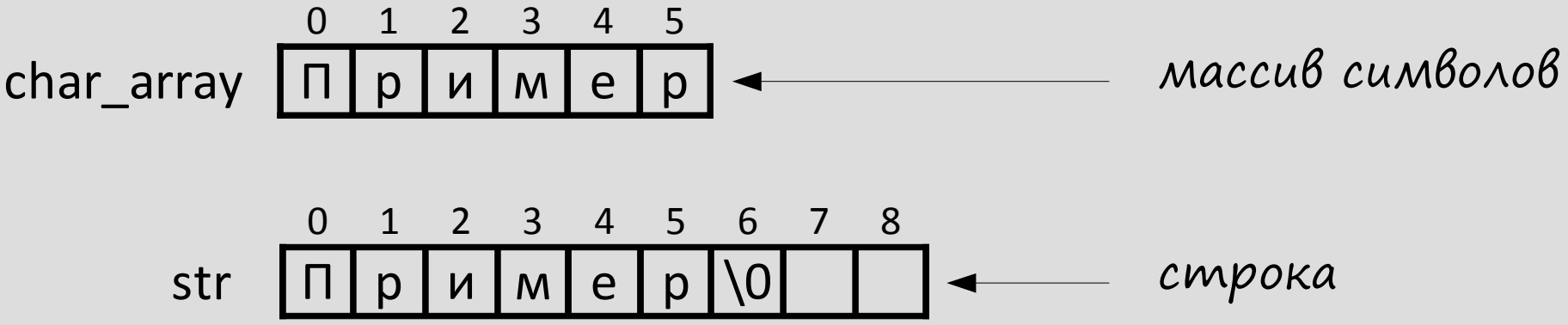
Лекция №6. Строки

- Представление строк в языке Си
- Представление многострочного текста
- Ввод-вывод строк
- Обработка строк

Представление строк в языке Си

- В языке Си **строка** – это **массив символом**, который заканчивается **нуль-символом** (символ с кодом 0).
- **Нуль-символ** сигнализирует о фактическом конце строки.



Объявление строк

char <имя массива>**[**<размер массива>**];**

<размер массива> – целочисленное выражение

<размер массива> = **<кол-во символов> + 1**

Размер памяти, выделяемой под строку:

Кол-во байт = <размер массива>

Задание

Объявите строку для хранения своей фамилии,

имени и отчества

Решение задания

- Определим длину ФИО: пусть ФИО составляет 34 буквы вместе с 2-мя пробелами

- Объявим строку длиной 35 символов (34 символа для ФИО + 1 символ для нуль-символа)

```
char FIO[35];
```

Инициализация строк

- Инициализация символьными константами

char <имя массива>**[**[<размер массива>]**] =**
{<символ. константа 1> [**,** <символ. константа 2>]**, '\0'};**

- Инициализация строковой константой

char <имя массива>**[**[<размер массива>]**] =** <строковая константа>**;**

Задание

Объявите строку для хранения 30 символов, различными способами запишите в нее «пустую» строку (которая не содержит символов)

Решение задания

- Так как строка должна хранить 30 значимых символов, то длина строки составляет 31 символ (дополнительный символ для нуль-символа).
- «Пустая» строка не содержит значащих символов, но содержит нуль-символ.

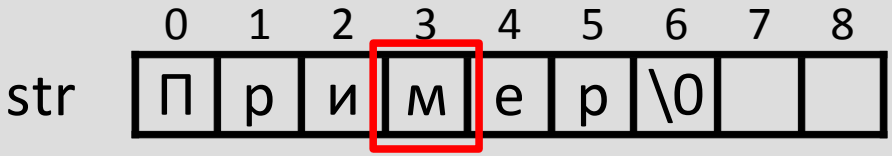
```
1) char EmptyStr[31] = {'\0'};   // 1-й вариант

2) char EmptyStr[31] = "";       // 2-й вариант
```

Обращение к символу строки

- Обращение к символу строки – это обращение к элементу массива:

<имя массива>[<индекс символа>]

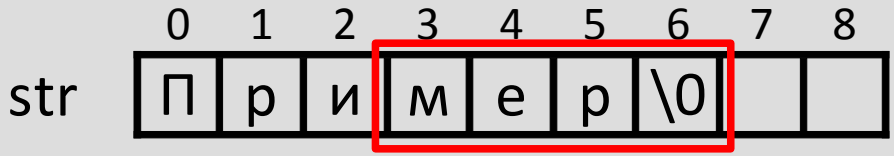


Обращение к подстроке

- Обращение к подстроке – это обращение ко 2-й части строки:

<имя массива> + <индекс символа>

<индекс символа> – индекс символа, с которого начинается подстрока



Задание

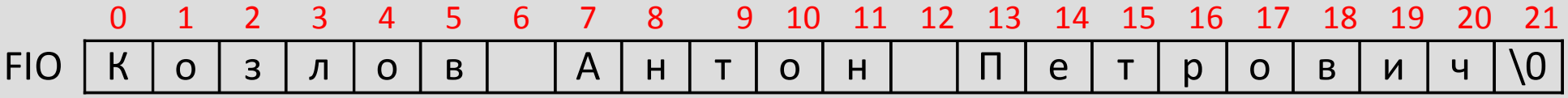
Дана строка FIO, содержащая Ваше ФИО, напечатайте первые буквы фамилии, имени и отчества

Используя конструкцию **printf("%s", <строка>)**, напечатайте свое имя и отчество, используя строку FIO

Решение задания

- Пусть объявлена строка, содержащая ФИО

```
char FIO[] = "Козлов Антон Петрович";
```



- Первые буквы фамилии, имени и отчества располагаются в позициях с индексами 0, 7 и 13

```
printf("%c%c%c", FIO[0], FIO[7], FIO[13]);
```

- Имя и отчество располагаются, начиная с индекса 7

```
printf("%s", FIO+7);
```

Представление многострочного текста в языке Си

- Многострочный текст представляется двумерным массивом типа char, каждая строка которого соответствует строке текста

> Написание хороших программ требует ума, вкуса и терпения.
>
> Б. Страуструп

Text

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	…	30
0	Н	а	п	и	с	а	н	и	е		х	о	р	о	ш	и	х	\0					…	
1	п	р	о	г	р	а	м	м		т	р	е	б	у	е	т		у	м	а	,	\0	…	
2	в	к	у	с	а		и		т	е	р	п	е	н	и	я	.	\0					…	
3	\0																						…	
4				Б	.		С	т	р	а	у	с	т	р	у	п	\0						…	

Объявление и инициализация многострочного текста

- Инициализация текста строковыми константами

char <имя массива>**[**[<кол-во строк>]**][**<кол-во симв. в строке+1>**] =**

{<строковая константа 1> [**,** <строковая константа 2>]**};**

Задание

Создайте массив для хранения списка группы: ФИО каждого студента должно хранится в отдельной строке. Всего в группе 25 студентов, самое длинное ФИО составляет 28 букв.

Запишите в список группы ФИО 3-х студентов.

Решение задания

- Объявим массив для хранения списка студентов и запишем ФИО 3-х студентов

```
char students[25][29]= {"Козлов Антон Петрович",
                        "Абрамов Иван Иванович",
                        "Жук Петр Петрович"};
```

Обращение к символам многострочного текста

- Обращение к символу текста:

  <имя массива>[<индекс строки>][<индекс симв. в строке]

- Обращение к строке текста:

  <имя массива>[<индекс строки>]

Задание

Замените в ФИО 2-го студента пробелы на символы подчеркивания '_'

Напечатайте ФИО 3-го студента из списка

Решение задания

students

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	К	о	з	л	о	в		А	н	т	о	н		П	е	т	р	о	в	и	ч	\0
1	А	б	р	м	о	в		И	в	а	н		И	в	а	н	о	в	и	ч	\0	
2	Ж	у	к		П	е	т	р		П	е	т	р	о	в	и	ч	\0				

- Пробелы в ФИО 2-го студента располагаются в позициях с индексами 6 и 11

```
FIO[1][6]= '_';
FIO[1][11]= '_';
```

- ФИО 3-го студента — это строка текста с индексом 2

```
printf("%s", FIO[2]);
```

Ввод строк (библиотека stdio.h)

```
char *gets( char *str );
```

Прочитать строку с клавиатуры

- В конец строки добавляется нуль-символ.
- Возвращается NULL, если возникла ошибка ввода.

```
// Вводим название города
// (название не более 20 символов)
char city[21];
gets( city );
```

Вывод строк (библиотека stdio.h)

```
int puts( const char *str );
```

Напечатать строку на экране

- В конце строки записывается символ новой строки.
- Возвращается значение <0, если возникает ошибка вывода.

```
// Выводим название города
char city[21]= "Volgograd";
puts( "Our city is" );
puts( city );
```

```
Our city is
Volgograd
```

Вывод строк (библиотека stdio.h)

printf() со спецификатором вывода **%s**

Форматированная печать строки на экране

```
// Выводим название города
char city[21]= "Volgograd";
printf("Our city is %s" , city );
```

Our city is Volgograd

Задание

Дан двумерный символьный массив `char txt[3][21]`, запросите у пользователя его ФИО и запишите в первую строку текста фамилию, во вторую — имя, в третью — отчество. Напечатайте на экране фамилию и инициалы пользователя, например,

Иванов И. К.

Решение задания

```
char txt[3][21]; // текст для хранения ФИО

// Вводим ФИО
puts("Input your family name");
gets(txt[0]);

puts("Input your name");
gets(txt[1]);

puts("Input your patronymic");
gets(txt[3]);

// Выводим фамилию и инициалы
printf("%s %c. %c.", txt[0], txt[1][0], txt[2][0]);
```

Обработка строк

В языке Си нет встроенных операций для работы со строками, но имеются библиотечные функции для обработки строк

Обработка строк (библиотека string.h)

strcmp()	Сравнивает две строки (аналог операции ==)
strncmp()	Сравнивает части двух строк
strcat()	Склеивает две строки (аналог операции +)
strncat()	Добавляет к строке заданный символ
strchr()	Находит первое вхождение в строку заданного символа
strrchr()	Находит последнее вхождение заданного символа
strstr()	Находит первое вхождение заданной строки
strcpy()	Копирует одну строку в другую (аналог операции =)
strncpy()	Копирует часть одной строки в другую
strlen()	Определяет длину строки
strlwr()	Переводит всю строку в нижний регистр
strupr()	Переводит всю строку в верхний регистр
strnset()	Создает строку из N заданных символов
strtok()	Разбиение строки на подстроки

Обработка строк

(библиотеки string.h, tchar.h, stdio.h, ctype.h)

`strspn()`	Возвращает индекс первого символа заданной строки, который не принадлежит указанной последовательности символов
`_strspnp()`	Возвращает подстроку (заданной строки), которая начинается с символа отсутствующего в другой строке
`sprintf()`	Форматированное преобразование данных в строку
`sscanf()`	Форматированное преобразование строки в данные
`_strrev()`	Переворачивает строку
`_memccpy()`	Копирует символы из одной строки в другую до заданного символа, но не более заданной длины
`memcpy()`	Копирует из одной строки в другую заданное кол-во символов
`isalpha()`	Проверяет, является ли символ буквой
`isalnum()`	Проверяет, является ли символ буквой или цифрой
`iscntrl()`	Проверяет, является ли символ управляющим
`islower()`	Проверяет, находится ли символ в нижнем регистре
`isupper()`	Проверяет, находится ли символ в верхнем регистре
`isspace()`	Проверяет, является ли символ разделителем
`ispunct()`	Проверяет, является ли символ символом пунктуации

Сравнение строк

```
int strcmp(const char *str1, const char *str2);
```

Сравнивает две строки str1 и str2, возвращает

< 0 – str1 < str2,

0 – str1 = str2,

> 0 – str1 > str2.

```
// Сравниваем ФИО
char FIO_1[31]= "Ivanov Ivan";
char FIO_2[31]= "Ivanov Roman";
char relation;      // соотношение ФИО

if      (strcmp(FIO_1, FIO_2) < 0)  relation = '<';
else if(strcmp(FIO_1, FIO_2) = = 0)   relation = '=';
else if(strcmp(FIO_1, FIO_2) > 0)  relation = '>';

printf("%s %c %s", FIO_1, relation, FIO_2);
```

Склеивание строк

```
char *strcat(char *strDestination,
             const char *strSource);
```

Склеить две строки strDestination *и* strSource, *результат поместить в* strDestination

```
// Склеиваем ФИО
char FIO[31]= "";
char LastName[]= "Ivanov", Name[]= "Ivan",
     Patronymic[]= "Ivanovich";

strcat(FIO, LastName);
strcat(FIO, " ");
strcat(FIO, Name);
strcat(FIO, " ");
strcat(FIO, Patronymic);
puts(FIO);
```

Ivanov Ivan Ivanovich

Копирование строк

```
char *strcpy(char *strDestination,
             const char *strSource);
```

Скопировать содержимое строки strSource *в строку* strDestination

```
// Создаем копию строки
char str[]= "Example";
char copy[31]= "";

strcpy(copy, str);
puts(copy);
```

```
Example
```

Поиск подстроки

```
char *strstr(const char *str,
             const char *strSearch);
```

Найти в строке str **первое вхождение подстроки** strSearch

- Результатом является 2-я часть строки str, начинающаяся с подстроки strSearch.

```
// Поиск имени и отчества в ФИО
char FIO[31]= "Ivanov Ivan Ivanovich";

puts( strstr(FIO, " ") + 1 );
```

```
Ivan Ivanovich
```

Разбиение строки

```
char *strtok(char *strToken,
             const char *strDelimit);
```

Разбивает строку strToken **на подстроки, ориентируясь на разделители** strDelimit

- Результатом однократного вызова функции является замена первого разделителя в строке на нуль-символ.
- Дополнительно функция возвращает начало следующей подстроки (если подстрок уже нет, то возвращается NULL).

Разбиение строки

- При повторных вызовах функции **strtok()** в качестве первого параметра указывается NULL. В этом случае следующий разделитель заменяется нуль-символом и возвращается начало следующей подстроки.

- ВНИМАНИЕ: Результирующие строки находятся рядом друг с другом и имеют разную длину. Соблюдайте осторожность при их модификации: в случае превышения размера будет испорчена соседняя строка.

Разбиение строки

```
char string[] = "A string\tof,,,tokens"; // исходная строка
char seps[]   = " ,\t";    // строка разделителей
char *token;               // ссылка на начало подстроки

token = strtok(string, seps);  // выделение первой подстроки
while(token != NULL)
{
  puts(token);                 // печать подстроки
  token = strtok(NULL, seps); // выделение следующей подстроки
}
```

исходная строка	A		s	t	r	i	n	g	\t	o	f	,	,	,	t	o	k	e	n	s	\0
строка после разбиения	A	\0	s	t	r	i	n	g	\0	o	f	\0	,	,	t	o	k	e	n	s	\0

```
A
string
of
tokens
```

Преобразование строк в числа и наоборот (библиотека stdlib.h)

`atof()`	Преобразует строку в `double` (`alphabetical to float`)
`atoi()`	Преобразует строку в `int` (`alphabetical to integer`)
`atol()`	Преобразует строку в `long int` (`alphabetical to long`)
`_itoa()`	Преобразует `int` в строку (`integer to alphabetical`)
`_ltoa()`	Преобразует `long int` в строку (`long to alphabetical`)
`_gcvt()`	Преобразует `double` в строку

Преобразование строк в числа и наоборот (библиотека stdlib.h)

```
char str[]= "1.2";  // вещественное число
                    // строковой константой
char buffer[21];

// Преобразование строки в число
double d= atof(str);  // d = 1.2
int    i= atoi(str);  // i = 1

// Преобразование числа в строку
_itoa(i, buffer, 10); // 10 - десятичная
puts( buffer );       // система счисления

_gcvt(d, 5, buffer);  // 5 - кол-во значащих
puts( buffer );       // разрядов
```

```
1
1.2
```