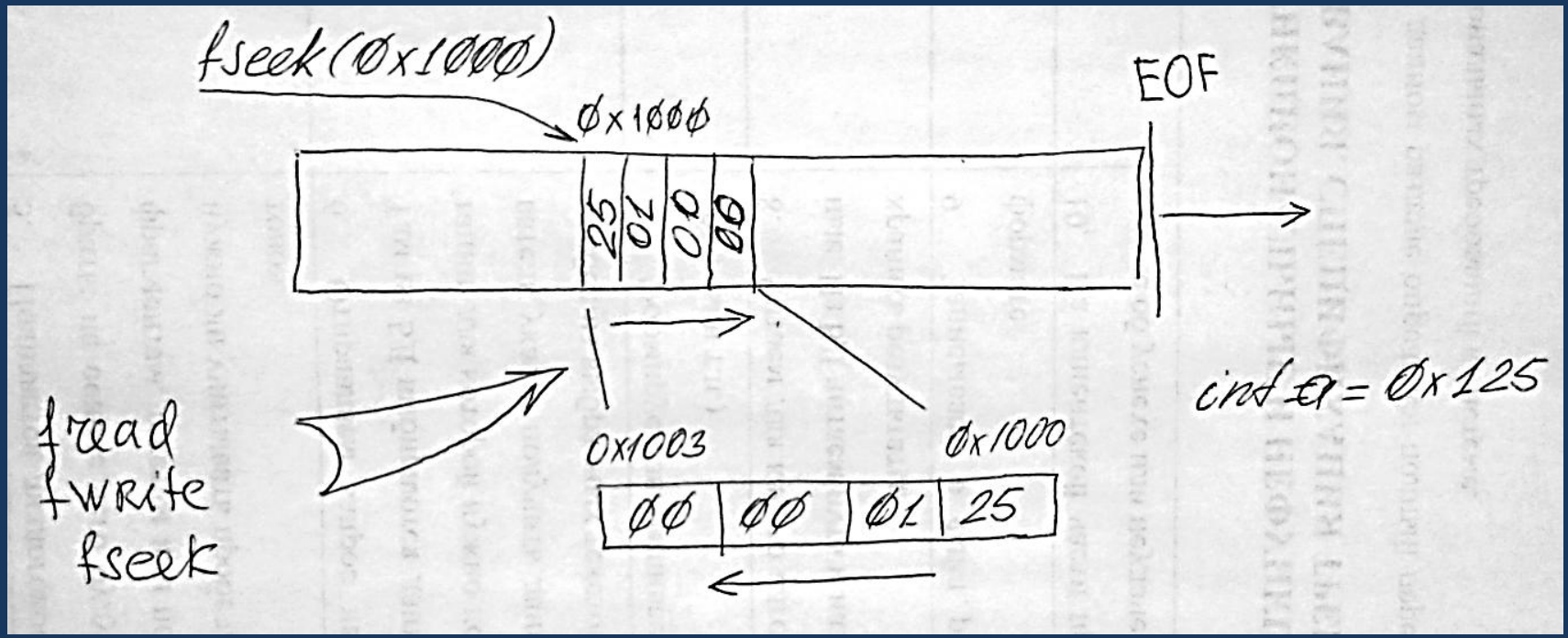
Двоичные файлы

Двоичный файл – это файл, в котором используется двоичный поиск

Вариант ответа на экзаменационный тест

Фабула:

Файл – модель файла – неограниченный массив байтов (аналог ОП, память данных) – архитектурные форматы БТД – двоичная система – **двоичный файл**



Двоичный файл (ДФ)

Правила:

- Открыть файл «как двоичный» **образ памяти** fopen(“a.dat”,”rb+wb”), иначе по умолчанию преобразование конца строки (Windows- \r\n - \n, Linux- \r-\n)
- **Последовательный файл**: две операции чтения (записи) читают данные, расположенные **последовательно**
- **Файл прямого (произвольного) доступа** (random access file) – установка внутреннего указателя в файле (позиция) и чтение/запись с этого места
- Адрес в ДФ – номер байта в неограниченном массиве
- Чтение/запись «на низком уровне» - прозрачная (без преобразований) передача массива байтов – void* - адрес, int – размерность
- Интерпретация содержимого – преобразование типа указателя, например – void*-double*
- Особенности представления данных в разных архитектурах - младший байт вперед (little endian, Intel,Windows) – старший байт вперед (Sun,Java) **https://habr.com/ru/post/233245/**

Задачи:

- Распределение памяти в ДФ
- Сбор мусора (сжатие)
- Форматы файлов

Двоичный файл (ДФ)

Чтение дампа ДФ

```
//-------------------------------------------------94-00.cpp
// Формирование ДАМПА для чтения файла
void main(){
FILE *fd=fopen("94-00.dat","wb");
char k=10,m=4;
short A[10]={6,3,7,3,4,8,300,5,23,64};
int   B[4]={6,3,7,3};
long p=0,offset;
fwrite(&p,sizeof(long),1,fd);         // Занять место под указатель
fwrite(&k,1,1,fd);             // Записать один байт - счетчик
fwrite(A,sizeof(short),k,fd);         // Записать массив коротких целых (2B)
p=ftell(fd);                          // Получить значение указателя
fwrite(&m,1,1,fd);                    // Записать один байт - счетчик
fwrite(B,sizeof(int),m,fd); // Записать массив целых
fseek(fd,0,SEEK_SET);                 // К началу файла
fwrite(&p,sizeof(long),1,fd);         // Обновить указатель на второй массив
fclose(fd);}
```

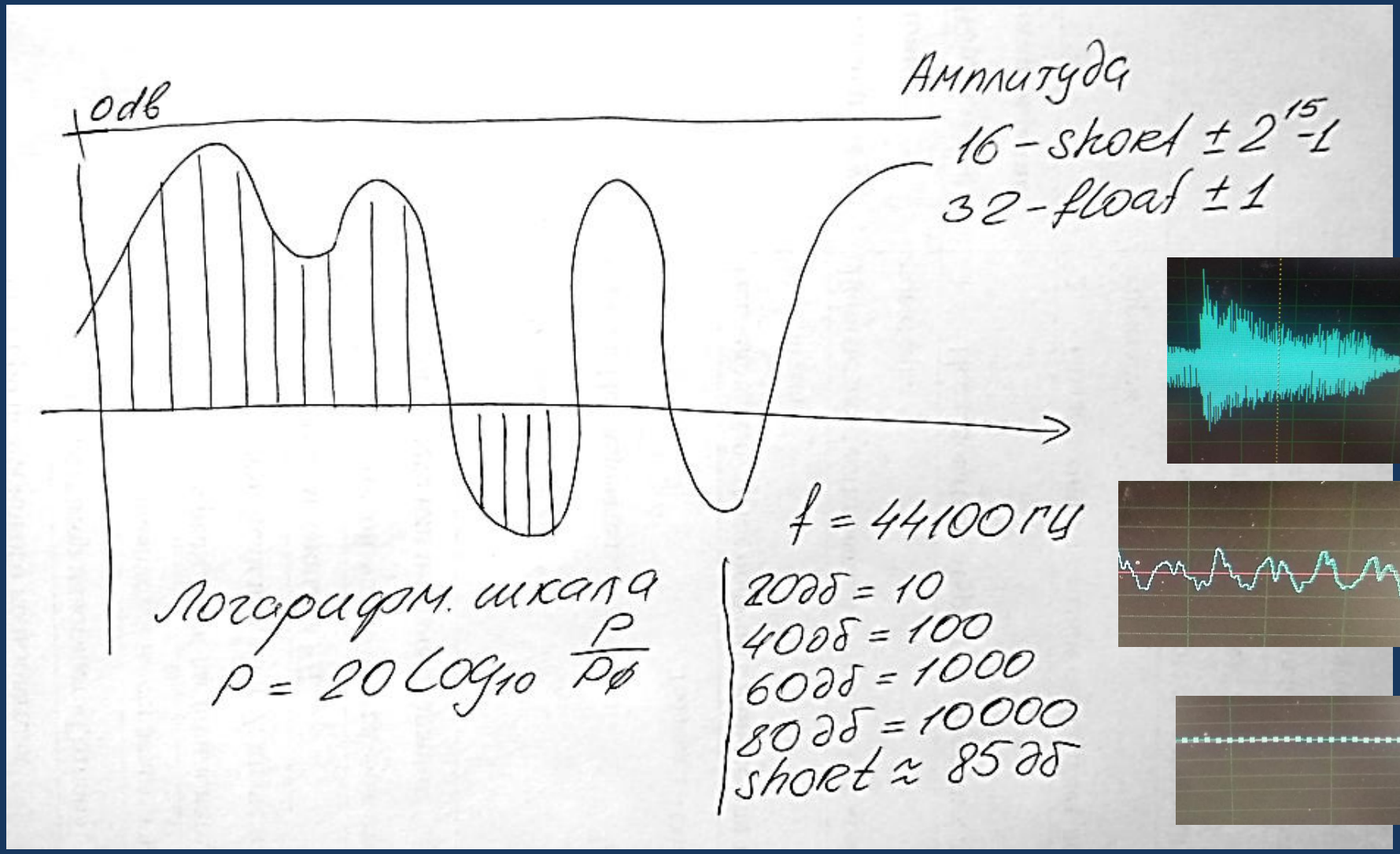
Двоичный файл (ДФ)

Форматы ДФ – «скатерть-самобранка», саморазворачивающаяся последовательность:

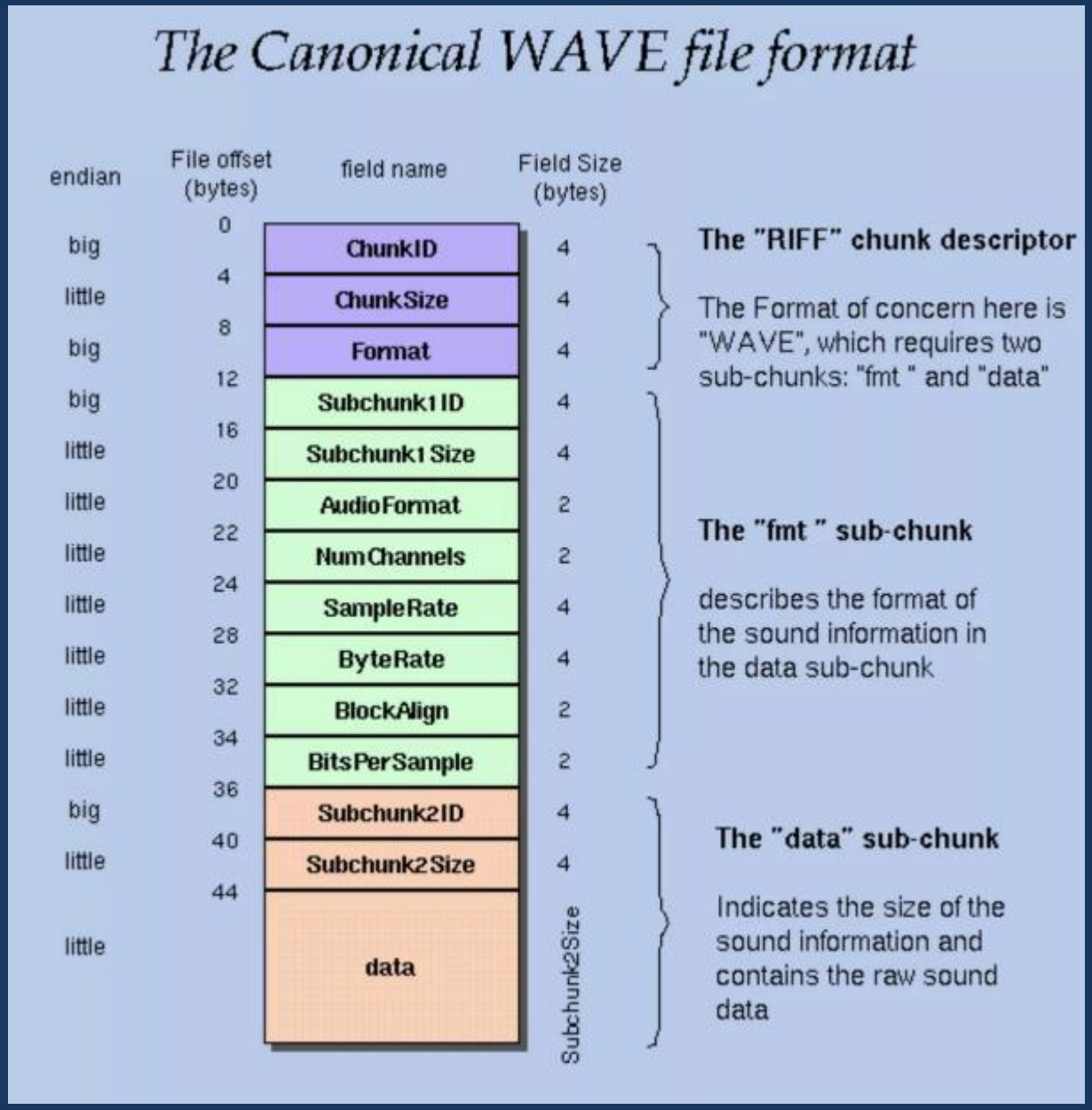
- Счетчик элементов + массив
- Идентификатор типа следующих за ним данных
- Структура с фиксированным порядком и размерностями полей
- Относительный адрес в ДФ (смещение)
- Абсолютный адрес в ДФ

Звуковой файл (wav)

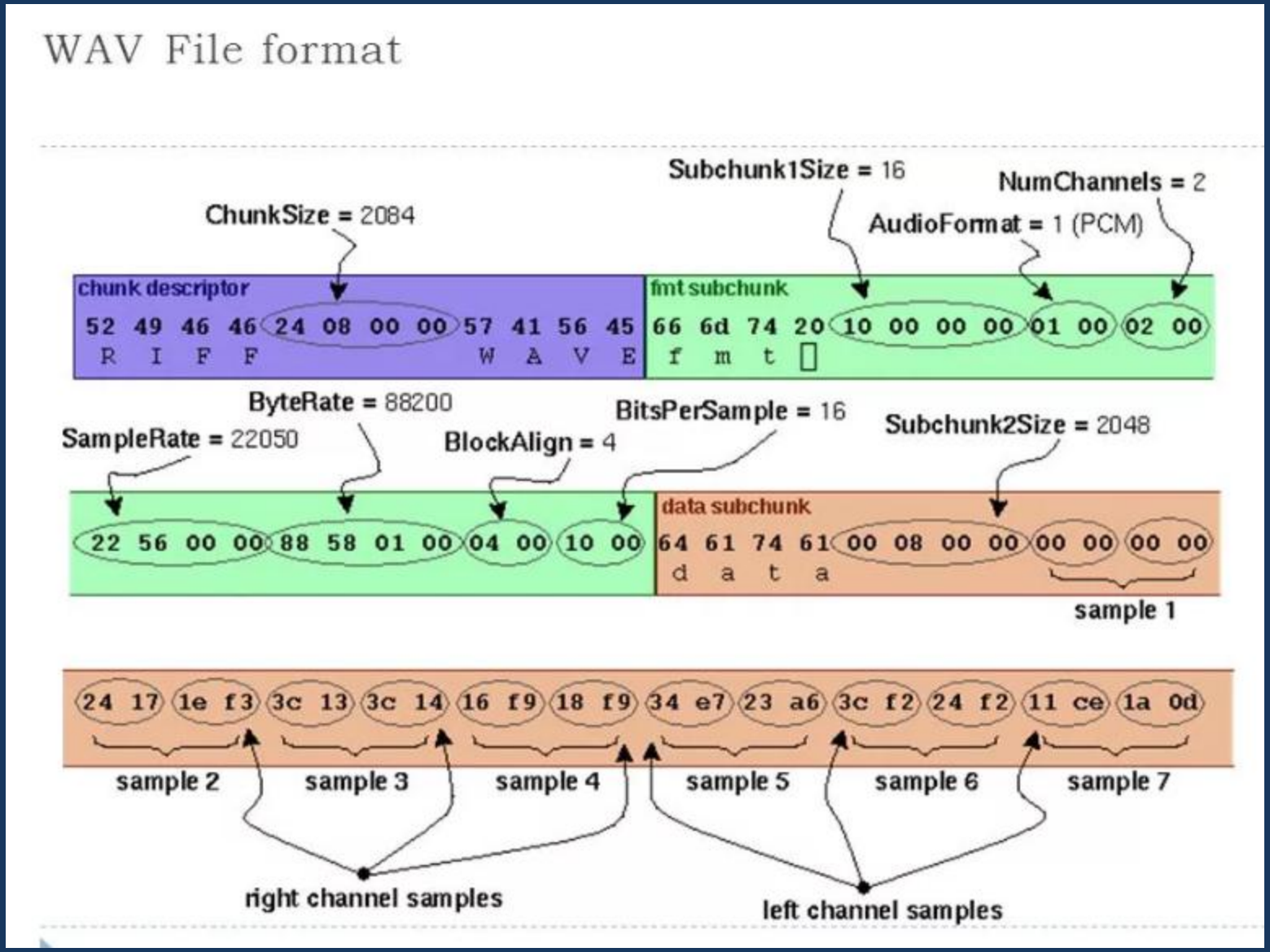
Массив амплитуд звукового сигнала (волна)



Звуковой файл (wav)



Звуковой файл (wav)

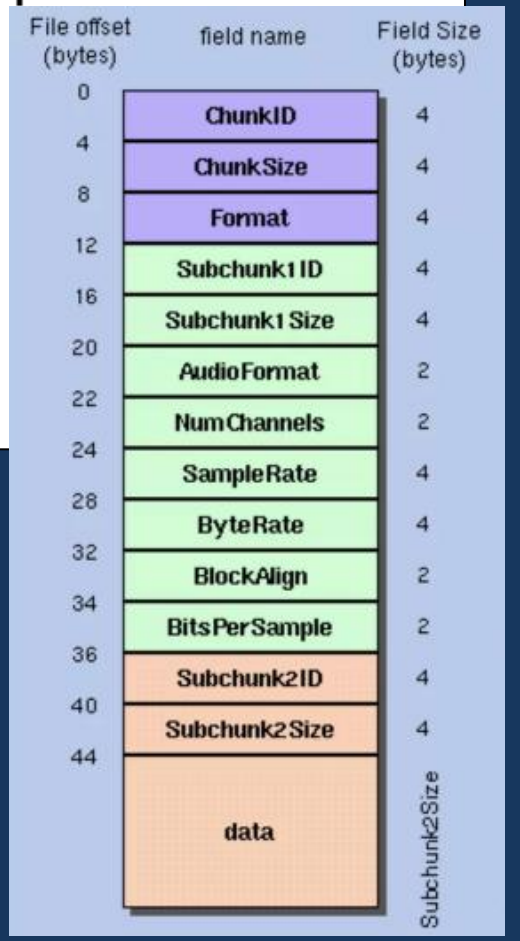


Звуковой файл (wav)

Lister - [d:\Temp\Track1.dat]

Файл Правка Вид Кодировка Справка

```
00000000: 52 49 46 46 78 FD 00 00|57 41 56 45 66 6D 74 20 | RIFFxэ..WAVEfmt
00000010: 10 00 00 00 01 00 01 00|80 3E 00 00 00 7D 00 00 | ........Ъ>...}..
00000020: 02 00 10 00 64 61 74 61|54 FD 00 00 1D FE 5B FE | ....dataTэ...ю[ю
00000030: B4 FE 60 FE D7 FD D7 FD|04 FE 0F FE DE FD 95 FD | ґю`юЧэЧэ.ю.юЮэ•э
00000040: 67 FD 5C FD 71 FD 1A FD|9E FC C5 FC B5 FC 86 FC
00000050: 88 FC 56 FC 4F FC 50 FC|3C FC 56 FC 5D FC 86 FC
00000060: 9D FC 6B FC 2F FC 30 FC|6B FC 4A FC 3E FC 36 FC
00000070: 63 FC A5 FC 6C FC 65 FC|A6 FC C4 FC DC FC A9 FC
00000080: D7 FC 05 FD E4 FC E5 FC|C1 FC AA FC 0D FD 29 FD
00000090: 3A FD 13 FD 27 FD 79 FD|98 FD CF FD E8 FD EF FD
000000A0: 09 FE 23 FE 26 FE EE FD|F2 FD 12 FE 11 FE 11 FE
000000B0: 2C FE 92 FE 58 FE 69 FE|A2 FE A1 FE ED FE D1 FE
000000C0: A3 FE 66 FE 86 FE 9B FE|A2 FE 68 FE 1C FE 26 FE
000000D0: 0E FE EA FD EC FD DC FD|B5 FD 9B FD 55 FD 52 FD
```



- Размер **следующей части** 0xFD78 + 8 = 0xFD80
- AudioFormat=1
- NumChannels=1 моно
- SampleRate=0x3E80 = 16000 гц
- ByteRate=0x7D00 = 32000 (байт/с)
- BlockAlign = 2 – выравнивание блока
- BitsPerSample = 0x10 = 16
- Размер субблока = 0xFD54
- Данные = FE1D,FE5B,FEB4……

Звуковой файл (wav)

```
struct HD1{
    char id_riff[4];
    long len_riff;
    } hd1;
struct HD2{
    char id_chuck[4];
    char fmt[4];
    long len_chuck;      // Поч
    } hd2;
struct HD3{
    short  type;
    unsigned short  channels;
    unsigned long freq;
    unsigned long bytes;
    unsigned short  align;
    unsigned short  bits;
    } hd3;
struct HD4{
    char id_data[4];
    long len_data;
} hd4;
```

```
void    main(){
    FILE * f;
    char c;
    f=fopen("16bit.wav","rb");
    if (f==NULL) {  return; }
    fread(&hd1,sizeof(hd1),1,f
    printf("LEN RIFF\t - %ld\n
    if (strncmp(hd1.id_riff,"R
        printf("invalid RIFF i
    fread(&hd2,sizeof(hd2),1,f
    printf("LEN Chuck\t - %ld\
    if (strncmp(hd2.id_chuck,"
        printf("invalid  CHUCK
    if (strncmp(hd2.fmt,"fmt "
        printf("invalid  FMT i
    // int k=hd2.len_chuck-siz
    // while(k--!=0) fread(&c,1,1,f);
    fread(&hd3,sizeof(hd3),1,f);
    printf("Type\t\t - %d\n", hd3.type );
    printf("Channels\t - %d\n", hd3.channels );
    printf("Sample Per Sec\t - %d\n", hd3.freq );
    printf("Bytes Per Sec\t - %d\n", hd3.bytes );
    printf("Bits\t\t - %d\n", hd3.bits );
    printf("Aligned\t\t - %d\n", hd3.align );
    fread(&hd4,sizeof(hd4),1,f);
    printf("LEN Data\t - %ld\n", hd4.len_data );
    if ( strncmp(hd4.id_data,"data",4)!=0 )
        printf("invalid DATA id\n");
```

```
Для 1 канала и 32 бит
(hd3.bits==32){
 int sz=hd4.len_data/sizeof(float);
 float *data=new float[sz];
 fread(data,sz,sizeof(float),f);
 for (int i=0;i<sz;i++){
     printf("%f\n",data[i]);
     if (i%100==0) getch();
     }
 }
else
if (hd3.bits==16){
    int sz1=hd4.len_data/sizeof(short);
    short *data1=new short[sz1];
    fread(data1,sz1,sizeof(short),f);
    for (int i=0;i<sz1;i++){
        printf("%d\n",data1[i]);
        if (i%100==0) getch();
        }
    }
fclose(f);
```

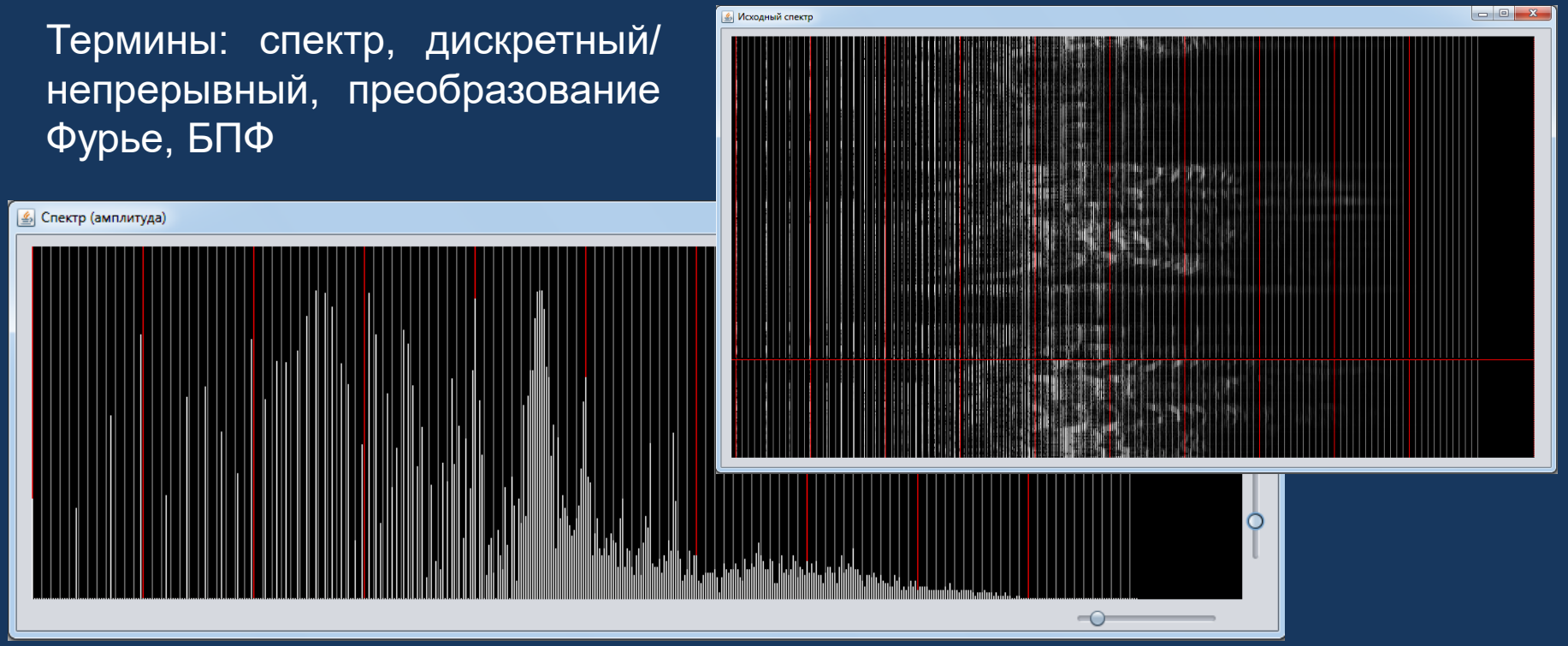
Звуковой файл (wav)

Обработка - амплитуда:

- Громкость (нормирование)
- Динамический диапазон (компрессия)
- Задержки (отражения) – реверберация, эхо, delay. Реализация: «прямая задержка», «обратная связь» - циклическая очередь с добавлением амплитуды задержанного сигнала A[i] = A[i]вх + A[i-n]*K
- Стерео – панорама: громкость и **задержка** звука между каналами

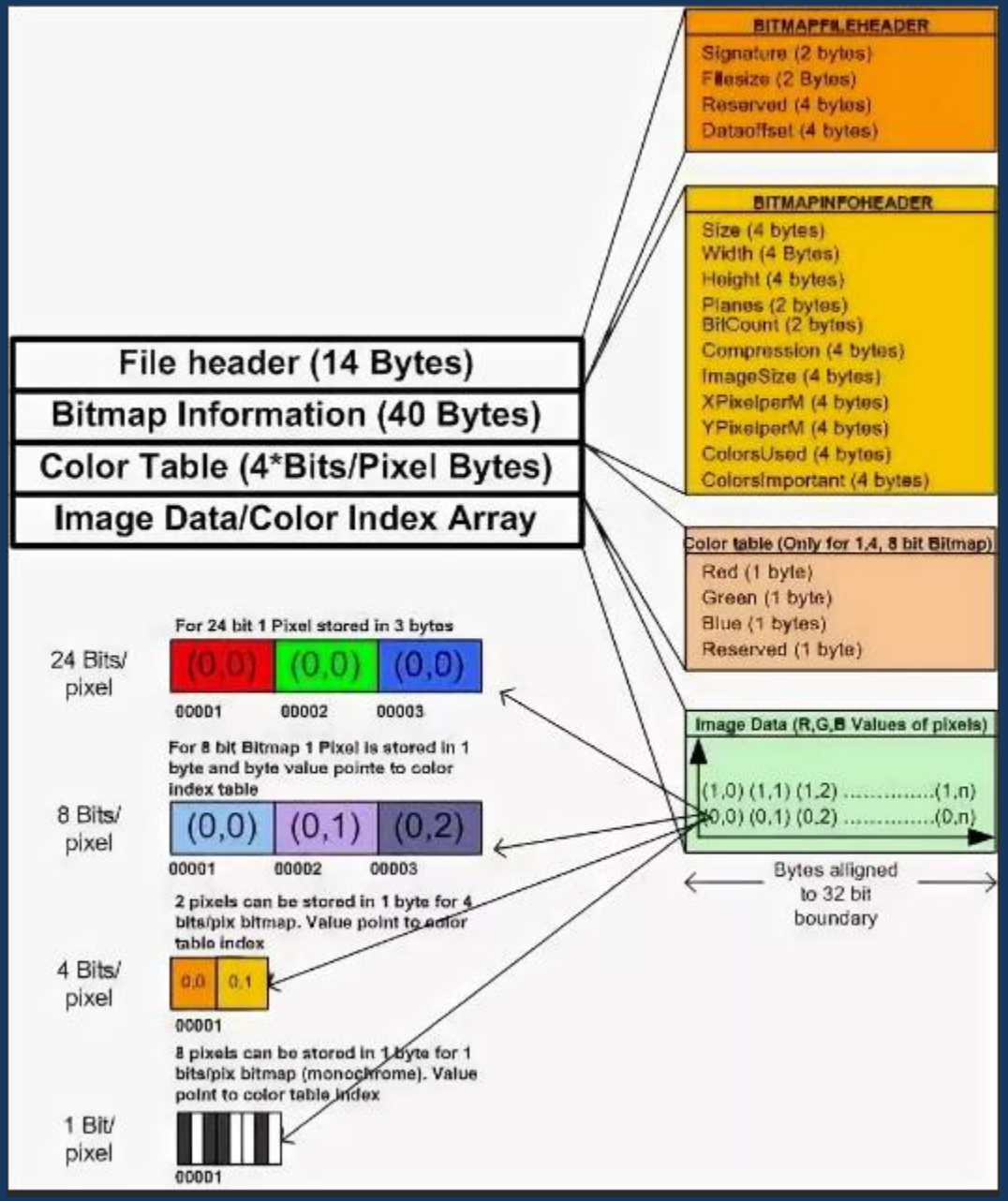
Обработка – частота: спектр сигнала

Термины: спектр, дискретный/ непрерывный, преобразование Фурье, БПФ



Битмап-изображение (bmp)

- Карта пикселов N x M
- Вариации форматов – прямые цвета/палитра



Битмап-изображение (bmp)

Пример – Java
Побайтное чтение (little endian)

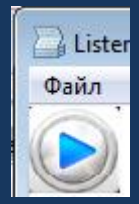
```
class BFile  {
    RandomAccessFile W;
    BFile(String path) throws FileNotFoundException{
        W = new RandomAccessFile(path, "rw");
        }
byte  []read4Byte(String ss) throws IOException{
    byte s[]=ss.getBytes();
    byte b[]=new byte[4];
    for (int i=0;i<4;i++) b[i]=W.readByte();
    for (int i=0;i<4;i++) if (b[i]!=s[i]) throw new IOException("Illegal format: not "+ss);
    return b;
    }
short readShort()throws IOException{
    return (short)((W.read()&0xFF) | ((W.read()&0xFF)<<8));
    }
float readFloat(RandomAccessFile W)throws IOException{
    return Float.intBitsToFloat(W.readInt());
    }
 float readFloat()throws IOException{ return Float.intBitsToFloat(readInt()); }
int readInt()throws IOException{
   return (int)(((W.read()&0xFF) | ((W.read()&0xFF))<<8) | ((W.read()&0xFF)<<16) | ((W.read
    }
}
```

Битмап-изображение (bmp)

```
class RGB {
    byte    rgbRed;     //интенсивность красного
    byte    rgbGreen;   //интенсивность зеленого
    byte    rgbBlue;    //интенсивность голубого
    byte    rgbRserved; //не используется
void load(BFile W) throws IOException{
    rgbRed=W.W.readByte();
    rgbGreen=W.W.readByte();
    rgbBlue=W.W.readByte();
    rgbRserved=W.W.readByte();
}}
```

```
//-------------- Заголовок файла ----------------------------------------
short bfType;           //тип файла (для битового образа - BM)
int   bfSize;           //размер файла в dword
short bfReserved1;      //не используется
short bfReserved2;      //не используется
int   bfOffbits;        //СМЕЩЕНИЕ данных битового образа от заголовка в байтах
int   biSize;           //число байт, занимаемых структурой BITMAPINFOHEADER
int   biWidth;          //ширина битового образа в пикселах
int   biHeight;         //высота битового образа в пикселах
short biPlanes;         //число битовых плоскостей устройства
short biBitCount;       //число битов на пиксель
int   biCompression;    //тип сжатия
int   biSizeImage;      //размер картинки в байтах
int   biXPelsPerMeter;  //горизонтальное разрешение устройства, пиксел/м
int   biYPelPerMeter;   //вертикальное разрешение устройства, пиксел/м
int   biClrUsed;        //число используемых цветов
int   biClrImportant;   //число "важных" цветов
//
```

Битмап-изображение (bmp)



Lister - [d:\Romanow\Форматы и описания\wave+bmp\24bit.bmp]

Файл Правка Вид Кодировка Справка

```
00000000: 42 4D 0E 17 00 00 00 00|00 00 36 00 00 00 28 00  | BM........6...(.
00000010: 00 00 2D 00 00 00 2B 00|00 00 01 00 18 00 00 00  | ..-...+.........
00000020: 00 00 D8 16 00 00 00 00|00 00 00 00 00 00 00 00  | ..Ш.............
00000030: 00 00 00 00 00 00 79 79|79 A9 A9 A9 BF BF BF CB  | ......yyy©©©ïïïЛ
00000040: CB CB D0 D0 D0 D1 D1 D1|D1 D1 D1 D1 D1 D1 D1 D1  | ЛЛРРРССССССССССС
00000050: D1 D1 D1 D1 D1 D1 D1 D2|D2 D2 D1 D1 D1 D0 D0 D0  | ССССССТТТССРРР
00000060: CA CA CA BB BB BB AC AB|AB 95 95 95 79 79 7A 78  | КККЖЖЖ¬¬«•••yyzx
00000070: 78 7A 7D 7E 80 80 80 82|81 81 83 7D 7E 81 77 77  | xz}~ЂЂЂ‚ЃЃѓ}~Ѓww
00000080: 79 72 72 73 8D 8D 8D A8|A8 A8 B8 B8 B8 C8 C8 C8  | yrrsЌЌЌЁЁЁёёёИИИ
00000090: CF CF CF D1 D1 D1 D2 D2|D2 D1 D1 D1 D1 D1 D1 D1  | ПППССТТТССССССС
000000A0: D1 D1 D1 D1 D1 D1 D1 D1|D1 D1 D1 D1 D1 D1 CD CD  | ССССССССССССССНН
```

```
//-------------- Заголовок файла ---------------------------
short bfType;           //тип файла (для битового образа - BM)
int   bfSize;           //размер файла в dword
short bfReserved1;      //не используется
short bfReserved2;      //не используется
int   bfOffbits;        //СМЕЩЕНИЕ данных битового образа от з
int   biSize;           //число байт, занимаемых структурой BI
int   biWidth;          //ширина битового образа в пикселах
int   biHeight;         //высота битового образа в пикселах
short biPlanes;         //число битовых плоскостей устройства
short biBitCount;       //число битов на пиксель
int   biCompression;    //тип сжатия
int   biSizeImage;      //размер картинки в байтах
int   biXPelsPerMeter;  //горизонтальное разрешение устройства
int   biYPelPerMeter;   //вертикальное разрешение устройства,
int   biClrUsed;        //число используемых цветов
int   biClrImportant;   //число "важных" цветов
```

Offset = 0x36
BiSize=0x28=40
W=0x2D=45
H=0x2B=43
BitCount=0x18=24
SizeImage=16D8=5848=43*136
Строка=45*3+1=136
Выравнивание до int
Палитры нет, область данных = 0x36, смещение =0x36 от начала первого блока

Битмап-изображение (bmp)

- Чтение палитры (если надо)
- Чтение bitmap – байтный массив

24 бита – 3 байта RGB, цвет из 3 байтов bitmap

```
case 24:     for (i=0,k=0; i<biHeight;i++){
                 for (j=0; j<biWidth;j++,k+=3){
                     Color c=new Color(MAP[k+2]&0xFF,MAP[k+1]&0xFF,MAP[k]&0xFF);
                     G.setColor(c);
                     G.fillRect(5+j,5+(biHeight-i),1,1);
                 }    while(k%4!=0) k++;     // Выравнивание строки по 4-байтной границе
             }
             break;
         }}
```

4 – ½ байта, цвет из палитры по индексу цвета

```
case 4:      for (i=0,k=0; i<biHeight;i++){
                 for (j = 0; j < biWidth; j++) {
                     int m = MAP[k] & 0xFF; // Номер цвета
                     m= (j%2==0 ? m : m>>4) & 0xF;
                     Color c = new Color(PAL[m].rgbBlue & 0xFF, PAL[m].rgbGreen & 0xFF, PAL[m
                     G.setColor(c);
                     G.fillRect(5 + j, 5 + (biHeight - i), 1, 1);
                     if (j%2==0) k++;
                     } while (k % 4 != 0) k++; // Выравнивание строки по 4-байтной границе
                     }
             break;
```

Двоичный файл (ДФ)

```
int     fread  (void *buf, int size, int nrec, FILE *fd);
int     fwrite (void *buf, int size, int nrec, FILE *fd);
```

```
// Прочитать  целую переменную и следующий за ней
// динамический массив из n переменных типа double
int n;                              // в целой переменной – размерность массива
fread(&n, sizeof(int),1,fd);        // указатель на переменную int
double   *pd = new double[n];
fread(pd, sizeof(double),n,fd);     // преобразование к void* - неявное
```

Прямой доступ – функция позиционирования (fseek)

```
int     fseek(FILE *fp, long pos, int mode)
```

```
long ftell(FILE *fp)
```

```
#define SEEK_SET 0    // Относительно начала файла
                      // начало файла - позиция 0
#define  SEEK_CUR 1   // Относительно текущей позиции,
                      // >0 - вперед, <0 - назад
#define SEEK_END 2    // Относительно конца файла
                      // (значение pos - отрицательное)
```

```
long     fsize;
fseek(fl,0L,SEEK_END); // Установить позицию на конец файла
fsize = ftell(fd);                  // Прочитать значение текущей позиции
```

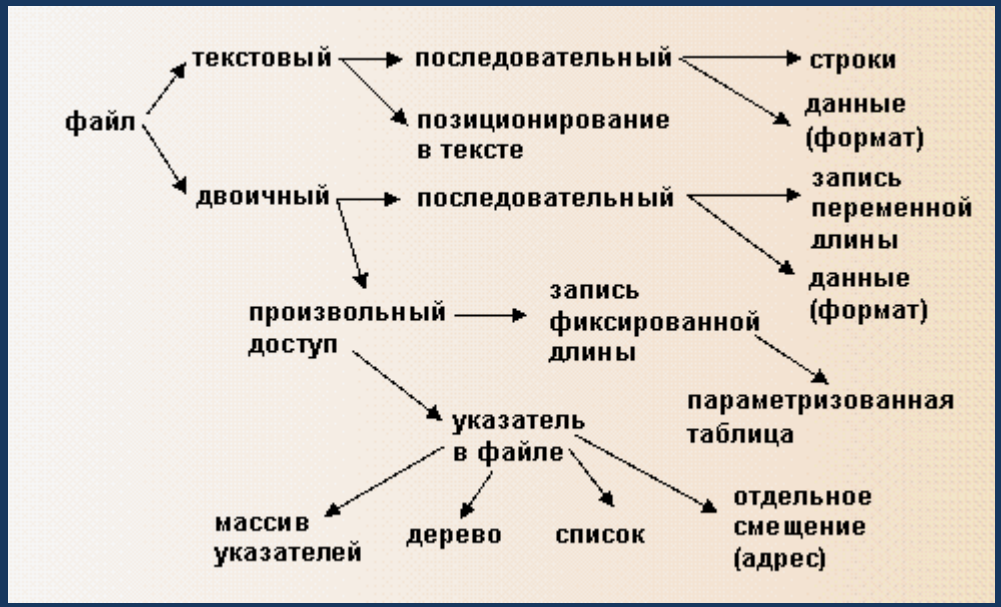
```
fseek(fd,100,SEEK_SET);         // По адресу 100 находится смещение
fread(&P,sizeof(long),1,fd);    // Читается P=46, после чтения текущая позиция
fseek(fd,i,SEEK_CUR);           // 100+sizeof(long)=104, позиционирование 104+46=150
```

Двоичный файл (ДФ)

Техника работы с ДФ

- Последовательный ДФ (поток), сериализация СД. Чтение/запись файла полностью в память
- Распределение памяти в ДФ – добавить в ДФ: позиционирование 0,SEEK_END, чтение абсолютной позиции (адреса) – ftell, запись в ДФ
- Обновление данных (update) – чтение, изменение в памяти, позиционирование, запись
- Обновление с увеличением размерности – добавление новой копии, коррекция ссылок на нее (адресов), старые данные - мусор
- Сжатие, сбор мусора – переписывание актуальных данных из входного файла в выходной

Структуры данных в ДФ



Двоичный файл (ДФ)

Двоичная сериализация: сохранение дерева в ДФ (95-01)

```
//-----------------------------------------------------9
// Базовые алгоритмы над деревьями
#include <stdio.h>
#include <conio.h>
#define N 4
struct tree{
    int val;
    int cnt;          // Счетчик вершин поддерева
    int n;            // Счетчик потомков в ch
    tree *ch[N];      // Массив указателей на потомков
    };

//-----------------------------------------------------
// Полный рекурсивный обход дерева
void scan(tree *p,int level){
    if (p==NULL) return;
    printf("l=%d cnt=%d val=%d\n",level,p->
    for (int i=0;i<p->n;i++)
        scan(p->ch[i],level+1);
    }
tree *create(int vv){
    tree *q=new tree;
    q->val=vv;
    q->n=0;
    q->cnt=1;
    return q;
    }
```

```
//-----------------------------------------
// Вставка в поддерево с минимальным количеством вершин
void insert_min(tree *&p,int vv){
    if (p==NULL) { p=create(vv); return; }
    p->cnt++;
    if (p->n!=N) {
        p->ch[p->n++]=create(vv);
        return;
        }
    int i,k;
    for (i=1,k=0; i<N;i++)          // Количество потомк
        if (p->ch[i]->cnt < p->ch[k]->cnt)
            k=i;                    // Искать потомка с
    insert_min(p->ch[k],vv);        // числом вершин и в
    }
```

Двоичный файл (ДФ)

Двоичная сериализация: сохранение дерева в ДФ (95-01)

```
//-------------Сохранение в двоичный последовательный поток
void save(tree *p, FILE *fd){
    fwrite(&p->cnt,sizeof(int),1,fd);
    fwrite(&p->val,sizeof(int),1,fd);
    fwrite(&p->n,sizeof(int),1,fd);
    for (int i=0;i<p->n;i++)
        save(p->ch[i],fd);
    }
//-------------Загрузка из двоичного последоват
tree *load(FILE *fd){
    tree *p=new tree;
    fread(&p->cnt,sizeof(int),1,fd);
    fread(&p->val,sizeof(int),1,fd);
    fread(&p->n,sizeof(int),1,fd);
    for (int i=0;i<p->n;i++)
        p->ch[i]=load(fd);
    return p;
    }
```

```
void main(){
    tree *pp=NULL,*pp2=NULL,*pp3=NULL;
    int A1[]={4,7,2,56,78,3,7,4,7,4,7,5,8,23,7,4};
    int n=sizeof(A1)/sizeof(int);
    for (int i=0;i<n;i++) insert_min(pp,A1[i]);
    puts("-- Recursive scan -----------");
    scan(pp,0);
    puts("-- Save to FILE -------------");
    FILE *fd=fopen("95-01.dat","wb");
    save(pp,fd);
    fclose(fd);
    fd=fopen("95-01.dat","rb");
    tree *pp4=load(fd);
    scan(pp4,0);
    }
```

L=0 cnt=16 val =4 n=4
L=1 cnt=4 val =7 n=3
L=2 cnt=1 val =3 n=0
L=2 cnt=1 val =4 n=0
L=2 cnt=1 val =23 n=0
L=1 cnt=4 val =3 n=3

Lister - [d:\Temp\95-01.dat]

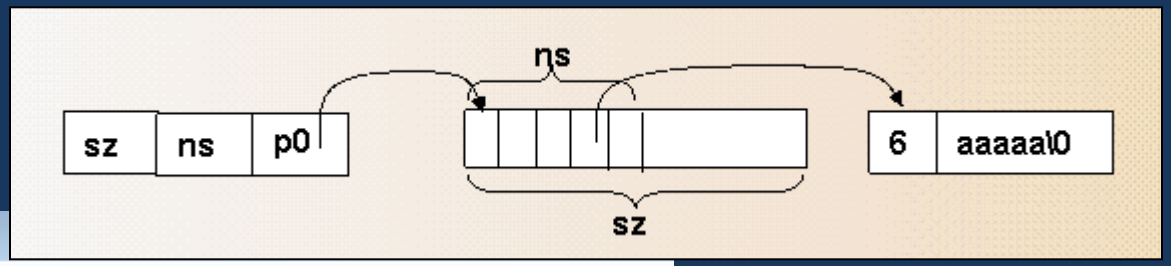
Файл Правка Вид Кодировка Справка

```
00000000: 10 00 00 00 04 00 00 00|04 00 00 00 04 00 00 00
00000010: 07 00 00 00 03 00 00 00|01 00 00 00 03 00 00 00
00000020: 00 00 00 00 01 00 00 00|04 00 00 00 00 00 00 00
00000030: 01 00 00 00 17 00 00 00|00 00 00 00 04 00 00 00
00000040: 02 00 00 00 03 00 00 00|01 00 00 00 07 00 00 00
00000050: 00 00 00 00 01 00 00 00|07 00 00 00 00 00 00 00
00000060: 01 00 00 00 07 00 00 00|00 00 00 00 04 00 00 00
00000070: 38 00 00 00 03 00 00 00|01 00 00 00 04 00 00 00
00000080: 00 00 00 00 01 00 00 00|05 00 00 00 00 00 00 00
00000090: 01 00 00 00 04 00 00 00|00 00 00 00 03 00 00 00
000000A0: 4E 00 00 00 02 00 00 00|01 00 00 00 07 00 00 00
000000B0: 00 00 00 00 01 00 00 00|08 00 00 00 00 00 00 00
```

Двоичный файл (ДФ)

Обновление данных в прямом доступе (95-06…95-09:

- Записи переменной длины (строки, кадры…)
- Массив адресов (файловых указателей на записи, МФУ)
- Резерв свободного места в МФУ
- Переполнение МФУ – переразмещение в конце файла
- Заголовок файла – размер МФУ, кол-во записей, адрес МФУ



Lister - [d:\Temp\95-06.dat]

Файл Правка Вид Кодировка Справка

```
00000000: 2A 00 00 00 27 00 00 00 0C 00 00 00 B4 00 00 00 | *...'.......´...
00000010: CC 00 00 00 E5 00 00 00 FA 00 00 00 41 01 00 00 | М...е...ъ...А...
00000020: 66 07 00 00 4D 07 00 00 86 01 00 00 C8 01 00 00 | f...M...‡...И...
00000030: F1 01 00 00 16 02 00 00 5B 07 00 00 42 02 00 00 | с.......[...В...
00000040: 42 07 00 00 6F 02 00 00 9E 02 00 00 E3 02 00 00 | B...o...ћ...г...
00000050: 1F 03 00 00 66 03 00 00 A6 03 00 00 D0 03 00 00 | ....f...¦...Р...
00000060: 1B 04 00 00 67 04 00 00 B3 04 00 00 FC 04 00 00 | ....g...і...ь...
00000070: 45 05 00 00 81 05 00 00 BE 05 00 00 D7 05 00 00 | Е...Ѓ...s...Ч...
00000080: 12 06 00 00 38 06 00 00 53 06 00 00 5D 06 00 00 | ....8...S...]...
00000090: A9 06 00 00 E8 06 00 00 FC 06 00 00 02 07 00 00 | ©...и...ь.......
000000A0: 15 07 00 00 3A 07 00 00 00 00 00 00 00 00 00 00 | ....:...........
000000B0: 00 00 00 00 14 00 00 00 23 69 6E 63 6C 75 64 65 | ........#include
000000C0: 20 3C 73 74 64 69 6F 2E 68 3E 0A 00 15 00 00 00 |  <stdio.h>......
000000D0: 23 69 6E 63 6C 75 64 65 20 3C 73 74 72 69 6E 67 | #include <string
000000E0: 2E 68 3E 0A 00 11 00 00 00 23 64 65 66 69 6E 65 | .h>......#define
000000F0: 20 46 4E 55 4C 4C 20 30 0A 00 43 00 00 00 2F 2F |  FNULL 0..C.../
00000100: 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D | ----------------
```

sz = 0x2A=42
n = 0x27=39
Offset=0xC=12
МФУ: 12+42*4=180
Данные: 180 =0xB4

Двоичный файл (ДФ)

```
#define FNULL 0
//-----------------------------------------------------95-06.cpp
//---- Создание файла с массивом указателей из текстового файла
// Формат файла: ns - размерность МУ (количество указателей)
// p0 - смещение массива указателей
 void save(char *in, char *out)
{ FILE *fdi,*fdo; char c[80]; int ns,i;
 if ((fdi=fopen(in,"r"))==NULL) return;
 if ((fdo=fopen(out,"wb"))==NULL) return;
 for (ns=0; fgets(c,80,fdi)!=NULL; ns++);    // Количество строк
 int sz=ns*1.2;                              // Размерность с учетом резерва
 fseek(fdi,0l,SEEK_SET);                     // Вернуться к началу
 long *pp = new long[sz];                    // Массив указателей в файле
 for (i=0;i<sz;i++) pp[i]=FNULL;
 long p0=2*sizeof(int)+sizeof(long);         // Начальное смещением МУ
 fwrite(&sz,sizeof(int),1,fdo);              // Записать размерность МУ
 fwrite(&ns,sizeof(int),1,fdo);              // Записать размерность МУ
 fwrite(&p0,sizeof(long),1,fdo);             // Записать смещение МУ
 fwrite(pp,sizeof(long),sz,fdo);             // Записать "пустой" МУ
 for (i=0; i<ns; i++) {                      // Повторное чтение строк
      pp[i]=ftell(fdo);                      // Получить адрес i-ой строки
      fgets(c,80,fdi);
      int l=strlen(c)+1;                     // Переписать в формате ЗПД
      fwrite(&l,sizeof(int),1,fdo);
      fwrite(c,l,1,fdo);
      }
 fseek(fdo,p0,SEEK_SET);                     // Обновить в файле массив
 fwrite(pp,sizeof(long),sz,fdo);             // файловых указателей
 fclose(fdo);}

 void main(){
 save("95-06.cpp","95-06.dat");
 }
```

Двоичный файл (ДФ)

Извлечение отдельной строки по ЛН без загрузки всего файла

```
//-------------------------------------------------------95-08.cpp
//---- Массив указателей на строки, чтение по логическому номеру
 char *load(char *name, int num)          // Возвращается строка =
{ FILE *fd; int i,n,sz; long p0,pp;       // динамический массив
 if ((fd=fopen(name,"rb"))==NULL)
     return NULL;                         // Режим чтения двоичного файла
 fread(&sz,sizeof(int),1,fd);             // Считать размерность МУ
 fread(&n,sizeof(int),1,fd);              // Считать количество указателей
 fread(&p0,sizeof(long),1,fd);            // и начальный адрес МУ в файле
 if (num>=n) return NULL;                 // Нет записи с таким номером
 fseek(fd,p0+sizeof(long)*num,SEEK_SET);
 fread(&pp,sizeof(long),1,fd);            // Прочитать указатель с номером num
 fseek(fd,pp,SEEK_SET);                   // Установиться на запись
 fread(&sz,sizeof(int),1,fd);             // Прочитать длину записи
 char *p=new char[sz];                    // Создать динамический массив
 fread(p,sz,1,fd);                        // Прочитать запись - строку
 fclose(fd); return p; }                  // Возвратить указатель на строку

void main(){
for (int i=20; i>=10; i--){
   char *s=load("95-06.dat",i); if (s!=NULL) {
       CharToOemA(s,s);
       printf("[%d] %s",i,s);
       delete [] s;
       }}
}
```

Двоичный файл (ДФ)

Прямой доступ: в данных класса – МФУ, строки загружаются по требованию.
Перераспределение памяти под МФУ

```
//-------------------------------------------------------95-09.cpp
//---- Массив указателей на строки - загрузка управляющих данных
// Для приведения двоичного файла в исходный вид выполнить 95-06.cpp
struct DMUS{
    FILE *fd;                          // Файл в stdio
    int sz,ns;
    long p0, *pp;                      // Образ массива файловых указателей
// Открытие файла и загрузка управляющих структур
int open(char name[]){
    if ((fd=fopen(name,"rb+"))==NULL)
        return 0;                      // Режим чтения/записи/добавления
    fread(&sz,sizeof(int),1,fd);       // Считать размерность МУ
    fread(&ns,sizeof(int),1,fd);       // Считать количество указателей
    fread(&p0,sizeof(long),1,fd);      // и начальный адрес МУ в файле
    pp=new long[sz];                   // Создать образ МУ
    fseek(fd,p0,SEEK_SET);             // и прочитать его содержимое из файла
    fread(pp,sizeof(long),sz,fd);
    return 1;
    }
// Обновление управляющих структур
void updateSys(){
    fseek(fd,0,SEEK_SET);              //
    fwrite(&sz,sizeof(int),1,fd);      //
    fwrite(&ns,sizeof(int),1,fd);
    fwrite(&p0,sizeof(long),1,fd);
    fseek(fd,p0,SEEK_SET);
    fwrite(pp,sizeof(long),sz,fd);
    }
```

```
    }
// Перераспределение памяти под массив файловы
void extend(){
    if (ns!=sz) return;
    sz*=2;                                  // расшири
    pp=(long*)realloc(pp,sizeof(long)*sz);
    fseek(fd,0,SEEK_END);                   // Установ
    p0=ftell(fd);                           // и получ
    updateSys();                            // Обновит
    }
```

Двоичный файл (ДФ)

```
// Чтение по логическому номеру
char *get(int k){
    if (fd==NULL || k>=ns) return NULL;
    fseek(fd,pp[k],SEEK_SET);
    int ll;
    fread(&ll,sizeof(int),1,fd);     // Прочитать длину записи
    char *p=new char[ll];            // Создать динамический массив
    fread(p,ll,1,fd);                // Прочитать запись - строку
    return p;
    }
// вставка по логическому номеру
int insert(char *s, int k){
    if (fd==NULL || k>=ns) return 0;
    extend();                        // Отработать переполнение
    fseek(fd,0,SEEK_END);            // Спозиционироваться в конец файла
    for(int j=ns-1;j>=k;j--)
        pp[j+1]=pp[j];               // Сдвинуть указатели в массиве
    ns++;
    pp[k]=ftell(fd);                 // Записать адрес новой строки
    int ll=strlen(s)+1;
    fwrite(&ll,sizeof(int),1,fd);    // Записать с
    fwrite(s,ll,1,fd);
    return 1;
    }
```

Продолжение следует: обновить по ЛН:

- Читать размер старой строки
- Больше новой – обновить на месте
- Добавить в конец
- Новый адрес поместить в МФУ

```
void main(){
    DMUS FF={NULL};
    if (!FF.open("95-06.dat")) return;
    FF.insert("aaaaa\n",10);
    FF.insert("bbbbbbbb\n",5);
    for (int i=20; i>=0; i--){
        char *s=FF.get(i); if (s!=NULL) {
        CharToOemA(s,s);
        printf("[%d] %s",i,s);
        delete []s;}
        }
    FF.close();
}
```

Двоичный файл (ДФ)

Вопросы без ответов: формат ДФ

```
//-----------------------------------------
void *F7(int n, FILE *fd)
{ int sz; void *p; long p0;
fseek(fd,0L,SEEK_SET);
fread(&sz,sizeof(int),1,fd);
fread(&p0,sizeof(long),1,fd);
p = (void*)new char[sz];
fseek (fd, p0 + sizeof(long)*n, SEEK_SET);
fread (&p0, sizeof(long),1,fd);
fseek(fd, p0, SEEK_SET);
fread(p, sz, 1, fd);
return p; }
//-----------------------------------------
char *F8(int n, FILE *fd)
{ char *p; long fp; int i;
fseek(fd, sizeof(long)*n,SEEK_SET);
fread(&fp,sizeof(long),1,fd);
fseek(fd,fp,SEEK_SET);
n = 80; p = new char [n];
     for (i=0;; i++) {
     if (i==n) p = (char*)realloc(p, n=n*2);
     fread(p+i,1,1,fd);
     if (p[i]=='\0') return p;
     }
return p; }
```

```
//---------------------------------------------------- 9
#define FNULL -1L
char *F9(int n, FILE *fd)
{ long p0; int sz; char *p;
fseek(fd,0L,SEEK_SET);
fread(&p0,sizeof(long),1,fd);
     for (; p0!=FNULL && n!=0; n--) {
     fseek(fd,p0,SEEK_SET);
     fread(&p0,sizeof(long),1,fd);
     }
if (p0==FNULL) return(NULL);
fread(&sz,sizeof(int),1,fd);
p = new char[sz+1];
fread(p,sz,1,fd);
p[sz]='\0'; return p;}
```

```
//---------------------------------------------------- 19
man *F19(FILE *fd)
{ man *p; int n;
fread(&n,sizeof(int),1,fd);
p = new man;
fread (p, sizeof(man),1,fd);
n = n - sizeof(man);
p->addr = new char[n];
fread(p->addr,n,1,fd);
return p; }
//---------------------------------------------------- 20
void F20(FILE *fd, man *p)
{ int n = sizeof(man)+strlen(p->addr)+1;
fseek(fd,0L,SEEK_END);
fwrite(&n,sizeof(int),1,fd);
fwrite (p, sizeof(man),1,fd);
n = n - sizeof(man);
fwrite (p->addr, n,1,fd ); }
```