# Элементы ЯПВУ. СИМВОЛЫ

Pascal

C / C++

**В языках программирования существуют средства для работы с текстами. Текстовые данные представляются в виде отдельных символов или в виде строк текста.**

**Символ – это один элемент алфавита языка. В Pascal и C символьные данные описываются типом данных char, размер 8 бит (1 байт) – описывает символы с кодом от 0 до 255 (расширенный ASCII).**

**Char - порядковый тип данных – то есть:**

- **представляет собой конечное упорядоченное множество,**
- **всегда доступен порядковый номер конкретного значения в данном типе (элемента),**
- **всегда доступны значения предыдущего и последующего элементов.**

## Операции и функции для типа char

- **операции отношения: <, <=, >, >=, =, <>, - сравниваются коды символов**
- **функция ord('S') - возвращает порядковый № символа S**
- **chr(№) - возвращает символ**
- **pred('S') - предыдущий символ**
- **succ('S') - последующий символ**
- **upcase('S') - перевод в верхний регистр**

- **операции – доступны все операции C, с учетом преобразования типов данных**
- **функция getchar() - читает символ из буфера ввода клавиатуры (б-ка stdio.h)**
- **putchar() - отображает символ на экран (stdio.h)**
- **функции библиотеки ctype.h (isalnum, isalpha, isblank, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit, tolower, toupper)**

**Функции работы с символами изучить самостоятельно!**

# Элементы ЯПВУ. СИМВОЛЫ

Pascal

C / C++

## Символьные константы (литералы)

**Три формы записи символьных констант:**

- Символ в апострофах ('a', 'W', '4', '?');
- Знак # перед десятичным кодом символа ASCII (#97, #186, #254) – обычно для символов, отсутствующих на клавиатуре;
- Буква, предваряемая знаком ^ (^M, ^V) – для представления управляющих символов (первые 32 символа ASCII) – код буквы на 64 больше кода управляющего символа (соответственно, 13, 22).

**Форма записи символьных констант:**

Один или несколько символов в апострофах ('a', 'W', '4', '?', 'vb').

- Двухсимвольные константы занимают 2а байта и имеют тип данных int;
- Для многобайтовых символов (для работы с набором символов требующих больше одного байта, например, Unicode) существует тип wchar_t - расширенный символьный. Эти константы записываются с префиксом L (wchart_t=wr; wr=L'A';);
- Специальные символьные константы (ESC-последовательности) служат для представления специальных символов. Их отличительный признак символ обратной косой черты - \ (\n – новая строка, \" – кавычка, \0ddd – восьмеричная константа, \x0ddd – шестнадцатеричная).
- Пустая символьная константа – недопустима.

# Элементы ЯПВУ.
# Таблица кодировки символов
## Символы с кодами 0 - 127

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 - | 16 - ▶ | 32 - | 48 - 0 | 64 - @ | 80 - P | 96 - ' | 112 - p |
| 1 - ☺ | 17 - ◀ | 33 - ! | 49 - 1 | 65 - A | 81 - Q | 97 - a | 113 - q |
| 2 - ☻ | 18 - ↕ | 34 - " | 50 - 2 | 66 - B | 82 - R | 98 - b | 114 - r |
| 3 - ♥ | 19 - ‼ | 35 - # | 51- 3 | 67 - C | 83 - S | 99 - c | 115 - s |
| 4 - ♦ | 20 - ¶ | 36 - $ | 52 - 4 | 68 - D | 84 - T | 100 - d | 116 - t |
| 5 - ♣ | 21 - § | 37 - % | 53 - 5 | 69 - E | 85 - U | 101 - e | 117 - u |
| 6 - ♠ | 22 - ▬ | 38 -& | 54 - 6 | 70 - F | 86 - V | 102 - f | 118 - v |
| 7 - | 23 - ↨ | 39 - ' | 55 - 7 | 71 - G | 87 - W | 103 - g | 119 - w |
| 8 - | 24 - ↑ | 40 - ( | 56 - 8 | 72 - H | 88 - X | 104 - h | 120 - x |
| 9 - | 25 - ↓ | 41 - ) | 57 - 9 | 73 - I | 89 - Y | 105 - i | 121 - y |
| 10 - | 26 - → | 42 - * | 58 - : | 74 - J | 90 - Z | 106 - j | 122 - z |
| 11 - | 27 - ← | 43 - + | 59 - ; | 75 - K | 91 - [ | 107 - k | 123 - { |
| 12 - | 28 - ∟ | 44 - , | 60 - < | 76 - L | 92 - \ | 108 - l | 124 - \| |
| 13 - | 29 - ↔ | 45 - - | 61 - = | 77 - M | 93 - j | 109 - m | 125 - } |
| 14 - ♫ | 30 - ▲ | 46 - . | 62 -> | 78 - N | 94 - ^ | 110 - n | 126 - ~ |
| 15 - ☼ | 31 - ▼ | 47 - / | 63 - ? | 79 - O | 95 - ÷ | 111 - o | 127 - ⌂ |
| 16 - ▶ | 32 - | 48 -0 | 64 - @ | 80 - P | 96 - | 112 - p | |

# Элементы ЯПВУ.
# Таблица кодировки символов
# Символы с кодами 128 - 255

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 128 -А | 144 - Р | 160 - а | 176 - ▒ | 192 - └ | 208 - ╨ | 224 -р | 240 - Ё |
| 129 - Б | 145 - С | 161 - б | 177 - ▓ | 193 - ┴ | 209 - ╤ | 225 -с | 241- ё |
| 130 -В | 146 - Т | 162 - в | 178 - | 194 - ┬ | 210 - ╥ | 226 -т | 242 - Є |
| 131 - Г | 147 - У | 163 - г | 179 - │ | 195 - ├ | 211 - ╙ | 227 -у | 243 - є |
| 132 -Д | 148 - Ф | 164 - д | 180 - ┤ | 196 - ─ | 212 - ╘ | 228 -ф | 244 - Ї |
| 133 -Е | 149 - Х | 165 - е | 181 - ╡ | 197 - ┼ | 213 - ╒ | 229 - х | 245 - ї |
| 134 - Ж | 150 - Ц | 166 - ж | 182 - ╢ | 198 - ╞ | 214 - ╓ | 230 - ц | 246 - Ў |
| 135 -З | 151 - Ч | 167 - з | 183 - ╖ | 199 - ╟ | 215 - ╫ | 231 - ч | 247 - ў |
| 136 -И | 152 -Ш | 168 - и | 184 - ╕ | 200 - | 216 - ╪ | 232 - ш | 248 - ° |
| 137 - Й | 153 -Щ | 169 - й | 185 - ╣ | 201 - ╔ | 217 - ┘ | 233 - щ | 249 - ● |
| 138- К | 154 - Ъ | 170 - к | 186 - ║ | 202 - ╩ | 218 - ┌ | 234 - ъ | 250 - ∙ |
| 139 -Л | 155 - Ы | 171 - л | 187 - ╗ | 203 - ╦ | 219 - ■ | 235 - ы | 251 - √ |
| 140 -М | 156 - Ь | 172 - м | 188 - ╝ | 204 - ╠ | 220 - ▄ | 236 - ь | 252- № |
| 141 -Н | 157 - Э | 173 - н | 189 - ╜ | 205 - ═ | 221 - ▌ | 237 - э | 253 - ¤ |
| 142 -О | 158 - Ю | 174 - о | 190 - ╛ | 206 - ╬ | 222 - ▐ | 238 - ю | 254 - ▪ |
| 143- П | 159 - Я | 175 - п | 191 - ┐ | 207 - ╧ | 223 - ▀ | 239 - я | 255 - |
| 144 - Р | 160 - а | 176 - ▒ | 192 - └ | 208 - ╨ | 224 - р | 240 - Ё | |

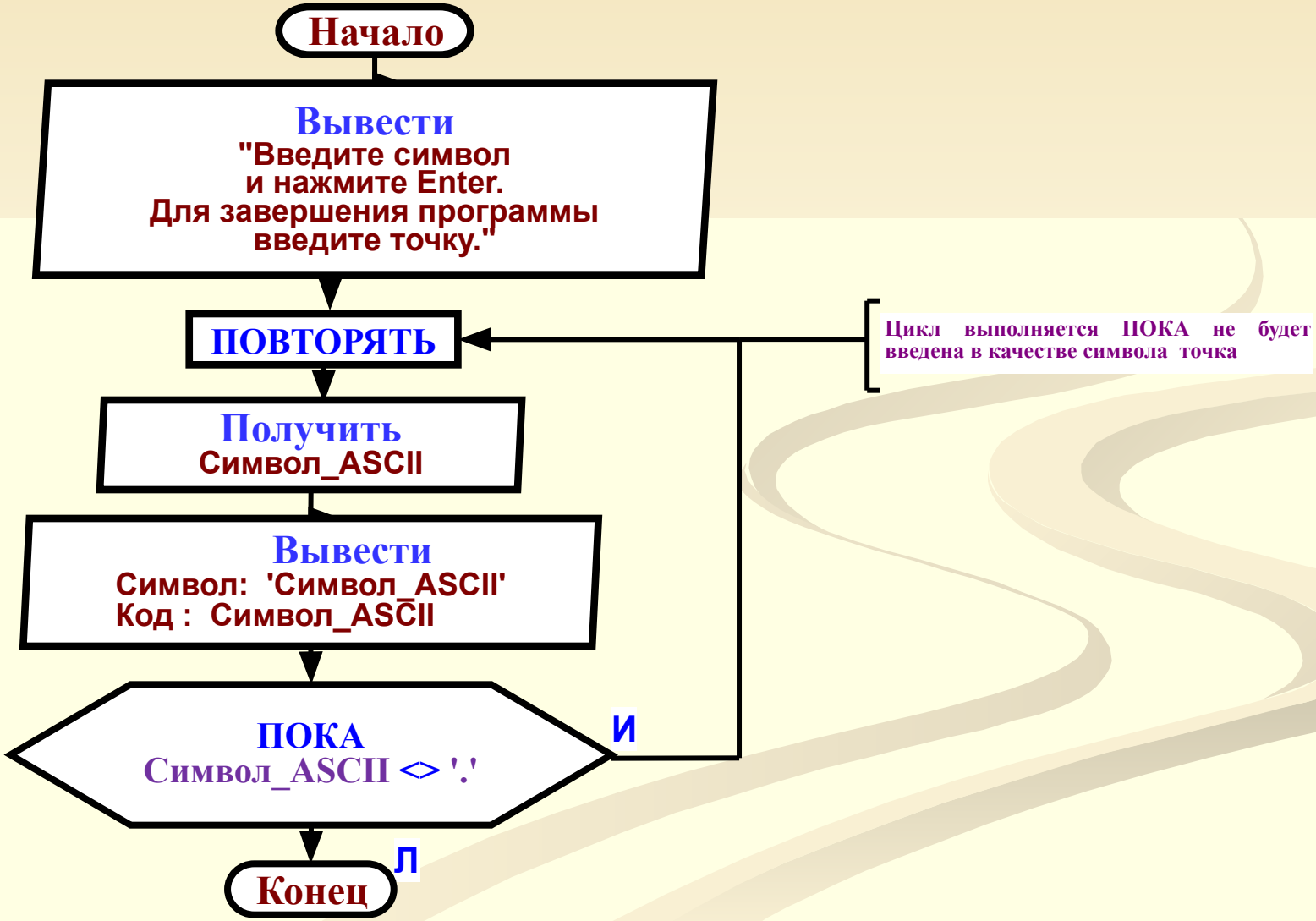
Pascal

# Символы

## Практическое занятие

C / C++

**Задание:** **Выводить на экран код введенного символа, для завершения ввода ввести точку.**



# Символы

**Pascal** | **Практическое занятие** | **C / C++**

**Задание:** **Вывести на экран код введенного символа.** Требуемый вид экрана:

Введите символ.
Для завершения программы введите точку.
-> 1
Символ: 1 Код: 49 *и т.д.*

```
(* Выводит код введенного символа *)
Program sym_cod;
Var    sym: char;
       code: integer;
begin
   writeln ('Введите символ и нажмите
                  Enter.');
  writeln ('Для завершения программы
введите точку и нажмите Enter.');
 repeat
  begin
    write('-> ');
     readln(sym);
     code := Ord(sym);
   writeln('Символ: ', sym,', Код: ', code);
  end
   until  sym = '.';
end.
```

```
// Вывод кода введенного символа
#include <stdio.h>
#include <conio.h>
void main()
  {
   char ch;
   printf("\nВведите символ и нажмите
          Enter\n");
   printf ("Для завершения программы
      введите точку.\n");
  do
   {
    ch = getch();
    printf("Символ: %c Код: %i\n",ch,ch);
   }
  while (ch != '.');
      printf("\nДля  завершения  нажмите
Enter\n");
   getch();
  }
```

# Элементы ЯПВУ. СТРОКИ

**Pascal** | **C / C++**

**Строка – это конечная последовательность символов, цепочка символов.**

В Pascal существует стандартный тип данных **String - строка.**

```
var <имя_строки>:
    string[[<длина>]]
[='<строка_инициализации>'];
```

Длина строки по умолчанию - 255 символов плюс нулевой символ, хранящий значение длины строки (всего 256 байт)

Можно определить собственный тип строки:

```
type <имя_типа>=string[[дл]];
```

Длина строки – константа или константное выражение.

Строка в С – это одномерный символьный массив с **нуль-символом - \0** в конце.

При объявлении размера такого массива надо к количеству символов строки добавлять одну позицию (байт) для нуль-символа.

Стандартного строкового типа и строковых переменных в С - нет.

Функции работы со массивами-строками собраны в заголовочном файле **string.h**.

В С++ для работы со строками определен класс **string** (изучите в ООП).

Главный недостаток – необходимо вручную проверять выход за границу строкового массива!

# Элементы ЯПВУ. СТРОКИ

**Pascal**

**C / C++**

## Примеры:

```
Type srt4 = string [4];
Const n = 5;
Var
  s : string;  (*строка 255 символов*)
  s1 : str4;    (*строка типа str4*)
  s2 : string [n]; (*прямое описание*)
```

**Инициализация строк – в разделе const:**

```
Const  s3 : string[6] = 'good';
```

*Вид строки:*

| 4 | g | o | o | d | | |
|---|---|---|---|---|---|---|

*- 7 байт.*

```
char str[6] = "good";
```

**выделено 6 элементов массива (0÷5)**

| g | o | o | d | \0 | |
|---|---|---|---|---|---|

**Оператор char str[ ] = "good"; - создаст и заполнит массив размерностью 5.**

| g | o | o | d | \0 |
|---|---|---|---|---|

## Строковые константы (литералы)

**Строковая константа – это последовательность любых ASCII символов, заключенных в**

**Апострофы – 'abc'**

**Кавычки – "abc"**

```
Const  Text = 'Простая программа'
WriteLn(Text);
```

Максимальная длина – 126 символов.

Апостроф в константе – дублируется:

'Don''t right'

"Текстовая константа"

Кавычка внутри константы - \"

"ООО \"Рога и копыта\""

Знак переноса длинной константы - \

# Элементы ЯПВУ. СТРОКИ



**Pascal**

**C / C++**

## Операции для строк

- **присваивание строк** := - при разной длине лишние символы отбрасываются

```
type str4 = string[4];
var s1 = str4;   s2 = string[10]
s2 := 'qwertyuiop';
s1 := s2; (* строка s1 содержит qwer *)
```

- **конкатенация** + - склеивание (сцепление) строк

```
st1 := 'пар';  st2:= 'воз';  st := st1+'о'+st2;
```

- **обращение к компонентам строки**

_<имя_строки>[<индекс>]

{s='15.47'} c:= s[3]; {c='.'} cr:=s[2]; {cr='5'}

но в отличие от массива, нельзя напрямую заменять символы в строке.

- **операции отношения** <, <=, >, >=, =, < > - сравнивают строки, из двух строк меньшая та, чей первый различный символ меньше.

'abc'<'xyz', 'a'<'abc', '120'<'45', 'Anny'<'anny'

- **ввод-вывод строк** – имя строки может исп. в процедурах readln(s1, s2); и writeln(s1);

*При вводе* в строку считывается количество символов равное длине строки или меньше, если раньше встретиться команда Enter (LF+CR – перевод строки+возврат каретки).

- поскольку строка в С – это массив, то над ними возможны все те же операции, что и над массивами, например, операция присваивания одной строки другой выполняется с помощью цикла или функций стандартной библиотеки.

# Элементы ЯПВУ. СТРОКИ



**Pascal**

## Функции и процедуры для строк

**C / C++**

**Функции** (возвращают простое значение)

- **Concat (s1[,s2,…,sn])** – последовательная конкатенация **s1, s2…sn**,
- **Copy (s, start, len)** – возвращает подстроку длиной **len** строки **s, начиная** позиции **start** (**len** и **start** – целого типа),
- **Length (s)** – текущая длина строки **s** (тип byte),
- **Pos (substr, s)** – ищет подстроку **substr** в строке **s** и возвращает номер первого символа **substr** в **s** или 0, если такой подстроки в **s** нет.

**Процедуры** (возвращают преобразованную строку)

- **Delete (s, start, len)** – удаляет подстроку длиной **len** из строки **s**, начиная с позиции **start**,
- **Insert (substr, s, start)** – вставляет подстроку **subst** в строку **s**, начиная с позиции **start**,
- **Str (x, s)** – преобразует числовое значение **x** в строку **s** (**x** может иметь формат – **x:6:2**),
- **Val (s, x, errcode)** – преобразует строку **s** в числовое значение **x**, **errcode** содержит номер позиции первого ошибочного символа или 0.

**Некоторые функции стандартных библиотек для работы со строками:**

❖ **библиотека stdio.h**

- **gets(s)** - читает символ с клавиатуры в строку **s** до Enter, возвращает указатель на **s**.
- **puts(s)** – выводит строку **s** на экран.

❖ **библиотека string.h**

- **strcpy(s1,s2)** – копирует **s2** в **s1**,
- **strcat(s1,s2)** – конкатенация **s2** в конец **s1**,
- **strlen(s1)** – возвращает длину строки **s1**,
- **strcmp(s1,s2)** – сравнение строк, возвращает 0 (false), если **s1** и **s2** совпадают, отрицательное значение, если **s1<s2** и положительное, если **s1>s2**,
- **strchr(s1,ch)** – ищет символ в строке, возвращает указатель на первое вхождение символа **ch** в строку **s1**,
- **strstr(s1,s2)** – ищет подстроку в строке, возвращает указатель на первое вхождение строки **s2** в строку **s1**,
- **и другие.**

**Формат функций и процедур работы со строками изучить самостоятельно!**

# Элементы ЯПВУ. СТРОКИ

Pascal

~~C / C++~~

## ASCIIZ-строки

**ASCIIZ-строки – это строки символов "с нулевым окончанием"- признак окончания строки - символ с нулевым кодом #0. Такие строки введены в Pascal для обеспечения возможности использовать длинные строки (до 65 535 символов) и для работы под Windows.**

**Для реализации механизма ASCIIZ-строк введен новый предопределенный тип данных PChar: Type PChar = ^char, т.е. указатель на символьное значение. Переменным типа PChar можно присваивать строковые значения, а также индексировать их, как при работе с символьными массивами. Переменная типа PChar ссылается на первый элемент некоторой ASCIIZ-строки неопределенной длины, завершающейся #0.**

**Для поддержки работы с ASCIIZ-строками используется специальный модуль (Uses) String. Он доступен в BP и BPW и содержит процедуры и функции для:**

- **динамического создания и уничтожения строк (StrNew и StrDispose),**
- **копирования строк (StrCopy, StrECopy, StrLCopy, StrPCopy, StrMove),**
- **конкатенации строг (StrCat, StrLCat),**
- **сравнения строк (StrComp),**
- **и другие (StrIComp, StrLComp, StrLIComp, StrPos, StrIPos, StrScan, StrRScan, StrEnd, StrLen, StrLower, StrUpper, StrPas).**

# Строки

Pascal

C / C++

## Практическое занятие:

**Объяснить работу программ и что они выводят на экран.**

```
Program Simple;
const
  Text = Простая программа ';
var s : string[4];
begin
  WriteLn(Text);
  readln  (s);
  writeln (s);
  writeln('s[0]=', s[0]);
  writeln('s[0]=', length(s));
   end.
```

```
#include <stdio.h>
#include <conio.h>
 void main()
{
   unsigned char st[80];
   int i;
 printf("\n Введите строку текста и нажмите Enter\n");
     printf("-> ");
     gets(st);
     i = 0;
     while ( st[i] )
     {
         if (((st[i] >= 'a') && (st[i] <= 'z')) || ((st[i] >= 'a')
                                  && (st[i] <= 'п')))
               st[i] -= 32;
      else
        if (st[i] >= 'p' && st[i] <= 'я')
           st[i] -= 80;
        i++;
     }
    puts(st);
   printf("\n Для завершения нажмите Enter");
   getch();
}
```