КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СИСТЕМ СВЯЗИ

1. :Кловский Д. Д., Теория электрической связи М.: Радиотехника, 2009. – 648 с.
2. Прокис Дж. Цифровая связь. Пер с англ. / Под ред. Д.Д. Кловского. – М.: Радио и связь. 2000.
3. Язык С. Руководство для начинающих. Уэйт М., Прата С., Мартин Д., М., Мир, 1988.
4. Объектно-ориентированное программирование на С++. А. Пол Спб., М., «Невский диалект», «Изд. БИНОМ» 1999., 462 с.
5. Справочник по математике для инженеров и учащихся втузов. Бронштейн И. Н., Семендяев К.А. – М. Наука. 1981.
6. Компьютерное моделирование передачи и приёма двоичных сигналов в канале с гауссовским шумом с использованием объектно-ориентированного программирования на С++. Методическая разработка к практическим занятиям для студентов дневной формы обучения по дисциплине «объектно-ориентированное программирование на С++» Составители: Алышев Ю. В., Борисенков А. В., Самара 2006.
7. Лагутенко О.И. Модемы. Справочник пользователя. Спб.: «Лань», 1997, – 368 с.
8. М. Абрамовиц, И. Стиган. Справочник по специальным функциям М.: «Наука» 1979. – 832 с.

Абстрактная модель цифровой системы связи

a_i – сообщение на передаче, $\hat{a}_i$ – оценка сообщения на приёме, i – индекс, соответствующий отдельному сообщению (символу) и характеризующий порядковый номер временной последовательности.

Задачей передатчика является преобразование сообщений в сигналы, с помощью которых эти сообщения можно передавать по каналу связи. В реальных условиях в канале связи сигналы искажаются под воздействием различных внешних помех и шумов. Приёмное устройство принимает решение в пользу одного из передаваемых сигналов.

Искажение сигналов из-за внешних воздействий может привести к неправильным решениям приёмного устройства. В некоторых случаях вероятность ошибки приёмного устройства можно рассчитать теоретически. Например, если искажающее воздействие представляет собой белый гауссовский шум, то здесь практически всегда можно теоретически решить поставленную задачу.

Однако, не всегда удаётся теоретически определить вероятность ошибки. Например, в случае нескольких разных воздействий и, особенно, в случаях, когда имеются корреляционные связи, как в самих воздействиях, так и в передаваемых сигналах. Разрешение этого затруднения возможно при использовании адекватных моделей системы связи, то есть максимально точного описания как сигналов, так и мешающих воздействий.

Модели системы связи бывают математические, компьютерные и реальные в виде тестирующих устройств.

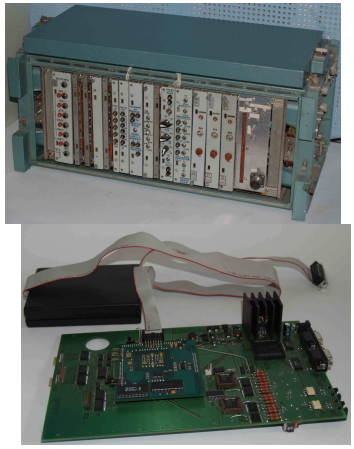
Математические модели рассмотрены в курсе «Теория электрической связи».

Для проверки и отладки разрабатываемых макетов устройств связи необходимы другие устройства, которые, например, имитируют радиоканал, могут сдвигать по частоте спектр сигнала в радиоканале, устройства для выявления ошибок и т. п.

Имитатор радиоканала ИРК-2

Приборы выявления ошибок

Радиомодемы

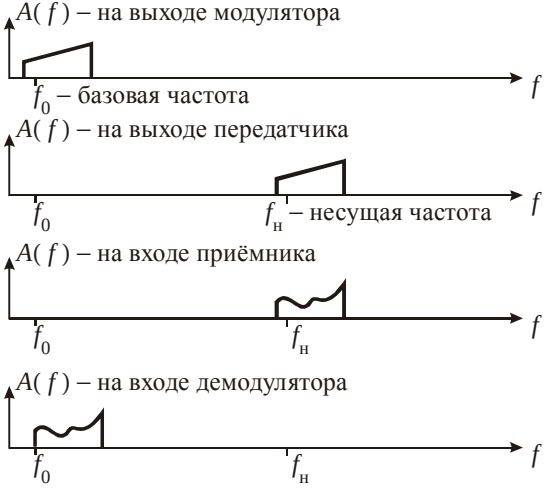


Компьютерные модели представляют собой программную модель, или отдельные программные элементы реальных устройств и радиоканала.

Анализ качества цифровой системы связи

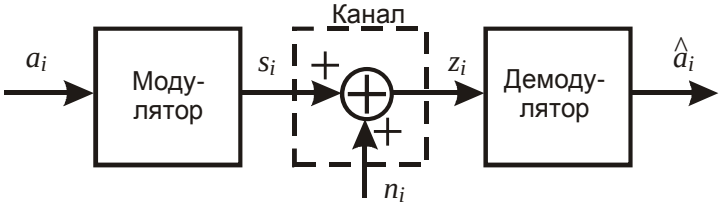
Рассмотрим подробнее процесс передачи цифрового сообщения.

Для передачи по каналам связи цифровое сообщение преобразуют в сигнал. Для того, чтобы этот сигнал передать по радиолинии необходимо принять во внимание его спектр. Спектр сигнала должен быть ограничен практически, то есть иметь конечную ширину. Далее этот сигнал (его спектр) необходимо перенести в область частот, на которой целесообразно передавать его через радиоканал. На приёмной стороне происходит обратное преобразование спектра сигнала. Далее производится анализ принятого колебания, в результате которого выносится решение в пользу символа, наиболее вероятного по заданному критерию качества.



При переносе спектра сигнала вверх на передающей стороне и вниз на приёмной в устройствах происходят линейные преобразования, которые можно не учитывать при постановке задачи анализа качества полученной оценки $\hat{a}_i$. Анализ качества оценки $\hat{a}_i$ можно производить на базе и математической, и компьютерной модели. Но искажения спектра сигнала в радиоканале можно моделировать, не производя переноса его в верхнюю область частот. Для этого, в простейшем случае, искажения в канале могут быть получены с помощью добавления гауссовского шума.

Модель системы связи, содержащей канал с аддитивным гауссовским шумом



модулятор – устройство, в котором параметр переносчика (несущей) меняется по закону первичного (входного) сигнала;

демодулятор – устройство, которое принимает решение в пользу той или иной гипотезы при анализе сигнальной реализации, искажённой в канале связи.

Под гипотезами понимаются символы, которые были использованы в качестве информационного сообщения на передаче. Количество гипотез обычно соответствует количеству символов на передающей стороне. В простейшем случае рассматривают передачу двоичных символов (бит).

В реальных устройствах связи сигнал на выходе модулятора и на входе демодулятора аналоговый. Однако при моделировании этих устройств на ЭВМ необходимо перейти к дискретному или цифровому сигналу. С учётом того, что над спектром сигнала не производится преобразование вверх и, согласно условию теоремы Котельникова, сигнал на выходе модулятора можно представить в виде последовательности отсчётов с заданной частотой дискретизации: $f_{\text{д}} \geq 2 f_{\text{в}}$, где $f_{\text{в}}$ – верхняя частота в спектре сигнала.

Конкретную частоту дискретизации выбирают из ряда стандартизованных частот и являющейся наиболее эффективной для поставленной задачи.

В рамках поставленной задачи гауссовский шум также можно представить в виде случайного сигнала, дискретные значения которого имеют гауссовское распределение и следуют с той же частотой дискретизации.

При моделировании на ЭВМ понятию «вероятность ошибки» $p_{\text{ош}}$ соответствует понятие *частость ошибки*, то есть насколько часто появляются ошибки в текущем сеансе связи. Частость ошибки определяется как $p'_{\text{ош}} = \frac{n_{\text{ош}}}{N}$.

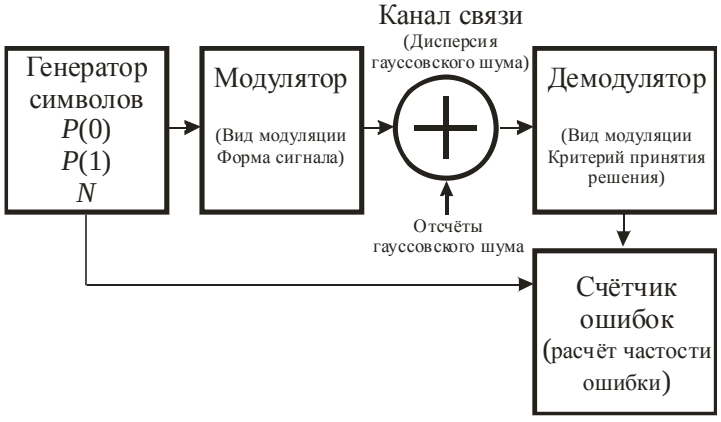
где $n_{\text{ош}}$ – число ошибок, которые произошли за текущий сеанс связи при передаче N символов.

Модель для статистических испытаний

При многократном повторении эксперимента с передачей информационной последовательности заданной длины можно получить ряд случайных значений параметра $n_{\text{ош}}$. В рамках теории математической статистики речь идёт о проведении статистических испытаний, где число передаваемых символов N в каждом эксперименте является численным параметром, характеризующим объём выборки.

Для проведения статистических испытаний должна быть построена соответствующая модель. Эту модель можно эффективно реализовать на персональном компьютере.

Простейшая модель для статистических испытаний содержит источник сигнала, выдающий двоичные символы, модель модулятора, модель канала связи, вносящего искажения в сигнал, модель демодулятора и счётчик ошибок.



P(0) – вероятность выдачи генератором символа «0»,
P(1) – вероятность выдачи генератором символа «1»,
N – число переданных символов.

Вероятность ошибки для фазовой модуляции (ФМ), модуляции ортогональными сигналами (МОС), частотной модуляции (ЧМ) и амплитудной модуляции (АМ) при когерентном приёме в однолучевом канале:

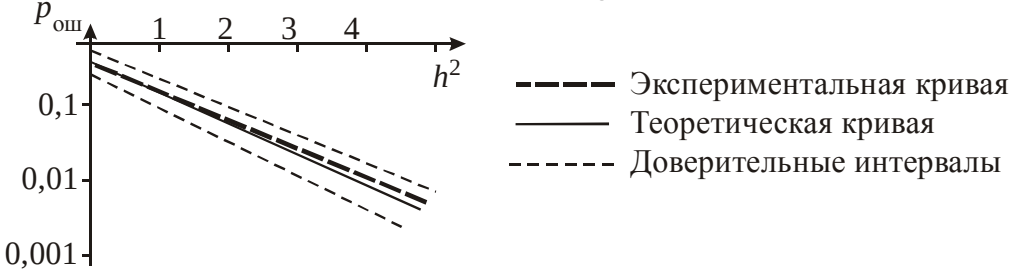
$$p_{\text{ФМ}} = Q\left(\sqrt{2h^2}\right)$$

$$p_{\text{МОС или ЧМ}} = Q\left(\sqrt{h^2}\right)$$

$$p_{\text{АМ}} = Q\left(\sqrt{h^2/2}\right)$$

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^{\infty} \mathrm{e}^{-t^2/2}dt \ - \text{функция ошибок}$$

График кривых помехоустойчивости



ДОВЕРИТЕЛЬНЫЕ ИНТЕРВАЛЫ:

$$p_{1,2} = p \pm \Phi^{-1}\left(P_{\text{дов}}\right)\sqrt{\frac{p(1-p)}{n}},$$

Где доверительная вероятность: $P_{\text{дов}} = 0,95$, число испытаний: $n = 100000$,

p – рассчитанная вероятность ошибки, $\Phi^{-1}(x)$ — функция, обратная функции Крампа:

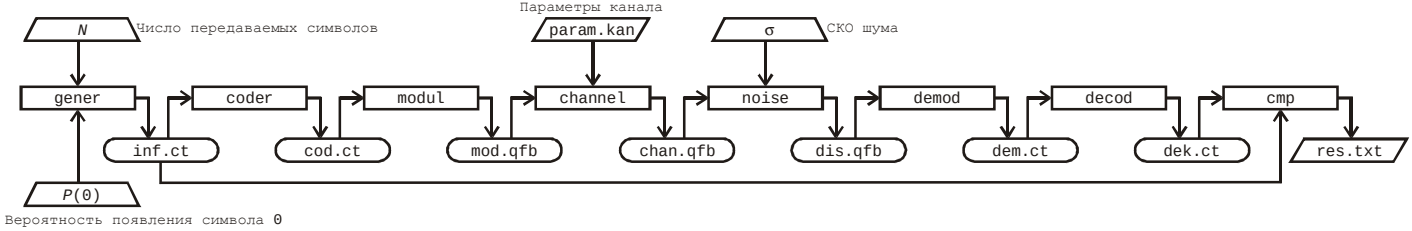
$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-x}^{x} \mathrm{e}^{-\frac{t^2}{2}}dt.$$

Для нахождения значений функций $Q(x)$ и $\Phi(x)$ воспользуйтесь литературой [5, стр. 76–77], где табулированы значения функции $\Phi_0(x)$. При этом

$$\Phi_0(x) = \frac{1}{\sqrt{2\pi}}\int_{0}^{x} \mathrm{e}^{-\frac{t^2}{2}}dt = \frac{1}{2}\Phi(x) = \frac{1}{2} - Q(x).$$

Блочная модель системы связи

Моделируя отдельно каждое устройство системы связи, впоследствии потребуется осуществить взаимосвязь между ними. Эта взаимосвязь может быть, например, осуществлена через файлы данных. Кроме того, модель какого-либо устройства может иметь ряд параметров. При статистическом моделировании потребуется выставлять численные значения этих параметров. В компьютерной модели и файлы данных, и численные значения параметров, должны быть явно прописаны. Таким требованиям удовлетворяет блочная модель системы связи:



Здесь прямоугольниками обозначены исполняемые файлы, фигуры с двумя закруглёнными сторонами являются файлами данных, трапеции — числовые параметры устройств, параллелограммы — текстовые файлы, содержащие численные параметры устройств или содержащие результаты моделирования.

Чтобы получить статистический результат, потребуется поочерёдно запустить каждый исполняемый файл. Это делается через командную строку операционной системы. В командной строке прописывается имя исполняемого файла, а также параметры, которые запрашивает исполняемый файл.

Для автоматического поочерёдного запуска исполняемых программ можно воспользоваться командными файлами, где сразу прописать все вызовы, которые необходимы для получения статистического результата.

Правила работы с командной строкой. Передача параметров командной строки в программу. Вызов краткой подсказки. Обоснование выбора формата файла. Выходной файл.

Командная оболочка — это отдельный программный продукт, который обеспечивает прямую связь между пользователем и операционной системой. Текстовый пользовательский интерфейс командной строки предоставляет среду, в которой выполняются приложения и служебные программы с текстовым интерфейсом. В командной оболочке программы выполняются, и результат выполнения отображается на экране.

Командная оболочка Windows использует интерпретатор команд **cmd.exe**, который загружает приложения и направляет поток данных между приложениями, для перевода введенной команды в понятный системе вид. Консоль командной строки присутствует во всех версиях операционных систем Windows. Отличием работы из командной строки является полное отсутствие больших и громоздких графических утилит.

Для каждого приложения, поддерживающего командную строку, предусмотрен специальный набор команд, которые может обрабатывать программа. Параметры команд могут иметь самый разный формат. Чтобы передать программе параметры, необходимо ввести в командной строке имя приложения и параметры команд, разделённые пробелом (пробелами). После нажатия **Enter** запустится приложение (или выполнится команда) с введёнными параметрами.

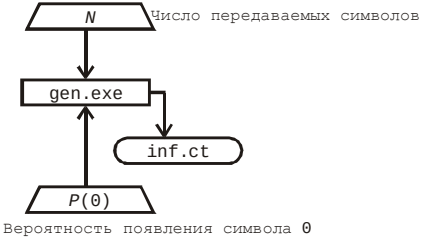
Генератор двоичных символов: (BorlandC++ DOS)

```
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
int main(int mn,char* nm[])
{
long i,N;
char h,c;
float P,r;
randomize();
if(mn!=4) cerr<<"gen.exe inf.ct N P(0)\n",exit(1);
strlwr(nm[1]);
if(nm[1][strlen(nm[1])-1]=='t') h=0x30; else h=0;
N=atol(nm[2]); if(N<1) cerr<<"Ошибка N<0!\n",exit(2);
P=atof(nm[3]); if(P>1||P<0) cerr<<"Ошибка P(0)!=(0..1)!\n",exit(3);
r=32768*P;
ofstream out(nm[1],ios::binary);
if(!out) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(4);
for(i=0;i<N;i++) c=(rand()>=r)+h,out.put(c);
out.close();
return 0;
}
```

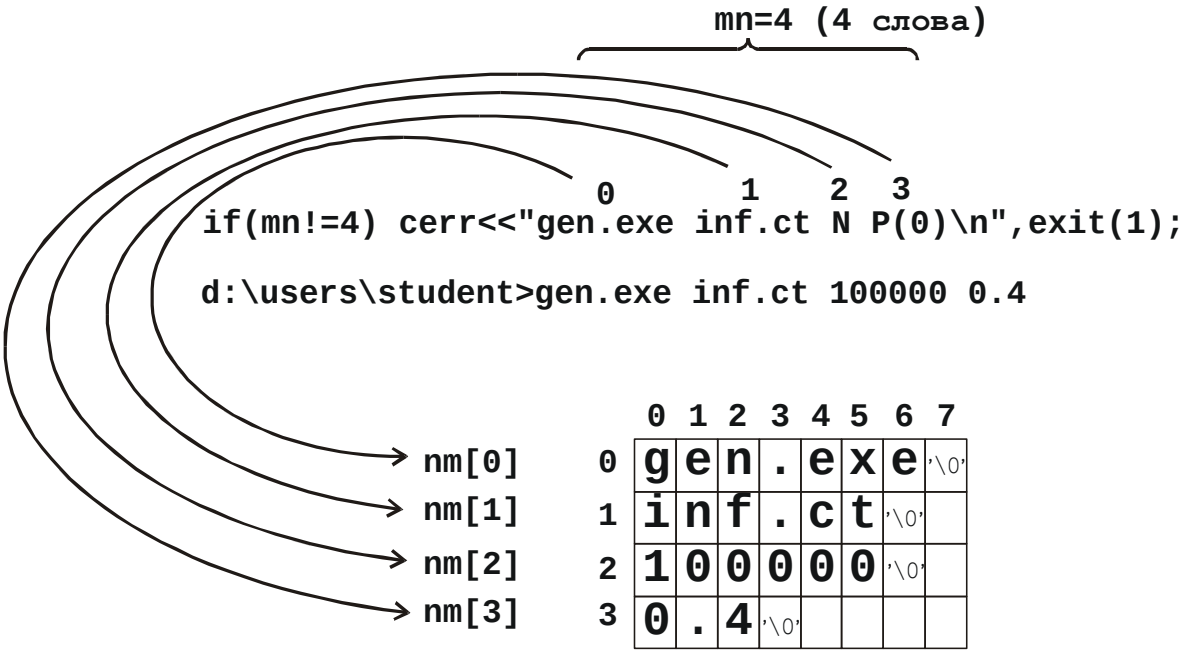
(Geany C++ Linux)

```
#include<iostream>
#include<fstream>
#include<cstdlib>
#include<cstring>
#include<ctime>
using namespace std;
int main(int mn,char* nm[])
{
long i,N; char h,c; float P,r;
srand(time(NULL));
if(mn!=4) cerr<<"./gen inf.ct N P(0)\n",exit(1);
if(nm[1][strlen(nm[1])-1]=='t') h=0x30; else h=0;
N=atol(nm[2]); if(N<1) cerr<<"Ошибка N<0!\n",exit(2);
P=atof(nm[3]); if(P>1||P<0) cerr<<"Ошибка P(0)!=(0..1)!\n",exit(3);
r=RAND_MAX*P;
ofstream out(nm[1],ios::binary);
if(!out) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(4);
for(i=0;i<N;i++) c=(rand()>=r)+h,out.put(c);
out.close();
return 0;
}
```

Генератор в блочной компьютерной модели

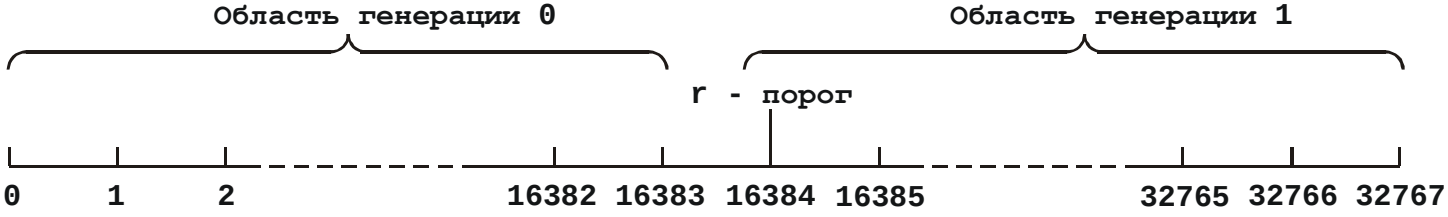


Передача информации из командной строки в программу



Используется двумерный массив типа char.

r=32768*P при P=0.5
r=16384



c=(rand()>=r)+h

rand() - генерирует целые псевдослучайные равномерно распределённые числа в диапазоне 0...32767

Для rand()>=16384 это диапазон 16384...32767, иначе 0...16383

Для диапазона 16384...32767: (rand()>=16384) есть “Истина” или равно 1.
Для диапазона 0...16383: (rand()>=16384) есть “Ложь” или равно 0.

Проверка распределения выпадения двоичных символов:

Источник: Пересчёт коэффициента p и изменение вида формулы (7.1.26., стр. 122, [8])

Проверка генератора осуществляется исполняемым файлом «prov.exe». Производится подсчёт нулей и общего количества записанных символов в файле «inf.ct». Исполняемый файл имеет также входной параметр вероятности выпадения нуля $P(0)$. Кроме того, задаётся значение доверительной вероятности $P_{\text{дов}}$.

Выполняется расчёт доверительных границ для частости появления нуля.

$$p_{\text{в,н}} = p(0) \pm \Phi^{-1}\left(P_{\text{дов}}\right)\sqrt{\frac{p(0)\left(1-p(0)\right)}{N}},$$

где $\Phi^{-1}(x)$ — функция обратная функции Крампа.

Для нахождения значения функции обратной функции Крампа, необходимо задать величину доверительной вероятности. Расчёт выполняется с использованием формул

$$\Phi_0(x) = \frac{1}{\sqrt{2\pi}}\int_0^x \mathrm{e}^{-\frac{t^2}{2}}dt = \frac{1}{2}\Phi(x) = \frac{1}{2} - Q(x) \qquad \text{и} \qquad \operatorname{erf} x = \frac{2}{\sqrt{\pi}}\int_0^x \mathrm{e}^{-t^2}dt$$

где erf x вычисляется с помощью аппроксимационных формул:

7.1.26. $\operatorname{erf} x = 1 - (a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5)\, e^{-x^2} + \varepsilon(x)$,

$$t = \frac{1}{1+px}, \quad |\varepsilon(x)| \leqslant 1.5 \cdot 10^{-7},$$

$p = 0.32759\ 11$, $a_1 = 0.25482\ 9592$,

$a_2 = -0.28449\ 6736$, $a_3 = 1.42141\ 3741$,

$a_4 = -1.45315\ 2027$, $a_5 = 1.06140\ 5429$.

Ниже приведён листинг программы, реализующей работу программы, проверяющей правильность частости выпадения символа «0» в генераторе информационных символов:

```
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>

long double fn_q(long double x)
{ long double sc,t; t=1/(1+.2316419*x);
sc=(.254829592+(-.284496736+(1.421413741+(-1.453152027+1.061405429*t)*t)*t)*t)*t;
return(.5*sc*exp(-x*x/2));
}
long double fn_qm1(long double y)
{ long double x0,x=0; if(y<0) y=-y;
while(y<fn_q(x+=5));
for(x0=x-5;;)
 {
 if(fabs(x0-x)<0.000001) break;
 if(y<fn_q((x+x0)/2)) x0=(x+x0)/2; else x=(x+x0)/2;
 }
return((x+x0)/2);
}
```

```
int main(int mn,char* nm[])
{ long i,N,s; char h,c; float P,Pd,d,f;
if(mn!=4) cerr<<"test.exe inf.ct P(0) Pдов\n",exit(1);
if(nm[1][strlen(nm[1])-1]=='t') h=0x30; else h=0;
P=atof(nm[2]); if(P>1||P<0) cerr<<"Ошибка P(0) вне диапазона (0..1)!\n",exit(2);
Pd=atof(nm[3]); if(Pd>1||Pd<0) cerr<<"Ошибка Pдов вне диапазона (0..1)!\n",exit(3);
ifstream in(nm[1],ios::binary);
if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(4);
in.seekg(0,ios::end); N=in.tellg(); in.seekg(0,ios::beg);
cout << "N=" << N;
for(s=i=0;i<N;i++) c=in.get()-h,s+=c;
cout<<"\tQ^-1="<<fn_qm1((1-Pd)/2)<<'\t';
d=fn_qm1((1-Pd)/2)*sqrt(P*(1-P)/N);
s=N-s; f=float(s)/N;
cout << P-d << " < " << f << " < " << P+d;
if(P-d<f&&f+d>P); else cout << " Тест не пройден!", cerr <<"Тест не пройден!\n";
cout << endl;
in.close();
return 0;
}
```

Командный файл для проверки:

```
del res.txt
del err.txt
for /l %%i in (1,1,100) do gen.exe 1.ct 10000 0.1 | test.exe 1.ct 0.1 0.95 >> res.txt 2>> err.txt
```

Но так как ГПСЧ не будет успевать инициализировать новым значением, то генератор требует доработки:

```
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#include<sys\timeb.h>
int main(int mn,char* nm[])
{
long i,N; char h,c; float P,r;
time_t t=time(NULL);
struct timeb y,z;
for(ftime(&z),z=y;z.millitm==y.millitm;ftime(&y));
z=y;
srand((t<<9)+y.millitm);
if(mn!=4) cerr<<"gen.exe inf.ct N P(0)\n",exit(1);
strlwr(nm[1]);
if(nm[1][strlen(nm[1])-1]=='t') h=0x30; else h=0;
N=atol(nm[2]); if(N<1) cerr<<"Ошибка N<0!\n",exit(2);
P=atof(nm[3]); if(P>1||P<0) cerr<<"Ошибка P(0)!=(0..1)!\n",exit(3);
r=32768l*P;
ofstream out(nm[1],ios::binary);
if(!out) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(4);
for(i=0;i<N;i++) c=(rand()>=r)+h,out.put(c);
out.close();
return 0;
}
```

Аналогичные программы для среды Geany в Linux

```
#include<iostream>
#include<fstream>
#include<cstdlib>
#include<cstring>
#include<cmath>
using namespace std;
long double fn_q(long double x)
{ long double sc,t; t=1/(1+.2316419*x);
sc=(.254829592+(-.284496736+(1.421413741+(-1.453152027+1.061405429*t)*t)*t)*t)*t;
return(.5*sc*exp(-x*x/2));
}
long double fn_qm1(long double y)
{ long double x0,x=0; if(y<0) y=-y;
while(y<fn_q(x+=5));
for(x0=x-5;;)
 {
 if(fabs(x0-x)<0.000001) break;
 if(y<fn_q((x+x0)/2)) x0=(x+x0)/2; else x=(x+x0)/2;
 }
return((x+x0)/2);
}
```

```
int main(int mn,char* nm[])
{ long i,N,s; char h,c; float P,Pd,d,f;
if(mn!=4) cerr<<"test.exe inf.ct P(0) Pдов\n",exit(1);
if(nm[1][strlen(nm[1])-1]=='t') h=0x30; else h=0;
P=atof(nm[2]); if(P>1||P<0) cerr<<"Ошибка P(0) вне диапазона (0..1)!\n",exit(2);
Pd=atof(nm[3]); if(Pd>1||Pd<0) cerr<<"Ошибка Pдов вне диапазона (0..1)!\n",exit(3);
ifstream in(nm[1],ios::binary);
if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(4);
in.seekg(0,ios::end); N=in.tellg(); in.seekg(0,ios::beg);
cout << "N=" << N;
for(s=i=0;i<N;i++) c=in.get()-h,s+=c;
cout<<"\tQ^-1="<<fn_qm1((1-Pd)/2)<<"\t";
d=fn_qm1((1-Pd)/2)*sqrt(P*(1-P)/N);
s=N-s; f=float(s)/N;
cout << P-d << " < " << f << " < " << P+d;
if(P-d<f&&f+d>P); else cout << " Тест не пройден!", cerr <<"Тест не пройден!\n";
cout << endl;
in.close();
return 0;
}
```

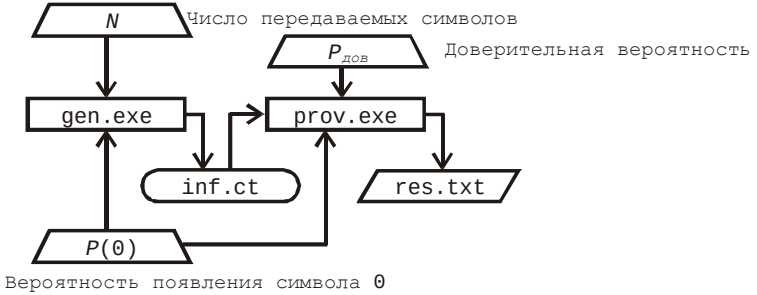
командная строка

```
rm res.txt
rm err.txt
for((i=0;i<100;i++)); do ./gen i.ct 1000000 0.1; ./2 i.ct 0.1 0.95 >> res.txt 2>> err.txt; done
```

и модифицированный генератор

```
#include<iostream>
#include<fstream>
#include<cstdlib>
#include<cstring>
#include<ctime>
using namespace std;
int main(int mn,char* nm[])
{
long i,N; char h,c; float P,r;
struct timespec z;
clock_gettime(CLOCK_REALTIME,&z);
srand(z.tv_nsec);
if(mn!=4) cerr<<"gen.exe inf.ct N P(0)\n",exit(1);
if(nm[1][strlen(nm[1])-1]=='t') h=0x30; else h=0;
N=atol(nm[2]); if(N<1) cerr<<"Ошибка N<0!\n",exit(2);
P=atof(nm[3]); if(P>1||P<0) cerr<<"Ошибка P(0)!=(0..1)!\n",exit(3);
r=RAND_MAX*P;
ofstream out(nm[1],ios::binary);
if(!out) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(4);
for(i=0;i<N;i++) c=(rand()>=r)+h,out.put(c);
out.close();
return 0;
}
```

Тестирование генератора



Основная задача состоит в том, чтобы получить последовательность, которая похожа на случайную.

Главная мысль сказанного заключается в том, что мы не можем доверять себе в оценке, случайна или нет данная последовательность чисел. Необходимо использовать какие-то непредвзятые механические тесты и статистическая теория даёт нам некоторые количественные критерии случайности.

Рассмотрим один из эмпирических тестов, который применяется для проверки равномерности (проверки частот) целочисленной последовательности. В этом случае проверке подвергается следующая последовательность:

$$\langle Y_n \rangle = Y_0, Y_1, Y_2, ...,$$

где n — длина последовательности (размер выборки)

Входящие в эту последовательность целые числа распределены равномерно между 0 и $d-1$. Значение d может быть любым. Чтобы тест был представительным, значение d должно быть достаточно большим, но не слишком, чтобы на вычислительной машине это не возникало затруднений во временных рамках. Для каждого целого r, $0 \le r < d$, подсчитать число значений $Y_j = r$ при $0 \le j < n$ и применить критерий χ^2 с числом категорий, равным $k = d$ и вероятностями $p_s = \frac{1}{d}$.

Пусть все возможные испытания разделены на k категорий, то при проведении n независимых испытаний исход каждого испытания абсолютно не влияет на исход остальных. Пусть p_s — вероятность того, что результат испытания попадёт в категорию s, и пусть Y_s — число испытаний, тогда можно определить величину V, которая называется статистикой χ^2, соответствующей значениям $Y_1, Y_2, ..., Y_k$, полученным в эксперименте:

$$V = \sum_{1 \le s \le k} \frac{\left(Y_s - np_s\right)^2}{np_s} = \frac{1}{n} \sum_{1 \le s \le k} \left(\frac{Y_s^2}{p_s}\right) - n.$$

Используя тождество $\left(Y_s - np_s\right)^2 = Y_s^2 - 2np_sY_s + n^2p_s^2$ и равенства $Y_1 + Y_2 + ... + Y_k = n$ и $p_1 + p_2 + ... + p_k = 1$

Как по значению V определить: является ли равномерной рассматриваемая целочисленная последовательность?

Ответ на этот вопрос даёт таблица, в которой приведено «распределение χ^2 с ν степенями свободы» при разных значениях ν. Следует пользоваться строкой таблицы с $\nu = k-1$; *число «степеней свободы» равно* $k-1$, *то есть на единицу меньше числа категорий.*

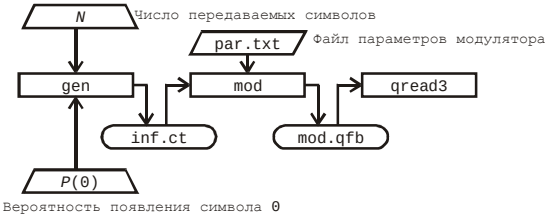
Если в таблице в строке ν и колонке p находится число x, то это означает, что значение V, определяемое по формуле, будет больше x с вероятностью p.

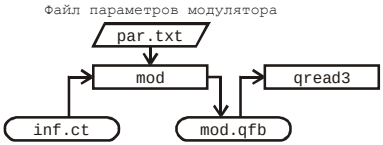
	$p=99\%$	$p=95\%$	$p=75\%$	$p=50\%$	$p=25\%$	$p=5\%$	$p=1\%$
$\nu=1$	0.00016	0.00393	0.1015	0.4549	1.323	3.841	6.635
$\nu=2$	0.00201	0.1026	0.5753	1.386	2.773	5.991	9.210
$\nu=3$	0.1148	0.3518	1.213	2.366	4.108	7.815	11,34
$\nu=4$	0.2971	0.7107	1.923	3.357	5.385	9.488	13.28
$\nu=5$	0.5543	1.1455	2.675	4.351	6.626	11.07	15.09
$\nu=6$	0.8720	1.635	3.455	5.348	7.841	12.59	16.81
$\nu=7$	1.239	2.167	4.255	6.346	9.037	14.07	18.48
$\nu=8$	1.646	2.733	5.071	7.344	10.22	15.51	20.09
$\nu=9$	2.088	3.325	5.899	8.343	11.39	16.92	21.67
$\nu=10$	2.558	3.940	6.737	9.342	12.55	18.31	23.21
$\nu=11$	3.053	4.575	7.584	10.34	13.70	19.68	24.73
$\nu=12$	3.571	5.226	8.438	11.34	14.84	21.03	26.22
$\nu=15$	5.229	7.261	11.04	14.34	18.25	25.00	30.58
$\nu=20$	8.260	10.85	15.45	19.34	23.83	31.41	37.57
$\nu=30$	14.95	18.49	24.48	29.34	34.80	43.77	50.89
$\nu=50$	29.71	34.76	42.94	49.33	56.33	67.50	76.15
$\nu>30$	*приблизительно* $\nu+2\sqrt{\nu}x_p+\frac{4}{3}x_p^2-\frac{2}{3}$						
$x_p=$	-2.33	-1.64	$-.675$	0.00	0.675	1.64	2.33

Проверка с помощью критерия χ^2 заключается в следующем. Проводится n независимых испытаний, где n — достаточно большое число. Подсчитывается число испытаний, результат которых относится к каждой из k категорий, и по формулам вычисляется значение V. Затем V сравнивается с числами из таблицы при $\nu=k-1$. Если V меньше значения, соответствующего $p=99\%$, или больше значения, соответствующего $p=1\%$, то результаты бракуются как недостаточно случайные. Если p лежит между 99 и 95% или между 5 и 1%, то результаты считаются «подозрительными»; при значениях p, полученных интерполяцией по таблице, заключённых между 95 и 90% или 10 и 5%, результаты «слегка подозрительны». Часто с помощью критерия χ^2 проверяют по крайней мере три раза разные части исследуемого ряда чисел, и, если не менее двух раз из трёх результаты оказываются подозрительными, числа отбрасываются как недостаточно случайные. (для проверки равномерности: как недостаточно равномерные).

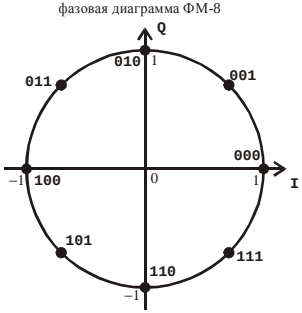
МОДЕЛЬ МОДУЛЯТОРА ДЛЯ ЛИНЕЙНЫХ ВИДОВ МОДУЛЯЦИИ.

Тестирование модулятора





Текстовый файл параметров



Пример содержимого файла par.txt
для фазовой диаграммы ФМ-8
и соответствие строк трибитам

000	1	0
001	0,707	0,707
010	0	1
011	–0,707	0,707
100	–1	0
101	–0,707	–0,707
110	0	–1
111	0,707	–0,707

Ниже приведён листинг программы, реализующей работу модулятора

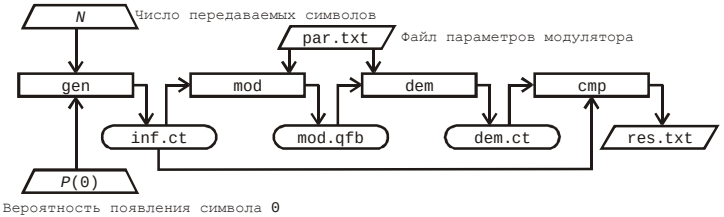
```
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
int main(int mn,char* nm[])
{ long i,N,m; char h,c; float r,*x,*y; strlwr(nm[1]);
if(mn!=4) cerr<<"mod.exe inf.ct mod.qfb mod.txt\n",exit(1);
if(nm[1][strlen(nm[1])-1]=='t') h=0x30; else h=0;
ifstream in(nm[1],ios::binary);
if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end); N=in.tellg(); in.seekg(0,ios::beg);
cout <<"N="<<N<<endl;
ifstream inp(nm[3]); if(!inp) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(3);
for(m=-1;!inp.eof();m++) inp>>r;
if(m&1) cerr<<"Нечётное количество координат!\n",exit(4);
m/=2; inp.close();
cout<<"m="<<m<<endl;
x=new float[m]; if(!x) cerr<<"Нет памяти для массива x!\n",exit(5);
y=new float[m]; if(!y) cerr<<"Нет памяти для массива y!\n",exit(6);
inp.open(nm[3]); if(!inp) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(7);
for(i=0;i<m;i++) inp>>x[i]>>y[i];
inp.close();
for(i=0;i<m;i++) cout<<x[i]<<'\t'<<y[i]<<endl;
ofstream out(nm[2],ios::binary);
if(!out) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(8);
for(i=0;i<N;i++)
 { c=in.get()-h;
 out.write((char*)&x[c],sizeof(float));
 out.write((char*)&y[c],sizeof(float));
 }
out.close(); in.close(); delete [] x; delete [] y;
return 0;
}
```

Программа модулятора для среды Geany в Linux

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cstring>
using namespace std;
int main(int mn,char* nm[])
{ long i,N,m; char h,c; float r,*x,*y;
if(mn!=4) cerr<<"./mod inf.ct mod.qfb mod.txt\n",exit(1);
if(nm[1][strlen(nm[1])-1]=='t') h=0x30; else h=0;
ifstream in(nm[1],ios::binary);if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end); N=in.tellg(); in.seekg(0,ios::beg);
cout <<"N="<<N<<endl;
ifstream inp(nm[3]); if(!inp) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(3);
for(m=-1;!inp.eof();m++) inp>>r;
if(m&1) cerr<<"Нечётное количество координат!\n",exit(4);
m/=2; inp.close(); cout<<"m="<<m<<endl;
x=new float[m]; if(!x) cerr<<"Нет памяти для массива x!\n",exit(5);
y=new float[m]; if(!y) cerr<<"Нет памяти для массива y!\n",exit(6);
inp.open(nm[3]); if(!inp) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(7);
for(i=0;i<m;i++) inp>>x[i]>>y[i];
for(i=0;i<m;i++) cout<<x[i]<<'\t'<<y[i]<<endl;
inp.close(); ofstream out(nm[2],ios::binary);
if(!out) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(8);
for(i=0;i<N;i++)
 { c=in.get()-h;
 out.write(reinterpret_cast<char*>(&x[int(c)]),sizeof(float));
 out.write(reinterpret_cast<char*>(&y[int(c)]),sizeof(float));
 }
out.close(); delete [] x; delete [] y; return 0;
}
```

5. Модель демодулятора линейных видов модуляции.

Тестирование демодулятора



Ниже приведён листинг программы, реализующей работу демодулятора

```
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
int main(int mn,char* nm[])
{ long j,i,N,m,R; char h,c;
float r,*x,*y,X,Y,min;
strlwr(nm[1]);
if(mn!=4) cerr<<"dem.exe mod.qfb dem.ct mod.txt\n",exit(1);
if(nm[2][strlen(nm[2])-1]=='t') h=0x30; else h=0;
ifstream in(nm[1],ios::binary);
if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end); N=in.tellg()/8; in.seekg(0,ios::beg);
cout <<"N="<<N<<endl;
ifstream inp(nm[3]); if(!inp) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(3);
for(m=-1;!inp.eof();m++) inp>>r;
if(m&1) cerr<<"Нечётное количество координат!\n",exit(4);
m/=2; inp.close();
cout<<"m="<<m<<endl;
x=new float[m]; if(!x) cerr<<"Нет памяти для массива x!\n",exit(5);
```

```
y=new float[m]; if(!y) cerr<<"Нет памяти для массива y!\n",exit(6);
inp.open(nm[3]); if(!inp) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(7);
for(i=0;i<m;i++) inp>>x[i]>>y[i];
inp.close();
for(i=0;i<m;i++) cout<<x[i]<<'\t'<<y[i]<<endl;
ofstream out(nm[2],ios::binary);
if(!out) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(8);
for(i=0;i<N;i++)
 {
 in.read((char*)&X,sizeof(float));
 in.read((char*)&Y,sizeof(float));
 min=(X-x[0])*(X-x[0])+(Y-y[0])*(Y-y[0]),R=0;
 for(j=1;j<m;j++)
  { r=(X-x[j])*(X-x[j])+(Y-y[j])*(Y-y[j]);
  if(r<min) min=r,R=j;
  }
 c=R+h; out.put(c);
 }
out.close(); in.close(); delete [] x; delete [] y; return 0;
}
```

Программа демодулятора для среды Geany в Linux

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cstring>
using namespace std;
int main(int mn,char* nm[])
{ long j,i,N,m,R; char h,c;
float r,*x,*y,X,Y,min;
if(mn!=4) cerr<<"./dem mod.qfb dem.ct mod.txt\n",exit(1);
```

```
if(nm[2][strlen(nm[2])-1]=='t') h=0x30; else h=0;
ifstream in(nm[1],ios::binary);
if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end); N=in.tellg()/8; in.seekg(0,ios::beg);
cout <<"N="<<N<<endl;
ifstream inp(nm[3]); if(!inp) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(3);
for(m=-1;!inp.eof();m++) inp>>r;
if(m&1) cerr<<"Нечётное количество координат!\n",exit(4);
m/=2; inp.close();
cout<<"m="<<m<<endl;
x=new float[m]; if(!x) cerr<<"Нет памяти для массива x!\n",exit(5);
y=new float[m]; if(!y) cerr<<"Нет памяти для массива y!\n",exit(6);
inp.open(nm[3]); if(!inp) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(7);
for(i=0;i<m;i++) inp>>x[i]>>y[i];
inp.close();
for(i=0;i<m;i++) cout<<x[i]<<'\t'<<y[i]<<endl;
ofstream out(nm[2],ios::binary);
if(!out) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(8);
for(i=0;i<N;i++)
 {
 in.read((char*)&X,sizeof(float));
 in.read((char*)&Y,sizeof(float));
 min=(X-x[0])*(X-x[0])+(Y-y[0])*(Y-y[0]),R=0;
 for(j=1;j<m;j++)
  { r=(X-x[j])*(X-x[j])+(Y-y[j])*(Y-y[j]);
  if(r<min) min=r,R=j;
  }
 c=R+h; out.put(c);
 }
out.close(); in.close(); delete [] x; delete [] y; return 0;
}
```

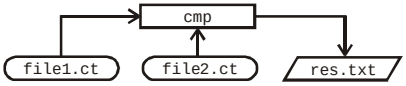
//Преобразователь двоичных символов в m-позиционные, где $m = 2^N$

```
#include<iostream>
#include<fstream>
using namespace std;
void ExeFile(char*& n)
{ int j,k=0; for(;n[k];k++); for(k--;k>=0;--k) if(n[k]=='\\') break;
for(j=0;n[++k];n[j++]=n[k]); n[j]=0;
}
int main(int mn,char* nm[])
{
char h1,h2,t,c;
long i,j,N,size;
ExeFile(nm[0]);
if(mn!=4) cerr<<nm[0]<<" in(2).ct(.cb) out(N).ct(.cb) N\n",exit(1);
ifstream in(nm[1],ios::binary); if(!in) cerr<<"file \""<<nm[1]<<"\" not open!\n",exit(1);
in.seekg(0,ios::end);size=in.tellg();in.seekg(0,ios::beg);
if((nm[1][strlen(nm[1])-1]|0x20)=='t') h1=0x30; else h1=0;
if((nm[2][strlen(nm[2])-1]|0x20)=='t') h2=0x30; else h2=0;
N=atol(nm[3]);  if(N>256) cerr<<" N>256 не реализован\n",exit(1);
if(h2&&N>3)
 h2=0,cerr<<"тип выходного файла изменён на \'cb\'\n",nm[2][strlen(nm[2])-1]='b';
cout<<nm[2]<<endl;
ofstream ou(nm[2],ios::binary); if(!ou) cerr<<"file \""<<nm[1]<<"\" not open!\n",exit(1);
if(size%N) cerr<<"количество входных символов не кратно "<<N<<"\n";
for(i=0;i<size/N;i++,ou.put(char(t+h2))) for(t=j=0;j<N;j++) c=in.get()-h1,t<<=1,t|=c;
ou.close();in.close();
return 0;
}
```

//Преобразователь *m*-позиционных символов в двоичные, где $m = 2^N$

```
#include<iostream>
#include<fstream>
using namespace std;
void ExeFile(char*& n)
{ int j,k=0; for(;n[k];k++); for(k--;k>=0;--k) if(n[k]=='\\') break;
for(j=0;n[++k];n[j++]=n[k]); n[j]=0;
}
template<class TYPE> TYPE powq(TYPE a,int k)
{ TYPE b=1; while(k) { if(k%2) b*=a; k/=2; a*=a; } return b;
}
int main(int mn,char* nm[])
{
char h1,h2,c;
long t,i,j,N,size;
ExeFile(nm[0]);
if(mn!=4) cerr<<nm[0]<<" in(N).ct(.cb) out(2).ct(.cb) N\n",exit(1);
ifstream in(nm[1],ios::binary); if(!in) cerr<<"file \""<<nm[1]<<"\" not open!\n",exit(1);
ofstream ou(nm[2],ios::binary); if(!ou) cerr<<"file \""<<nm[1]<<"\" not open!\n",exit(1);
in.seekg(0,ios::end);size=in.tellg();in.seekg(0,ios::beg);
if((nm[1][strlen(nm[1])-1]|0x20)=='t') h1=0x30; else h1=0;
if((nm[2][strlen(nm[2])-1]|0x20)=='t') h2=0x30; else h2=0;
N=atol(nm[3]);  if(N>256) cerr<<" N>256 не реализован\n",exit(1);
for(i=0;i<size;i++) for(c=in.get()-h1,t=1,t<<=N-1,j=0;j<N;j++,t>>=1)
ou.put(char(!!(t&c)+h2));
ou.close();in.close();
return 0;
}
```

Тестирование сравнивающего устройства



Ниже приведён листинг программы, реализующей работу сравнивающего устройства

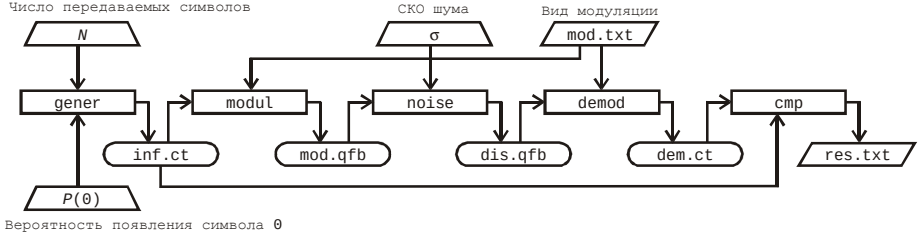
```
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
int main(int mn,char** nm)
{ long i,N1,N2,err=0; char h1,h2;
strlwr(nm[1]); strlwr(nm[2]);
if(mn!=3) cerr<<"cmp.exe inf1.ct(cb) inf2.ct(cb)\a\a"<<endl,exit(1);
ifstream in1(nm[1],ios::binary);
if(!in1) cerr<<"Файл "<<nm[1]<<" не открыт!\a\a"<<endl,exit(2);
in1.seekg(0,ios::end); N1=in1.tellg(); in1.seekg(0,ios::beg);
ifstream in2(nm[2],ios::binary);
if(!in2) cerr<<"Файл "<<nm[1]<<" не открыт!\a\a"<<endl,exit(3);
in2.seekg(0,ios::end); N2=in2.tellg(); in2.seekg(0,ios::beg);
if(nm[1][strlen(nm[1])-1]=='t') h1=0x30; else h1=0;
if(nm[2][strlen(nm[2])-1]=='t') h2=0x30; else h2=0;
if(N1!=N2) cerr << "Сравниваемые файлы разной длины!\n";
if(N2<N1) N1=N2;
for(i=0;i<N1;i++)
 if(in1.get()-h1!=in2.get()-h2) err++;
cout<<"Ошибок: "<<err<<" из "<<N1<<endl;
return 0;
}
```

Программа сравнивающего устройства для среды Geany в Linux

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cstring>
using namespace std;
int main(int mn,char** nm)
{ long i,N1,N2,err=0; char h1,h2;
if(mn!=3) cerr<<"./cmp inf1.ct(cb) inf2.ct(cb)\a\a"<<endl,exit(1);
ifstream in1(nm[1],ios::binary);
if(!in1) cerr<<"Файл "<<nm[1]<<" не открыт!\a\a"<<endl,exit(2);
in1.seekg(0,ios::end); N1=in1.tellg(); in1.seekg(0,ios::beg);
ifstream in2(nm[2],ios::binary);
if(!in2) cerr<<"Файл "<<nm[1]<<" не открыт!\a\a"<<endl,exit(3);
in2.seekg(0,ios::end); N2=in2.tellg(); in2.seekg(0,ios::beg);
if(nm[1][strlen(nm[1])-1]=='t') h1=0x30; else h1=0;
if(nm[2][strlen(nm[2])-1]=='t') h2=0x30; else h2=0;
if(N1!=N2) cerr << "Сравниваемые файлы разной длины!\n";
if(N2<N1) N1=N2;
for(i=0;i<N1;i++)
 if(in1.get()-h1!=in2.get()-h2) err++;
cout<<"Ошибок: "<<err<<" из "<<N1<<endl;
return 0;
}
```

МОДЕЛЬ ГАУССОВСКОГО ШУМА

Блочная схема для включения устройства, добавляющего гаусовский шум к сигналу



Реализация шума на основе центральной предельной теоремы с ограничением суммы

$$\zeta = \sum_{i=1}^{12} \xi_i ,$$

где $\xi_i = (-0{,}5 \ldots 0{,}5)$,

Дисперсия суммарной (гауссовской) случайной величины

$$D_{\xi_i} = \int_{-\infty}^{\infty} w(x) x^2 dx = \int_{-0{,}5}^{0{,}5} 1 \cdot x^2 dx = \frac{x^3}{3}\Big|_{-0{,}5}^{0{,}5} = \frac{1}{3}\left(\frac{1}{8} + \frac{1}{8}\right) = \frac{1}{12},$$

$$D_\zeta = 1.$$

Ниже приведён листинг программы, реализующей работу устройства, добавляющего гауссовский шум

```
#include <fstream.h>
#include <stdlib.h>
float noise(float s)
{ long r; int i;
for(r=-32768l*6,i=0;i<12;i++) r+=rand();
s*=r/32768.; return s;
}

int main(int mn, char* nm[])
{
long i,size;
if(mn!=4) cerr<<"\nnoise.exe in.qfb out.qfb СКО\n",exit(1);
randomize();
float x,y,s=atof(nm[3]);
ifstream in(nm[1],ios::binary);
if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end);size=in.tellg()/8;in.seekg(0,ios::beg);
ofstream out(nm[2],ios::binary);
if(!out) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(3);
for(i=0;i<size;i++)
 {
 in.read((char*)&x,4);in.read((char*)&y,4);
 x+=noise(s); y+=noise(s);
 out.write((char*)&x,4);out.write((char*)&y,4);
 }
out.close(); in.close();
}
```

Программа реализующая работу устройства, добавляющего гауссовский шум, для среды Geany в Linux

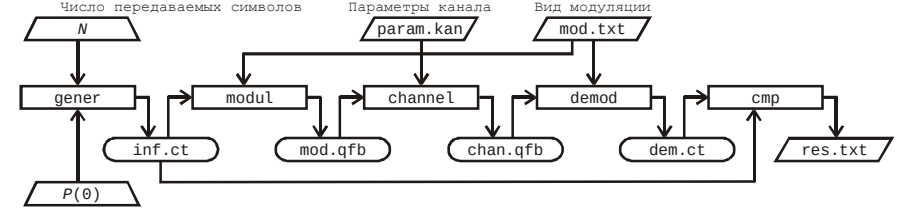
```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
using namespace std;
float noise(float s)
{ int i; int64_t r;
for(r=(-RAND_MAX-1)*6ll,i=0;i<12;i++) r+=rand();
s*=r/(RAND_MAX+1.); return s;
}
int main(int mn, char* nm[])
{ long i,size;
struct timespec z;
clock_gettime(CLOCK_REALTIME,&z);srand(z.tv_nsec);
if(mn!=4) cerr<<"./noise in.qfb out.qfb CKO\n",exit(1);
float x,y,s=atof(nm[3]);
ifstream in(nm[1],ios::binary);
if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end);size=in.tellg()/8;in.seekg(0,ios::beg);
ofstream out(nm[2],ios::binary);
if(!out) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(3);
for(i=0;i<size;i++)
 { in.read((char*)&x,4);in.read((char*)&y,4);
 x+=noise(s); y+=noise(s);
 out.write((char*)&x,4);out.write((char*)&y,4);
 }
out.close(); in.close();
}
```

Программа реализующая работу устройства, добавляющего гауссовский шум, для среды Geany в Linux используя генератор на основе числе Мерсена.

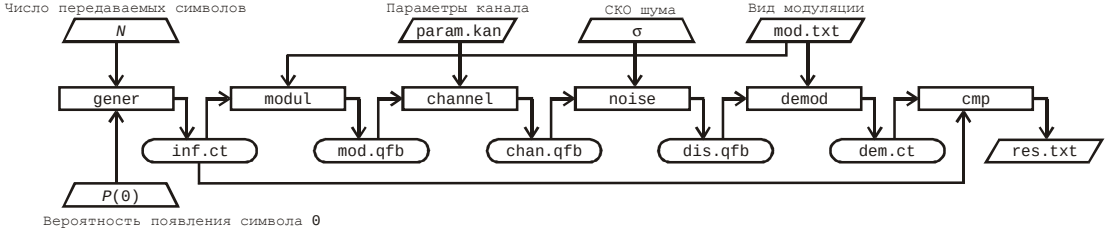
```
#include <iostream>
#include <fstream>
#include <random>
using namespace std;
int main(int mn, char* nm[])
{ long i,size;
random_device rd;
mt19937 gen(rd());
normal_distribution<> dist(0,1);
if(mn!=4) cerr<<"./noise in.qfb out.qfb h^2\n",exit(1);
float x,y,s=atof(nm[3]);s=1/sqrt(2*s);
ifstream in(nm[1],ios::binary);
if(!in) cerr<<"file \""<<nm[1]<<"\" нnot open!\n",exit(2);
in.seekg(0,ios::end);size=in.tellg()/8;in.seekg(0,ios::beg);
ofstream out(nm[2],ios::binary);
if(!out) cerr<<"file \""<<nm[2]<<"\" not openн!\n",exit(3);
for(i=0;i<size;i++)
 { in.read((char*)&x,4);in.read((char*)&y,4);
 x+=dist(gen)*s;  y+=dist(gen)*s;
 out.write((char*)&x,4);out.write((char*)&y,4);
 }
out.close(); in.close();
}
```

5. Модель многолучевого канала

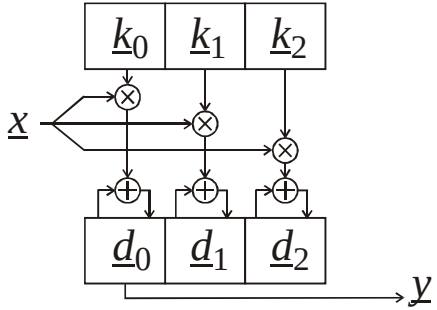
Блочная схема для включения модели канала



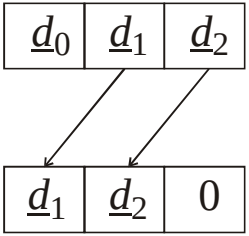
Блочная схема для включения модели канала и устройства, добавляющего гаусовский шум к сигналу



модель многолучевого канала на базе
линии задержки квадратурных компонент сигнала



сдвиг квадратурных компонент сигнала
в линии задержки



Ниже приведён листинг программы, реализующей работу многолучевого канала

```
#include <fstream.h>
#include <stdlib.h>
int main(int mn, char* nm[])
{
long i,j,k,size;
if(mn!=4) cerr<<"noise in.qfb out.qfb path.txt\n",exit(1);
float x,y,*px,*py,*dx,*dy;
ifstream ip(nm[3]),ir(nm[3]),in(nm[1],ios::binary);
if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
if(!ir||!ip) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end);size=in.tellg()/8;in.seekg(0,ios::beg);
ofstream out(nm[2],ios::binary);
if(!out) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(3);
for(k=-1;!ip.eof();k++) ip>>x;
if(k&1) cerr<<"Нечётное количество чисел в \""<<nm[3]<<"\"\n",exit(4);
k/=2; cout<<"k="<<k<<endl;
px=new float[k];py=new float[k];dx=new float[k];dy=new float[k];
if(!px||!py||!dx||!dy) cerr<<"Недостаточно памяти!\n",exit(5);
for(i=0;i<k;i++) ir>>px[i]>>py[i],dx[i]=dy[i]=0;
for(i=0;i<k;i++) cout<<px[i]<<'\t'<<py[i]<<endl;
for(j=0;j<size;j++)
 { in.read((char*)&x,4);in.read((char*)&y,4);
 for(i=0;i<k;i++) dx[i]+=x*px[i]-y*py[i],dy[i]+=y*px[i]+x*py[i];
 out.write((char*)&dx[0],4);out.write((char*)&dy[0],4);
 for(i=1;i<k;i++) dx[i-1]=dx[i],dy[i-1]=dy[i]; dx[i-1]=dy[i-1]=0;
 }
out.close(); in.close();
delete [] px; delete [] py; delete [] dx; delete [] dy;
}
```

Программа, реализующая работу многолучевого канала, для среды Geany в Linux

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
int main(int mn, char* nm[])
{
long i,j,k,size;
if(mn!=4) cerr<<"\n./noise in.qfb out.qfb path.txt\n",exit(1);
float x,y,*px,*py,*dx,*dy;
ifstream ip(nm[3]),ir(nm[3]),in(nm[1],ios::binary);
if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
if(!ir||!ip) cerr<<"Файл \""<<nm[3]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end);size=in.tellg()/8;in.seekg(0,ios::beg);
ofstream out(nm[2],ios::binary);
if(!out) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(3);
for(k=-1;!ip.eof();k++) ip>>x;
if(k&1) cerr<<"Нечётное количество чисел в \""<<nm[3]<<"\"\n",exit(4);
k/=2; cout<<"k="<<k<<endl;
px=new float[k];py=new float[k];dx=new float[k];dy=new float[k];
if(!px||!py||!dx||!dy) cerr<<"Недостаточно памяти!\n",exit(5);
for(i=0;i<k;i++) ir>>px[i]>>py[i],dx[i]=dy[i]=0;
for(i=0;i<k;i++) cout<<px[i]<<'\t'<<py[i]<<endl;
for(j=0;j<size;j++)
 { in.read((char*)&x,4);in.read((char*)&y,4);
 for(i=0;i<k;i++) dx[i]+=x*px[i]-y*py[i],dy[i]+=y*px[i]+x*py[i];
 out.write((char*)&dx[0],4);out.write((char*)&dy[0],4);
 for(i=1;i<k;i++) dx[i-1]=dx[i],dy[i-1]=dy[i]; dx[i-1]=dy[i-1]=0;
 }
out.close(); in.close(); delete [] px; delete [] py; delete [] dx; delete [] dy;
}
```

Помехоустойчивое кодирование, код Хемминга (7,4)

Линейный код (7, 4) построен по матрице

$$\mathbf{G}=\begin{vmatrix}1&0&0&0&0&1&1\\0&1&0&0&1&0&1\\0&0&1&0&1&1&0\\0&0&0&1&1&1&1\end{vmatrix}.$$

Составим проверочную матрицу и покажем процесс исправления ошибок в произвольном разряде корректирующего кода, информационная часть которого представляет собой 4-разрядные комбинации примитивного двоичного кода.

Кодовые комбинации формируются согласно выражению $\mathbf{b}=\mathbf{aG}$, где $\mathbf{a}$ — 4-разрядные комбинации примитивного двоичного кода.

Согласно теории, из

$$\mathbf{G}=\begin{vmatrix}1&0&0&\ldots&0&0&\gamma_{1,k+1}&\gamma_{1,k+2}&\ldots&\gamma_{1,k+r}\\0&1&0&\ldots&0&0&\gamma_{2,k+1}&\gamma_{2,k+2}&\ldots&\gamma_{2,k+r}\\0&0&1&\ldots&0&0&\gamma_{3,k+1}&\gamma_{3,k+2}&\ldots&\gamma_{3,k+r}\\\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots\\0&0&0&\ldots&0&1&\gamma_{k,k+1}&\gamma_{k,k+2}&\ldots&\gamma_{k,k+r}\end{vmatrix}=\left|\mathbf{1}_k,\boldsymbol{\gamma}\right|$$

следует

$$\mathbf{H}=\begin{vmatrix}\gamma_{1,k+1}&\gamma_{2,k+1}&\ldots&\gamma_{k,k+1}&1&0&\ldots&0\\\gamma_{1,k+2}&\gamma_{2,k+2}&\ldots&\gamma_{k,k+2}&0&1&\ldots&0\\\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots\\\gamma_{1,k+r}&\gamma_{2,k+r}&\ldots&\gamma_{k,k+r}&0&0&\ldots&1\end{vmatrix}=\left|\boldsymbol{\gamma}^T,\mathbf{1}_r\right|.$$

В нашем случае

$$\mathbf{H}=\begin{vmatrix}0&1&1&1&1&0&0\\1&0&1&1&0&1&0\\1&1&0&1&0&0&1\end{vmatrix}$$

Допустим, что была передана комбинация **b**=|1000011|. Если ошибок в ней нет, то выполняется соотношение $\mathbf{bH}^T=0$.

Если в каком-либо, разряде происходит ошибка, то это условие не выполняется. По виду результата в этом случае можно указать разряд, в котором произошла ошибка. Например, при ошибке в первом разряде для $\mathbf{bH}^T$ получаем результат 001 для любой кодовой комбинации.

Код Хемминга (7,4), режим исправления ошибок.

Для кода с производящей матрицей

$$\mathbf{G}=\begin{vmatrix}1&0&0&0&1&1&1\\0&1&0&0&1&1&0\\0&0&1&0&1&0&1\\0&0&0&1&0&1&1\end{vmatrix}$$

составим таблицу синдромов и покажем, каким ошибочным разрядам они соответствуют. Проверочная матрица:

$$\mathbf{H}=\begin{vmatrix}1&1&1&0&1&0&0\\1&1&0&1&0&1&0\\1&0&1&1&0&0&1\end{vmatrix}$$

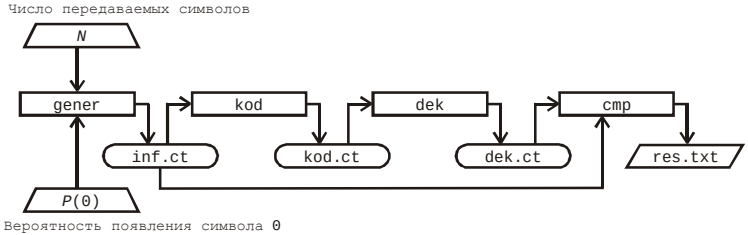
Произведение $\mathbf{bH}^T$, где в **b** для нулевой комбинации сделаны ошибки, даёт следующие результаты:

1000000 → 111
0100000 → 110
0010000 → 101
0001000 → 011
0000100 → 100
0000010 → 010
0000001 → 001

Синдромы, соответствующие одиночным ошибкам в различных разрядах кодовых комбинаций, приведены в таблице:

Синдром	001	010	011	100	101	110	111
Разряд кодовой комбинации подлежащий исправлению	7	6	4	5	3	2	1

БЛОЧНАЯ СХЕМА ПРОВЕРКИ КОДЕРА И ДЕКОДЕРА



Кодер Хемминга (7,4)

```
#include<stdlib.h>
#include<fstream.h>
#include<string.h>
int main(int mn, char* nm[])
{ int i,j,k,s,t;
char h1,h2,a[4],b[7],G[4][7]=
{1,0,0,0,0,1,1, 0,1,0,0,1,0,1, 0,0,1,0,1,1,0, 0,0,0,1,1,1,1};
strlwr(nm[1]),strlwr(nm[2]);
if(mn!=3) cerr<<"kod.exe in.ct out.ct\n",exit(1);
ifstream in(nm[1],ios::binary); if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end);s=in.tellg();in.seekg(0,ios::beg);
ofstream ou(nm[2],ios::binary); if(!ou) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(3);
if(nm[1][strlen(nm[1])-1]=='t') h1=0x30; else h1=0;
if(nm[2][strlen(nm[2])-1]=='t') h2=0x30; else h2=0;
for(t=s%4,s/=4,k=0;k<s;k++)
 {
 for(i=0;i<4;a[i++]=in.get()-h1);
 for (i=0;i<7;i++) for (b[i]=j=0;j<4;j++) b[i]^=a[j]&G[j][i];
 for(i=0;i<7;i++) ou.put(b[i]+h2);
 }
if(t)
 {
 for(i=0;i<t;a[i++]=in.get()-h1);
 for(i=0;i<4;a[i++]=0);
 for (i=0;i<7;i++) for (b[i]=j=0;j<4;j++) b[i]^=a[j]&G[j][i];
 for(i=0;i<7;i++) ou.put(b[i]+h2);
 }
ou.close(),in.close(); return 0;
}
```

Декодер Хемминга (7,4), работающего в режиме исправления ошибок

```
#include<stdlib.h>
#include<fstream.h>
#include<string.h>
int main(int mn, char* nm[])
{ int  i,j,k,s,l,e[8]={0};
char h1,h2,b[7],c[3],HT[7][3]={0,1,1, 1,0,1, 1,1,0, 1,1,1, 1,0,0, 0,1,0, 0,0,1};
strlwr(nm[1]),strlwr(nm[2]);
if(mn!=3) cerr<<"dek.exe in.ct out.ct\n",exit(1);
ifstream in(nm[1],ios::binary); if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end);s=in.tellg();in.seekg(0,ios::beg);
ofstream ou(nm[2],ios::binary); if(!ou) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(3);
if(nm[1][strlen(nm[1])-1]=='t') h1=0x30; else h1=0;
if(nm[2][strlen(nm[2])-1]=='t') h2=0x30; else h2=0;
for(j=0;j<7;e[k]=j++) for(k=i=0;i<3;i++) k|=(HT[j][i]<<i);
for(s/=7,l=0;l<s;l++)
 {
 for(i=0;i<7;b[i++]=in.get()-h1);
 for(i=0;i<3;i++) for(c[i]=j=0;j<7;j++) c[i]^=b[j]&HT[j][i];
 for(k=i=0;i<3;i++) k|=(c[i]<<i);
 if(k) b[e[k]]=!b[e[k]];
 for(i=0;i<4;i++) ou.put(b[i]+h2);
 }
ou.close(),in.close(); return 0;
}
```

Кодер Хемминга (7,4) для среды Geany в Linux

```
#include<cstdlib>
#include<iostream>
#include<fstream>
#include<cstring>
using namespace std;
int main(int mn, char* nm[])
{ int i,j,k,s,t; char h1,h2,a[4],b[7],G[4][7]=
{{1,0,0,0,0,1,1}, {0,1,0,0,1,0,1}, {0,0,1,0,1,1,0}, {0,0,0,1,1,1,1}};
if(mn!=3) cerr<<"./kod in.ct out.ct\n",exit(1);
ifstream in(nm[1],ios::binary); if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end);s=in.tellg();in.seekg(0,ios::beg);
ofstream ou(nm[2],ios::binary); if(!ou) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(3);
if(nm[1][strlen(nm[1])-1]=='t') h1=0x30; else h1=0;
if(nm[2][strlen(nm[2])-1]=='t') h2=0x30; else h2=0;
for(t=s%4,s/=4,k=0;k<s;k++)
 { for(i=0;i<4;a[i++]=in.get()-h1);
 for (i=0;i<7;i++) for (b[i]=j=0;j<4;j++) b[i]^=a[j]&G[j][i];
 for(i=0;i<7;i++) out.put(b[i]+h2);
 }
if(t)
 { for(i=0;i<t;a[i++]=in.get()-h1);
 for(i=0;i<4;a[i++]=0);
 for (i=0;i<7;i++) for (b[i]=j=0;j<4;j++) b[i]^=a[j]&G[j][i];
 for(i=0;i<7;i++) ou.put(b[i]+h2);
 }
ou.close(),in.close(); return 0;
}
```

Декодер Хемминга (7,4), работающего в режиме исправления ошибок, для среды Geany в Linux

```
#include<cstdlib>
#include<iostream>
#include<fstream>
#include<cstring>
using namespace std;
int main(int mn, char* nm[])
{ int  i,j,k,s,l,e[8]={0};
char h1,h2,b[7],c[3],HT[7][3]={{0,1,1},{1,0,1},{1,1,0},{1,1,1},{1,0,0},{0,1,0},{0,0,1}};
if(mn!=3) cerr<<"dek.exe in.ct out.ct\n",exit(1);
ifstream in(nm[1],ios::binary);if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
in.seekg(0,ios::end);s=in.tellg();in.seekg(0,ios::beg);
ofstream ou(nm[2],ios::binary);if(!ou) cerr<<"Файл \""<<nm[2]<<"\" не создан!\n",exit(3);
if(nm[1][strlen(nm[1])-1]=='t') h1=0x30; else h1=0;
if(nm[2][strlen(nm[2])-1]=='t') h2=0x30; else h2=0;
for(j=0;j<7;e[k]=j++) for(k=i=0;i<3;i++) k|=(HT[j][i]<<i);
for(s/=7,l=0;l<s;l++)
 { for(i=0;i<7;b[i++]=in.get()-h1);
 for(i=0;i<3;i++) for(c[i]=j=0;j<7;j++) c[i]^=b[j]&HT[j][i];
 for(k=i=0;i<3;i++) k|=(c[i]<<i); if(k) b[e[k]]=!b[e[k]];
 for(i=0;i<4;i++) ou.put(b[i]+h2);
 }
ou.close(),in.close(); return 0;
}
```

Вероятность ошибки для оптимального приёма сигналов ФМ-2 согласно критерию Котельникова для заданных априорных вероятностей передачи символов $P(0)$ **и** $P(1)=1-P(0)$**:**

$$p_{\text{ош дем}} = P(0)Q\left(\frac{a-U_0}{\sigma}\right)+P(1)Q\left(\frac{a+U_0}{\sigma}\right),$$

$U_0 = \frac{\sigma^2}{2a}\ln\frac{P(1)}{P(0)}$ **— порог принятия решения согласно критерию Котельникова,**

a **— амплитуда сигнала ФМ-2 (нормированное значение равно 1),**

$\sigma^2 = \frac{1}{\sqrt{2h^2}}$ **— дисперсия гауссовского шума,**

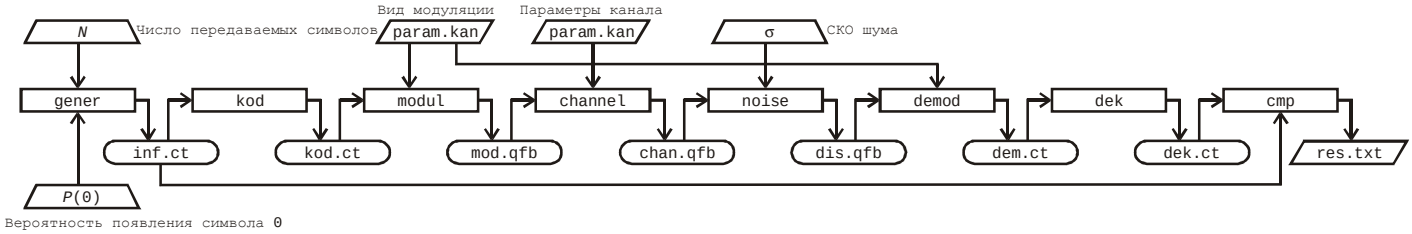
h^2 **— отношение сигнал-шум.**

Вероятность ошибки для декодирования систематического линейного блочного кода Хемминга (7,4), работающего в режиме исправления ошибки

$$p_{\text{ош дек}} = 9p^2 - 26p^3 + 38p^4 - 12p^5,$$

p **— вероятность ошибки в канале связи (вероятность ошибки демодуляции).**

Блочная схема включения кодера и декодера в систему связи.



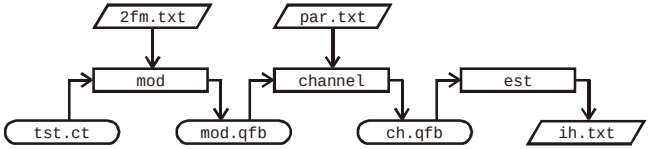
Оценивание параметров канала связи

Оцениватель построен на базе обучающей последовательности из стандарта GSM
00100101110000100010010111
Такая обучающая последовательность позволяет оценивать ИХ рассеянием до 6*T*.
Принцип оценивания:

```
00100 10111000010000100 10111
++-++ -+---++++-+++-++ -+---

Сдвиг 0:
++-++-+---++++-+++-++-+---
     -+---++++-+++-++
     +++++++++++++++ = 16
Сдвиг 1
++-++-+---++++-+++-++-+---
      -+---++++-+++-++
      --++-+++--++--+- = 0
Сдвиг 2
++-++-+---++++-+++-++-+---
       -+---++++-+++-++
       +-+--++-+-+-+--+ = 0
Сдвиг 3
++-++-+---++++-+++-++-+---
        -+---++++-+++-++
        +----+-++--++++- = 0
Сдвиг 4
++-++-+---++++-+++-++-+---
         -+---++++-+++-++
         ++----++++++---- = 0
Сдвиг 5
++-++-+---++++-+++-++-+---
          -+---++++-+++-++
          -+--++++--+-++-- = 0
```

БЛОЧНАЯ СХЕМА ПРОВЕРКИ ОЦЕНИВАТЕЛЯ



Листинг оценивателя

```
//00100101110000100010010111
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
int main(int mn,char* nm[])
{
long j,i,k,N;
float x[26],y[26],X[6],Y[6];
float u[16]={-1,1,-1,-1,-1,1,1,1,1,-1,1,1,1,-1,1,1};
randomize();
strlwr(nm[1]);
if(mn!=3) cerr<<"est.exe in.qfb ih.txt\n",exit(1);
ifstream in(nm[1],ios::binary); if(!in) cerr<<"Файл \""<<nm[1]<<"\" не открыт!\n",exit(2);
for(i=0;i<26;i++) in.read((char*)&x[i],4),in.read((char*)&y[i],4);
ofstream out(nm[2],ios::binary); if(!out) cerr<<"Файл \""<<nm[2]<<"\" не
открыт!\n",exit(8);
for(i=0;i<6;i++) X[i]=Y[i]=0;
for(i=0;i<6;i++) for(j=i+5,k=0;k<16;k++,j++) X[i]+=x[j]*u[k],Y[i]+=y[j]*u[k];
for(i=0;i<6;i++) X[i]/=16,Y[i]/=16;
for(i=0;i<6;i++) out<<X[i]<<'\t'<<Y[i]<<endl;
out.close();
in.close();
return 0;
}
```

Оценивание импульсной характеристики (ИХ) можно производить на основе метода минимума среднеквадратической ошибки, то есть в матричном виде можно описать принимаемую информационную комбинация следующим образом:

$$\dot{\mathbf{Z}}_i = \dot{\mathbf{D}}\dot{\mathbf{G}}_i + \dot{\mathbf{N}}_i,$$

или

$$\begin{bmatrix} \dot{Z}_1 \\ \dot{Z}_2 \\ \dots \\ \dot{Z}_M \end{bmatrix} = \begin{bmatrix} \dot{D}_{11} & 0 & \dots & 0 \\ \dot{D}_{21} & \dot{D}_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \dot{D}_{M,Q+1} \end{bmatrix} \times \begin{bmatrix} \dot{G}_1 \\ \dot{G}_2 \\ \dots \\ \dot{G}_{Q+1} \end{bmatrix} + \begin{bmatrix} \dot{N}_1 \\ \dot{N}_2 \\ \dots \\ \dot{N}_M \end{bmatrix},$$

где $\dot{\mathbf{Z}}$ — матрица-столбец представляющую собой упорядоченную последовательность входного сигнала; $\dot{G}_i$ — набор известных элементов ИХ, $\dot{D}_{i,k}$ — образуют матрицу информационных элементов $\dot{\mathbf{D}}$; $\dot{\mathbf{N}}$ — матрица-столбец описывающая собой шум в канале.

Используя метод минимума среднеквадратической ошибки матрицу $\dot{G}_k$ можно вычислить по формуле:

$$\hat{\dot{G}} = (\dot{D}^{\mathrm{T}}\dot{D})^{-1}\dot{D}^{\mathrm{T}}\dot{Z}_k.$$

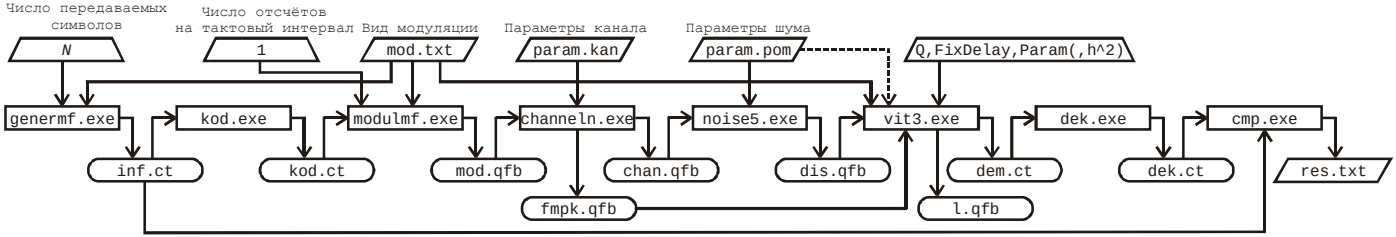
Здесь $\hat{\dot{G}}$ рассматривается как оценка ИХ. Матрица $(\dot{D}^{\mathrm{T}}\dot{D})^{-1}\dot{D}^{\mathrm{T}}$ известна, и её можно вычислить:

$$\dot{F} = (\dot{D}^{\mathrm{T}}\dot{D})^{-1}\dot{D}^{\mathrm{T}}$$

Следовательно:

$$\hat{\dot{\mathbf{G}}} = \dot{\mathbf{F}}\dot{\mathbf{Z}}$$

Блочная схема системы связи.



Файл формата *.kan

```
2       // количество_лучей
4       // порядок_интерполяции для замираний
1000    // произведение длительности посылки на частоту несущей (ед) (f0/fд)
1       // направление_потока для ФМПК (0 - входной[N/R], 1 - выходной(без заголовка), 2 -
выходной(заголовок 16bit), 3 - выходной(заголовок 32bit), 4 - выходной(заголовок 64bit) )
1       // число_sample в течение которого ИХ=const[N/R]
0       // Srand 0, 1, .. - srand(), -1 - randomize()
0       // вывод_выходных_данных_в_текстовом_виде : 1 - вывод, 0 - запрет вывода
1       // форма_чтения_данных_из_файла_ФМПК  0 - МЗИХ (мгновенные значения ИХ)[N/R], 1 -
МЗСК (мгновенные значения состояния канала)
0       // тип_интерполяции 0 - по Котельникову, 1 - по Лагранжу
1       // тест_float [N/R]

[луч_№0]
0       // задержка (в sample)
0.0     // доплеровский_коэффициент (v/c)
1.0     // mx математическое_ожидание_компоненты_x
1.0     // my математическое_ожидание_компоненты_y
0.0     // бx СКО_компоненты_x
0.0     // бy СКО_компоненты_y
```

```
100    // квазиполупериод_замираний
1.0 0.0   0.0 0.0   // коэффициенты_нерекурсивной_ветви_ОФ  a
0.0 0.0  -0.0 0.0   // коэффициенты_рекурсивной_ветви_ОФ    b
[end]

[луч_№1]
1      // задержка (в sample)
0.0     // доплеровский_коэффициент (v/c)
1.0     // mx математическое_ожидание_компоненты_x
1.0     // my математическое_ожидание_компоненты_y
0.0     // бx СКО_компоненты_x
0.0     // бy СКО_компоненты_y
100    // квазиполупериод_замираний
1.0 0.0   0.0 0.0    // коэффициенты_нерекурсивной_ветви_ОФ
0.0 0.0  -0.0 0.0   // коэффициенты_рекурсивной_ветви_ОФ
[end]
```

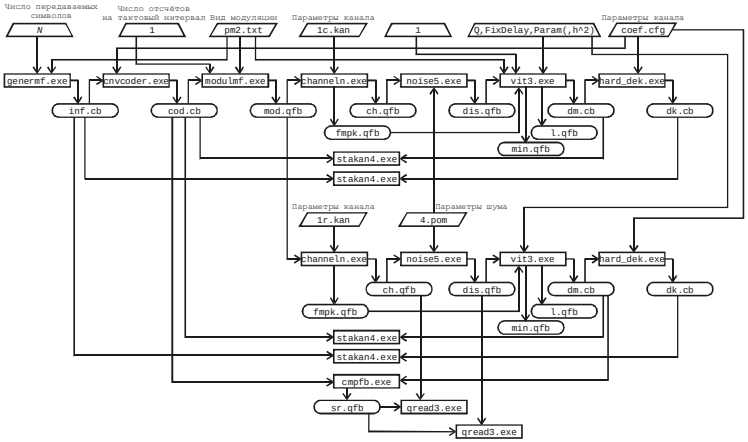
Файл формата *.pom

```
-1      // отношение_сигнал/шум h2
1        // СКО_шума
-1       // Srand  - srand(), -1 - randomize()
0       // вывод_выходных_данных_в_текстовом_виде : 1 - вывод, 0 -запрет вывода
```

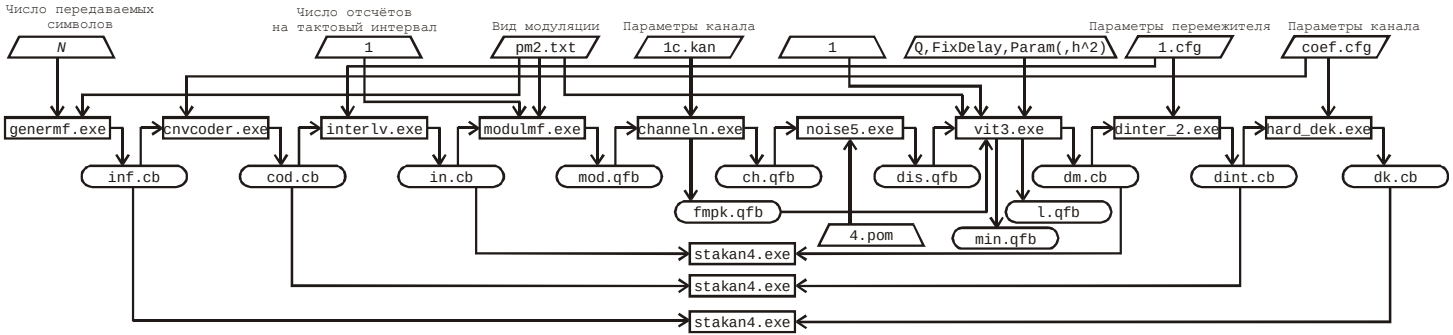
Демодулятор vit3.exe

```
vit3.exe z.qfb ou.cb(ct) fmpk.qfb Q sm.txt noise.pom|h^2 min.fb FixDelay Param l.fb
<ou2.cb(ct) fmpk2.qfb noise2.pom|h^2 <...>>
             Демодулятор
z.qfb        - файл входного синала;
ou.cb(ct)    - файл решений (два возможных формата "cb" и "ct");
fmpk.qfb     - файл мгновенных параметров канала (16-битный заголовок);
Q            - интервал анализа - 1 (память канала);
sm.txt       - файл сигналов модуляции;
noise.pom|h^2 - файл параметров шума (расширение ".pom") или величина h^2;
```

```
min.fb          - файл с минимальными метриками (при отсутствии фиксированной
                  задержки сбрасываются текущие значения задержки);
FixDelay        - фиксированная задержка >= Q
                  (-1 - отсутствие фиксированной задержки);
Param  bit0=1 - тест входного файла на float;
       bit1=1 - рекуррентное вычисление метрик;
       bit2=1 - отбрасывание траекторий по АКН;
       bit3=1 - SOVA: 0 - exp, 1 - max;
l.fb            - файл надежностей.
vit3.exe z.cb(ct) ou.cb(ct) cod.txt noise.pom|h^2 m min.fb FixDelay Param l.fb
                Декодер жестких решений
z.cb(ct)        - файл входного синала (два возможных формата "cb" и "ct");
ou.cb(ct)       - файл решений (два возможных формата "cb" и "ct");
cod.txt         - файл с таблицей коэффициентов свертосного кода;
noise.pom|h^2 - файл параметров шума (расширение ".pom") или величина h^2;
m               - позиционность кода;
min.fb          - файл с минимальными метриками (при отсутствии фиксированной
                  задержки сбрасываются текущие значения задержки);
FixDelay        - фиксированная задержка >= Q
                  (-1 - отсутствие фиксированной задержки);
Param  bit0=1 - тест входного файла на float;
       bit1=1 - рекуррентное вычисление метрик;
       bit2=1 - отбрасывание траекторий по АКН;
       bit3=1 - SOVA: 0 - exp, 1 - max;
l.fb            - файл надежностей.
```




```
genermf.exe pm2.txt inf.cb 10000
cnvcoder.exe inf.cb cod.cb 0 coef.cfg
modulmf.exe cod.cb mod.qfb pm2.txt 1
channeln.exe mod.qfb ch.qfb 1c.kan fmpk.qfb
noise5.exe ch.qfb dis.qfb 4.pom
vit3.exe dis.qfb dm.cb fmpk.qfb 0 pm2.txt 1 min.fb -1 2 l.fb
hard_dec.exe dm.cb dk.cb 0 coef.cfg 100
stakan4.exe cod.cb dm.cb 2
stakan4.exe inf.cb dk.cb 2
channeln.exe mod.qfb ch.qfb 1r.kan fmpk.qfb
noise5.exe ch.qfb dis.qfb 4.pom
vit3.exe dis.qfb dm.cb fmpk.qfb 0 pm2.txt 1 min.fb -1 2 l.fb
hard_dec.exe dm.cb dk.cb 0 coef.cfg 10
stakan4.exe cod.cb dm.cb 2
stakan4.exe inf.cb dk.cb 2
cmpfb.exe cod.cb dm.cb sr.qfb
qread3 ch.qfb sr.qfb
qread3 dis.qfb sr.qfb
```




```
genermf.exe pm2.txt inf.cb 100000
cnvcoder.exe inf.cb cod.cb 0 coef.cfg
interlv.exe cod.cb in.cb 1.cfg
modulmf.exe in.cb mod.qfb pm2.txt 1
channeln.exe mod.qfb ch.qfb 1r.kan fmpk.qfb
noise5.exe ch.qfb dis.qfb 4.pom
vit3.exe dis.qfb dm.cb fmpk.qfb 0 pm2.txt 1 min.fb -1 2 l.fb
dinter_2.exe dm.cb dint.cb 1.cfg
hard_dec.exe dint.cb dk.cb 0 coef.cfg 100
stakan4.exe in.cb dm.cb 1
stakan4.exe cod.cb dint.cb 1
stakan4.exe inf.cb dk.cb 1
```