывания вероятностей их встречаемости. Затем их разделяют на две группы так, чтобы суммы вероятностей встречаемости букв в каждой из групп были бы по возможности одинаковыми. Всем буквам верхней половины в качестве первого символа записывается – 1, а всем нижним – 0. Каждая из полученных групп, в свою очередь, разбивается на две подгруппы с одинаковыми суммарными вероятностями и т. д. Процесс повторяется до тех пор, пока в каждой подгруппе не останется по одной букве.

Все буквы будут закодированы различными последовательностями символов из 0 и 1 так, что ни одна более длинная кодовая комбинация не будет начинаться с более короткой, соответствующей другой букве. Код, обладающий этим свойством, называется префиксным. Это позволяет вести запись текста без разделительных символов и обеспечивает однозначность декодирования.

6.2 Метод Хаффмана

Метод Хаффмана гарантирует однозначное построение кода с наименьшим для данного распределения вероятностей средним числом символов на букву. Для двоичного кода методика сводится к следующему.

Буквы алфавита сообщений выписываются в первый столбец в порядке убывания вероятностей. Две последние вероятности объединяются в одну вспомогательную, которой приписывается суммарная вероятность. Вероятности букв, не участвующих в объединении, и полученная суммарная вероятность снова располагаются в порядке убывания вероятностей в дополнительном столбце, а две последние вероятности снова объединяются. Процесс продолжается до тех пор, пока не получим единственную вспомогательную вероятность равную единице (таблица 6.1).

Для получения кодовой комбинации, соответствующей данной букве, прослеживается путь перехода вероятности по строкам и столбцам. Строится кодовое дерево. Из точки, соответствующей вероятности 1, направим две ветви, причем ветви с большей вероятностью присвоим символ 1, а с меньшей – 0. Такое последовательное ветвление продолжим до тех пор, пока не дойдем до вероятности каждой буквы. Кодовое дерево приведено на рисунке 6.1.

Таблица 6.1 – Метод Хаффмана

Буква	Вероятности	Вспомогательные столбцы						
		1	2	3	4	5	6	7
A_1	0.22	0.22	0.22	0.26	0.32	0.42	0.58	1
A_2	0.20	0.20	0.20	0.22	0.26	0.32	0.42	
A_3	0.16	0.16	0.16	0.10	0.22	0.26		
A_4	0.16	0.16	0.16	0.16	0.20			
A_5	0.10	0.10	0.16	0.16				
A_6	0.10	0.10	0.10					
A_7	0.04	0.06						
A_8	0.02							

Теперь, двигаясь по кодовому дереву от единицы через промежуточные вероятности к вероятностям каждой буквы, можно записать соответствующую ей кодовую комбинацию: A_1 – 01, A_2 – 00, A_3 – 111, A_4 – 110, A_5 – 100, A_6 – 1011, A_7 – 10101, A_8 – 10100. При этом получим $l_{ср} = 2{,}80$ символа на букву.

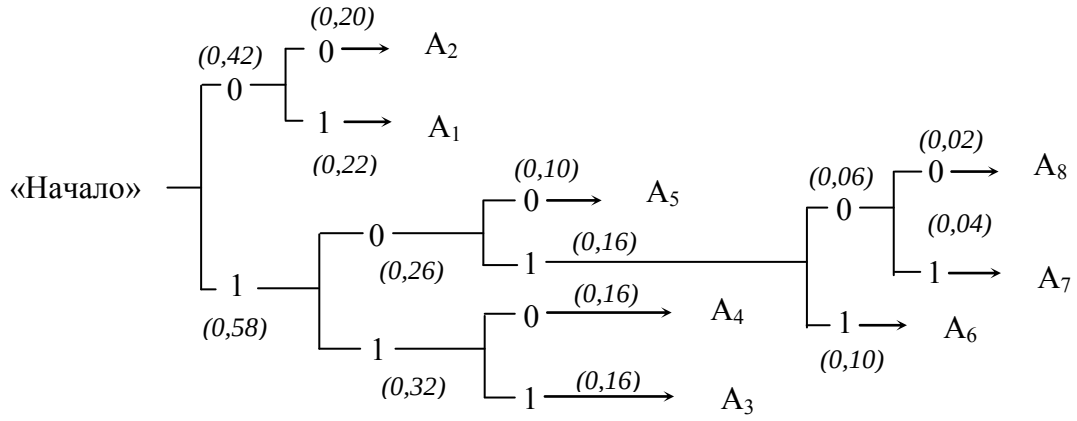

Рисунок 6.1 – Кодовое дерево

Отметим в заключение особенности эффективного кодирования.

Одна из особенностей обусловлена различием в длине кодовых комбинаций для разных букв. Если буквы выдаются через равные промежутки времени, то кодирующее устройство через равные промежутки времени выдает комбинации различной длины. Поскольку линия связи используется эффективно только в том случае, когда символы поступают в нее с постоянной скоростью, то на выходе кодирующего устройства должно быть предусмотрено буферное устройство («упругая» задержка). Оно запасает символы по мере их поступления и выдает их в линию связи с постоянной скоростью. Аналогичное устройство необходимо и на приемной стороне.

Вторая особенность связана с возникновением задержки в передаче информации. Наибольший эффект достигается при кодировании длинными бло-

ками, а это приводит к необходимости накапливать буквы, прежде чем сопоставить с ними определенную последовательность символов. При декодировании задержка возникает снова. Общее время задержки может быть велико, особенно при появлении блока, вероятность которого мала. Это следует учитывать при выборе длины кодируемого блока.

Еще одна особенность заключается в специфическом влиянии помех на достоверность приема. Одиночная ошибка может перевести передаваемую кодовую комбинацию в другую, не равную ей по длительности. Это повлечет за собой неправильное декодирование целого ряда последующих комбинаций, который называют треком ошибки. Специальными методами построения эффективного кода трек ошибки стараются свести к минимуму [5, 6].

6.3 Практическая работа № 6
«Эффективное кодирование. Метод Шеннона – Фано»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ кодирования методом Шеннона – Фано.

Используемые приемы и технологии: метод Шеннона – Фано, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: эффективное кодирование, метод Шеннона – Фано.

6.3.1 Варианты заданий

Вариант 1. Плохой работник с инструментами не в ладу.
Вариант 2. Добрая слава лучше богатства.
Вариант 3. Хорошее начало обеспечивает хороший конец.
Вариант 4. Верный как сталь.
Вариант 5. Весёлый словно сверчок.
Вариант 6. Чистый как новая булавка.
Вариант 7. Древний как холмы.
Вариант 8. Лающие собаки редко кусают.
Вариант 9. Пустая посуда звенит громче.
Вариант 10. Крайности сходятся.
Вариант 11. Доброе здоровье дороже богатства.
Вариант 12. Нечестно нажитое впрок не идёт.
Вариант 13. Словно кошка на горячих кирпичах.
Вариант 14. Свой своего ищет.

Вариант 15. У маленьких кувшинов большие ручки.
Вариант 16. Маслом каши не испортишь.
Вариант 17. Не сразу Москва строилась.
Вариант 18. Сколько стран, столько и обычаев.
Вариант 19. Куй железо, пока горячо.
Вариант 20. Исключение подтверждает правило.
Вариант 21. Свет клином не сошёлся.
Вариант 22. Что с возу упало, то пропало.
Вариант 23. В лес ведёт не одна дорога.
Вариант 24. Последняя капля переполняет чашу.
Вариант 25. Больше дела, меньше слов.

6.3.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее кодирование сообщения методом Шеннона – Фано.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

6.3.3 Контрольные вопросы

1 Что называется эффективным кодированием?

2 Какое условие выполнения кодирования методом Шеннона – Фано?

3 Какой алгоритм метода Шеннона – Фано?

4 Как формулируется теорема Шеннона о кодировании в дискретном канале без помех?

5 Какая структура кодовой таблицы?

6.4 Практическая работа № 7
«Эффективное кодирование. Метод Хаффмана»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ расчёта количества информации и определения целесообразности информации.

Используемые приемы и технологии: метод Хаффмана, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: кодовая таблица, метод Хаффмана.

6.4.1 Варианты заданий

Вариант 1. У маленьких кувшинов большие ручки.
Вариант 2. Маслом каши не испортишь.
Вариант 3. Не сразу Москва строилась.
Вариант 4. Сколько стран, столько и обычаев.
Вариант 5. Куй железо, пока горячо.
Вариант 6. Исключение подтверждает правило.
Вариант 7. Свет клином не сошёлся.
Вариант 8. Что с возу упало, то пропало.
Вариант 9. В лес ведёт не одна дорога.
Вариант 10. Последняя капля переполняет чашу.
Вариант 11. Больше дела, меньше слов.
Вариант 12. Плохой работник с инструментами не в ладу.
Вариант 13. Добрая слава лучше богатства.
Вариант 14. Хорошее начало обеспечивает хороший конец.
Вариант 15. Верный как сталь.
Вариант 16. Весёлый словно сверчок.
Вариант 17. Чистый как новая булавка.
Вариант 18. Древний как холмы.
Вариант 19. Лающие собаки редко кусают.
Вариант 20. Пустая посуда звенит громче.
Вариант 21. Крайности сходятся.
Вариант 22. Доброе здоровье дороже богатства.
Вариант 23. Нечестно нажитое впрок не идёт.
Вариант 24. Словно кошка на горячих кирпичах.
Вариант 25. Свой своего ищет.

6.4.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее кодирование сообщения методом Хаффмана.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

6.4.3 Контрольные вопросы

1 Что называется оптимальным префиксным кодом?

2 Какие существуют свойства оптимального кода для вероятностных распределений?

3 Какой этапы включает алгоритм метода Хаффмана?

4 Что понимается под кодовым деревом?

5 Какие выделяют отличия метода кодирования Хаффмана от метода кодирования Шеннона – Фано?

7 УНИВЕРСАЛЬНОЕ КОДИРОВАНИЕ ИСТОЧНИКОВ

7. 1 Постановка задачи универсального кодирования

Универсальное кодирование предполагает, что входом кодера является последовательность сообщений $x = (x_1,..., x_n)$ источника X. Алфавит кодера, длина последовательности n, алгоритм работы кодера известны декодеру. Статистические свойства источника заранее неизвестны. Задача состоит в том, чтобы обеспечить эффективное кодирование: найти представление последовательности источника двоичной последовательностью наименьшей длины.

Постановка задачи формулируется следующим образом: пусть $\Omega = \{\omega\}$ представляет множество моделей источников. Например, множество дискретных постоянных источников. Элемент множества определяется одномерным распределением вероятностей на X. Пусть H_ω энтропия на сообщение источника для заданной модели $\omega \in \Omega$ и заданный алгоритм кодирования обеспечивает для этой модели при кодировании последовательностей длины n среднюю скорость $\overline{R}_{n(\omega)}$. Тем самым определена избыточностью $r_{n(\omega)-}\overline{R}_n(\omega) - H_\omega$. Избыточностью кодирования для данного алгоритма для класса моделей Ω называется величина $r_n(\Omega) = \sup_{\omega\in\Omega}[\overline{R}_n(\omega) - H_\omega]$. Требуется построить алгоритм, минимизирующий избыточность $r_n(\Omega)$ для заданного класса моделей Ω.

7.2 Практическая работа № 8
«Универсальное кодирование. Двухпроходное побуквенное кодирование»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ алгоритма двухпроходного побуквенного кодирования.

Используемые приемы и технологии: метод двухпроходного побуквенного кодирования, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: избыточность, универсальное кодирование, двухпроходное побуквенное кодирование.

7.2.1 Варианты заданий

1 A little pot is soon hot.
2 A drop in the bucket.
3 A dog in the manger.
4 A word is enough to the wise.
5 A rolling stone gathers no moss.
6 All is well that ends well.
7 All is fish that comes to his net.
8 All that glitters id not gold.
9 Brevity is the soul of wit.
10 Custom is a second nature.
11 Better to do well than to say well.
12 Every country has its customs.
13 Hawks will not pick hawks’ eyes out.
14 It is never too late to learn.
15 Lost time is never found again.
16 Never put off till tomorrow what you can do today.
17 Speech is silver but silence is good.
18 Seconds thoughts are best.
19 Plenty is no plague.
20 The least said, the soonest mended.
21 The first blow is half the battle.
22 The mountain has brought forth a mouse.
23 To get out of bed on the wrong side.

24 To give a lark to catch a kite.

25 To make the cup run over.

7.2.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее двухпроходное побуквенное кодирование.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

7.2.3 Контрольные вопросы

1 Что называется универсальным кодированием?

2 Что называется двухпроходным побуквенным кодированием?

3 Как формулируется постановка задачи универсального кодирования источника?

4 Какие ограничения накладываются на множество моделей источника?

5 Какие ограничения накладываются на множество допустимых алгоритмов?

8 АЛГОРИТМЫ КОДИРОВАНИЯ ИСТОЧНИКОВ, ПРИМЕНЯЕМЫЕ В АРХИВАТОРАХ

Методы интервального кодирования и «стопки книг», монотонные коды применяются в более сложных и эффективных алгоритмах кодирования.

8.1 Метод Лемпела – Зива

Алгоритм LZ77 был опубликован в 1977 г. Разработан израильскими математиками Якобом Зивом (англ. *Jacob Ziv*) и Абрахамом Лемпелом (англ. *Abraham Lempel*). Многие программы сжатия информации используют модификацию LZ77. Одной из причин популярности алгоритмов LZ является простота при высокой эффективности сжатия. Основная идея LZ77 состоит в том, что второе и последующие вхождения некоторой строки символов в сообщении заменяются ссылками на ее первое вхождение. Алгоритм LZ77 использует про-

смотренную часть сообщения как словарь. Чтобы добиться сжатия, пытается заменить очередной фрагмент сообщения на указатель в содержимое словаря.

Алгоритм LZ77 использует «скользящее» по сообщению окно, разделенное на две неравные части. Первая, большая по размеру, включает просмотренную часть сообщения. Вторая, намного меньшая, является буфером, содержащим еще незакодированные символы входного потока. Размер окна составляет несколько килобайт, а размер буфера – не более ста байт. Алгоритм пытается найти в словаре (большей части окна) фрагмент, совпадающий с содержимым буфера.

Алгоритм LZ77 выдает коды, состоящие из трех элементов:

- смещение в словаре относительно его начала подстроки, совпадающей с началом содержимого буфера;
- длина этой подстроки;
- первый символ буфера, следующий за подстрокой [7].

8.2 Метод Лемпела – Зива – Велча

Алгоритм Лемпела – Зива – Велча (англ. Lempel – Ziv – Welch, LZW) – это универсальный алгоритм сжатия данных без потерь, созданный Абрахамом Лемпелем (англ. *Abraham Lempel*), Якобом Зивом (англ. *Jacob Ziv*) и Терри Велчем (англ. *Terry Welch*). Он был опубликован Велчем в 1984 году, в качестве улучшенной реализации алгоритма LZ78, опубликованного Лемпелем и Зивом в 1978 году.

Алгоритм при сжатии (кодировании) динамически создаёт таблицу преобразования строк: определённым последовательностям символов (словам) ставятся в соответствие группы бит фиксированной длины (обычно 12-битные). Таблица инициализируется всеми 1-символьными строками (в случае 8-битных символов – это 256 записей). По мере кодирования алгоритм просматривает текст символ за символом и сохраняет каждую новую уникальную 2-символьную строку в таблицу в виде пары код/символ, где код ссылается на соответствующий первый символ. После того как новая 2-символьная строка сохранена в таблице, на выход передаётся код первого символа. Когда на входе читается очередной символ, для него по таблице находится уже встречавшаяся строка максимальной длины, после чего в таблице сохраняется код этой строки со следующим символом на входе; на выход выдаётся код этой строки, а следующий символ используется в качестве начала следующей строки.

Алгоритму декодирования на входе требуется только закодированный текст, поскольку он может воссоздать соответствующую таблицу преобразования непосредственно по закодированному тексту.

Алгоритм Лемпела – Зива – Велча включает этапы:

1) инициализация словаря всеми возможными односимвольными фразами. Инициализация входной фразы *w* первым символом сообщения;

2) считать очередной символ *K* из кодируемого сообщения;

3) если *КОНЕЦ_СООБЩЕНИЯ*, то выдать код для *w*, иначе закончится предложение;

4) если фраза *wK* уже есть в словаре, присвоить входной фразе значение *wK* и перейти на 2-ой этап, иначе выдать код *w*, добавить *wK* в словарь, присвоить входной фразе значение *K* и перейти на 2-ой этап;

5) конец.

Алгоритм LZW даёт лучший коэффициент сжатия для большинства приложений, чем любой другой хорошо известный метод того времени. В 1987 году алгоритм стал частью стандарта на формат изображений GIF. Он также может (опционально) использоваться в формате TIFF. В настоящее время алгоритм содержится в стандарте PDF [7].

Пример представляет алгоритм LZW в действии, показывая состояние выходных данных и словаря на каждой стадии, как при кодировании, так и при раскодировании сообщения. С тем чтобы сделать изложение проще, ограничимся простым алфавитом – только заглавные буквы, без знаков препинания и пробелов. Сообщение, которое нужно сжать, выглядит следующим образом:

TOBEORNOTTOBEORTOBEORNOT#

Маркер # используется для обозначения конца сообщения. Тем самым, в нашем алфавите 27 символов (26 заглавных букв от A до Z и #). Компьютер представляет это в виде групп бит, для представления каждого символа алфавита нам достаточно группы из 5 бит на символ. По мере роста словаря, размер групп должен расти, с тем чтобы учесть новые элементы. 5-битные группы дают $2^5 = 32$ возможных комбинаций бит, поэтому, когда в словаре появится 33-е слово, алгоритм должен перейти к 6-битным группам. Заметим, что поскольку используется группа из всех нолей 00000, то 33-я группа имеет код **32**. Начальный словарь будет содержать:

= 00000

A = 00001

B = 00010

C = 00011

.

.

.

Z = 11010

Без использования алгоритма LZW, при передаче сообщения как есть – 25 символов по 5 бит на каждый – оно займёт 125 бит. Сравним это с тем, что получается при использовании LZW:

Символ: Битовый код: Новая запись словаря:

(на выходе)

T 20 = 10100

O 15 = 01111 27: TO

B 2 = 00010 28: OB

E 5 = 00101 29: BE

O 15 = 01111 30: EO

R 18 = 10010 31: OR <--- со следующего символа начинаем использовать 6-битные группы

N 14 = 001110 32: RN

O 15 = 001111 33: NO

T 20 = 010100 34: OT

TO 27 = 011011 35: TT

BE 29 = 011101 36: TOB

OR 31 = 011111 37: BEO

TOB 36 = 100100 38: ORT

EO 30 = 011110 39: TOBE

RN 32 = 100000 40: EOR

OT 34 = 100010 41: RNO

0 = 000000 42: OT#

Общая длина = 6*5 + 11*6 = 96 бит.

Применяя алгоритм LZW, сократили сообщение на 29 бит из 125 – это почти 22 %. Если сообщение будет длиннее, то элементы словаря будут представлять всё более и более длинные части текста, благодаря чему повторяющиеся слова будут представлены очень компактно.

Декодирование: нужно знать начальный словарь, а последующие записи словаря можно реконструировать уже на ходу, поскольку они являются просто конкатенацией предыдущих записей.

Данные:	На выходе:	Новая запись:	
		Полная:	Частичная:
10100 = 20	T		27: T?
01111 = 15	O	27: TO	28: O?
00010 = 2	B	28: OB	29: B?
00101 = 5	E	29: BE	30: E?
01111 = 15	O	30: EO	31: O?

10010 = 18 R 31: OR 32: R? <---- начинаем использовать 6-битные группы

001110 = 14 N 32: RN 33: N?

001111 = 15 O 33: NO 34: O?

010100 = 20 T 34: OT 35: T?

011011 = 27 TO 35: TT 36: TO? <---- для 37, добавляем только первый элемент

011101 = 29 BE 36: TOB 37: BE? следующего слова словаря

011111 = 31 OR 37: BEO 38: OR?

100100 = 36 TOB 38: ORT 39: TOB?

011110 = 30 EO 39: TOBE 40: EO?

100000 = 32 RN 40: EOR 41: RN?

100010 = 34 OT 41: RNO 42: OT?

000000 = 0 #

В приведённом примере декодирования, когда декодер встречает первый символ **T**, он знает, что слово 27 начинается с T, но необходимо знать и чем заканчивается. Проиллюстрируем проблему следующим примером. Мы декодируем сообщение **ABABA**:

Данные: На выходе: Новая запись:

Полная: Частичная:

…

011101 = 29 AB 46: (word) 47: AB?

101111 = 47 AB? <--- что нам с этим делать?

На первый взгляд, для декодера это неразрешимая ситуация. Мы знаем наперёд, что словом 47 должно быть **ABA**, но как декодер узнает об этом? Заметим, что слово 47 состоит из слова 29 плюс символ идущий следующим. Таким образом, слово 47 заканчивается на «символ идущий следующим». Но поскольку это слово посылается немедленно, то оно должно начинаться с «символа идущего следующим», и поэтому оно заканчивается тем же символом, что и начинается, в данном случае – **A**. Этот трюк позволяет декодеру определить, что слово 47 – это **ABA**. В общем случае такая ситуация появляется, когда кодируется последовательность вида *cScSc*, где *c* – это один символ, а *S* – строка, причём слово *cS* уже есть в словаре. На алгоритм LZW и его вариации был выдан ряд патентов как в США, так и в других странах. На LZ78 был выдан американский патент U.S. Patent 4,464,650 (англ.), принадлежащий Sperry Corporation, позднее ставшей частью Unisys Corporation. На LZW в США были выданы два патента: U.S. Patent 4,814,746 (англ.), принадлежащий IBM, и патент Велча U.S. Patent 4,558,302 (англ.) (выдан 20 июня 1983 года), принадлежащий Sperry Corporation, позднее перешедший к Unisys Corporation.

8.3 Практическая работа № 9
«Метод Лемпела – Зива»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ Лемпела – Зива.

Используемые приемы и технологии: метод Лемпела – Зива, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: монотонные коды, метод Лемпела – Зива.

8.3.1 Варианты заданий

1 Cut your coat according to your cloth.
2 Between two stools one goes to the ground.
3 East or west, home is best.
4 Empty vessels make the greatest sound.
5 Many a true word is spoken in jest.
6 Mutual admiration society.
7 One today is worth two tomorrow.
8 New brooms sweep clean.
9 No sweet without some sweat.
10 Take care of the pence and the pounds will take care of themselves.
11 Promise little, but do much.
12 Small rain lays great dust.
13 The mill cannot grind with the water that is past.
14 To cry with one eye and laugh with the other.
15 To pour water into a sieve.
16 To measure other people’s corn by one’s own bushel.
17 To play with fire.
18 To plough the sand.
19 To pay one back in one’s own coin.
20 When the fox preaches, take care of your geese.
21 Who has no tested bitter, knows not what is sweet.
22 Two heads are better then one.
23 To work with the left hand.
24 Zeal without knowledge is a run way horse.
25 You cannot judge a tree by its hark.

8.3.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее кодирование методом Лемпела – Зива.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

8.3.3 Контрольные вопросы

1 Как формулируется определение монотонного кода?

2 Какие существуют виды монотонного кода?

3 Какие этапы включает алгоритм метода Лемпела – Зива?

4 Какие преимущества имеет алгоритм Лемпела – Зива?

5 Какие выделяют аспекты практической реализации алгоритма Лемпела – Зива?

8.4 Практическая работа № 10
«Метод Лемпела – Зива – Велча»

Цель: получить теоретические знания и практические навыки в формализации на языке Visual C++ алгоритма Лемпела – Зива – Велча.

Используемые приемы и технологии: метод Лемпела – Зива – Велча, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: интервальное кодирование, метод Лемпела – Зива – Велча.

8.4.1 Варианты заданий

1 When the fox preaches, take care of your geese.

2 Who has no tested bitter, knows not what is sweet.

3 Two heads are better then one.

4 To work with the left hand.

5 Zeal without knowledge is a run way horse.

6 You cannot judge a tree by its hark.

7 To pour water into a sieve.

8 To measure other people’s corn by one’s own bushel.

9 To play with fire.

10 To plough the sand.

11 To pay one back in one’s own coin.

12 Take care of the pence and the pounds will take care of themselves.

13 Promise little, but do much.

14 Small rain lays great dust.

15 The mill cannot grind with the water that is past.

16 To cry with one eye and laugh with the other.

17 Many a true word is spoken in jest.

18 Mutual admiration society.

19 One today is worth two tomorrow.

20 New brooms sweep clean.

21 No sweet without some sweat.

22 Cut your coat according to your cloth.

23 Between two stools one goes to the ground.

24 East or west, home is best.

25 Empty vessels make the greatest sound.

8.4.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее кодирование методом Лемпела – Зива – Велча.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

8.4.3 Контрольные вопросы

1 Что понимается под интервальным кодированием?

2 Какие этапы включает алгоритм метода «стопка книг»?

3 Какие этапы включает алгоритм метода Лемпела – Зива – Велча?

4 Какие преимущества имеет метод Лемпела – Зива?

5 Какие существуют аспекты практического применения метода Лемпела – Зива – Велча?

9 КОДИРОВАНИЕ ИНФОРМАЦИИ ПРИ ПЕРЕДАЧЕ ПО ДИСКРЕТНОМУ КАНАЛУ С ПОМЕХАМИ

9.1 Код Хэмминга

Коды Хэмминга – самоконтролирующиеся и самокорректирующиеся коды. Построены они применительно к двоичной системе счисления.

Это алгоритм, который позволяет закодировать информационное сообщение определённым образом и после передачи (например по сети) определить, появилась ли какая-то ошибка в этом сообщении (к примеру из-за помех), и при возможности восстановить это сообщение. Код Хэмминга состоит из двух частей. Первая часть кодирует исходное сообщение, вставляя в него в определённых местах контрольные биты (вычисленные особым образом). Вторая часть получает входящее сообщение и заново вычисляет контрольные биты (по тому же алгоритму, что и первая часть). Если все вновь вычисленные контрольные биты совпадают с полученными, то сообщение получено без ошибок. В противном случае выводится сообщение об ошибке и при возможности ошибка исправляется.

Например, сообщение «habr» необходимо передать без ошибок. Для этого сначала нужно сообщение закодировать при помощи кода Хэмминга. Оно представляется в бинарном виде (рисунок 9.1).

Символ	ACSII код	Бинарное представление
h	68	01000100
a	61	00111101
b	62	00111110
r	72	01001000

Рисунок 9.1 – Представление кода Хэмминга в бинарном виде

Определяют длину информационного слова. Пусть длина слова равна 16. Таким образом, необходимо разделить исходное сообщение («habr») на блоки по 16 бит, которые потом кодировать отдельно друг от друга. Так как один символ занимает в памяти 8 бит, то в одно кодируемое слово помещается ровно два ASCII символа. Получаются две бинарные строки по 16 бит (рисунок 9.2).

h	a		b	r
01000100	00111101	и	00111110	01001000

Рисунок 9.2 – Бинарные строки по 16 бит

Процесс кодирования распараллеливается, и две части сообщения («ha» и «br») кодируются независимо друг от друга. Необходимо вставить контрольные биты. Они вставляются в строго определённых местах – это позиции с номерами, равными степеням двойки. В примере (при длине информационного слова в 16 бит) это будут позиции 1, 2, 4, 8, 16. Соответственно, получилось 5 контрольных бит (рисунок 9.3).

Было:

h	a
01000100	00111101

Стало:

h	a
000010000100	001011101

Рисунок – 9.3 Контрольные биты

Длина всего сообщения увеличилась на 5 бит. До вычисления самих контрольных бит, мы присвоили им значение 0.

Вычисление контрольных бит: необходимо вычислить значение каждого контрольного бита. Значение каждого контрольного бита зависит от значений информационных бит, но не от всех, а только от тех, которые этот контрольный бит контролирует. Для того, чтобы проследить, за какие биты отвечает каждый контрольный бит, необходимо понять закономерность: контрольный бит с номером *N* контролирует все последующие *N* бит через каждые *N* бит, начиная с позиции *N* (рисунок 9.4).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
X		X		X		X		X		X		X		X		X		X		X	1
	X	X			X	X			X	X			X	X			X	X			2
			X	X	X	X					X	X	X	X					X	X	4
							X	X	X	X	X	X	X	X							8
															X	X	X	X	X	X	16

Рисунок 9.4 – Биты, контролирующие контрольные биты

Знаком «X» обозначены биты, которые контролирует контрольный бит, номер которого справа. Например, бит номер 12 контролируется битами с номерами 4 и 8. Чтобы узнать какими битами контролируется бит с номером *N* необходимо разложить *N* по степеням двойки.

Вычисление значения контрольного бита: определяют сколько среди контролируемых им битов единиц, получают целое число и, если оно чётное, ставим ноль, в противном случае ставим единицу (рисунки 9.5, 9.6).

h	a
100110000100	001011101

Рисунок 9.5 – Первая часть

b	r
100101101110	010101000

Рисунок 9.6 – Вторая часть

Декодирование и исправление ошибок. Закодированное сообщение с ошибкой (11-ый бит передался неправильно) представлено на рисунке 9.7.

h	a
100110000110	001011101

Рисунок 9.7 – Ошибка закодированного сообщения

Вся вторая часть алгоритма заключается в том, что необходимо заново вычислить все контрольные биты (так же как и в первой части) и сравнить их с контрольными битами, которые получили. Расчёт контрольных бит с неправильным 11-ым битом представлен на рисунке 9.8.

h	a
010110010110	001011101

Рисунок 9.8 – Расчёт контрольных бит

Контрольные биты под номерами: 1, 2, 8 не совпадают с такими же контрольными битами, которые получили. Сложив номера позиций неправильных контрольных бит (1 + 2 + 8 = 11) получаем позицию ошибочного бита. Инвертировав его и отбросив контрольные биты, получим исходное сообщение в первозданном виде. Аналогично поступают со второй частью сообщения.

9.2 Практическая работа № 11
«Помехоустойчивое кодирование. Код Хэмминга»

Цель: получение теоретических знаний и практических навыков в формализации на языке Visual C++, кодирование сообщения и восстановление сообщения с обнаружением 1 ошибки и исправлением.

Используемые приемы и технологии: код Хэмминга, технология визуального проектирования и событийного программирования, среда программирования Microsoft Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: исходное сообщение, код Хэмминга, закончить предложения.

9.2.1 Варианты заданий

1 A dog in the manger.
2 A drop in the bucket.
3 A fly in the ointment.
4 A hard nut to crack.
5 A storm in a tea cup.
6 All covet, all lose.
7 As black as a crow.
8 As bold as brass.
9 As busy as a bee.
10 As clear as day.
11 As fit as a butter.
12 As like as two peas.
13 As large as life.
14 As fit as a fiddle.
15 As melancholy as a cat.
16 As pale as a ghost.
17 As poor as Job.
18 As slippery as an eel.
19 As true as steel.
20 As thin as a rake.
21 As ugly as sin.
22 As old as the hills.
23 As merry as a cricket.
24 As neat as a picked bone.
25 As mad as a March here.

9.2.2 Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее кодирование и восстановление сообщения с обнаружением и исправлением 1 ошибки с использованием кода Хэмминга.

3 Оформить отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

9.2.3 Контрольные вопросы

1 Что называется помехоустойчивым кодированием?

2 Какие виды кодов помехоустойчивого кодирования выделяются?

3 Как формулируется теорема Шеннона о кодировании для канала с помехами?

4 Что называется кодом Хэмминга?

5 Какие существуют этапы кодирования и декодирования сообщения с использованием кода Хэмминга?

10 КОНТРОЛЬНАЯ РАБОТА

10.1 Назначение, цели и задачи контрольной работы

Цель выполнения контрольной работы – закрепление теоретических знаний методов эффективного кодирования, изучение принципов построения и технической реализации кодирующих и декодирующих устройств эффективных кодов, приобретение практических навыков в настройке кодера и декодера и разработке программных приложений, формализующих алгоритмы методов эффективного кодирования.

Задачи, решаемые студентом в процессе выполнения контрольной работы:

– изучение алгоритмов методов эффективного кодирования;

– разработка программного приложения, реализующего алгоритм метода эффективного кодирования;

– изучение устройства и принципов работы кодера и декодера;

– настройка кодера и декодера;

– документирование контрольной работы.

10.2 Требования к контрольной работе

Контрольная работа включает программное приложение и пояснительную записку, содержащую разделы:

– введение;

– постановка задачи;

– алгоритмы методов эффективного кодирования:

– дерево Хаффмана;

– описание устройства и принципа работы кодера;

– описание устройства и принципа работы декодера;

– результаты работы программ кодирования сообщения и настройки кодера и декодера Effectcoding.exe (скриншоты);

– заключение;

– список использованных источников.

10.3 Порядок выполнения контрольной работы

1 Изучить методы построения и технической реализации эффективных кодов.

2 Построить эффективный код, используя методы Шеннона – Фано и Хаффмана.

3 Вычислить энтропию источника и среднюю длину комбинации полученного кода.

4 Используя построенный код, по методу Шеннона – Фано и Хаффмана настроить кодер и декодер.

5 Проверить работоспособность системы передачи.

10.4 Требования к отчету

Отчет должен включать:

1 кодовую таблицу построения эффективного кода;

2 схемы шифратора и дешифратора для построенного кода;

3 кодовое дерево Хаффмана;

4 результаты расчетов энтропии источника и среднюю длину кода для буквы, заданного алфавита из 8 букв и текста;

5 скриншоты программы;

6 выводы.

10.5 Варианты контрольной работы «Программные и технические средства кодирования и декодирования эффективных кодов»

Вариант 1. Дело мастера боится.

Вариант 2. Видна птица по полёту.

Вариант 3. Собака на сене.

Вариант 4. Капля в море.

Вариант 5. Ясный как день.

Вариант 6. Худой как щепка.

Вариант 7. Алмаз алмазом режется.

Вариант 8. Посмотрим, сказал слепой.

Вариант 9. Нужны дела, а не слова.
Вариант 10. Нет худа без добра.
Вариант 11. Крайности сходятся.
Вариант 12. Счастье благоприятствует смелым.
Вариант 13. Молвишь – не воротишь.
Вариант 14. Предупреждение – тоже сбережение.
Вариант 15. Друзья – воры времени.
Вариант 16. Яичница без яиц.
Вариант 17. На всех не угодишь.
Вариант 18. Он пороху не выдумает.
Вариант 19. Нужда всему научит.
Вариант 20. Голод – лучший повар.
Вариант 21. Если бы, да кабы
Вариант 22. Чёрным по белому.
Вариант 23. Выносить сор из избы.
Вариант 24. Последний, но не наименьший.
Вариант 25. Это и кошку рассмешит.

10.6 Контрольные вопросы

1 Когда целесообразно использовать эффективное кодирование?

2 Каковы сложности в реализации систем передачи с применением эффективных кодов?

3 До какого предела может быть сокращена средняя длина кодовой комбинации?

4 Каковы преимущества методики Хаффмана?

5 Какой код называется префиксным?

6 Как учесть взаимосвязь букв в тексте? Что произойдет с энтропией, если учесть взаимосвязь букв?

ЗАКЛЮЧЕНИЕ

Теория информации – научное направление, изучающее и исследующее методы измерения, преобразования, передачи, хранения и применения информации.

Современные достижения в теории информации позволили разработать эффективные криптографические системы, системы массового обслуживания, алгоритмы и программы для сжатия файлов, кодирования, обнаружения и исправления ошибок в полученных данных. Теория информации помогла создать

эффективные способы ослабления помех, действующих в каналах связи, позволила эффективно решить множество прикладных вопросов: создать модемы для телефонных каналов, скорость передачи информации которых приблизилась к теоретической пропускной способности каналов связи.

В методическом указании приводятся задания к выполнению практических и контрольных работ по дисциплине «Теория информации». Задания сопровождаются кратким теоретическим обоснованием. Выполнение практических и контрольных работ позволит приобрести практические навыки разработки приложений кодирования, передачи, восстановления информации и закрепить теоретические знания, полученные студентами на лекциях по дисциплине «Теория информации».

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1 Белов В. М. Теория информации. Курс лекций: учебное пособие для вузов/В. М. Белов, С. Н. Новиков, О. И. Солонская. – Москва: Горячая линия – Телеком, 2017. – 144 с.

2 Павлов Ю. Н. Теория информации для бакалавров/Ю. Н. Павлов, Е. В. Смирнова. – Москва: МГТУ им. Баумана, 2016. – 176 с.

3 Осокин А. Н. Теория информации: учебное пособие для прикладного бакалавриата/А. Н. Осокин. – Москва: Юрайт, 2016. – 205 с.

4 Цымбал В. П. Задачник по теории информации и кодированию/В. П. Цымбал. – Москва: Ленанд, 2014. – 280 с.

5 Панин В. В. Основы теории информации: учебное пособие для вузов/ В. В. Панин. – Москва.: БИНОМ. Лаборатория знаний, 2014. – 438 с.

6 Духин А. А. Теория информации/А. А. Духин – Москва: Гелиос АРВ, 2007. – 248 с.

7 Кудряшов Б. Д. Теория информации: учебное пособие/Б. Д. Кудряшов. – Санкт-Петербург: СПбГУ ИТМО, 2010. – 188 с.

Семахин Андрей Михайлович

ТЕОРИЯ ИНФОРМАЦИИ

Методические указания
к выполнению практических и контрольных работ
для студентов направлений подготовки 09.03.04, 09.03.03

Редактор В. С. Никифорова

Подписано в печать 03.06.22 | Формат 60x84 1/16 | Бумага 80 г/см^2
Печать цифровая | Усл. печ. л. 2,875 | Уч.-изд. л. 2,875
Заказ 46 | Тираж 25

БИЦ Курганского государственного университета.
640020, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.