


[⬆ \(http://cppstudio.com\)](http://cppstudio.com)[/ Стандартные заголовочные файлы из Си в C++ \(http://cppstudio.com/cat/309/\)](http://cppstudio.com/cat/309/)[/ Заголовочный файл cstring \(string.h\) \(http://cppstudio.com/cat/309/325/\)](http://cppstudio.com/cat/309/325/) / Функция memcpy

# Функция memcpy

 Оценка: **4,00** ( голосов: 1 )

Чтобы проголосовать, вы должны зарегистрироваться.

## Прототип функции memcpy:

```
1 void * memcpy( void * destptr, const void * srcptr, size_t num );
```

## Заголовочный файл:

Название	Язык
string.h	C
cstring	C++

## Описание

Функция memcpy копирует num байтов первого блока памяти, на который ссылается указатель srcptr , во второй блок памяти, на который ссылается указатель destptr .

Тип данных (/uchebniki/yazyk-programmirovaniya-s/tipy-dannyx-s/) объектов, на которые указывают как srcptr так и destptr не имеют никакого значения. Так как эта функция работает с бинарными данными.

Функция не проверяет, есть ли символ завершения в srcptr , она всегда копирует количество байтов, указанное в num .

Чтобы избежать переполнения блока памяти destptr , размер destptr должен быть не менее num байтов. Однако, может возникнуть ситуация, когда destptr и srcptr пересекутся. Поэтому, для перекрытия блоков памяти, функция memmove (/spravochnik/standartnye-zagolovochnye-fajly-iz-si-v-s/zagolovochnyj-fajl-cstring-string-h/funkciya-memmove/) является более безопасным подходом.

## Параметры:

- **destptr**

Указатель на блок памяти назначения (куда будут копироваться байты данных), имеет тип данных void .

- **srcptr**

Указатель на блок памяти источник (т. е., откуда будут копироваться байты данных), имеет тип данных `void`.

- **num**

Количество копируемых байтов.

## Возвращаемое значение

Указатель на блок памяти назначения.

## Пример: исходный код программы

```
1 //пример использования функции memcpy
2
3 #include <iostream>
4 #include <cstring>
5
6 int main()
7 {
8     char str1[] = "Пример строки";
9     char str2[40];
10    char str3[60];
11
12    memcpy (str2, str1, strlen(str1)+1); // копируем строку str1 и
13    memcpy (str3, "Копирование успешно выполнено", 60); // копируем 60 байт строки
14    std::cout << "str1: " << str1
15              << "nstr2: " << str2
16              << "nstr3: " << str3 << std::endl;
17    return 0;
18 }
```

## Пример работы программы

CppStudio.com

str1: Пример строки

str2: Пример строки

str3: Копирование успешно выполнено

Обсудить на форуме (/topics/)

Автор: admin (/forums/users/admin/)

Дата: 31.08.2012

Поделиться:

### Похожие статьи:

1. Функция memmove (<http://cppstudio.com/post/682/>)
2. Функция strncpy (<http://cppstudio.com/post/690/>)
3. Функция strcpy (<http://cppstudio.com/post/686/>)
4. Функция strncat (<http://cppstudio.com/post/698/>)

## 5. Класс, реализующий операции со строками (<http://cppstudio.com/post/1530/>)

## Комментарии



kpanat . (<https://plus.google.com/100539009002642920953>)

03.11.2015 (/post/678/comment-page-1/#comment-2967)

Ну собственно как раз эта функция и не работает. Особенно усовершенствованная модель... Только для неперекрывающихся блоков. Хотя раньше она работала и для перекрывающихся также, но только когда копирование назад делалось. Вперёд она затирает ещё нескопированные данные. Поэтому она и не работает. Но если копировать не от начала к концу строки а наоборот, то будет копировать вперёд. Но назад будет затирать. А если варьировать направление копирования в зависимости от направления сдвига блока, то можно сделать корректное и быстрое копирование для всех случаев. Но в функции тетсру этот параметр(направление копирования) прописан внутри, жёстко фиксирован вперёд и не меняется... Надо писать свой вариант тетсру.

Войдите, чтобы ответить ([http://cppstudio.com/wp-login.php?redirect\\_to=http%3A%2F%2Fcppstudio.com%2Fpost%2F678%2F](http://cppstudio.com/wp-login.php?redirect_to=http%3A%2F%2Fcppstudio.com%2Fpost%2F678%2F))






## Оставить комментарий

Вы должны войти ([http://cppstudio.com/wp-login.php?redirect\\_to=http%3A%2F%2Fcppstudio.com%2Fpost%2F678%2F](http://cppstudio.com/wp-login.php?redirect_to=http%3A%2F%2Fcppstudio.com%2Fpost%2F678%2F)), чтобы оставить комментарий.

## Translation



(/post/678/)Русский (/post/678/)  
(/uk/post/678/)Українська (/uk/post/678/)  
(/en/post/678/)English (/en/post/678/)

 (/de/post/678/)Deutsch (/de/post/678/)  
 (/be/post/678/)Беларуская (/be/post/678/)  
 (/kk/post/678/)Қазақ тілі (/kk/post/678/)  
 (/uz/post/678/)O'zbek tili (/uz/post/678/)  
 (/tr/post/678/)Türkçe (/tr/post/678/)

## Новое

- Особенности Qt: слоты и сигналы, описание QObject и QApplication, виды окон и т.д.  
(<http://cppstudio.com/post/11167/>)
- Первая программа на Qt:  
(<http://cppstudio.com/post/11127/>)
- Введение – графическая библиотека Qt  
(<http://cppstudio.com/post/11097/>)
- Наследование классов  
(<http://cppstudio.com/post/10103/>)
- Перегрузка операторов в C++ (часть 2)  
(<http://cppstudio.com/post/10058/>)

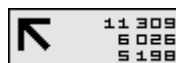
## Популярное

Sorry. No data so far.

© 2022 CppStudio – Программирование для начинающих на C++



(<https://plus.google.com/u/0/106109650739084338784>)



(<http://www.liveinternet.ru/click>)



(<http://orphus.ru>)