

Типове (структури) данни.

Оператори.



## Типове (структури) данни



## Скаларни данни

Скалара е единична данна, която не може да бъде разглеждана като състояща се от съставни компоненти. Скалари са числата, низовете и указателите. Пред имената на скаларните променливи се поставя префикс: \$ .

```
$int_num = 256;           # цяло число
$name     = 'Iva Popova'; # символен низ
$name_ref = \ $name;     # указател към променлива
```

## Числови литерали

---

256	# цяло число
12.34	# число, с плаваща запетая
1_000_000	# за по-лесно разчитане
0644	# осмично число (префикс - 0 )
0xAF	# шестнадесетично число (префикс - 0x )
0b1011	# двоично число (префикс - 0b )

## Низови литерали

Съвкупност от символи, обикновено заградени в единични ( ' ') или двойни ( " ") кавички. Низ, заграден в единични кавички се възприема буквално. Низът в двойни кавички подлежи на интерполация - променливите се заместват със техните стойности, а ескейп последователностите се заместват с тяхното значение.

```
$name='Iva';  
  
print 'Hi $name';           < Hi $name >  
  
print "Hi $name";         < Hi Iva >  
  
print '10000\210';        < 10000 \210 >  
  
print "10000\210";        < 10000 € >
```

## Защо низът да е заграден точно в кавички ?

TMTOWTDI :

' ' <=> q//

" " <=> qq//

Вместо наклонените черти (//) може да се използва кой да е символ (без цифри, букви и спейс) от клавиатурата:

```
print '2\'1, 2\'2, 2\'3';           < 2'1, 2'2, 2'3 >
```

```
print q(2'1, 2'2, 2'3);           < 2'1, 2'2, 2'3 >
```

# Масиви

Нареден списък от елементи (скалари, списъци)

@plodove

круша	ябълка	лимон	череша	ягода	стойности
0	1	2	3	4	индекси

Инициализират се със **списък** от литерали/променливи

```
@plodove = ('круша', 'ябълка', 'лимон', 'череша', 'ягода');
```

Или записано по друг начин:

```
@plodove = qw(круша ябълка лимон череша ягода);
```

## Масиви - достъп до елементите

```
# @plodove = qw(круша ябълка лимон череша ягода);
```

Стандартно индексирание:

```
$plodove[0];      # круша
```

```
$plodove[4];      # ягода
```

Отрицателни индекси:

```
$plodove[-1];    # ягода
```

```
$plodove[-3];    # лимон
```

Определяне на последния индекс :

```
$#plodove;      # 4
```

Масив в скаларен контекст:

```
$foo = @plodove;          # $foo = 5
```

## Операции над масиви

---

### **push** ARRAY,LIST

добавя *LIST* в **края** на масива и връща броя на елементите на новия масив.

### **pop** ARRAY

връща **последния** елемент на масива като го премахва и намалява размера на масива с 1.

### **unshift** ARRAY,LIST

добавя *LIST* в **началото** на масива и връща броя на елементите на новия масив.

### **shift** ARRAY

връща **първия** елемент на масива като го премахва и намалява размера на масива с 1.

## Хешове (Асоциативни масиви)

Неподредена съвкупност от елементи (няма начало и край)

`%pazar`

35	12	43	9	16	стойности
круша	ябълка	лимон	череша	ягода	КЛЮЧОВЕ

Инициализират се чрез **списък** двойки от вида:

**' КЛЮЧ ' - СТОЙНОСТ :**

Например:

```
%pazar = ( 'круша' => 35 ,  
           'ябълка' => 12 ,  
           'лимон'  => 43 ,  
           'череша' => 9  ,  
           'ягода'  => 16 ,  
)
```

## Хешове - достъп до елементите, операции

```
# %pazar=('круша'=>35,'ябълка'=>12,'лимон'=>43,'череша'=>9,'ягода'=>16)
```

```
$pazar{'круша'}; # 35
```

```
$pazar{'ягода'}; # 16
```

Добавяне на елементи:

```
$pazar{'ананас'} = 8;
```

```
# %pazar=('круша'=>35,'ябълка'=>12,'лимон'=>43,'череша'=>9,'ягода'=>16,'ананас'=>8)
```

Изтриване на елементи:

```
delete $pazar{'круша'};
```

```
# %pazar=('ябълка'=>12,'лимон'=>43,'череша'=>9,'ягода'=>16,'ананас'=>8)
```

**keys** HASH - връща списък от ключовете в хеш

**values** HASH - връща списък от стойностите в хеш

```
print keys %pazar);  
print values %pazar;
```

Perl

# Оператори



## Аритметични оператори

Аритметичните оператори в Perl са аналогични на тези в C.

оператор	пример	резултат	описание
<b>**</b>	5 ** 3	125	повдигане на степен
<b>*</b>	5 * 3	15	умножение
<b>/</b>	5 / 3	1.666666666666667	делене
<b>%</b>	5 % 3	2	модул (остатък)
<b>+</b>	5 + 3	8	събиране
<b>-</b>	5 - 3	2	изваждане

# Оператори за низове

Конкатенация: `.`

```
print 'iva'.'' ''.''popova';      # <iva popova>
```

```
$foo = 'Привет '';
```

```
$bar = 'Свят'';
```

```
print $foo.'$bar;
```

Повторение: `x`

```
print 'iva' x 3;      # <ivaivaiva >
```

```
print '*' x 20;      # <*****>
```

## Оператор за присвояване: =

<лява\_част> = <дясна\_част>

Дясната част (rvalue) може да бъде константа, константен израз или израз, чиято стойност може да бъде изчислена (т.е ако участват променливи, то те да имат стойност).

Лявата част (lvalue) може да бъде всичко, което обозначава конкретно място от паметта: променлива, елемент от масив и пр.

*Семантика:* изчислената стойност на израза в дясната част се записва в мястото от паметта, обозначено от лявата част.

*Връщан резултат:* оператора връща (като lvalue, не като константа) лявата част:

```
$a = $b = $c = 0;    # $a=0, $b=0, $c=0
($b = 2) = 3;      # $b=3
```

## Логически оператори

оператор	пример	резултат	описание
<code>&amp;&amp;</code>	<code>\$a &amp;&amp; \$b</code>	<code>\$a</code> - ако <code>\$a</code> е лъжа <code>\$b</code> - ако <code>\$a</code> е истина	логическо 'И'
<code>  </code>	<code>\$a    \$b</code>	<code>\$a</code> - ако <code>\$a</code> е истина <code>\$b</code> - ако <code>\$a</code> е лъжа	логическо 'ИЛИ'
<code>!</code>	<code>! \$a</code>	<i>истина</i> - ако <code>\$a</code> е лъжа <i>лъжа</i> - ако <code>\$a</code> е истина	логическо 'НЕ'
<code>and</code>	<code>\$a and \$b</code>	<code>\$a</code> - ако <code>\$a</code> е лъжа <code>\$b</code> - ако <code>\$a</code> е истина	логическо 'И'
<code>or</code>	<code>\$a or \$b</code>	<code>\$a</code> - ако <code>\$a</code> е истина <code>\$b</code> - ако <code>\$a</code> е лъжа	логическо 'ИЛИ'
<code>not</code>	<code>not \$a</code>	<i>истина</i> - ако <code>\$a</code> е лъжа <i>лъжа</i> - ако <code>\$a</code> е истина	логическо 'НЕ'

## Оператори за сравнение

ЧИСЛОВИ	НИЗОВИ	значение
>	gt	по-голямо
>=	ge	по-голямо или равно
<	lt	по-малко
<=	le	по-малко или равно
==	eq	равно
!=	ne	различно
<=>	cmp	сравнение с цифров резултат,

## Низове и числа

### Преобразуване на низ в число:

- ако низа е изцяло съставен от цифри, то той се преобразува до съответстващото му число:

```
print "100" + 1;      # 100+1 => <101>
```

- ако низа не съдържа цифри, то той се преобразува до нула:

```
print "iva" + 1;     # 0+1 => <1>
```

- ако низа започва с цифри, а след това продължава с не-цифрови символи, то тези цифри формират числото, а след тях се спира преобразуването, дори и след това отново да се срещат цифри:

```
print "79x11" + 1;   # 79+1 => <80>
```

### Кога се извършва автоматично преобразуване:

- когато низа участва като операнд в аритметичен оператор или в числов оператор за сравнение.

```
$pass = 'abracadabra';  
$new_pass = 'sezam';  
$res = $pass == $new_pass;  
print $res;
```