Строки (продолжение); работа с файлами

А. Г. Фенстер, `fenster@fenster.name`

12 марта 2010 г.

Конспект семинара №5 по программированию для студентов 1 курса ММФ НГУ. Помните, что чтение конспекта не делает посещение соответствующего семинара необязательным! О найденных ошибках и неточностях, пожалуйста, пишите мне по адресу `fenster@fenster.name`. Также буду рад получить любые комментарии по поводу этого текста.

1 Повторение

На предыдущем занятии мы рассмотрели две функции из `string.h`: функцию подсчёта длины строки `strlen` и функцию копирования строк `strcpy`. Сейчас мы разберём ещё несколько функций, но принцип работы каждой из них, в общем-то, такой же, как у двух ранее разобранных: строка (строки) проходятся в цикле до тех пор, пока не будет найден символ с кодом 0.

Задача. Реализуйте функцию `strcat`, добавляющую вторую строку в конец первой:

```
char *strcat(char *destination, char *source);
```

Функция возвращает адрес начала строки `destination`.

Задача. Реализуйте функцию, «разворачивающую» строку, переданную её в качестве аргумента. Необходимо определить длину строки и понять, какой индекс будет у символа, симметричного символу с индексом i (помните, что нумерация начинается с нуля).

2 Сравнение строк

Пусть нам нужно определить, какая из двух строк расположена ранее в лексикографическом порядке. Очевидно, что для сравнения строк не могут использоваться стандартные операторы <, > и прочие, потому как эти операторы будут сравнивать адреса нулевых элементов строк (считая эти адреса числами), что к сравнению содержимого строк не имеет никакого отношения (вообще, операция сравнения адресов имеет смысл в том и только в том случае, когда оба адреса являются адресами элементов одного и того же массива). Раз использовать операторы сравнения нельзя, нужна специальная функция сравнения строк, и эта функция — `strcmp`.

Нужно пояснить, что сравнение выполняется не строго в алфавитном порядке, а согласно кодам символов в используемой кодировке. Цифры всегда будут расположены раньше, чем буквы, а заглавные латинские буквы будут расположены раньше, чем строчные. Утверждать что-то про русские буквы нельзя: например, в кодировке KOI8-R для них даже не будет сохранён алфавитный порядок.

Функция `strcmp` возвращает целое число, которое имеет следующий смысл:

- если строки одинаковы, результат функции равен нулю;
- если первая строка «меньше» второй, результат функции меньше нуля;
- если первая строка «больше» второй, результат функции больше нуля.

Так, `strcmp("vasya", "vasya") == 0`, а `strcmp("vasya", "asya") > 0`. Если одно слово является началом другого слова, оно считается меньшим, т. е. `strcmp("ab", "abc") < 0`.

В попытках написать реализацию этой функции можно создать весьма громоздкий код, однако задача решается достаточно просто:

```
int strcmp(char *a, char *b)
{
    while (*a && *a == *b)
    {
        a++;
        b++;
    }
```

```
    return (*a - *b);
}
```

Вспомним, что `*a` и `*b` — символы, на которые указывают указатели `a` и `b` соответственно и что операция `++` сдвигает указатель вправо на один элемент массива.

Чтобы понять идею реализации, необходимо рассмотреть пять случаев:

1. Текущие символы строк совпадают и не являются символами с кодом 0 (завершающими) — выполнены оба условия, происходит переход к следующим символам обеих строк;

2. Обе строки закончились одновременно — тогда `a` и `b` одновременно указывают на символы с кодом 0 (завершающие символы строк), не выполняется первое условие (`*a` равно нулю, т. е. ложно); результат равен $0 - 0 = 0$;

3. Одна из строк закончилась раньше другой — тогда либо `*a` равно нулю и не выполнено первое условие, либо `*a` не равно нулю, но равно нулю `*b` (ведь одна из строк закончилась) и не выполнено второе условие. В любом случае, из цикла мы выйдем и вернём либо число больше нуля (если закончилась вторая строка и `*b == 0`), либо число меньше нуля (если закончилась первая строка);

4. Текущие символы строк не совпадают — не выполнено второе условие, результатом будет положительное число в случае, если символ первой строки имеет больший код, чем символ второй строки, и отрицательное в противном случае.

3 Работа с файлами

Программа на C может прочитать данные из файла или записать данные в файл. Как и в других языках программирования, при работе с файлами выполняются два действия:

1. Реальный файл связывается с какой-либо переменной программы;

2. Производятся операции чтения или записи, одним из аргументов которых является эта переменная.

Имя файла необходимо указывать только один раз: при связывании файла с переменной (пункт 1). В дальнейшем при работе с файлом (при помощи переменной) имя файла не используется.

Файловые переменные в C имеют тип `FILE *`. Никто и никогда не разыменовывает эти переменные, поскольку все функции работы с файлами принимают параметром именно указатель на тип `FILE` (потому что каждая из функций должна иметь возможность изменить содержимое этой переменной).

Для «открытия» файла (связывания файла с переменной) используется функция `fopen`, для закрытия — `fclose`). Обе объявлены в `stdio.h` следующим образом:

```
FILE *fopen(char *filename, char *mode);
int fclose(FILE *f);
```

Первый параметр функции `fopen` — имя файла, который нужно открыть, второй параметр — строка, описывающая режим; нам понадобятся следующие режимы: `"r"` для открытия на чтение, `"w"` для открытия на запись (если файл существует, его содержимое пропадёт), `"a"` для открытия на дозапись в конец. Возвращает функция файловую переменную, которой можно пользоваться. Если файл не был открыт (например, неверно указано имя), функция вернёт `NULL` — нулевой указатель. Рекомендую всегда проверять, был ли открыт файл: указание неверного имени — очень частая ошибка.

Функция `fclose` «закрывает» файл и возвращает 0, если эта операция была успешной. Проверять этот код возврата в учебных программах не так важно: вряд ли вы сможете сделать что-либо осмысленное, даже если узнаете, что файл не был закрыт. Если забыть закрыть файл, открытый на запись, данные могут не сохраниться.

В Windows существует понятие открытия файла в текстовом или в двоичном режиме (к строке `mode` добавляется буква `b`), нас сейчас этот момент не интересует, т. к. мы работаем в Linux.

Список основных функций ввода-вывода представлен в отдельном файле.

Пример работы с файлами:

```
FILE *f1 = fopen("input.txt", "r");
if (!f1)
{
    perror("fopen");  /* стандартное сообщение об ошибке */
}
```

```
FILE *f2 = fopen("output.txt", "w");
if (!f2)
{
    perror("fopen");  /* стандартное сообщение об ошибке */
}
fscanf(f1, "%d", &a);
fprintf(f2, "%d", a);
fclose(f1);
fclose(f2);
```

4 Проверочное задание («пятиминутка»)

Тема: подсчёт количества символов или строк в файле (см. задание).