

# Учебный курс. Часть 5. Директивы объявления данных

Автор: **xrnd** | Рубрика: [Учебный курс](#) | 14-03-2010 | 

[Распечатать запись](#)

Практически любая программа кроме машинных команд содержит также какие-то данные. Например, числа, текстовые строки, идентификаторы, различные ресурсы и т.д. Данные могут быть как константами, не меняющими своё значение во время выполнения программы, так и переменными, в которых хранятся всякие промежуточные результаты.

Прежде всего нужно научиться объявлять данные в программе. Для этого в ассемблере существуют директивы объявления данных.

Размер (в байтах)	Объявление	Резервирование
1	db	rb
2	dw du	rw
4	dd	rd
6	dp df	rp rf
8	dq	rq
10	dt	rt
N	file	

В учебном курсе для нас самыми полезными будут директивы *db*, *dw* и *dd*.

## Синтаксис объявления данных

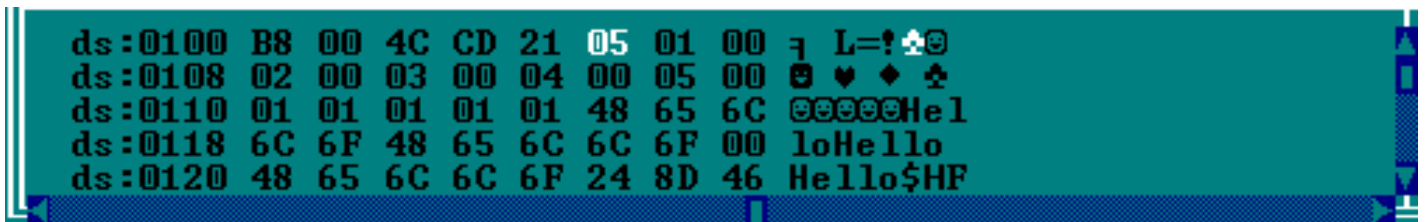
Объявлять данные очень просто — например, чтобы объявить байт со значением 5 достаточно написать:

```
x db 5
```

где *x* — название нашей переменной или константы, *db* — директива объявления байта, а *5* — значение. С помощью названия в программе можно будет обращаться к ячейке памяти, содержащей наш байт. Вообще, название не обязательно и можно его не писать, если оно не требуется:

```
db 5
```

Если запустить программу в отладчике Turbo Debugger, то в окне дампа можно увидеть результат работы директивы *db*:



```
ds:0100 B8 00 4C CD 21 05 01 00  L=?♠  
ds:0108 02 00 03 00 04 00 05 00  ♠♥♦♣  
ds:0110 01 01 01 01 01 48 65 6C  @@@@Hel  
ds:0118 6C 6F 48 65 6C 6C 6F 00  loHello  
ds:0120 48 65 6C 6C 6F 24 8D 46  Hello$HF
```

## Объявление последовательностей (массивов)

Иногда в программе требуется объявить массив, то есть несколько переменных одинакового размера, расположенных в памяти друг за другом. Например, чтобы объявить массив из 5 двухбайтных чисел можно написать:

```
array1 dw 1,2,3,4,5
```

где *array1* — название массива, *1,2,3,4,5* — значения элементов. Вместо *array1* компилятор FASM будет подставлять в программу адрес начала массива, то есть адрес первого элемента.

Дамп памяти будет выглядеть следующим образом (обратите внимание, младший байт каждого слова расположен перед старшим):



```
ds:0100 B8 00 4C CD 21 05 01 00  L=?♠  
ds:0108 02 00 03 00 04 00 05 00  ♠♥♦♣  
ds:0110 01 01 01 01 01 48 65 6C  @@@@Hel  
ds:0118 6C 6F 48 65 6C 6C 6F 00  loHello  
ds:0120 48 65 6C 6C 6F 24 8D 46  Hello$HF
```

Для объявления повторяющихся элементов можно использовать такую запись (объявляем массив из 5 байтов, равных 1):

```
array2 db 5 dup(1)
```



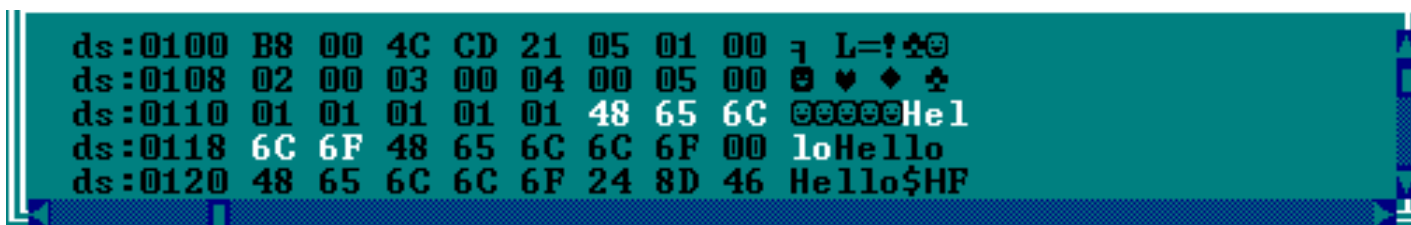
А ещё можно вот так объявить массив (догадайтесь сами, что тут получается):

```
array3 dd 4 dup(3,7,0)
```

## Объявление строк

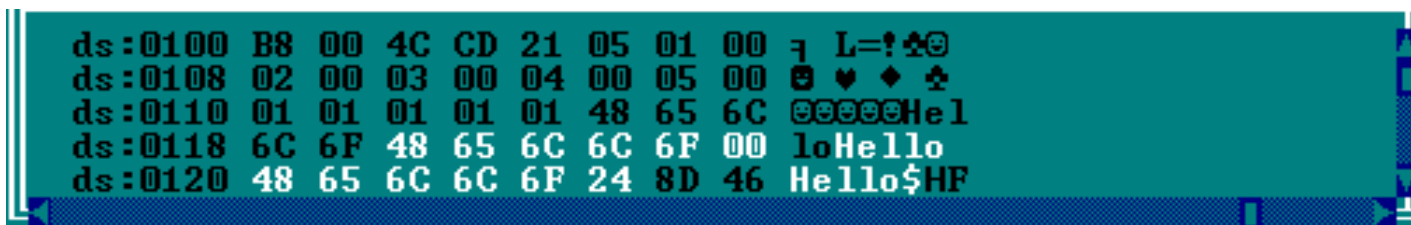
Строка представляет собой массив байтов-символов и записывается в одинарных кавычках:

```
str1 db 'Hello'
```



Для обозначения конца строки используется специальный символ. Обычно это нулевой байт, но для функций DOS используется символ '\$'.

```
str2 db 'Hello',0      ;Обычно так  
str3 db 'Hello$'       ;Для DOS
```



## Резервирование данных (точнее памяти для них)

Можно объявлять переменные, не имеющие определённого начального значения. Такие переменные называются неинициализированными. Например, их можно использовать в программе для хранения временного или промежуточного значения. Фактически под переменную просто резервируется место в памяти. Объявлять такие переменные можно с помощью директив *db*, *dw*, *dd*, ... и знака вопроса вместо значения.

```
x1 db ?  
x2 dw ?, ?, ?  
x3 dd 10 dup(?)
```

Кроме того, FASM поддерживает специальные директивы резервирования данных. Число после директивы обозначает количество резервируемых элементов. То же самое можно объявить вот так:

```
x1 rb 1  
x2 rw 3  
x3 rd 10
```

С неинициализированными переменными следует быть внимательным. Не надо рассчитывать, что по умолчанию значение будет нулевым или ещё каким-то, иначе это может привести к ошибке.

### Директива *file*

*file* — это особая директива объявления данных, которая позволяет добавить в исполняемый файл последовательность байтов из внешнего файла. Иногда это может быть очень удобно. Например, если вы хотите добавить изображение в исполняемый файл (в виде данных), или большой кусок текста, или даже код из другого файла. Директива используется следующим образом:

```
data1 file 'data.bin'      ;Добавить файл data.bin целиком.  
data2 file 'data.bin':20   ;Добавить байты из файла data.bin, начиная со с  
data3 file 'data.bin':20,5 ;Добавить 5 байтов из файла data.bin, начиная с
```

[Следующая часть »](#)

# Комментарии:

RoverWWorm

04-04-2010 20:21

aaa, я совсем запутался, выше в таблице db -это же два байта, а не 1 байт,  
dw-4 байта, а не 2 байта, вроде

[\[Ответить\]](#)

[xrnd](#)

05-04-2010 02:56

db — это 1 байт (Define Byte), а dw (Define Word) — 2 байта или слово (word = слово).

[\[Ответить\]](#)

bebe

08-04-2010 21:07

>array3 dd 4 dup(3,7,0)

А что получается? 😊 По мне так первый байт = 3, второй = 7, третий =0, четвертый снова =3 😊

А как на самом деле?

[\[Ответить\]](#)

[xrnd](#)

10-04-2010 17:25

Не совсем так. dd это Define Dword, поэтому первое двойное слово = 3, второе = 7, третье = 0, и всё это повторяется 4 раза 😊

[\[Ответить\]](#)

[mc-black](#)

08-05-2010 22:27

В masm (32) насколько я его знаю, нет встроенного средства определения файла (не считая include, includelib — это совсем другое). Также нет группы rb, rw, rd — используют вариант со знаками вопроса.

[\[Ответить\]](#)

[xrnd](#)

09-05-2010 16:50

Да, именно так. У FASM другая идеология. Он сразу создаёт исполняемый файл. Исключается этап компоновки EXE из объектных файлов. Хотя можно создать и объектный файл специальными директивами.

В этом плане FASM более гибкий. С его помощью можно создать вообще любой файл, не обязательно исполняемый. Или можно просто скомпилировать кусок кода, без какого либо формата. Например, это очень удобно для создания всяких шеллкодов и т.п. 😊

[\[Ответить\]](#)

GENN

12-11-2010 12:57

Спасибо

[\[Ответить\]](#)

Philin

11-12-2010 16:42

Привет... пара вопросов...

array3 dd 4 dup(3,7,0) — где \*dup\* это директива повторения...? или это вовсе не \*директива\*...?

и,

str1 db 'Hello' — в каком случае ставится директива \*db\*...? или, слово из скольки знаков можно записать под этой директивой \*db\*...?

спасибо...

[\[Ответить\]](#)

[xrnd](#)

12-12-2010 21:27

Да, dup — директива повторения.

После db можно писать строку любой длины.

[\[Ответить\]](#)

Cross

09-07-2011 07:04

Интересно почему в дампе памяти какие то картинки рожи :)?

[\[Ответить\]](#)

AD

14-07-2011 01:59

потому что ascii =)

[\[Ответить\]](#)

Sergey\_K

21-08-2011 16:36

Спасибо!

[\[Ответить\]](#)

Денис

10-11-2011 12:31

Что то я не понимаю.

1 Как узнать имя переменной в дампе памяти?

[\[Ответить\]](#)

vv20

12-01-2012 03:02

А ещё можно вот так объявить массив (догадайтесь сами, что тут получается):

array3 dd 4 dup(3,7,0)

Может это:

03 00 00 00 07 00 00 00 00 00 00 00 03 00 00 00 07 00 00 00 00 00 00 03  
00 00 00 07 00 00 00 00 00 00 00 00 03 00 00 00 07 00 00 00 00 00 00 00

[\[Ответить\]](#)

Vlad

11-02-2012 20:53

Подскажите, плз, по какому адресу помещается байт директивой db?

[\[Ответить\]](#)

magi

13-06-2012 11:18

В том месте где встретиться данная директива. Поэтому если размещать данные в коде надо предусмотреть обход данных, примерно так:

```
use16
```

```
org 100h
```

```
jmp kod
```

```
array2 db 5 dup(1)
```

```
kod:
```

```
mov ax,255
```

```
inc ax
```

```
nop
```

```
mov bx,ax
```

```
mov ah,4ch
```

```
int 21h
```

Если данный обход не сделать то объявленные данные будут трактоваться как код, что наверняка приведет к непредсказуемым результатам.

[\[Ответить\]](#)

[Мудрец](#)

30-05-2012 17:42

<3 ваш сайт ))

[\[Ответить\]](#)

Владислав

27-11-2012 09:51



Отличный ресурс, автору всяческие плюсы)

[\[Ответить\]](#)

Cross

02-12-2012 02:14

a kak skompilirovat v :

1) exe

2) com

??

[\[Ответить\]](#)

guest456234

11-01-2013 19:49

LOL

[\[Ответить\]](#)

Artem

28-02-2013 21:40

Здравствуйте. Есть вопрос.

1)Я проделал такой эксперимент. Если я указываю директиву управления адресным пространством ни с org 100h, а с org 150h! Почему при компиляции файл получается с расширением \*.bin??? в чём причина такого эффекта и как можно это исправить если например я хочу разместить файл по другому адресному пространству. Как я понимаю org указывает на смещение . Буду признателен если поможете в данном вопросе. С уважением Артём

[\[Ответить\]](#)

[xrnd](#)

19-03-2013 22:22

COM-программа должна начинаться с адреса 100h. Если ты указываешь 150h, то FASM уже не считает твой код COM-программой и генерирует бинарный файл с кодом, поэтому расширение .bin

[\[Ответить\]](#)

Роман

07-09-2013 19:45

а файл может хоть где находится или где он должен быть?

[\[Ответить\]](#)

[Роман](#)

06-11-2013 18:49

У меня такой вопрос. Может я чего-то не догоняю, но если мы зарезервируем например байт памяти без имени, то есть db 5, то как нам к нему обращаться

[\[Ответить\]](#)

хор eaх,eaх

17-10-2014 21:20

Имя — указатель на начало данных. Его можно и вручную указывать.

[\[Ответить\]](#)

Максим

21-12-2015 17:35

Плюсую. Мне тоже хотелось бы знать ответ на этот вопрос!

[\[Ответить\]](#)

Daniel Mor

04-07-2017 17:14

Так часто делают когда объявляют массив. Для элементов массива тоже не создаются указатели. Нужен всего-то указатель на начало, а остальные вычисляются суммированием начального адреса и номера элемента. Т.е. объявили элемент с меткой array db 'с', а остальные можно не «метить».

[\[Ответить\]](#)

Profi\_GMan  
19-07-2017 19:07

такой же вопрос

[\[Ответить\]](#)

Денис  
16-11-2015 04:13

Читал на хабре статью Пишем свою ОС.

Там был приведен код загрузчика на yasm. И так как код располагается в загрузочном секторе, то мы должны иметь в конце этого сектора сигнатуру 55 AA. На yasm это делается с помощью директивы times. А как реализовать инициализацию этих двух последних байт директивой ассемблера tasm? Я знаю что есть dup, но как им воспользоваться в данном случае?

Например

```
.code
org 7c00h
start:
;=====code=====
;=====data=====
;заполнение до конца сектора-2 байта нулями
finish:
times 0x1FE-finish+start db 0
magic db 55, AA
end start
```

если тем же образом это интерпретировать на tasm:

finish:

db 1FEh — \$ + start dup (0)

то от компилятора летит такой подзатыльник:

**\*\*Error\*\*** C:\MyOS\source\boot.asm(49) Can't subtract dissimilar relative quantities

Подскажите пожалуйста, если этот портал еще живой.

[\[Ответить\]](#)

Асилбек  
03-10-2016 16:43

После урока 11, у меня возник такой вопрос. Число в 64-бита можно записать в dd, а как написать произведение 64-битовых чисел?

[\[Ответить\]](#)

Гость

20-07-2017 13:14

Предположим, я вписал такой код в программу:

```
data2 file 'program.bin':512
```

И как тогда запустить его?

[\[Ответить\]](#)

Гость

20-07-2017 17:29

Как запустить добавленный код с помощью перехода на его начальный байт, как должен выглядеть переход?

[\[Ответить\]](#)

Arsentiy

30-12-2017 19:36

Напишите пожалуйста код полностью а то не получается объявлять данные

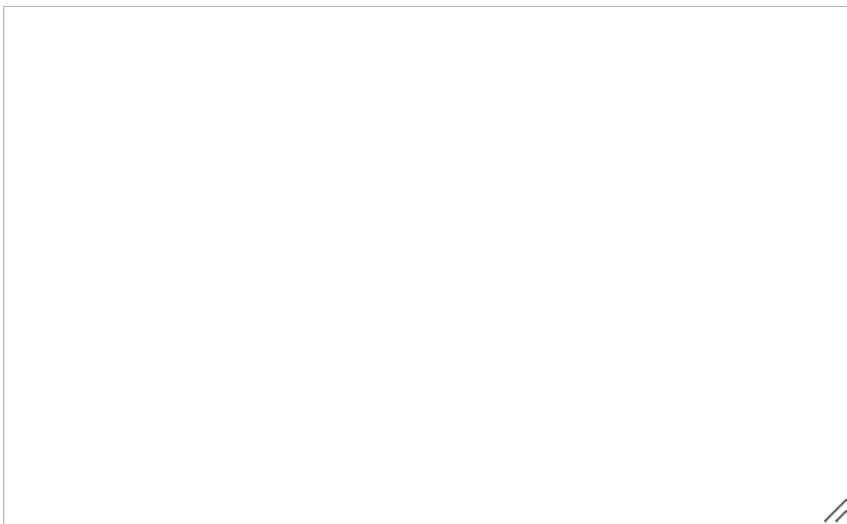
[\[Ответить\]](#)

## Ваш комментарий

Имя \*

Почта (скрыта) \*

Сайт



Добавить

- ☐ Уведомить меня о новых комментариях по email.
- ☐ Уведомлять меня о новых записях почтой.