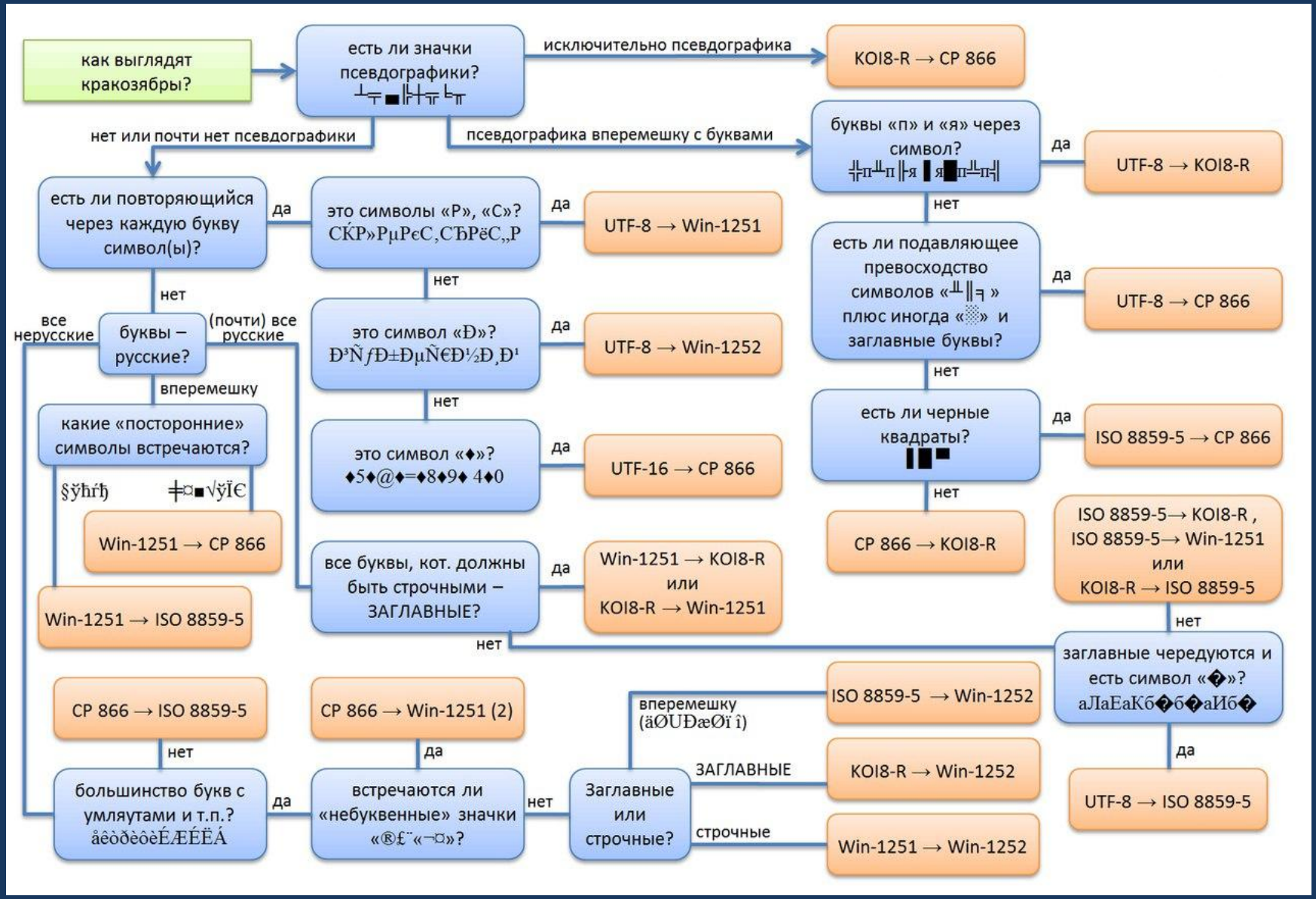
Символы. Строки. Текст

Медицинские факты:

- Представление символов и текста базируется на форматах внутреннего представления данных в компьютере - машинное слово, целое со знаком и без знака
- Символу соответствует машинное слово - байт (char), 2 байта (short) или переменной длины (1-2) в зависимости от способа кодирования = внутреннее представление символа или код символа
- Кодировка – правило соответствия символа его коду - кодовая таблица
- в Си,Java символы напрямую отображаются на базовые ТД, в Бейсик, Паскаль – функции псевдо-преобразования символа в код
- Анахронизмы – многообразие, сложности и нестыковки форматов представления исходят из исторических реалий представления текста и файлов
  - в файле не содержатся данные о его кодировке
  - тип (расширение) файла формально не связаны с его содержимым (на уровне соглашения о типах)
  - например, в html-файле есть тег о смене кодировки <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
  - в Си кодовая таблица назначается библиотекой

Кодировка символов



Кодировка символов

Байтная – 1 символ = 1 байт, 60-70 гг., 256 символов, кодовая таблица, классический Си, тип данных - char:

- первые 128 символов для все кодовых таблиц совпадают (латиница, цифры)
- кодовые таблицы для кириллицы:
    - кодовая таблица Windows **CP-1251**
    - кодовая таблица DOS **CP-866**;
    - кодовая таблица Международной организации стандартизации (**ISO**), используемая семействами мобильных ОС UNIX, Linux,FreeBSD и т.п. - **ISO-8859-5**;
    - кодовые таблицы «советских» стандартов кодов информационного обмена (**KOI-8)** - **CP KOI-8U** и **CP KOI-8R**
- консольное приложение в Windows в VisualStudio в DOS-кодировке, обработка – в Windows (CP-1251), для кириллицы требуется перекодировка, при работе в NetBeans – не требуется

Кодировка символов (байтная)

Таблица 2. Кодовая страница CP866

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
8.	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
9.	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A.	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
B.	▒	▓	▓	│	┤	┤	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
C.	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
D.	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▒	▄	▌	▐	▀
E.	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
F.	Ё	ё	Є	є	Ї	ї	Ў	ў	°	∙	·	√	№	¤	■	A0

Таблица 3. Кодовая страница CP1251

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
8.	Ђ	Ѓ	‚	ѓ	„	…	†	‡	€	‰	Љ	‹	Њ	Ќ	Ћ	Џ
9.	ђ	‘	’	“	”	•	–	—		™	љ	›	њ	ќ	ћ	џ
A.		Ў	ў	Ј	¤	Ґ	¦	§	Ё	©	Є	«	¬		®	Ї
B.	°	±	І	і	ґ	µ	¶	·	ё	№	є	»	ј	Ѕ	ѕ	ї
C.	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D.	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E.	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F.	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

Таблица 4. KOI8-R

	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
8.	└	┘	├	┤	┬	┴	┼	▀	▄	█	▌	▐
9.	▒	∙	√	≈	≤	≥		⌠	°	²	·	÷
A.	═	╒	╓	╕	╖	╘	╙	╛	╜	╝	╞	╟
B.	╠	╡	ё	╢	╣	╤	╥	╦	╧	╨	╩	©
C.	ю	а	б	ц	д	е	ф	г	х	и	й	к
D.	л	м	н	о	п	я	р	с	т	у	ж	в
E.	ь	ы	з	ш	э	щ	ч	ъ	Ю	А	Б	Ц
F.	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О
	П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш
	Э	Щ	Ч	Ъ								

DOS **CP-866**;

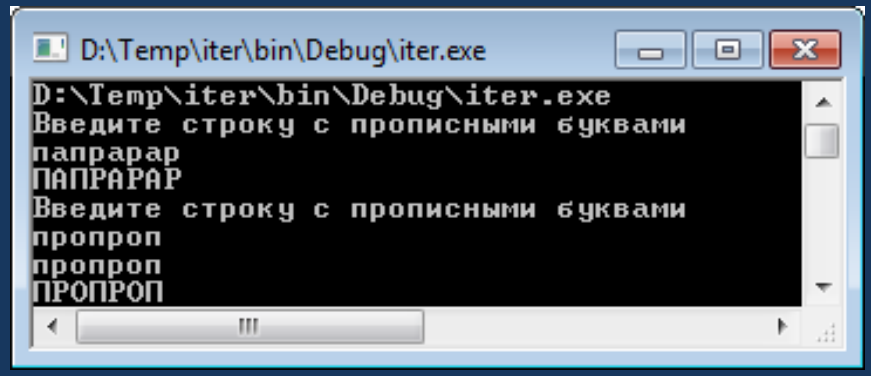
Windows **CP-1251**

«советские» стандарты (**KOI-8**) - **CP KOI-8U** и **CP KOI-8R**

Кодировка символов (байтная)

```
void russian(){
    char c[80];
    CharToOemA("Введите строку с прописными буквами",c);// Функция явного преобразования для вывода
    puts(c);
    gets(c);
    OemToCharA(c,c);                                    // Функция явного преобразования для вывода
    for (int i=0;c[i]!=0;i++)                           // Преобразования прописных в строчные
        if (c[i]>='a' && c[i]<='я') c[i]=c[i]-'a'+'A';
    CharToOemA(c,c);                                    // Функция явного преобразования для вывода
    puts(c);
    setlocale(LC_ALL,"Russian");                        // Установка преобразования при выводе
    puts("Введите строку с прописными буквами");
    gets(c);
    OemToCharA(c,c);                                    // Функция явного преобразования для ввода
    puts(c);
    for (int i=0;c[i]!=0;i++)                           // Преобразования прописных в строчные
        if (c[i]>='a' && c[i]<='я') c[i]=c[i]-'a'+'A';
    puts(c);
    }
```

Архив
cprog/programms/russian.cpp



Кодировка символов (Unicode-1,2)

UNICODE – универсальная система кодирования символов (коды как двоичные числа, имеется несколько вариантов преобразования во внутреннее представление и последовательные потоки). Кодовое пространство UNICODE расширяется
Обозначение U+1D788 – код в 16-ой СС

UNICODE-1 (1991) – 1 символ = 16 разрядов (4 цифры 16СС), 65536 символов. Первые 128 символов совпадают с байтной (латиница, цифры)
UNICODE-2 (1996) - 1 символ = 4 байта, в т.ч. иероглифы, эмодзи

0x0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
400		Ё	Ђ	Ѓ	Є	S	I	Ї	J	Љ	Њ	Ћ	Ќ		Ў	Џ
410	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
420	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
430	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
440	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
450		ё	ђ	ѓ	є	s	i	ї	j	љ	њ	ћ	ќ		ў	џ
490	Ґ	ґ	Ғ	ғ			Җ	җ			Қ	қ				
4A0			Ң	ң											Ү	ү
4B0	Ұ	ұ	Ҳ	ҳ							Һ	һ				
4D0									Ә	ә						
4E0									Ө	ө						

Кодировка символов (UTF-8,16,32)

UTF (UNICODE Transformation Format) - кодировка Unicode в формат последовательной передачи в потоке.
UTF8 - кодировка Unicode последовательностью переменной длины
0x00000000 — 0x0000007F: 0xxxxxxx – первые 128 -> 1 байт
0x00000080 — 0x000007FF: 110xxxxx 10xxxxxx – 2 байта -> 2^11 = 2048
0x00000800 — 0x0000FFFF: 1110xxxx 10xxxxxx 10xxxxxx
0x00010000 — 0x001FFFFF: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Символ		Двоичный код символа	UTF-8 в двоичном виде	UTF-8 в шестнадцатеричном виде
$	U+0024	0100100	00100100	24
¢	U+00A2	10100010	11000010 10100010	C2 A2
€	U+20AC	100000 10101100	11100010 10000010 10101100	E2 82 AC
𐍈	U+10348	1 00000011 01001000	11110000 10010000 10001101 10001000	F0 90 8D 88

UTF16 - двухбайтные слова (short), область 0xD800—0xDFFF первого слова, отведена для «суррогатных пар»

- Code <= $FFFF_{16}$ записываются 16-битным словом.
- Code $10000_{16}..10FFFF_{16}$ (больше 16 бит)
  - Code - 10000_{16}. = $0…FFFFF_{16}$, (20 бит)
- Старшие 10 бит + $D800_{16}$, = первое слово $D800_{16}..DBFF_{16}$.
- Младшие 10 бит + $DC00_{16}$, = второе $DC00_{16}..DFFF_{16}$.

Кодировка символов Unicode

UTF-32 – 1 символ = 4 байта (32 разряда) – прямое представление UNICODE

BOM – Byte Order Mark – идентификатор формата
UTF-8EF BB BF
UTF-16BE FE FF
UTF-16LE FF FE
UTF-32BE 00 00 FE FF
UTF-32LE FF FE 00 00

Кодировка символов

Пример перекодировки кириллицы из UTF8 в CP1251 – диапазон кириллицы из двухбайтного представления склеивается в байт и переносится в диапазон начала кириллицы в CP1251

```
void UTFtoChar(){
    char *s=strdup(str);
    int i,j;
    for(i=0,j=0; s[i]!=0;j++){
        if((s[i] & 0x0E0)!=0x0C0)    //0xC0 = [110]0 0000
            str[j]=s[i++];           //0xE0 = маска [111]0 0000
        else {                       //три старших разряда  = 110
            int v=s[i++] & 0x03;     //2 разряда из первого и 6 из второго
            v = (v<<6 | s[i++] & 0x03F) - 0x10 + 0xC0;
            str[j]=v;                // Начало кириллицы в CP1251 = 0xC0
        }                            // Начало кириллицы в UNICODE = 0x410 (0x10)
    }
    delete []s;
    str[j]=0;
    }
```

Одиозный??? пример

Выбор имени файла в Android с иероглифами (Java). Исходная строка:

1. Encoded=pGBYvlywWU5xX9JQ+IVSAcLuUdcWOJp1Fae1hoAekgQllsw5lg==
имя файла в Unicode, длинные коды передаются в Base64 –
3 байта -> 4 символа ASCII (по 2^6=64 символьной таблице)

2. Двоичные данные перекодируются в строку в кодировке UTF-16

3. Получаются иероглифы

Обычный файл – в исходной строке – ИМЯ ФАЙЛА

```
public Base64Coder() {}
public static void main(String argv[]){
    byte bb[]={1,2,3,4,5};
    System.out.println(encode(bb));
    String ss="pGBYvlywWU5xX9JQ+IVSAcLuUdcWOJp1Fae1hoAekgQllsw5lg==";
    byte dd[] = decode(ss);
    String mm = null;
    try {
        mm = new String(dd, charsetName: "UTF-16");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    System.out.println(mm);
}
```

```
Base64Coder
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
AQIDBAU=
[illegible]

Process finished with exit code 0
```

Строка

Строка – последовательность символов

Строка – последовательность символов в памяти

Символ перехода к следующей строке (конца строки)- **\n**

Си: последовательность, ограниченная символом с кодом 0 (\0) – конец строки в памяти

Тип данных – строка: целое (short) - счетчик символов (байтов) + последовательность байтов
Паскаль, Бейсик, Java-вывод в двоичный поток wtiteUTF

0	6	‘С’	‘т’	‘р’	‘о’	‘к’	‘а’

Строка в памяти

```
char c0[]={'q','q','q','q','q'};     // Нет символа конца строки
char c1[20]={'a','b','c','d',0};
char c2[]="querty";
char c3[20]="string1\nstring two";
char c4[]={'1','2','3','4','5'};     // Нет символа конца строки
puts(c1);
puts(c2);
puts(c3);
puts(c4);
puts(c0);
```

VisualStudio - выравнивание, стек «вниз головой»

```
abcd
querty
string1
string two
12345MMMMMMMMMMstring1
string two
qqqqqMMMMMMMMMM
abcd
```

c4	0x003efd88 "12345MMMMMMMMMMstring1str	16
dd	0x003efdf0 ""	
c2	0x003efdb4 "querty"	16
c1	0x003efdc4 "abcd"	16
c3	0x003efd98 "string1string two"	28
c0	0x003efde0 "qqqqqMMMMMMMMMM"	28

- при отсутствии \0 движется, пока не найдет, либо пока не свалится по защите памяти
- соответствие между размерностью массива и длиной строки проверяется программой
- при записи программа должна сама записывать \0 и контролировать размерность массива

Строка в памяти

```
char c0[]={'q','q','q','q','q'};     // Нет символа конца строки
char c1[20]={'a','b','c','d',0};
char c2[]="querty";
char c3[20]="string1\nstring two";
char c4[]={'1','2','3','4','5'};     // Нет символа конца строки
puts(c1);
puts(c2);
puts(c3);
puts(c4);
puts(c0);
```

c2	{...}
c2[0]	113 'q'
c2[1]	117 'u'
c2[2]	101 'e'
c2[3]	114 'r'
c2[4]	116 't'
c2[5]	121 'y'
c2[6]	0 '\\000'

CodeBlocks

```
abcd
querty
string1
string two
12345string1
string two
qqqqq
abcd
```

```
c0=28fefb
c1=28fee7
c2=28fee0
c3=28fecc
c4=28fec7
```

20 7 20 5

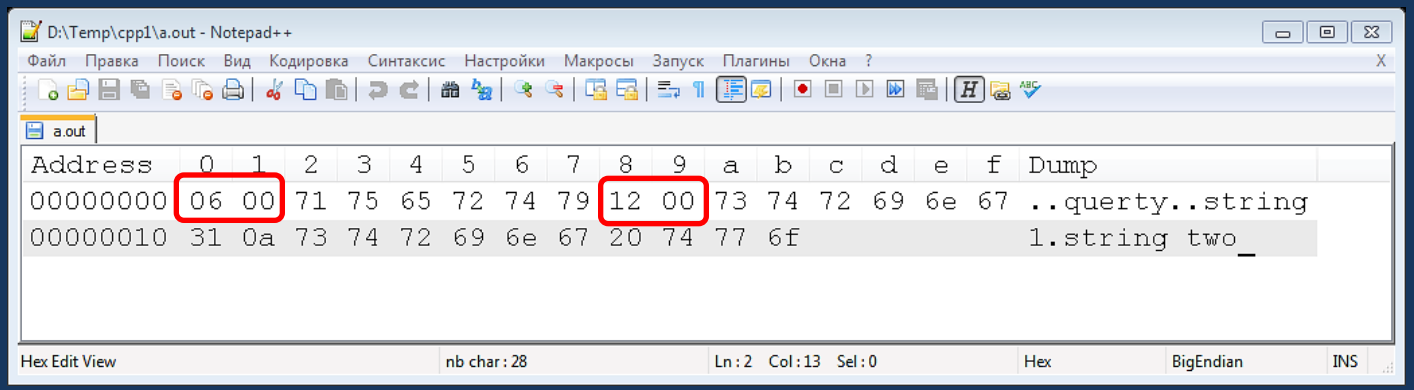
```
printf("c0=%x\nc1=%x\nc2=%x\nc3=%x\nc4%x\n",c0,c1,c2,c3,c4);
```

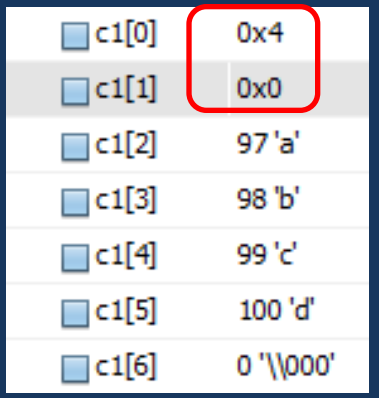
Строка со счетчиком

```
void writeTo(FILE *fd,char cc[]){
    short len = strlen(cc);
    fwrite(&len,sizeof(short),1,fd);
    fwrite(cc,len,1,fd);
    }
void convertFore(char cc[]){
    short len = strlen(cc);
    int i;
    for(i=len-1;i>=0;i--)
        cc[i+2]=cc[i];
    cc[0]=len;          // Младшим байтом вперед
    cc[1]=len<<8;       // Старший байт
    }
```

```
void convertBack(char cc[]){
    short len = cc[0] & 0x0FF | cc[1]<<8;
    int i;              // Арифметическое расширение
    for(i=0;i<len;i++)  // СТАРШЕГО РАЗРЯДА
        cc[i]=cc[i+2];
    cc[len]=0;
    }
```

Notepad++ -> Плагины -> PluginManager -> HEXEditor
Плагины -> HEXEditor -> View in Hex





Строка

Символ перехода к следующей строке (конца строки)

Используемые коды символов:

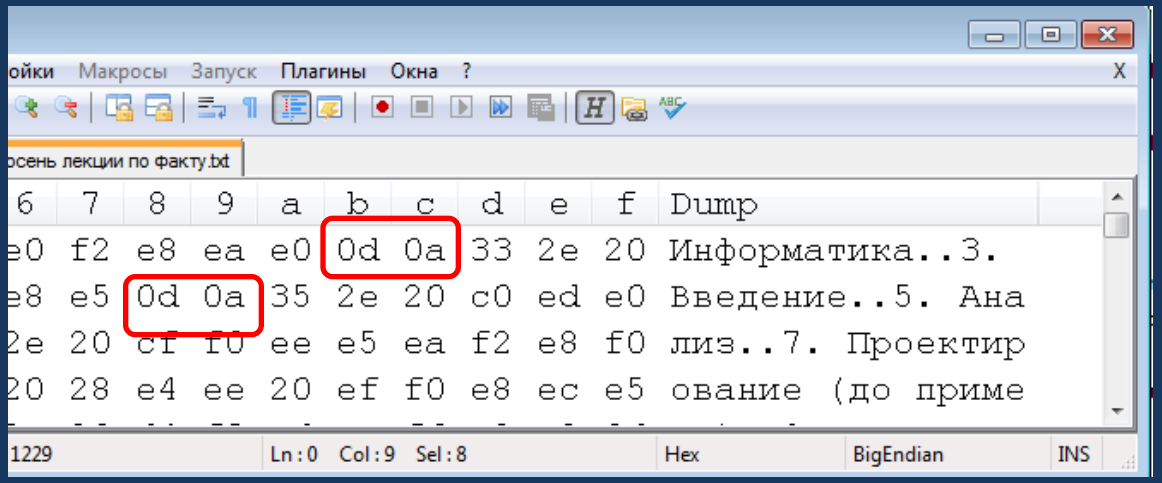
- 0x0D – CR (carriage return) – ВК (возврат каретки) - **\r**
- 0x0A – LF (line feed) – ПС (перевод строки) - **\n**

Строка и в/в в Си: **\n**

Символ ввода (enter): **\r**

Файл Unix(Linux): **\n**

Файл Windows: **\r\n**



Строка

Возврат каретки (англ. carriage return, CR) — управляющий символ ASCII (0x0D, 1310, '\r'), при выводе которого курсор перемещается к левому краю поля, не переходя на другую строку.

Перевод строки (от англ. line feed, LF — «подача [бумаги] на строку») — управляющий символ ASCII (0x0A, 10 в десятичной системе счисления, '\n'), при выводе которого курсор перемещается на следующую строку. В случае принтера это означает сдвиг бумаги вверх, в случае дисплея — сдвиг курсора вниз, если ещё осталось место, и прокрутку текста вверх, если курсор находился на нижней строке..

- LF (ASCII 0x0A) используется в Multics, **UNIX, UNIX-подобных операционных системах (GNU/Linux, AIX, Xenix, Mac OS X, FreeBSD и др.),** BeOS, Amiga UNIX, RISC OS и других
- CR (ASCII 0x0D) используется в 8-битовых машинах Commodore, машинах TRS-80, Apple II, системах **Mac OS до версии 9** и OS-9
- CR+LF (ASCII 0x0D 0x0A) используется в DEC RT-11 и большинстве других ранних не-UNIX- и не-IBM-систем, а также в CP/M, MP/M (англ.), MS-DOS, OS/2, **Microsoft Windows**, Symbian OS, протоколах Интернет

Внешняя и внутренняя форма числа

Внутренняя форма представления числа - представление числа в виде целой или вещественной переменной (двоичная)

Внешняя форма представления числа - представление числа в виде строки символов – цифр в *заданной системе счисления*

```
int StringToInt(char c[]){
int n,i;
for (i=0; !(c[i]>='0' && c[i]<='9'); i++)
    if (c[i]=='\0') return 0;                    // Поиск первой цифры
for (n=0; c[i]>='0' && c[i]<='9'; i++)           // Накопление целого
n = n * 10 + c[i] - '0';                         // "цифра за цифрой"
return n; }
```

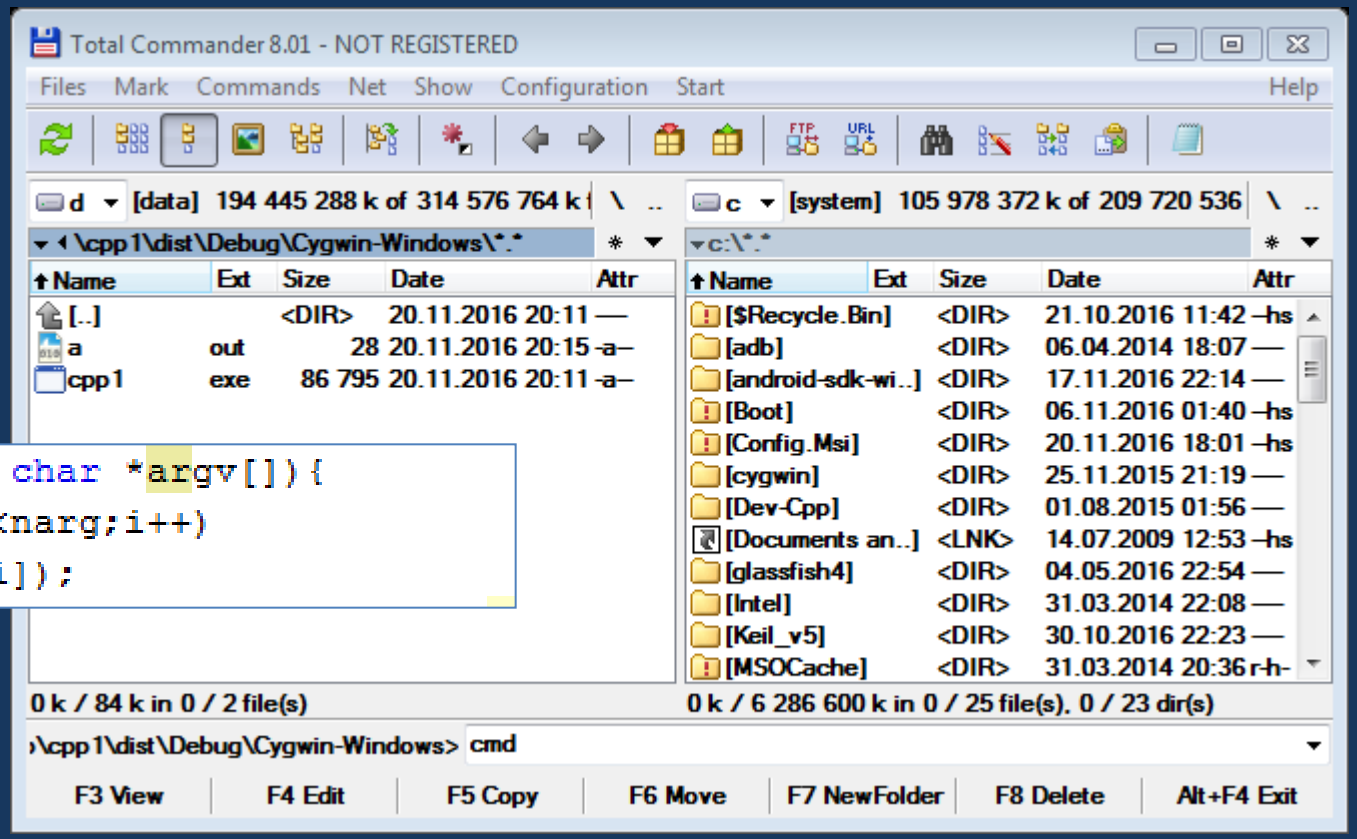
```
void IntToString(char c[], int n)
{ int nn,k;
for (nn=n, k=0; nn!=0; k++, nn/=10); // Подсчет количества цифр числа
c[k] = '\0';                         // Конец строки
for (k--; k >=0; k--, n /= 10)       // Получение цифр числа
c[k] = n % 10 + '0';                 // в обратном порядке
}
```

Внешняя и внутренняя форма числа

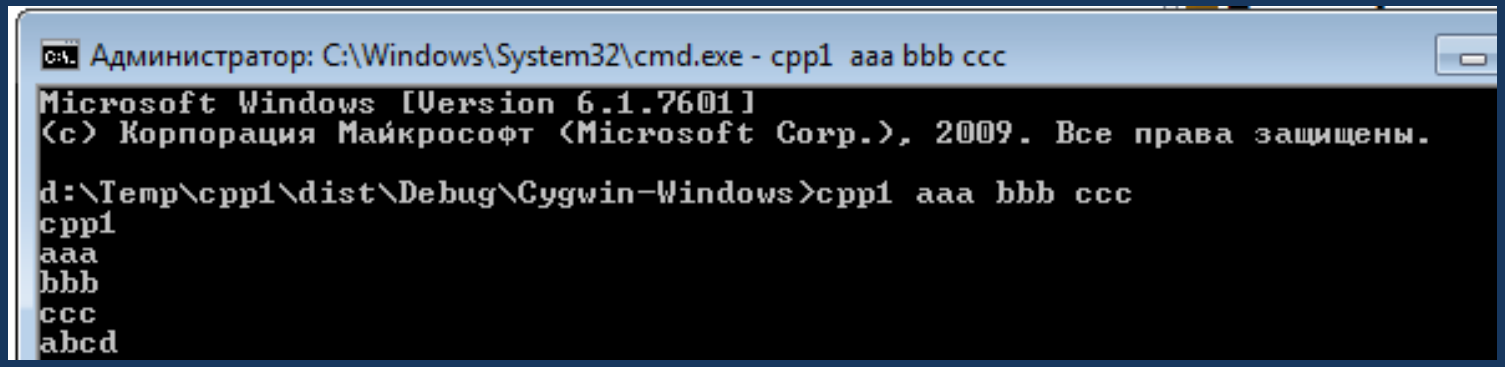
```
void FloatToString(char c[], double v)
{ int i,nn,k,kk;
for (nn=v, k=0; nn!=0; k++, nn/=10);    // Подсчет количества цифр
kk=k-1; c[k++] = '.';                   // целой части числа
for (nn=v; kk >=0; kk--, nn /= 10)      // Получение цифр числа
c[kk] = nn % 10 + '0';                  // в обратном порядке
v-=(int)v;                              // Убрать целую часть
for (i=0; i<6; i++) {
   v *= 10.;                            // *10 - очередная цифра
   c[k++]=(int)v + '0';                 // в целой части - записать
   v -= (int)v;                         // и отбросить
   }

c[k]=0;
}
```

```
double StringToFloat(char c[]){
int i;
double n,v;
for (i=0; !(c[i]>='0' && c[i]<='9'); i++)
    if (c[i]=='\0') return 0;              // Поиск первой цифры
for (n=0; c[i]>='0' && c[i]<='9'; i++)     // Накопление целого
n = n * 10 + c[i] - '0';                   // "цифра за цифрой"
if (c[i]!='.') return n;
for (i++, v=0.1; c[i]>='0' && c[i]<='9';i++){
   n+=v*(c[i] - '0');                      // Взвешивание цифр
   v=v/10;                                 // весом разряда дробной части
   }
return n; }
```

Интерфейс командной строки




```
int main(int narg, char *argv[]){
    for (int i=0;i<narg;i++)
        puts(argv[i]);
```



Стандартный ввод/вывод

Медицинские факты:

- стандартный ввод/вывод в библиотеке stdio.h – текстовый поток на консоль перенаправляется в файл (из файла)
- Функции работы с файлом идентичны функциям с консоли, дополнительный параметр – указатель (адрес) дескриптора файла

Функция		**Параметры и результат**
FILE *fopen(char *name, char *mode)	Открыть файл	Результат FILE* - указатель на описатель файла или NULL fd – идентификатор файла (указатель на описатель) name – строка с именем файла mode – строка режима работы с файлом
int fclose(FILE *fd)	Закрыть файл	
FILE *freopen(char *name, char *mode, FILE *fd)	Закрыть и открыть повторно	
FILE *tmpfile(void)	Создать и открыть временный с уникальным именем	

Стандартный ввод/вывод

Посимвольный ввод		**Параметры и результат**
int getc(FILE *fd)	Явно указанный файл	Код символа или EOF
inc getchar(void)	Стандартный ввод	
int ungetc(int ch, FILE *fd)	Возвратить символ в файл (повторно читается)	
Посимвольный вывод		
int putc(int ch, FILE *fd)	Явно указанный файл	Код символа или EOF
inc putchar(int ch)	Стандартный вывод	

Построчный ввод		**Параметры и результат**
char *fgets(char *str, int n, FILE *fd)	Явно указанный файл	n - максимальная длина строки str или NULL(ошибка)
char *gets(char *str)	Стандартный ввод	
Построчный вывод		
char *fputs(char *str, FILE *fd)	Явно указанный файл	str или NULL(ошибка)
char *puts(char *str)	Стандартный вывод	

Стандартный ввод/вывод

Форматированный ввод		**Параметры и результат**
int fprintf(FILE *fd, char format[],...)	Явно указанный файл	Число фактических параметров, для которых введены значения, или EOF
int printf(char format[],...)	Стандартный ввод	
int sprintf(char str[], char format[],...)	Строка в памяти	
Форматированный вывод		
int fprintf(FILE *fd, char format[],...)	Явно указанный файл	Число выведенных байтов или EOF
int printf(char format[],...)	Стандартный вывод	
int sprintf(char str[], char format[],...)	Строка в памяти	

Стандартный ввод/вывод

Проблемы форматированного ввода/вывода:

- ввод буферизуется до окончания строки
- при вводе по числовому формату символ – ограничитель **остается в потоке**
- **printf(“%s”,…)** - выводит строку
- **scanf(“%s”,…)** – вводит до первого разделителя (“ ”,\n,\t,”;”,”,”), но не пустой элемент

```
int aa=0;
printf("aa=");
scanf("%d",&aa);
char dd[20];
scanf("%s",dd);
printf("1.%s\n",dd);
scanf("%s",dd);
printf("2.%s\n",dd);
```

```
aa=12
aaaaa bbbbb
1.aaaaa
2.bbbbb
```

```
int aa=0;
printf("aa=");
scanf("%d",&aa);
char dd[20];
gets(dd);
printf("1.%s\n",dd);
gets(dd);
printf("2.%s\n",dd);
```

```
aa=125
1.
ffff
2.ffff
```

Первый gets читает пустую строку – остаток от ‘125\n’

Вводится одна строка до enter, scanf читает 2 строки (разделитель – пробел)

Альтернативы стандартному вводу/выводу

XML – *тегированный* структурированный (иерархический) формат описания переменных (обычных, структур, массивов и их значений) – имя тега = тип объекта

JSON – аналогичный, но не тегированный. Внешний тип должен быть задан

```
<model>
  <layers>
    <neurons name="Слой 2" type="Нейрон Гаврилова">
      <param name="DH" value="0.05"/>
      <param name="HMax" value="0.95"/>
      <param name="L" value="0.01"/>
    </neurons>
    <neurons name="Слой 1" type="Пороговый вход">
      <param name="spikeLevel" value="0.4"/>
    </neurons>
    <neurons name="Слой 3" type="Сглаживающий">
    </neurons>
  </layers>
  <link from="Слой 1" to="Слой 2" type="0" param="5"/>
  <link from="Слой 2" to="Слой 3" type="0" param="1"/>
  <link from="Вход" to="Слой 1" type="1" param="0"/>
  <statistics>
    <statistic name="Слой 3" layer="Слой 3"/>
    <statistic name="Слой 2" layer="Слой 2"/>
    <statistic name="вход" layer="Вход"/>
  </statistics>
  <input name="Слой 0"/>
  <output name="Слой 3"/>
</model>
```

Альтернативы стандартному вводу/выводу

JSON – аналогичный, но не тегированный. Внешний тип должен быть задан


```
[
{"type":0,"ways":[
        {"marsh":"3","name":"3","stopb":"пос. Северный",
                "stope":"Вокзал \"Новосибирск-Главный\""},
        {"marsh":"007","name":"7","stopb":"Белоусова ул.",
                "stope":"Микрорайон \"Щ\""},
        {"marsh":"008","name":"8э","stopb":"Цветной проезд",
                "stope":"М \"Речной вокзал\""},
        {"marsh":"016","name":"16","stopb":"Обл. больница",
                "stope":"Затон"},
        {"marsh":"021","name":"21","stopb":"УМ-3 (Инская)",
                "stope":"Вокзал \"Новосибирск-Главный\""},
        {"marsh":"023","name":"23","stopb":"Демакова ул.","stope":"ОРМЗ"},
        {"marsh":"27","name":"27","stopb":"Отделение связи №13 (Пашино)",
                "stope":"Площадь Калинина"}
        ]}
```

Формат. Парсинг

Формат - описание варьируемой последовательности элементов, в которой значение текущего элемента может определять порядок следования идущих за ним. (**саморазворачивающийся)**:

- последовательное размещение разных форматных единиц друг за другом
- повторение в цикле с использованием счетчика повторений или элемента-ограничителя
- выбор одной из нескольких форматных единиц в зависимости от значения элемента-селектора
- вложенность: последовательность форматных единиц является составной частью формата верхнего уровня

Виды элементов формата:

- данные
- ограничители
- счетчики повторений
- селекторы (идентификаторы последующего значения или формата)

Парсинг – чтение потока в формате и преобразование во внутреннюю стуктуру данных

Формат. Парсинг

```
double F1(char c[]){
    FILE *fd=fopen(c,"r");       // Открыть файл
    if (fd==NULL) return 0;
    double s=0;
    while(1){                    // Цикл чтения последовательности
        int v;
        double dd;
        fscanf(fd,"%d",&v);      // Читать очередное значение
        if (v==0) break;         // Ограничитель - 0
        if (v>0) s+=v;           // Если >0 - добавить к сумме
        else{                    // Если <0 - читать следующее
            fscanf(fd,"%lf",&dd);
            s+=dd;               // за ним - вещественное
            }}
        fclose(fd); return s;}
//-------------------------------------------
int F2(char c[],int d[]){
    FILE *fd=fopen(c,"r");       // Открыть файл
    if (fd==NULL) return 0;
    double s=0;
    int n=0;
    while(1){                    // Цикл чтения последовательности
        int v;
        double dd;
        fscanf(fd,"%d",&v);      // Читать очередное значение
        if (v==0) break;         // Ограничитель - 0
        if (v>0) d[n++]=v;       // Если >0 - добавить в массив
        else{
            int k=-v;            // Если <0 - счетчик повторений
            fscanf(fd,"%d",&v); // Следующее значение - повторяющееся
            while(k--!=0)        // Цикл копирования повторяющегося
                d[n++]=v;        // значения
            }}
        fclose(fd); return n;}
```

```
4 5 6 -1 2.55 3 3 -1 4.75 3 0
```

```
4 5 6 6 6 6 6 7 7 2 3 6 6 6 6 0     // до сжатия
4 6 -5 6 -2 7 2 3 -4 6 0            // после сжатия
```

Формат. Парсинг

Реализация формата в программе:

- Через переменные состояния, счетчики, признаки – автоматная модель
- Формат «зашивается» в текст программы

```
int find(char s[]) {
int i,n,lmax,imax;
    for (i=0,n=0,lmax=0,imax=-1; s[i]!=0; i++){
    if (s[i]!=' ') n++;              // символ слова увеличить счетчик
        else {                       // перед сбросом счетчика
            if (n > lmax) { lmax=n; imax=i-n; }
        n=0;                        // фиксация максимального значения
        }}                          // то же самое для последнего слова
if (n > lmax) { lmax=n; imax=i-n; }
return imax; }
```

```
//---- Поиск слова максимальной длины пословная обработка
 int find(char in[]){
 int i=0, k, m, b;
 b=-1; m=0;
    while (in[i]!=0) {              // Цикл пословного просмотра строки
        while (in[i]==' ') i++;     // Пропуск пробелов перед словом
        for (k=0;in[i]!=' ' && in[i]!=0; i++,k++); // Подсчет длины слова
            if (k>m){               // Контекст выбора максимума
            m=k; b=i-k; }           // Одновременно запоминается
    }                               // индекс начала
 return b; }
```

Двоичные и текстовые файлы

По содержимому - двоичные и текстовые файлы:

- символы текста
- данные во внутреннем представлении («образ памяти») – термин «двоичный» связан с внутренней формой представления
  - используются функции чтения/записи переменной (массива) как массива байтов fread/fwrite на уровне физического представления в памяти
  - файл открывается с модификатором «rb» или «wb»

По способу доступа:

- последовательный
- произвольного доступа – установка текущей позиции на заданный байт

Медицинский факт: **«из файла байта не выкинешь»** - вставка/удаление путем полного переписывания файла, в т.ч. и произвольного доступа

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

Двоичные и текстовые файлы

Компактная запись вещественного массива – с нулевой дробной частью и не превышающие 255 пишутся как байты (беззнаковые целые 0…254), остальные с байтным префиксом -1

```
void writeCompact(FILE *fd,double cc[], int sz){
    fwrite(&sz,sizeof(int),1,fd);    // Счетчик переменных
    for(int i=0;i<sz;i++){
        int vv;                      // Нет дробной и меньше 255
        if (cc[i]<255 && (cc[i]-(int)cc[i]==0)){
            vv=cc[i];                // Преобразовать в целое
            fwrite(&vv,1,1,fd);      // Записать 1 байт
            }
        else{
            vv=-1;
            fwrite(&vv,1,1,fd);      // Записать -1
            fwrite(&cc[i],sizeof(double),1,fd);
            }                        // Вещественное  - 8  байт
        }
    }
```

0x2d = 32+13=45
0x21 = 32+1 = 33

```
double dd[]={1,5,4,1.1,700,45,33};
fd = fopen("b.out","wb");
writeCompact(fd,dd,7);
fclose(fd);
```

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000000	07	00	00	00	01	05	04	ff	9a	99	99	99	99	99	f1	3f
00000010	ff	00	00	00	00	00	e0	85	40	2d	21					

Двоичные и текстовые файлы

```
int readCompact(FILE *fd,double cc[]){
    int sz=0;
    fread(&sz,sizeof(int),1,fd);
    for(int i=0;i<sz;i++){
        int vv=0;           // Читается ТОЛЬКО 1 байт (косяк был...)
        fread(&vv,1,1,fd);
        if (vv!=255)
            cc[i]=vv;
        else
            fread(&cc[i],sizeof(double),1,fd);
        }
    return sz;
    }
```