Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Информатика

Семинар 8

Строки С. Определение и инициализация строки. Ввод и вывод строки. Операции над строками.

Материал подготовили:

Горбунов Д. В.

Кудрявцев М. А.

Тассов К. Л.

2022 г.

Содержание

1. Общая информация

В языке С такого типа данных как строка нет. Язык С использует "ANSI C" стандарт организации строк, который говорит о том, что строка – это последовательность символов (символьный массив), заканчивающаяся символом с кодом 0 (терминальным символом).

Размер строки может составлять несколько ГБ. Ограничения накладывают операционная система и функции выделения памяти в языке С.

В работе со строками используются символы типа char. Также существует тип данных wchar_t w – word (машинное слово), t – type.

Для работы со строками необходимо рассмотреть следующие разделы:

- инициализация строки;
- ввод строки;
- вывод строки;
- операции, которые мы можем выполнять над строками.

2. Инициализации строки

1. char str[len] = {'O', 'K', '!', '\0'}; – память выделяется статически, инициализация в виде массива. Неудобная инициализация.
2. char str[len] = "OK!"; – память выделяется статически, затем происходит копирование литерала "OK!", где len – размер.
3. char* str = "OK!"; – указатель на литерал. Удобная инициализация (не происходит копирования).

3. Функции ввода строки

При считывании строки символ '\n' – Linux и "\n\r" – Windows заменяются на '\0' (нуль символ).

1. **int scanf(const char *format, arg-list);**

Для ввода строки используется спецификация %ns, где n максимальное количество введенных символов. Функция считывает строку до первого пробельного символа (пробел, знаки табуляции, символ конца строки). Чтобы считать строку до символа конца строки можно использовать спецификацию %[^\n].

2. **int scanf_s(const char* format, arg-list);**

   При вводе строки необходимо передавать два параметра: указатель на строку и максимальная длина строки.

   Например: scanf_s("%s", str, len);

3. **char* gets(char* str);**

   Функция считывает строку до символа конца строки.

4. **char* gets_s(char* str, size_t n);**

   Лучше использовать данную функцию, чем gets. Она принимает два параметра, одним из которых является максимальный размер строки, то есть, она будет считывать столько, сколько мы указали в параметре функции, либо меньше, если встретится символ конца строки.

4. Функции вывода строки

1. **int printf(const char *format, arg-list);**

```
printf("%-3.5s", str);
```

   1. Мы можем выравнивать строки по левому и по правому краю, если ставим минус идет выравнивание по левому краю, а без минуса по правому краю.

   2. Число 3 – минимальный размер символов под строку, число 5 –

максимальный. Если строка больше 3, то произойдет расширение зоны печати, но 5 ограничивает эту длину.

2. **int puts(const char* str);**

   Вывод строки.

5. Операции над строками

5.1 Определение стандартных функций

Для работы со строками используется библиотека string.h, в ней около 30 функций. Ниже перечислены основные функции для работы со строками:

1. **size_t strlen(const char* s)** – определение длины строки.

   Нужно отметить, что, если не подключить string.h, то size_t не будет доступен, вместо него можно использовать "unsigned".

2. **char* strcpy(char* dst, const char* src)** – копирование строки.

   dst - куда копируем. src- откуда копируем. Возвращает саму строку.

3. **char* strcat(char* s1, const char* s2)** – объединение двух строк.

   В конец строки s1 дописывается строка s2 и возвращается указатель на строку s1.

4. **int strcmp(const char* s1, const char* s2)** – сравнение строк.

   Происходит посимвольное сравнение строк. Возвращает целое число большее, равное или меньшее 0, в зависимости от того, является ли очередной символ строки s1 больше, равным или меньше очередного символа строки s2.

5. **char* strchr(const char* s, int c)** – поиск первого включения символа в строке. Возвращает указатель на первое включения символа в строке. Если в строке *s* нет символа с кодом *c* возвращается указатель на NULL.

6. **char* strstr(const char* s, const char* subs)** – поиск первого включения подстроки в строке. s – строка, в которой ищем подстроку, subs – подстрока. Возвращает указатель на первое включение в строку подстроки. Если в строке s нет подстроки subs возвращается указатель на NULL.
7. **char* strpbrk(const char* s, const char* sym)** – поиск первого включения символа из последовательности в строке. s – строка, в которой ищем символы, sym – строка с набором искомых символов.
8. **size_t strspn(const char* s, const char* sym)** – поиск длины начальной подстроки в строке, содержащей только символы из последовательности. s – строка, в которой ищем подстроку, sym – строка с набором искомых символов. Возвращает длину подстроки.

5.2 Определение дополнительных функций

Следующих функций нет в стандартной библиотеке, но они часто требуются на практике.

1. **char* strdel_my(char* s, unsigned n)** – удаление подстроки в строке. s – указатель на символ, с которого начинается удаление, n – количество удаляемых символов.
2. **char* strins_my(char* s, const char* subs)** – вставка подстроки в строку. В строке s должно быть достаточно места для добавления строки subs. s – указатель на символ, с которого начинается вставка, subs – подстрока.
3. **char* strsub_my(char* subs, const char* s, unsigned len)** – выделение из строки подстроки. subs - результирующая подстрока, s – исходная строка, len - длина.

6. Реализация функций

6.1 Реализация стандартных функций

Реализуемые функции являются атомарными.

1. **Определение длины строки**

   Перебирает все символы, пока не встретится **нуль** символ. Если в строке нет **нуль** символа, произойдет выход за пределы выделенной памяти. Строка должна заканчиваться нуль символом.

```
unsigned strlen(const char* s) {
    unsigned i;
    for (i = 0; s[i]; i++);
    return i;
}
```

2. **Копирование строки**

   Происходит посимвольное копирование из строки s2 в строку s1 до тех пор, пока не будет скопирован нуль символ. Перед вызовом функции необходимо обеспечить резервирование памяти для результирующей строки s1.

```
char* strcpy(char* s1, const char* s2) {
    for (unsigned i = 0; s1[i] = s2[i]; i++);
    return s1;
}
```

3. **Объединение двух строк**

   Объединяет две функции – нахождение длины и копирование строки. После нахождения длины s1 указатель p содержит адрес нуль символа,

далее происходит копирование s2 в конец s1 с указателя p. Перед вызовом функции необходимо обеспечить резервирование памяти для результирующей строки.

```
char* strcat(char* s1, const char* s2) {
    char* p;
    for (p = s1; *p; p++);
    for (unsigned j = 0; p[j] = s2[j]; j++);
    return s1;
}
```

4. Сравнение двух строк

Происходит посимвольное сравнение двух строк до тех пор, пока не встретятся разные символы или не закончится какая-либо строка. i - позиция, в которых символы отличаются.

```
int strcmp(const char* s1, const char* s2) {
    unsigned i;
    for (i = 0; s1[i] && s1[i] == s2[i]; i++);
    return s1[i] - s2[i];
}
```

5. Поиск символа в строке

Происходит посимвольное сравнение до тех пор, пока не закончится строка или не встретится нужный символ.

```
char* strchr(char* s, char ch) {
    char* p;
    for (p = s; *p && *p != ch; p++);
    return *p ? p : 0;
```

```
}
```

6. Поиск подстроки в строке

Указатель p хранит адрес текущей позиции в строке, с которой будет начинаться поиск подстроки. Флаг key используется для определения факта нахождения подстроки в строке.

```
char* strstr(char* s, const char* subs) {
   int key = 0;
   char *p;
   for (p = s; !key && *p;) {
      unsigned i;
      for (i = 0; subs[i] && p[i] == subs[i]; i++);
      if (!subs[i])
         key = 1;
      else
         p++;
   }
   return key ? p : 0;
}
```

7. Поиск первого включение символа из последовательности в строке

Указатель p хранит адрес символа из последовательности символов sym в исходной строке s. Флаг key используется для определения факта нахождения символа в строке.

```
char* strpbrk(char* s, const char* sym) {
   int key = 0;
   char *p;
```

```
    for (p = s; !key && *p;) {
        unsigned i;
        for (i = 0; sym[i] && *p != sym[i]; i++);
        if (sym[i])
            key = 1;
        else
            p++;
    }
    return key ? p : 0;
}
```

8. Поиск длины начальной подстроки содержащей только символы из последовательности

Флаг key определяет факт отсутствия очередного символа из строки s в подстроке sym. Переменная len определяет индекс первого символа из строки s, отсутствующего в строке sym. Len определяет длину подстроки содержащей символы из строки sym.

```
size_t strspn(const char* s, const char* sym) {
    int key = 0;
    size_t len;
    for (len = 0; !key && s[len];) {
        const char* p;
        for (p = sym; *p && s[len] != *p; p++);
        if (!*p)
            key = 1;
        else
            len++;
```

```
    }
    return len;
}
```

6.2 Реализация дополнительных функций:

Реализуемые функции являются не атомарными.

1. **Удаление подстроки в строке**

   Если n ≥ размера строки – ставится нуль символ в начало строки. Если n < размера строки – для удаления используется strcpy, в строку s копируется s + n.

```
char* strdel_my(char *s, unsigned n) {
    if (n >= strlen(s))
        *s = 0;
    else
        strcpy(s, s + n);
    return s;
}
```

2. **Вставка подстроки в строку**

   Необходима вспомогательная строка stemp, куда копируется исходная строка s. Затем в s копируется строка subs и происходит конкатенация s и stemp.

```
char* strins_my(char* s, const char* subs) {
    char* stemp = (char*) calloc(strlen(s) + 1, sizeof(char));
    strcpy(stemp, s);
    strcpy(s, subs);
    strcat(s, stemp);
```

```
    free(stemp);
    return s;
}
```

3. **Выделение из строки подстроки**

   Копирование len символов из строки s в строку subs. Если встречается нуль символ, копирование заканчивается.

```
char* strsub_my(char* subs, const char* s, unsigned len) {
    unsigned i;
    for (i = 0; i < len && subs[i] = s[i]; i++);
    subs[i] = '\0';
    return subs;
}
```

Лабораторная работа

Лабораторная работа №8