Технология программирования на ЭВМ
Строки и указатели

Баев А.Ж.

Казахстанский филиал МГУ

14 ноября 2018

Объявление

Строка — массив из ascii-символов, который заканчивается нулевым символом '\0' (он же просто 0). Например, для строки `math` необходим массив длины 5.

```
1 char s[5];
2 s[0] = 'm';
3 s[1] = 'a';
4 s[2] = 't';
5 s[3] = 'h';
6 s[4] = '\0';
```

Инициализация

Все три варианта эквивалентны:

```
1 char s1[] = "math";
2 char s2[] = {'m', 'a', 't', 'h', '\0'};
3 char s3[] = {109, 97, 116, 104, 0};
```

Рекомендуется использовать первый.

Инициализация

```
1    char s[7] = "math";
```

'm'	'a'	't'	'h'	0	0	0

Ввод-вывод через форматные строки

Вывод через `printf` со спецификатором `%s`:

```
1     char str[5] = "math";
2     printf("%s", str);
```

Ввод через `scanf` со спецификатором `%s`:

```
1     char str[5];
2     scanf("%s", str);
```

Обратите внимание, что у `str` нет амперсанда!
Считывает до пробельного символа не включительно (пробел, табуляция или перенос строки).
Небезопасный ввод, который приводит к неопределенному поведению, если ввод больше размера массива.

Ввод-вывод через специализированные функции

Вывод:

```
1     char str[5] = "math";
2     puts(str);
```

Добавляет перенос строки.

Ввод:

```
1     char str[5];
2     gets(str);
```

Считывает до переноса строки (не включительно).
Небезопасный ввод, который приводит к неопределенному поведению, если ввод больше размера массива.

Ввод-вывод через специализированные функции

Вывод:

```
1     char str[5] = "math";
2     fputs(str, stdout);
```

Не добавляет перенос строки.

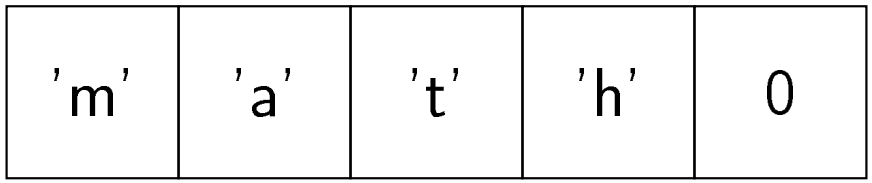
Ввод:

```
1     char str[5];
2     fgets(str, 5, stdin);
```

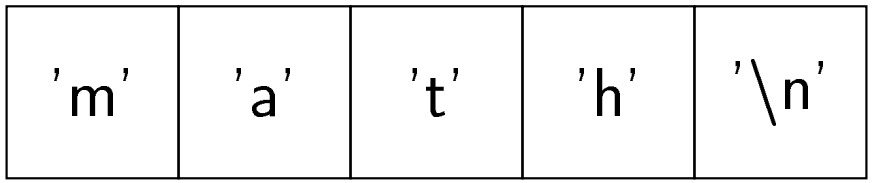
Считывает до переноса строки (включительно).
Безопасный ввод.

Небезопасный вариант

```
1 char s[5];
2 gets(s);
```

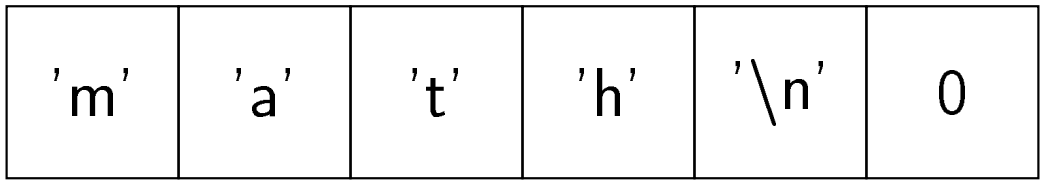



```
1 puts(s);
```

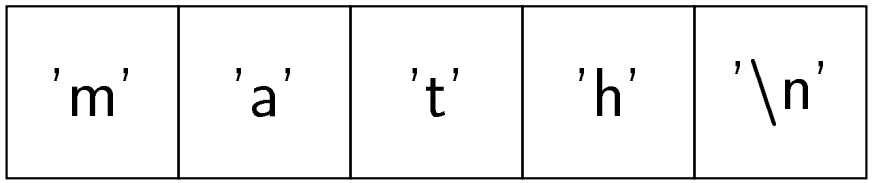


Безопасный вариант

```
1 char s[6];
2 fgets(s, 6, stdin);
```




```
1 fputs(s, stdout);
```



Заголовочный файл «ctype.h»

Функции для проверки значений (возвращает ненулевое значение, если проверка верна):

```
1    int isdigit(int c);
2    int isalpha(int c);
3    int islower(int c);
4    int isupper(int c);
5    int isalnum(int c);
6    int isgraph(int c);
7    int isprint(int c);
8    int ispunct(int c);
9    int isspace(int c);
```

Функции преобразования регистров букв:

```
1    int tolower(int c);
2    int toupper(int c);
```

Смена регистра

Дана строка, состоящая не более чем из 100 символов. Необходимо изменить все буквы на заглавные и вывести общее количество цифр.

Ввод:

```
Hello 2019!
```

Вывод:

```
HELLO 2019!
4
```

```c
#include <stdio.h>
#include <ctype.h>
int main() {
    int i = 0, digits = 0;
    char a[102];
    fgets(a, 102, stdin);
    for (i = 0; a[i] != 0; i++) {
        a[i] = toupper(a[i]);
        if (isdigit(a[i])) {
            digits++;
        }
    }
    fputs(a, stdout);
    printf("%d\n", digits);
    return 0;
}
```

Указатели

У каждой переменной есть адрес — номер ячейки оперативной памяти. Его значение вычисляется оператором взятия адреса

```
1       &var
```

Пример

```
1       int a = 5;
2       int *ptr;
3       ptr = &a;
4       printf("%p\n", ptr);
```

адрес	0x10002000	0x100020004
имя	a	ptr
значение	5	0x10002000

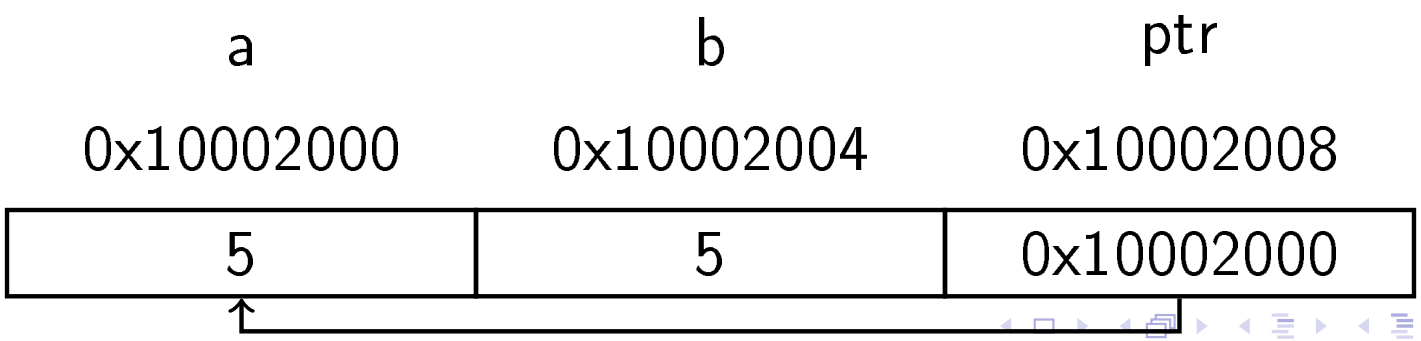
Указатели

У каждого адреса можно получить доступ к ячейке оперативной памяти. Доступ получается оператором разыменования адреса

```
1       *ptr
```

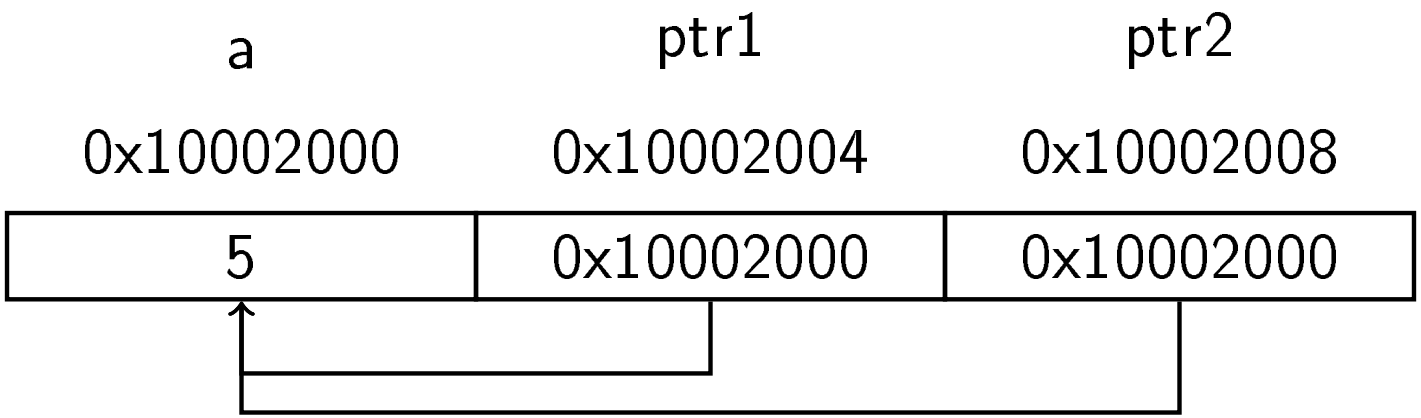
Пример

```
1       int a = 5, b;
2       int *ptr;
3       ptr = &a;   //ptr = 0x10002000
4       b = *ptr;   //b = 5
```



Указатели

```
1    int a = 5;
2    int *ptr1, *ptr2;
3    ptr1 = &a; //ptr1 = 0x10002000
4    ptr2 = &a; //ptr2 = 0x10002000
```




```
1  *ptr1 = 9;
2  printf("%d␣%d␣%d\n", a, *ptr1, *ptr2);
```

Арифметика указателей

Строка — указатель на начало массива.

`str` — адрес первой ячейки.

`str+4` — адрес пятой ячейки.

```
1 char str[] = "mechmath";
2 puts(str);
3 puts(str + 4);
```

Вывод:

```
1 mechmath
2 math
```

Заголовочный файл «string.h»

```
1  size_t strlen(char *s);
2  char *strcpy(char *dest,
3               char *src);
4  char *strncpy(char *dest,
5                char *src,
6                size_t n);
7  int strcmp(char *s1,
8             char *s2);
9  int strncmp(char *s1,
10             char *s2,
11             size_t n);
12 char *strstr(char *haystack,
13             char *needle);
```

Заголовочный файл «string.h»

Длина строки

```
1 int size;
2 char str[] = "mechmath";
3 size = strlen(str);
4 printf("%d\n", size);
```

Вывод:

```
1 8
```

Заголовочный файл «string.h»

Копирование строки

```
1 char src[9] = "mechmath";
2 char dst[20];
3 strcpy(dst, src);
4 puts(dst);
```

Вывод:

```
1 mechmath
```

Заголовочный файл «string.h»

Копирование строки

```
1 char src[9] = "mechmath";
2 char dst[20] = "supertext";
3 strcpy(dst + 5, src);
4 puts(dst);
```

Вывод:

```
1 supermechmath
```

Заголовочный файл «string.h»

Копирование строки

```
1 char src[9] = "mechmath";
2 char dst[20];
3 strncpy(dst, src, 4);
4 puts(dst);
```

Вывод:

```
1 mech
```

Заголовочный файл «string.h»

Копирование строки

```
1 char src[9] = "mechmath";
2 char dst[20];
3 strncpy(dst, src, 4);
4 strcpy(dst + 4, "-and-");
5 strcpy(dst + 9, src + 4);
6 puts(dst);
```

Вывод:

```
1 mech-and-math
```

Заголовочный файл «string.h»

Сравнение строк

```
1 int ans1, ans2, ans3;
2 ans1 = strcmp("math", "master");
3 ans2 = strcmp("math", "math");
4 ans3 = strcmp("math", "mathematics");
5 printf("%d␣%d␣%d", ans1, ans2, ans3);
```

Вывод:

```
1 1 0 -1
```

Заголовочный файл «string.h»

Поиск подстроки

```
1 char *ptr;
2 ptr = strstr("mechmath2018", "math");
3 puts(ptr)
```

Вывод:

```
1 math2018
```