

Различие между arr и &arr – как в C определить размер массива без sizeof

13.02.2017 / 16:39 от [kalterfx](#)
👤 Прочие языки c, си

Hey folks, Long time no C.

Обычно в C мы находим длину массива `arr` так:

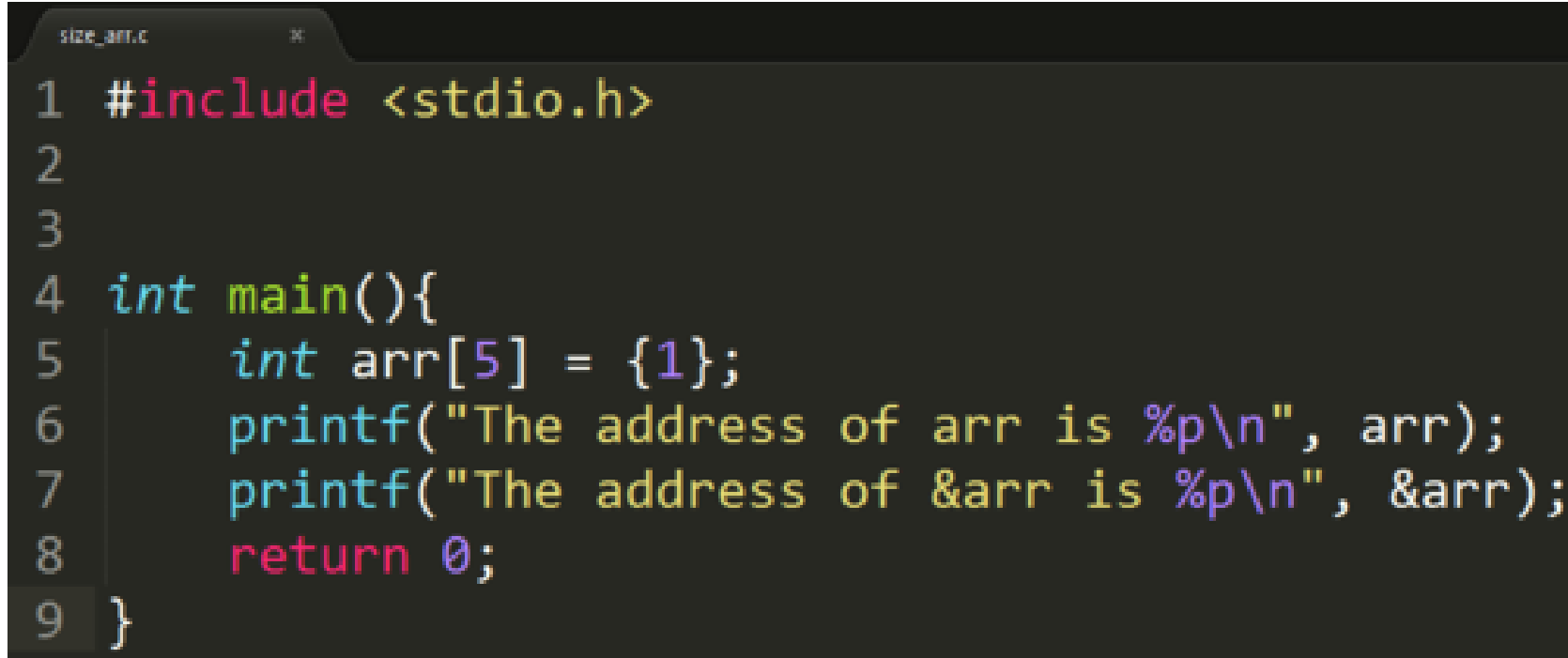
```
1. int n = sizeof(arr) / sizeof(arr[0]);
```

Здесь мы получаем размер массива в байтах; затем происходит деление этого размера на размер каждого элемента в массиве. *Давайте попробуем избавиться от `sizeof`.*

Никто из вас никогда не задавался вопросом насчёт разницы между `arr` и `&arr`? Это не одно и то же.

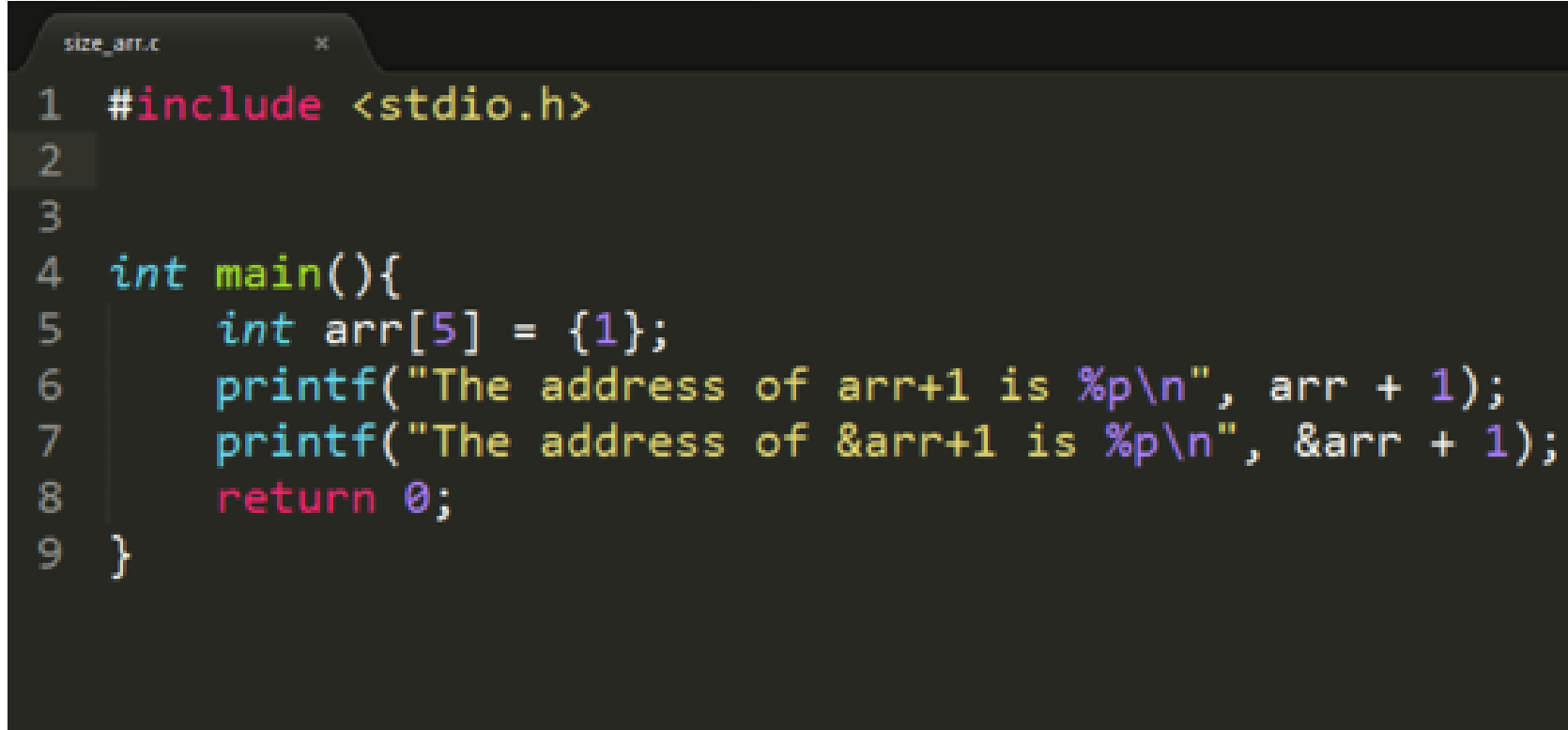
«1 из 3»

Давайте проверим это выведением адресов этих двух указателей



«1 из 2»

Теперь попробуем инкрементировать оба указателя на 1 и снова проверить их адреса в памяти



Мы можем определиться в следующем:

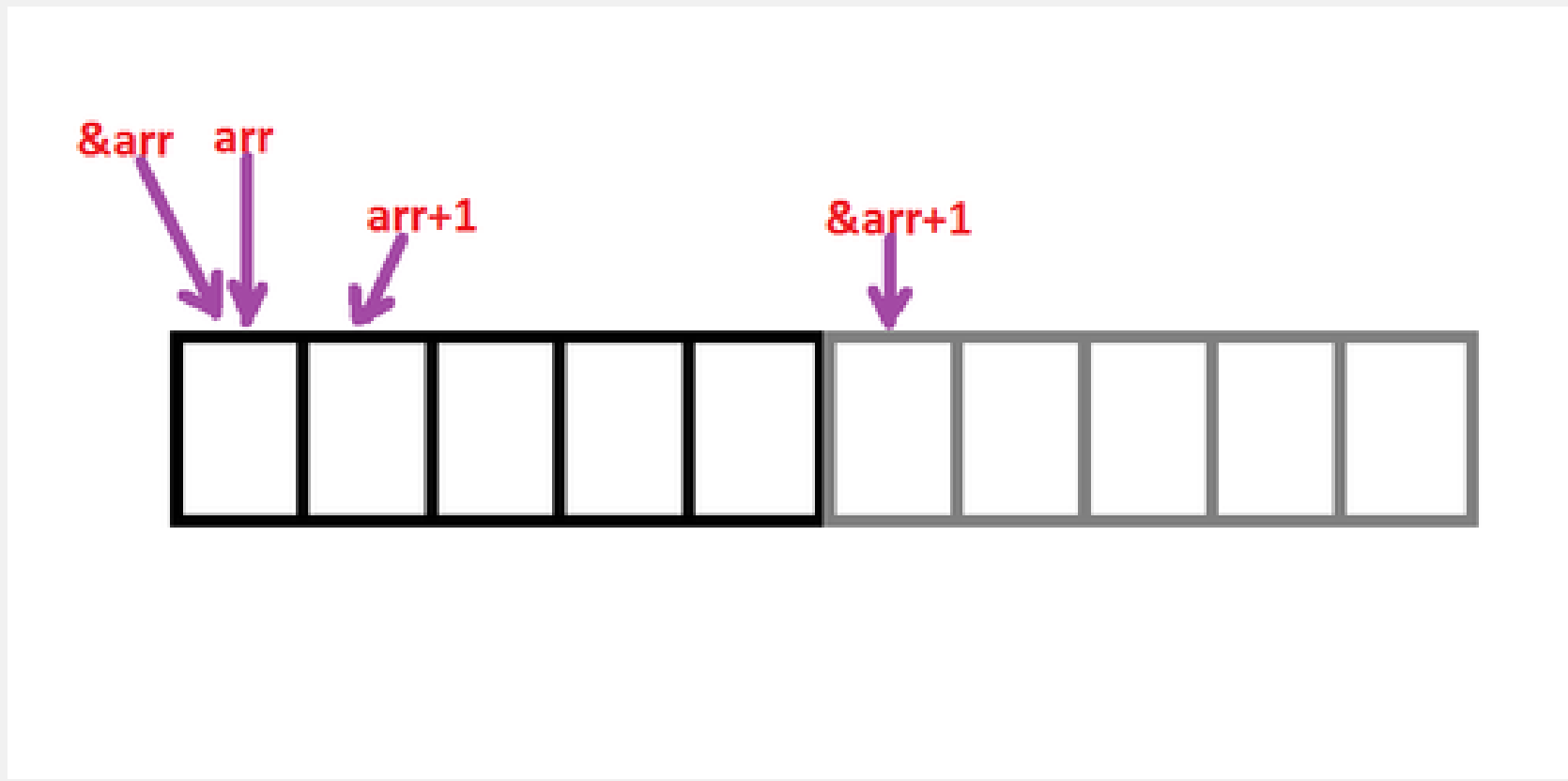
(arr + 1) указывает на **0x244fdc8** – это значение уходит на 4 байта вперёд от **arr**, который указывает на **0x244fdc4**.
Так как переменные типа `int` занимают 4 байта, **(arr + 1) указывает на второй элемент массива**.

(&arr + 1) указывает на **0x244fdd8** – это значение уходит на 20 байт вперёд от **arr**, который указывает на **0x244fdc4**.
(0x244fdd8 - 0x244fdc4 = 0x14 = 20)

Теперь можно сделать вывод о том, что не смотря на то, что **arr** и **&arr** указывают на одно и то же место в памяти, их применение совершенно разное.

arr имеет тип `int *`, в то время как **&arr** имеет тип `int (*)[size]`.

&arr указывает на весь массив, в то время как **arr** указывает на первый элемент массива.



Из этого можно извлечь один полезный опыт – получение длины массива.

***(&arr + 1)** даёт нам адрес за последним элементом массива, а **arr** адрес первого элемента массива.
Таким образом, вычитание второго из первого может дать нам длину массива.

```
1. int n = *(&arr + 1) - arr;
```

Мы можем это упростить, используя индексы массива (так как `x[1] == *(x + 1)`):

```
1. int n = (&arr)[1] - arr;
```

PS:
Этот метод работает только для массивов, но не для указателей (вроде `char *str`).

```
1. void reverseStr(char *str)
2. {
3.     // wrong
4.     int strlength = (&str)[1] - str;
5. }
```

В этом случае `&str` является указателем, который указывает на указатель `str`. Стоит запомнить, что массивы в C не являются указателями.

Послесловие:

Я люблю цитировать этот вопрос-ответ, который я нашёл на reddit / SO об обсужденной теме.

Q: Доступ к первому адресу после массива, кажется, предшествует неопределённому поведению (undefined behavior). Например, если массив располагается в самом конце всего адресного пространства, адрес, указывающий на первый элемент после массива, может вызвать переполнение и результирующий размер массива может быть любым. Как в этом случае можно обратиться к `(&arr)[1]`?

A: C не позволяет получить **доступ** к памяти за последним элементом массива. Тем не менее, C позволяет указателю указывать на один элемент вперёд за последним элементом массива. Разница важна.

Это мой вольный перевод статьи от Arjun Sreedharan, изначально написанной в блоге arjunsreedharan.org

 +4




 6079

 [kalterfx](#)

Комментарии (9)

- Прочие языки
- Категории

Поделиться

Похожие статьи

- Как эффективно учиться
- Как писать скрипты для Gimp 2.x? Осваиваем Scheme (Script-Fu). Часть 2.
- Как мы постепенно перешли на новые технологии
- Как удалить системные приложения на Xiaomi без рута
- Как писать скрипты для Gimp 2.x? Осваиваем Scheme (Script-Fu). Часть 1.

Другие статьи автора

- Писать в тот же файл, который читается
- Используем библиотеку JSOUP на примере бэкапа цитат из bash.im
- Long Flight: The history of development
- Структура мидлета [Java 2 ME]
- Посмотреть все (11 статей)

Дорогой посетитель

Реклама позволяет нашему ресурсу существовать и развиваться. Пожалуйста, отключите блокировщик рекламы или зарегистрируйтесь.

Спасибо. Приятного чтения. 😊