Работа со строками в C

Строка — массив символов (элементы типа `char`). Все функции работы со строками считают, что строка ограничена специальным символом, код которого равен нулю.

```
char s[100] = "vasya"; // { 'v', 'a', 's', 'y', 'a', 0, ... }
```

Каждый символ строки — значение типа `char`, для вывода **одного символа** используется функция `putchar` или модификатор `%c` в `printf`:

```
putchar(s[0]); printf("%c", s[0]); // обе инструкции печатают v
```

Для печати всей строки можно использовать функцию `puts` или модификатор `%s` в `printf`:

```
puts(s); printf("%s\n", s); // два раза печатается vasya
```

Для чтения одного символа используйте `getchar`:

```
int c = getchar(); // именно int, а не char!
```

Результат чтения — любой из возможных символов (тип `char`) или константа `EOF` (конец ввода), то есть количество возможных результатов функции `getchar` превышает количество значений типа `char`, поэтому приходится использовать `int`.

Для чтения **слова** (до ближайшего пробела или перевода строки) используйте `scanf` с модификатором `%s`. При чтении в массив не требуется `&`, т. к. имя массива уже является адресом его начала:

```
char kuda[100]; scanf("%s", kuda); // прочитали слово
```

Для чтения **строки** (до ближайшего перевода строки) используйте `fgets`. Эта функция принимает три параметра: куда читать (имя массива), сколько символов читать (чтобы не выйти за границу массива) и откуда читать (файловая переменная). Укажите `stdin` вместо файловой переменной, чтобы читать «с клавиатуры». Внимание: в конце строки, перед 0, будет сохранён перевод строки `'\n'`!

```
char kuda[100]; fgets(kuda, 100, stdin); // прочитали строку
```

Строки нельзя присваивать друг другу или сравнивать при помощи $>$ $<$ $==$ $!=$, потому что эти операции не будут работать для массивов. Необходимо использовать специальные функции. Все они легко реализуются простым циклом, так что если забыли что-то — не страшно:

```
#include <string.h>
```

`strlen(s)` — длина строки s (не считая завершающего 0)
`strcpy(kuda, chto)` — копирование строки (порядок аргументов — как в присваивании)
`strcat(kuda, chto)` — дописывание в конец строки
`strcmp(s1, s2)` — сравнение (результат — 0, если равны, >0 или <0, если первая $>$ или $<$ второй)

Если нужно хранить массив строк, заведите двумерный массив типа `char`:

```
char lines[10][100]; // 10 строк по 100 символов
```

Можно обращаться как к его отдельным элементам (`lines[3][2] = 'z';`), так и к целым строкам (`printf("%s", lines[2]);`).