Строки

Т.И. Комаров

НИЯУ МИФИ

2023

Структуры данных

Уровни представления информации:

- Интуитивный
- Логический (абстрактный)
- Конкретный (физический)

Текстовая информация — это частный случай информации, она может быть представлена на любом из уровней, например:

- Интуитивный — текст
- Логический — строка
- Конкретный — вектор

Представление строк в языке C

Важно

Отдельного встроенного типа данных в языке C для обработки текстовых данных нет
Для обработки строк в языке C предлагается использовать массивы символов

В векторах (т. е. массивах в языке C) хранятся лишь данные, подлежащие обработке, в них нет информации о длине (а она необходима).
Создателями языка C было принято следующее решение:

Важно

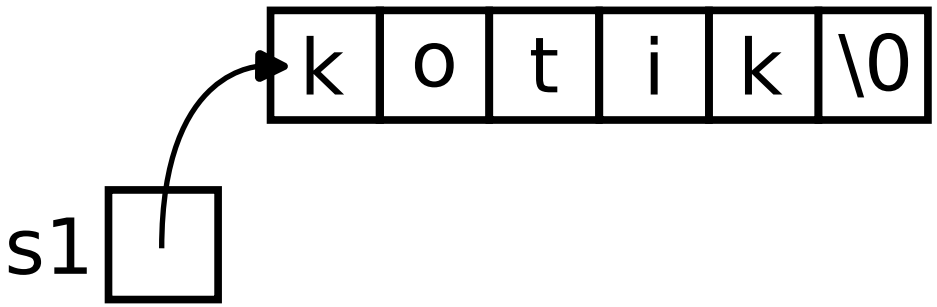
Строки в языке C являются нуль-терминированными — они завершаются символом `'\0'`

Таким образом, строки в языке C в памяти занимают на один символ больше, чем это необходимо для размещения данных

Объявление строк

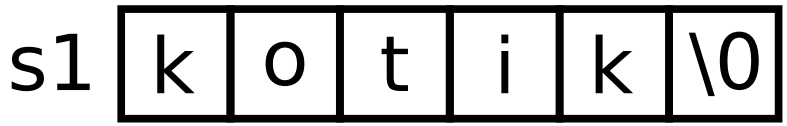
Указатель:

```
char *s1 = "kotik";
```



Массив:

```
char s1[] = "kotik";
```



Функции стандартной библиотеки I

Важно

Практически все функции стандартной библиотеки для работы со строками объявлены в заголовочном файле `<string.h>`

Определены две группы функций:

- функции с именами `str*` — предназначены непосредственно для обработки строк
- функции с именами `mem*` — предназначены, по сути, для манипулирования объектами, хранящимися в памяти, как байтовыми массивами

Функции стандартной библиотеки II

Важно

Исчерпывающий перечень функций, объявленных в заголовочном файле `<string.h>`, доступен с помощью встроенной справки:

```
$ man string.h
```

Информацию о работе каждой конкретной функции можно получить аналогичным образом, например:

```
$ man strlen
```

Важно

Функции копирования, за исключением `memmove()`, не должны обрабатывать объекты, хранящиеся в перекрываемых областях памяти, результат подобного использования данных функций не определён стандартом (undefined behavior)

Основные функции группы `str*` I

Параметры

- `s` и `t` — строки, имеют тип `char *`
- `cs` и `ct` — неизменяемые строки, имеют тип `const char *`
- `n` — беззнаковое целое, имеет тип `size_t`
- `c` — целое, имеет тип `int` (приводится к типу `unsigned char`)

- `char *strcpy(s, ct)` — копирует строку `ct` в строку `s`, включая нуль-байт; возвращает `s`
- `char *strncpy(s, ct, n)` — копирует не более `n` символов строки `ct` в `s`; возвращает `s`; дополняет результат нуль-байтами, если в `ct` меньше `n` символов
- `char *strcat(s, ct)` — присоединяет `ct` в конец строки `s`; возвращает `s`

Основные функции группы `str*` II

- `char *strncat(s, ct, n)` — присоединяет не более `n` символов строки `ct` к `s`, завершая `s` нуль-байтом; возвращает `s`
- `int strcmp(cs, ct)` — сравнивает строки `cs` и `ct`; возвращает значение меньше нуля (если `cs < ct`), ноль (если `cs == ct`) или значение больше нуля (если `cs > ct`)
- `int strncmp(cs, ct, n)` — сравнивает не более `n` символов строк `cs` и `ct`; возвращает значение меньше нуля (если `cs < ct`), ноль (если `cs == ct`) или значение больше нуля (если `cs > ct`)
- `char *strchr(cs, c)` — возвращает указатель на первое вхождение `c` в `cs` или `NULL` при отсутствии такового
- `char *strrchr(cs, c)` — возвращает указатель на последнее вхождение `c` в `cs` или `NULL` при отсутствии такового

Основные функции группы `str*` III

- `size_t strspn(cs, ct)` — возвращает длину начального сегмента строки `cs`, состоящего исключительно из символов, которые входят в строку `ct`
- `size_t strcspn(cs, ct)` — возвращает длину начального сегмента строки `cs`, состоящего исключительно из символов, которые не входят в строку `ct`
- `char *strpbrk(cs, ct)` — возвращает указатель на первый символ в строке `cs`, который совпадает с одним из символов, входящих в строку `ct`, или `NULL` при отсутствии таковых
- `char *strstr(cs, ct)` — возвращает указатель на первое вхождение строки `ct` в строку `cs` или `NULL` при отсутствии таковых
- `size_t strlen(cs)` — возвращает длину строки `cs`

Основные функции группы `str*` IV

- `char *strtok(s, ct)` — ищет в строке `s` лексемы, разделенные символами из строки `ct`

Основные функции группы mem* I

Параметры

- `s` и `t` — области памяти, имеют тип `void *`
- `cs` и `ct` — неизменяемые области памяти, имеют тип `const void *`
- `n` — беззнаковое целое, имеет тип `size_t`
- `c` — целое, имеет тип `int` (приводится к типу `unsigned char`)

- `void *memcpy(s, ct, n)` — копирует `n` байт из области памяти `ct` в область памяти `s` и возвращает `s`
- `void *memmove(s, ct, n)` — аналогична функции `memcpy()`, но обеспечивает корректную работу с перекрывающимися областями памяти

Основные функции группы mem* II

- `int memcmp(cs, ct, n)` — сравнивает первые n байт из областей памяти `cs` и `ct`; возвращает значение меньше нуля (если `cs < ct`), ноль (если `cs == ct`) или значение больше нуля (если `cs > ct`)
- `void *memchr(cs, c, n)` — возвращает указатель на первое вхождение символа c в области памяти `cs` размером n байт или `NULL` в случае отсутствия такового
- `void *memset(s, c, n)` — инициализирует первые n байт области памяти `s` значением байта c

Примеры реализации функции `strcpy` I

Примеры реализации функции strcpy II

Примеры

```
char *strcpy(char *s, const char *t) {
    int i = 0;
    while ((s[i] = t[i]) != '\0')
        i++;
    return s;
}

char *strcpy(char *s, const char *t) {
    char *s1 = s;
    while ((*s = *t) != '\0') {
        s++;
        t++;
    }
    return s1;
}
```