Символьные переменные

Консольный ввод/вывод символьных переменных

Одиночные символы можно задавать с клавиатуры или выводить на экран с помощью функций **scanf**() и **printf**(), указывая в строке форматирования "%c".

```
char symbol = 0;
scanf("%c", &symbol);           // считывание символа
printf("symbol = %c", symbol); // вывод на экран
```

Кроме этих универсальных функций в языке Си для ввода и вывода значений символьных переменных или просто единичных символов предусмотрены специальные функции **getchar**() и **putchar**().

```
char symbol = 0;
symbol = getchar();  // считывание символа
putchar(symbol);     // вывод на экран
```

Файловый ввод/вывод символьных переменных

Если считывание и вывод значения символьной переменной происходит из файла или в файл, то можно использовать функции **fscanf**() и **fprintf**().

```
char symbol = 0;
FILE * file;

file = fopen("input.txt", "r");
fscanf(file, "%c", &symbol);       // чтение символа из файла
fclose(file);

file = fopen("output.txt", "w");
fprintf("symbol = %c", symbol);    // вывод символа в файл
fclose(file);
```

Для файлового ввода/вывода символьных переменных также существуют специальные функции: **fgetc()** и **fputc()**.

```
char symbol = 0;
FILE * file;

file = fopen("input.txt", "r");
symbol = fgetc(file);    // чтение символа из файла
fclose(file);

file = fopen("output.txt", "w");
fputc(symbol, file);    // вывод символа в файл
fclose(file);
```

Символьные переменные	
Консольный ввод/вывод	**Файловый ввод/вывод**
`symbol = getchar();`	`symbol = fgetc(file);`
`scanf("%c", &symbol);`	`fscanf(file, "%c", &symbol);`
`putchar(symbol);`	`fputc (symbol, file);`
`printf("%c", symbol);`	`fprintf(file, "%c", symbol);`

Строки

Одномерные массивы чаще всего являются строками. Строка - это одномерный массив символов, который заканчивается нулевым символом конца строки. В качестве нулевого символа выступает символ `'\0'`. Таким образом, строка содержит символы, которые ее составляют, и нулевой символ.

При объявлении массива символов, который предназначен для хранения строки, необходимо предусмотреть место для нулевого символа, то есть указать размер этого массива при объявлении на единицу больше, чем число предполагаемых символов в массиве. Например, объявление массива из 10 символов должно выглядеть следующим образом:

```
char name[11];
```

При этом величина массива влияет всего лишь на количество выделяемой для него памяти, сам массив может быть меньше этой величины. Таким образом, размер массива при его объявлении — это только максимально возможная длина строки, но, ни в коем случае, не текущий ее размер. Таким образом, в отличие от числовых

массивов, длина строкового массива может как бы «изменяться» в пределах его размера.

При объявлении строки она также может быть сразу инициализирована:

```
char str1[21] = "строка\n";
char str2[30] = {"Test string"};
```

В первом случае выделяется память для 20 символов, а заполняются только семь из них (’\n’ – это один символ). Во втором случае заполняются одиннадцать символов из 29ти выделенных в памяти.

Существует возможность вообще не указывать размер строки при ее объявлении. В этом случае строку нужно инициализировать в обязательном порядке. Размер такой переменной-строки вычисляется, исходя из длины инициализирующей строки.

```
char special_size_str[] = "Это безразмерный массив";
```

Консольный ввод/вывод строк

Функция `scanf()`

Для того чтобы прочитать строку, введенную с клавиатуры, и записать ее в массив символов необходимо указать формат считываемой информации как `"%s"`. При указании переменной, в которую необходимо записать прочитанную строку знак амперсанта не ставится, так как имя строки это и есть ее адрес (указатель на нулевой элемент массива).

```
#define STR_LEN 250
char str[STR_LEN], str_1[STR_LEN / 2];
scanf("%s%s", str, str_1);
```

Ограничением функции **`scanf()`** можно считать то, что она считывает введенные символы до первого пробела и не всегда подходит для считывания строк большой длины, состоящих из нескольких слов или предложений.

Функция `printf()`

Вывод строк с помощью функции **`printf()`** может осуществляться двумя способами: напрямую (вывод константной строки или строки-переменной) или с указанием строки форматирования и ее параметров.

```
#define STR_LEN 250
char str[STR_LEN], str_1[STR_LEN / 2];
scanf("%s%s", str, str_1);
```

```
printf("Это вывод константной строки");
printf(str); //вывод без строки форматирования
printf("%s\n%s", str, str_1); //со строкой форматирования
```

Кроме `scanf()` и `printf()` для работы со строками существуют специальные функции `gets()` и `puts()`.

Функция `gets()`

Функция `gets()` позволяет вводить строки с клавиатуры. Параметром функции является имя массива символов, для которого считывается значение.

```
#define STR_LEN 250
char str[STR_LEN];
gets(str);
```

Важнейшим ее отличием от функции `scanf()` является то, что функция `scanf()` считывает строку до первого пробела, а функция `gets()` считывает строку до знака перехода на новую строку. В нижеследующем примере, если с клавиатуры ввести строку `"Первое слово съела корова"`, то в массив `str1` запишется только строка `"Первое"`, а в массив `str2` — полностью все предложение.

```
#define STR_LEN 250
char str1[STR_LEN], str2[STR_LEN];
scanf("%s", str1);
gets(str2);
```

Среди ограничений функции `gets()` можно отметить то, что она считывает только одну строковую переменную, в то время как `scanf()` может считывать значения сразу нескольких переменных, среди которых могут быть и строки и числовые значения. Кроме того, эта функция может приводить к ошибкам и многими современными компиляторами расценивается как потенциально опасная.

Функция `puts()`

Функция вывода строк `puts()` так же в качестве аргумента принимает строку, которая может быть представлена как переменной, так и константной текстовой строкой (в том числе макросом).

```
#define STR "Строка введена успешно!" //макрос
char stroka[21]; //строка как переменная
// вывод константной строки
puts("Введите строку не днинее 20ти символов\n");
gets(stroka);
```

```
puts(stroka); //вывод строки-переменной
puts(STR); //использование макроса в выводе
```

При выводе строк между функциями `puts()` и `printf()` существует различие: если строка выводится с помощью функции `puts()`, то после ее вывода произойдет обязательный переход на новую строку. При использовании функции `printf()` переместиться на новую строку можно, только если явно указать управляющую последовательность `"\n"`. Еще одним отличием является то, что функция `puts()` может выводить только одну строку, в то время как `printf()` способна выводить на экран любое количество объектов.

Файловый ввод/вывод строк

Для чтения/записи строк из файла существуют специальные функции `fgets()` и `fputs()`.

Функция `fgets()`

```
fgets(строка, max_длина_строки, файловый_поток);
```

Функция `fgets()` читает из потока строку, и делает это до тех пор, пока не будет прочитан символ новой строки или количество прочитанных символов не станет равным `max_длина_строки - 1`. Если был прочитан символ перехода на новую строку, то чтение заканчивается и все, что было прочитано, вместе с символом перехода на новую строку записывается в переменную-строку. Этим функция `fgets()` отличается от функции `gets()`, которая не записывает символ конца строки в переменную-строку. Полученная в результате переменная-строка будет иметь в конце символ перехода на новую строку `'\n'` и заканчиваться символом конца строки `'\0'`.

Пример. Вывести на экран первые `N` строк файла, указывая номер строки.

```
int i = 0;
char str[500];
for (i = 0; i < N; i++) {
    fgets(str, 500, file); //считываем строку из файла
    //выводим строку на экран
    printf("%dя строка - %s", i + 1, str);
}
```

Функция `fputs()`

```
fputs(строка, файловый_поток);
```

Функция `fputs()` записывает в файл переменную-строку в файловый поток. Ее функциональность полностью повторяет функцию `puts()` за исключением того, что после вывода переменной-строки (или константной строки) перехода на новую строку не происходит.

Пример. Записать в файл 5 строк введенных с клавиатуры.

```
int i = 0;
char str[500];
for (i = 0; i < 5; i++) {
    gets(str); //считываем строку с клавиатуры
    fputs(str, file); //выводим строку в файл
    //добавляем в файл переход на новую строку
    fputs("\n", file);
}
```

Общая таблица функций для работы со строками

Строковые переменные	
Консольный ввод/вывод	**Файловый ввод/вывод**
`gets(str);`	`fgets(str, SIZE, file);`
`scanf("%s", str);`	`fscanf(file, "%s", str);`
`puts(str);`	`fputs(str, file);`
`printf("%s", str);`	`fprintf(file, "%s", str);`

Особенности функций, работающих со строками.

Функция	**Особенности функции**
`scanf();`	Считывание символов до первого пробела
`fscanf();`	
`gets();`	`'\n'` не дописывается в строку
`fgets();`	`'\n'` дописывается в строку
`puts();`	`'\n'` добавляется после вывода
`fputs();`	`'\n'` не добавляется после вывода

Функции стандартной библиотеки `string.h`

Кроме функций ввод/вывода `gets()` и `puts()` для работы со строками в языке Си определено множество различных функций, которые содержатся в заголовочном файле библиотеки `string.h` ниже представлены несколько из них.

Функция	Описание
`strlen(str)`	Определение длины строки `str`
`strcmp(str_1, str_2)`	Сравнение строки `str_1` и строки `str_2`
`strcat(str_1, str_2)`	Дописать строку `str_2` в строку `str_1`
`strcpy(str_1, str_2)`	Копирование строки `str_2` в строку `str_1`

Пример. Задать в клавиатуры имя и фамилию и выяснить их длину.

```
char name[10], surname[20];
gets(name);
gets(surname);
printf("длина имени %s - %d, ", name, strlen(name));
printf("длина фамилии %s - %d", surname, strlen(surname));
```

Пример. Определить совпадают ли два введенных с клавиатуры адреса.

```
char address_Ann[25], address_Mary[25];
gets(address_Ann);
gets(address_Mary);
if(!strcmp(address_Ann, address_Mary))
    printf("Адреса совпадают");
else
    printf("Это разные адреса");
```

Так как функция `strcmp()` возвращает ноль, если строки одинаковы, то для выполнения условия ставим отрицание. Таким образом, если строки равны, то значение функции `strcmp()` ноль будет инвертирован в единицу и условие выполнится.

Массивы строк

Объявление массивов строк происходит аналогично двумерным числовым массивам. Индекс слева определяет количество строк, а правый — максимальное количество символов в строке.

Объявим массив из 25ти строк с максимальной длиной каждой строки 48 символов:

```
char str_array[25][49];
```

Для того, чтобы обращаться к отдельной строке массива строк, нужно указывать только индекс этой строки. Например, вот так выглядит ввод с клавиатуры третьей строки и вывод на экран пятой строки:

```
gets(str_array[2]);
puts(str_array[4]);
```

Пример. Вводить строки с клавиатуры и записывать их в массив строк str_arr[10][50] только тогда, когда длина введенной строки превышает 15 символов.

```
char str_arr[10][50];
for(i = 0; i < 10;  ) {
     gets(str_arr[i]);
     if (strlen(str_arr[i]) > 15) {
          i++;
     }
}
```

Пример. Прочитать из файла 7 строк и записать их в массив строк. Вывести массив строк на экран.

```
char str_arr[7][50], last_symbol;
int i = 0, len = 0;
FILE * file;
file = fopen("input.txt", "r");
for(i = 0; i < 7; i++) {
    //чтение строки
    fgets(str_arr[i], 50, file);

    //находим чему равен последний символ в строке
    len = strlen(str_arr[i]);
    last_symbol = str_arr[i][len - 1];
```

```
    // если последний символ = '\n' меняем его на '\0'
    if(last_symbol == '\n')
        str_arr[i][len - 1] = '\0';
}
fclose(file);

//вывод массива строк на экран
for(i = 0; i < 7; i++) {
    puts(str_arr[i]);
}
```

Пример. Файл `input.txt` содержит `50` строк. Используя массив строк, найти и вывести на экран самые короткие из них.

```
char str_arr[50][200];
int i = 0, len = 0;
char last_symbol;
FILE * file;
file = fopen("input.txt", "r");
for(i = 0; i < 50; i++) {
    fgets(str_arr[i], 200, file);
    len = strlen(str_arr[i]);
    last_symbol = str_arr[i][len - 1];
    if(last_symbol == '\n')
        str_arr[i][len - 1] = '\0';
}
fclose(file);

//начальное значение - длина первой строки
int min_len = strlen(str_arr[0]);

// находим минимальную длину строки
for(i  = 0; i < 50; i++) {
    if (strlen(str_arr[i]) > min_len) {
        min_len = strlen(str_arr[i]);
    }
}
```

```
//выводим на экран все строки, длина которых равна
минимальной
for(i  = 0; i < 50; i++) {
    if (strlen(str_arr[i]) == min_len) {
        puts(str_arr[i]);
    }
}
```