ЛЕКЦИЯ 9

ОСНОВЫ ПРОГРАММИРОВАНИЯ

СИМВОЛ ТЕКСТА

- Базовый тип данных **char** понимается трояко:

- 1. как байт - минимальная адресуемая единица представления данных в компьютере
- 2. как целое со знаком (в диапазоне –127…+127)
- 3. как **символ текста**.

СИМВОЛ ТЕКСТА

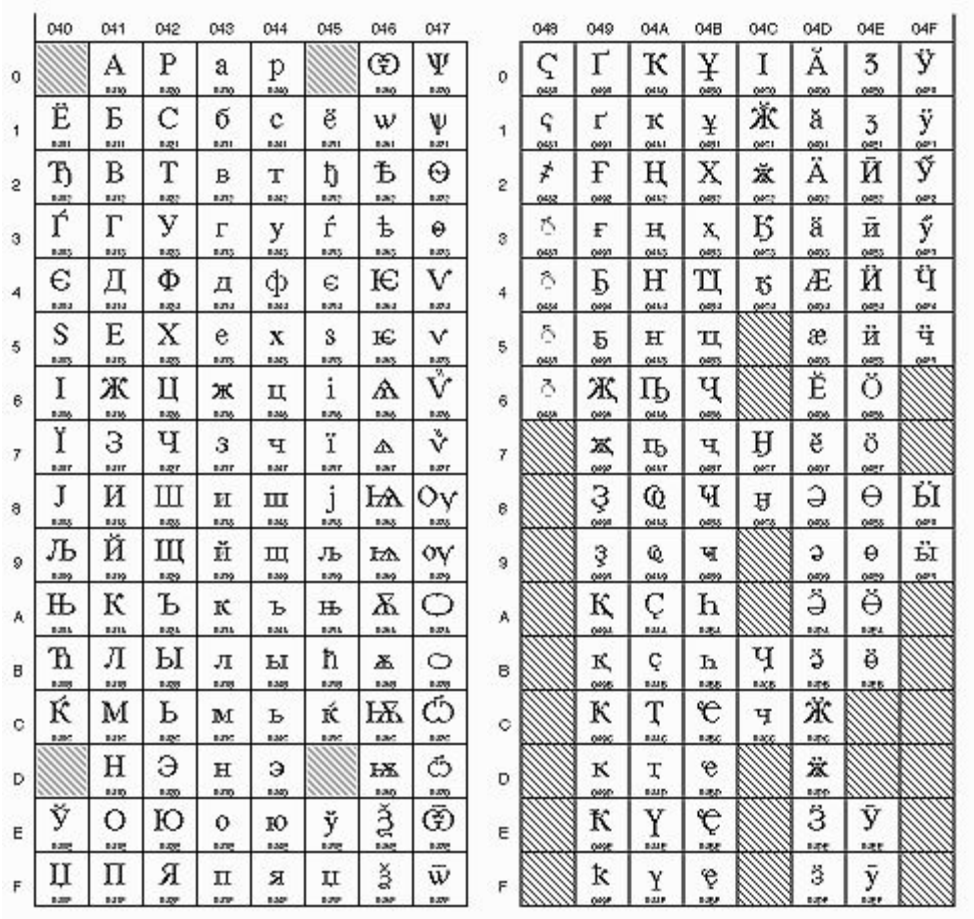
Стандартом установлено соответствие между символами и присвоенными им значениями целой переменной (**кодами**).

' '	- 0x20,	'B'	- 0x42,
'*'	- 0x2A,	'Y'	- 0x59,
'0'	- 0x30,	'Z'	- 0x5A,
'1'	- 0x31,	'a'	- 0x61,
'9'	- 0x39,	'b'	- 0x62,
'A'	- 0x41,	'z'	- 0x7A

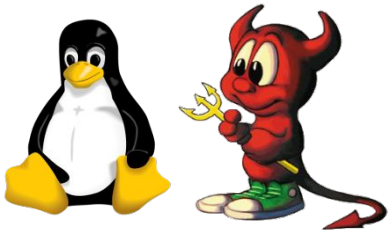
КОДОВЫЕ ТАБЛИЦЫ
КИРИЛЛИЦА В UNICODE

wchar_t

Unicode (U+0400 to U+04FF)



КОДОВЫЕ ТАБЛИЦЫ
КИРИЛЛИЦА В РАЗЛИЧНЫХ КОДИРОВКАХ



- ASCII – ISO/IEC 8859 (коды 128-255)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8		Ё	Ђ	Ґ	Є	S	І	Ї	J	Љ	Њ	Ћ	К	-	Ў	Ц
9	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

	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	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
A	№	ё	ђ	ґ	є	s	і	ї	j	љ	њ	ћ	к	§	ў	џ

- Семейство KOI8 (КОИ-8R)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	─	│	┌	┐	└	┘	├	┤	┬	┴	┼	▀	▄	█	▌	▐
9	═	‖	┌	ё	┌	┌	┐	┐	┐	└	└	└	┘	┘	┘	├
A	ю	а	б	ц	д	е	ф	г	х	и	й	к	л	м	н	о
B	Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	░	▒	▓	│	■	•	√	≈	≤	≥	□	⌠	°	²	·	÷
9	┤	┤	┤	Ё	┤	┤	┬	┬	┬	┴	┴	┴	┼	┼	┼	©
A	п	я	р	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ
B	П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ

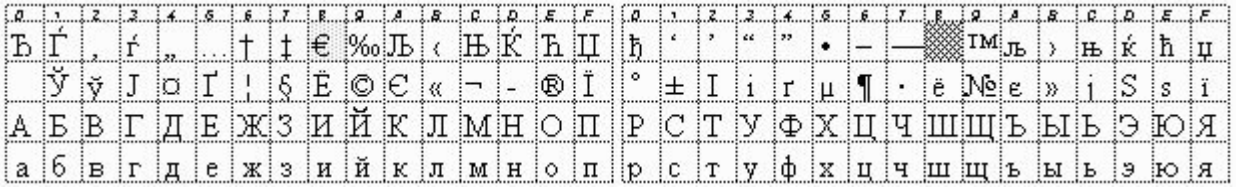
- Альтернативная (IBM code page 866)

	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	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
A	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
B	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

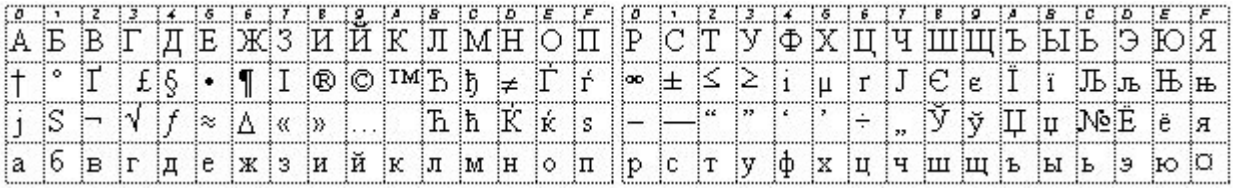
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
9	░	▒	▓	│	┤	┤	╡	╖	╕	╣	║	╗	╝	╜	╛	┐
A	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▀	▄	▌	▐	▀
B	Ё	ё	Є	є	Ї	і	Ў	ў	°	•	·	√	№	¤	■	□

КОДОВЫЕ ТАБЛИЦЫ
КИРИЛЛИЦА В РАЗЛИЧНЫХ КОДИРОВКАХ

- Windows-1251 (CP1251)



- x-mac-cyrillic (CP10007)



СИМВОЛЬНЫЕ (ЛИТЕРНЫЕ) КОНСТАНТЫ

- Для представления отдельных символов можно пользоваться ***символьными (литерными) константами***.

- Транслятор вместо такой константы всегда подставляет код соответствующего символа.

- char c;
- for (c= 'A'; c <= 'Z'; c++) ...
- for (c=0x41; c <=0x5A; c++) ...

СТРОКОВЫЕ ЛИТЕРАЛЫ

- **Строковые литералы** представляются массивами константных символов.

- char * ptr = "Hello"
- ptr[1]='a'; // Попытка записи в область памяти,
  // предназначенную для чтения

- char stackArray[] = "hello";
- stackArray[1] = 'a'; // эту копию можно
  // модифицировать

СТРОКА

- **Строкой** называется последовательность символов, ограниченная символом '\0'.

- Местом хранения строки является массив символов.
- char str1 [] = "Пример строки";

- Строка является *структурой данных*, а массив – *переменной*.

СТРОКА

- ***Строка хранится в массиве символов***, массив символов может быть инициализирован строкой, а может быть заполнен программно:

- char A[20] = { 'С','т','р','о','к','а','\0' };
- char B[80];
- for (int i=0; i<20; i++) B[i] = 'A';
- B[20] = '\0';

СТРОКА

- ***Строка имеет переменную размерность***, поэтому работать с ней нужно в цикле, ограниченном не размерностью массива, а условием обнаружения символа конца строки:

- for (i=0; B[i] !='\0'; i++)…

СТРОКА

- ***Соответствие размерности массива и длины строки*** транслятором не контролируется:

- char C[10],B[]="Строка слишком длинная";
- // следить за переполнением массива
- // и ограничить строку его размерностью
- for (i=0; i<9 && B[i]!='\0'; i++) C[i] = B[i];
- C[i]='\0';

НЕОТОБРАЖАЕМЫЕ СИМВОЛЫ

	Код	Действие
\a	0x07	Звуковой сигнал
\b	0x08	Курсор на одну позицию назад
\f	0x0C	Переход к началу (перевод формата)
\n	0x0A	Переход на одну строку вниз(перевод строки)
\r	0x0D	Возврат на первую позицию строки
\t	0x09	Переход к позиции, кратной 8 (табуляция)
\v	0x0B	Вертикальная табуляция по строкам

\\ \' \" \?	Представление символов \, ', ", ?
\Onn	Символ с восьмеричным кодом nn
\xnn	Символ с шестнадцатеричным кодом nn
\0	Символ с кодом 0

ВВОД-ВЫВОД ЦЕЛЫХ ЧИСЕЛ

- Кодирование:
- '0' - '9' 0x30 - 0x39
  'A' - 'Z' 0x41 - 0x5A
  'a' - 'z' 0x61 - 0x7A

- Преобразования при вводе и выводе целых чисел заключаются в переходе от символа-цифры к значению целой переменной, соответствующему этой цифре, и наоборот:
- char c; int n;
  n = c - '0';
  c = n + '0';

ПРЕОБРАЗОВАНИЕ СТРОКИ В ЦЕЛОЕ

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

ПРЕОБРАЗОВАНИЕ ЦЕЛОГО В СТРОКУ

```
void IntToString(char c[], int n)
{
  int nn,k;
// Подсчет количества цифр числа
  for (nn=n, k=1; nn!=0; k++, nn/=10);
  c[k] = '\0';
  for (k--; k > =0; k--, n /= 10)
// Получение цифр числа в обратном порядке
      c[k] = n % 10 + '0';
}
```

ПРЕДСТАВЛЕНИЕ ТЕКСТА

- Текст – упорядоченное ***множество строк.***

- char B[][40] ={”Строка”, ”Другая строка”};

- Первый индекс двумерного массива соответствует номеру строки, второй - номеру символа в нем:

```
int i,k;
for (k=0; A[i][k] !='\0'; k++)
     { … }        // Работа с i-й строкой
```

УПОРЯДОЧИВАНИЕ СТРОК

```
int Compare1(unsigned char s1[],unsigned char s2[])
{
 int n;
 for (n=0; s1[n]!='\0' &&  s2[n]!='\0'; n++)
       if (s1[n] != s2[n]) break;
 if (s1[n] == s2[n]) return 0;
 if (s1[n] <  s2[n]) return -1 ;
 return 1 ;
}
```

КОНТЕКСТНАЯ ЗАМЕНА

- ***Контекстная замена*** - поиск и замена в строке фрагментов, заданных одной строкой (контекста) на фрагмент, заданный другой.

- **Исходные данные и результат:**
- Строки заданы массивами символов s,s1,s2. Результирующая строка размещается в том же массиве, что и исходная. Контроль размерности не производится:

- void Context(char s[], char s1[], char s2[]) {...}

КОНТЕКСТНАЯ ЗАМЕНА

- **Основной цикл программы**

```
void Context(char s[], char s1[], char s2[])
{int n;
  for (n=0; s[n] !='\0'; n++)
  {
/* Если начиная с n-го символа расположена
   подстрока s1, заменить ее на s2 в строке */
  } };
```

КОНТЕКСТНАЯ ЗАМЕНА

- **Проверка утверждения,** что начиная с n-го символа в строке s расположена подстрока s1:

```
int i;
for (i=0; s[n+i] !='\0' & &  s1[i] !='\0'; i++) if (s[n+i] != s1[i]) break;
if (s1[i]=='\0')
{
 // заменить s1 на s2 в строке, начиная с s[n]
}
```

КОНТЕКСТНАЯ ЗАМЕНА

- **Замена подстроки s1 на s2,** начиная с n-го символа строки s, заключается перемещении "хвоста" строки s вправо или влево в зависимости от знака разности длин строк и в переписывании строки s2 на место строки s1.

- int l2,dd,k;
- l2 = strlen(s2); // получение длины строки
- dd = l2 - strlen(s1);
- if (dd != 0)
- { /* сдвинуть "хвост" строки s на dd символов */}
- for (k = 0; k < l2; k++) s[n+k] = s2[k];

КОНТЕКСТНАЯ ЗАМЕНА

- **Сдвиг всего "хвоста"** (начиная с n-го символа)

```
if (dd <  0)      // влево
{for (k=n; s[k+dd]!='\0'; k++) s[k] = s[k+dd];
  s[k]='\0';
}
else              //вправо
{for (k=n; s[k]!='\0'; k++); // найти конец строки
  for (; k != n; k--) s[k+dd] = s[k];
}
```

НЕДОСТАТКИ АЛГОРИТМА

- В случае удачной замены проверка возможности последующей замены производится, начиная со следующего символа.

- Тогда при наличии замен вида "nnn" на "nnnn..." программа будет расширять строку до бесконечности.

ФОРМАТИРОВАНИЕ СТРОКИ

- ***Форматирование строки*** - размещение ее в выходном массиве заданной размерности таким образом, чтобы интервалы между соседними словами отличались не более чем на 1.

- **Исходные данные и результат**.
- Входная строка произвольной длины в массиве IN[], отформатированная строка длины n в массиве OUT[].

ФОРМАТИРОВАНИЕ СТРОКИ

- Выходная строка отвечает следующим требованиям:

  1) слово - любая последовательность символов, кроме пробела ;

  2) после форматирования число пробелов между словами различается не более чем на 1; - первое и последнее слово расположены по краям строки.

ФОРМАТИРОВАНИЕ СТРОКИ

- **Форматирование включает** в себя последовательность из трех действий:

- void format(char IN[], char OUT[], int n)
- {
- // Собрать исходные данные по строке,
- // необходимые для форматирования;
- // Проверить возможность форматирования;
- // Разместить слова в выходной строке.
- }

ФОРМАТИРОВАНИЕ СТРОКИ

- **Данные по строке, необходимые для форматирования:**
- количество слов в строке - ***nw***;
- общее количество символов во всех словах - ***ns***;
- стандартное количество пробелов между словами при форматировании - ***np***;
- оставшееся количество пробелов, добавляемых по одному между словами - ***nr***.

- На этом шаге детализируется проверка возможности форматирования и определяются взаимосвязанные параметры.

ФОРМАТИРОВАНИЕ СТРОКИ

```
void format(char IN[], char OUT[], int n)
// длина OUT- n+1
{     int nw, ns, np, nr;
 // Определение nw, ns ...
 OUT[0] = '\0';
 if (nw <  2) return;        // Мало слов в строке
 np = (n - ns) / (nw - 1);
 if (np < = 0) return; // Много символов в словах
 nr = (n - ns) % (nw - 1);      // Ост. число пробелов
 // Размещение слов в выходной строке
 OUT[n]='\0';}
```

ФОРМАТИРОВАНИЕ СТРОКИ

- **Просмотр строки** при определении параметров и форматировании можно выполнить, используя:
  1) цикл в цикле: цикл просмотра всех слов, в который включен цикл посимвольного просмотра интервала между словами и самого слова;
  2) цикл посимвольного просмотра строки, с использованием признака нахождения внутри слова или вне его - ***inword***.

ФОРМАТИРОВАНИЕ СТРОКИ

- **Определение ns,nw:**

- ```
  for (i=0,ns=0,nw=0,inword =0; ; i++)
  {
  // Анализ символа IN[i] и подсчет параметров;
   if (IN[i] =='\0') break;
  }
  ```
  ```

ФОРМАТИРОВАНИЕ СТРОКИ

- **Размещение слов в выходной строке**:
- {
- for (i=0,j=0,inword =0; ; i++)
- {
- // Анализ символа IN[i] и форматирование
- // (перенос в OUT[j]);
- }
- if (IN[i] =='\0') break;
- }

ФОРМАТИРОВАНИЕ СТРОКИ

- **Анализ символа** состоит в выделении 4 вариантов по двум сравнениям – признака inword и типа символа IN[i] - разделителя или символа слова:

```
{if (IN[i]==' '  | |  IN[i]=='\0')
 if (inword)    { nw++; inword =0; } // Конец слова
 else    {}   // Продолжение разделителя
 else
 {    ns++;
 if (inword)    {}        // Продолжение слова
 else          { inword =1; }  // Начало слова
}}
```

ФОРМАТИРОВАНИЕ СТРОКИ

- **Анализ символа IN[i] и форматирование:**

```
{
  if (IN[i]==' ' || IN[i]=='\0')
        if (inword)
        {     // Конец слова
              inword =0;
              for (k=0; k< np; k++)   OUT[j++]=' '; // Включение "np" пробелов
              if (nr-- > 0) OUT[j++]=' '; // Включение дополнительного пробела
        }
        else
        {} // Продолжение разделителя
  else
        if (inword) OUT[j++]=IN[i]; // Продолжение слова
        else
        {     // Начало слова
              OUT[j++]=IN[i];
              inword =1;
        }
}
```

STRING

- string S, S1, S2; // Объявление трех строк
  cout<<"Как вас зовут? ";
  cin>>S1; // Считали строку S1
  S2="Привет, "; // Присвоили строке значение
  S=S2+S1; // Конкатенация строк
  cout<<S<<endl; // Вывод строки на экран
  cout<<S.length(); // Длина строки S

STRING

- При считывании строк из входного потока считываются все символы, кроме символов–разделителей (пробелов, табуляций и новых строк), которые являются границами между строками.

- ```
  string S1, S2, S3; // объявили 3 строки
  cin>>S1>>S2>>S3; // ввод «Мама мыла раму»
  ```

- В S1 будет записана строка "Мама",
- в S2 — "мыла",
- в S3 — "раму".
  ```

STRING

- Если нужно считать строку со всеми пробелами, то необходимо использовать функцию getline следующим образом:

```
string S;
getline(cin,S);
```

ЗАДАЧИ

- 1. Напишите программу, заменяющую в тексте вхождения цифр словами (1 - один) и выводящую результат на экран в отформатированном виде.

- 2. Напишите программу, упорядочивающую слова в тексте: а) по длине; б) по алфавиту.

- 3. Напишите программу, которая по описанию пути к определяет точные координаты, считая, что начало координат находится в начале пути, ось OX направлена на восток, ось OY – на север.

СТРОКИ В СТИЛЕ С
CSTRING (STRING.H)

Функции копирования	
memcpy	Скопировать блок данных из памяти.
memmove	Переместить блок данных в память.
strcpy	Скопировать строку.
strncpy	Скопировать n символов строки.

Функции объединения (конкатенации)	
strcat	Объединение строк.
strncat	Добавление n символов к строке.

СТРОКИ В СТИЛЕ С
CSTRING (STRING.H)

Функции отношения (сравнения)	
memcmp	Сравнение двух блоков памяти.
strcmp	Сравнение двух строк
strcoll	Сравнение двух строк по категориям.
strncmp	Сравнение n первых символов двух строк
strxfrm	Преобразование строки, с учетом локали.

Другие	
memset	Заполнить n байтов блока памяти указанным символом.
strerror	Интерпретация кодов ошибок в понятные сообщения об ошибках
strlen	Определить длину строки.

СТРОКИ В СТИЛЕ С
CSTRING (STRING.H)

Функции поиска	
memchr	Поиск символа в блоке памяти
strchr	Найти первое вхождение символа в строке.
strcspn	Выполняет поиск первого вхождения в строку str1 любого из символов строки str2, и возвращает количество символов до найденного первого вхождения.
strpbrk	Выполняет поиск первого вхождения в строку str1 любого из символов строки str2, и возвращает указатель на найденный символ.
strrchr	Поиск последнего вхождения указанного символа.
strspn	Поиск символов строки str2 в строке str1. Возвращает длину начального участка строки str1, который состоит только из символов строки str2.
strstr	Функция ищет первое вхождение подстроки str2 в строке str1.
strtok	Поиск лексем в строке, используя разделители.