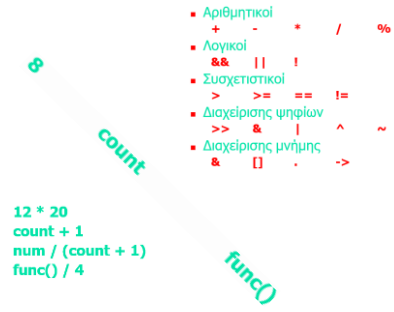
Εκφράσεις Τελεστές

Εισαγωγή στον Προγραμματισμό

(CEID_NY131)

Expressions Operators



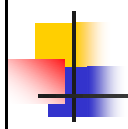
Kleanthis Thramboulidis
Prof. of Software and System Engineering
University of Patras
https://sites.google.com/site/thramboulidiskleanthis/

Έκφραση (Expression)

- **Είναι**
  - ένα τμήμα πηγαίου κώδικα που παράγει μια προσωρινή τιμή
- **αποτελείται**
  - από ένα συνδυασμό **τελεστέων** και **τελεστών** (optional)
- **Έχει ως δομικά στοιχεία**
  - σταθερές,
  - μεταβλητές, και
  - κλήσεις συναρτήσεων.

Τελεστής

- **είναι**
  - ένα **σύμβολο** ή μια **λέξη** της γλώσσας προγραμματισμού.
- **αναπαριστά**
  - συγκεκριμένη διεργασία, που εκτελείται πάνω σε ένα ή περισσότερα δεδομένα (τελεστέους - operands)
- **χρησιμοποιείται**
  - για τον σχηματισμό εκφράσεων (expressions).



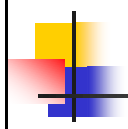
Παραδείγματα τελεστών

- Ο τελεστής *****,
  - όταν εφαρμόζεται σε δύο τελεστέους, αναπαριστά την διεργασία του πολλαπλασιασμού **num1 * num3**
  - Όταν εφαρμόζεται σε ένα τελεστέο που είναι δείκτης μας επιτρέπει να αναφερθούμε στο περιεχόμενο που δείχνει ο δείκτης ***num_ptr**
- Ο τελεστής **&**
  - εφαρμόζεται σε μία μεταβλητή και η προκύπτουσα έκφραση έχει ως τιμή την διεύθυνση μνήμης στην οποία αναφέρεται η μεταβλητή

    &num1

Κατηγορίες Τελεστών 1/2

- ανάλογα με τον αριθμό των τελεστέων στους οποίους δρουν, σε
  - μοναδιαίους (unary)
    π.χ. **-** , **(cast)**
  - δυαδικούς (binary)
    π.χ. **+**, **-**
  - τριαδικούς (ternary)
    π.χ. **? :**



Κατηγορίες Τελεστών 2/2

- ανάλογα με την διεργασία που εκτελούν
  - Αριθμητικοί
    + - * / %
  - Λογικοί
    && || !
  - Συσχετιστικοί
    > >= == !=
  - Διαχείρισης ψηφίων
    >> & | ^ ~
  - Διαχείρισης μνήμης
    & [] . ->

https://www.w3schools.com/c/c_operators.php

Απλές – Σύνθετες Εκφράσεις

- **απλές εκφράσεις**
  Τα δομικά στοιχεία (τελεστέοι) αποτελούν από μόνα τους εκφράσεις

Έκφραση → **8**
τιμή της έκφρασης → 8

count
Η τιμή της μεταβλητής

func()
Η επιστρεφόμενη τιμή της συνάρτησης

- **σύνθετες εκφράσεις**
  Σχηματίζονται συνδυάζοντας τα δομικά στοιχεία με τους τελεστές

```
12 * 20
count + 1
num / (count + 1)
func() / 4
```

Περιγράψτε την τιμή της έκφρασης

Συμβολισμοί σχηματισμού εκφράσεων

- σημειολογία **ένθεσης** ή ένθετου τελεστή (infix notation)
  y * y - 2 * x * z
- σημειολογία **επίθεσης** ή παρελκόμενου τελεστή (postfix notation)
  y y * 2 x * z * -
- σημειολογία **πρόθεσης** ή προπορευόμενου τελεστή (prefix notation)
  - * y y * * 2 x z

Τιμή έκφρασης

- **κάθε έκφραση έχει μια τιμή**

έκφραση	τιμή
-4 + 6	2
c=3+8	11
5>3	true (1)
8<1	false (0)
6 + (c=3+8)	17
(5> 1) && (6>7)	?

Υπολογισμός απλής έκφρασης 1/2

εκφρ1 ***τελεστής*** *εκφρ2*

- **εφαρμοστική σειρά** (applicative order)
  - Υπολογίζονται οι εκφρ1 και εκφρ2
  - Εφαρμόζεται ο τελεστής

Υπολογισμός απλής έκφρασης 2/2

εκφρ1 ***τελεστής*** *εκφρ2*

- **υπολογισμός περιορισμένης έκτασης** (short circuit evaluation)

  εκφρ1 ***and*** *εκφρ2* — **Δεν υπολογίζεται αν εκφρ1 false**

  εκφρ1 ***or*** *εκφρ2* — **Δεν υπολογίζεται αν εκφρ1 true**

Υπολογισμός σύνθετης έκφρασης 1/2

Πως υπολογίζω την τιμή;

y * y - 2 * x * z

- Για τον υπολογισμό της τιμής η γλώσσα ορίζει την **Προτεραιότητα** και **Προσεταιριστικότητα** στην εφαρμογή των τελεστών

C Operator Precedence Table

Προτεραιότητα - Προσεταιριστικότητα τελεστών

count = x + y – k *2;

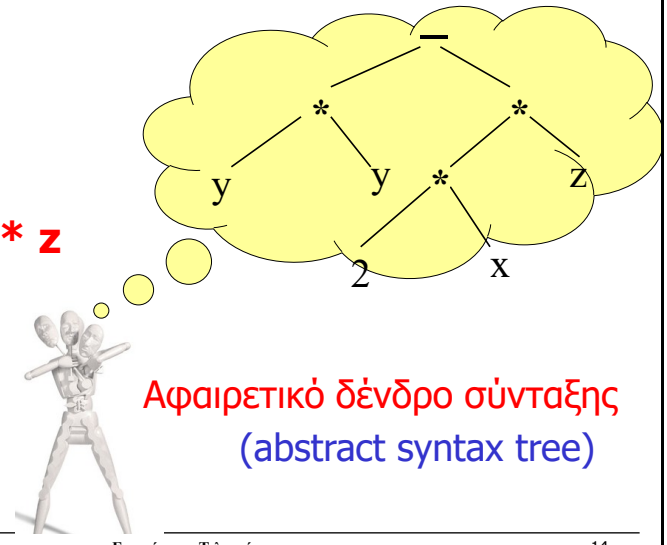
Precedence	Symbol	Name	Associativity
3	*	multiplication	left to right
	/	division	
	%	modulus(remainder)	
4	+	addition	left to right
	-	subtraction	

num1 = num2 = 12;

Precedence	Symbol	Name	Associativity
14	=	plain assignment	right to left

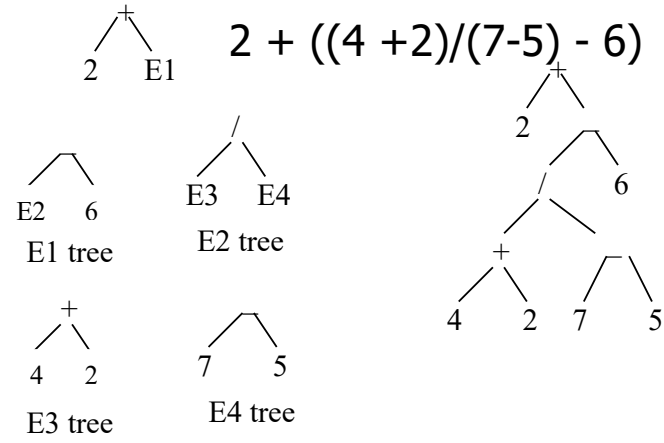
Υπολογισμός σύνθετης έκφρασης 2/2

y * y - 2 * x * z



Αφαιρετικό δένδρο σύνταξης
(abstract syntax tree)

Παράδειγμα



Τελεστές της C

- μοναδιαίοι **+** και **-** τελεστές
- δυαδικοί αριθμητικοί τελεστές **+ - * / %**
- αριθμητικοί τελεστές ανάθεσης **= += -= *=** ...
- αύξησης, ελάττωσης **++ --**
- λογικοί τελεστές **&& || !**
- συσχετιστικοί (relational) **< > <= >= == !=**
- bit-manipulation **>> << & | ^ ~**
- memory operators **& * [] . ->**

Increment-Decrement Operators

- προθεματικός

  num = 18 * **++count**;

- μεταθεματικός

  num = 18 * **count++**;

~~(x*y)++~~

Τιμή έκφρασης = **?**
Τιμή count = **?**

```
while( ++count<18.5);
while(count++<18.5);
```


Τελεστές Αντικατάστασης

εκφρ1 **=** (εκφρ1) **τελ** (εκφρ2)

εκφρ1 **τελ =** εκφρ2

i = i + k ----▸ **i += k**

i = i * k ----▸ **i *= k**

Δίνουν συνήθως πιο ευανάγνωστο και αποτελεσματικό κώδικα

m = (m + ((n+x)-y)) ----▸ **m += n+x-y**

Conditional Operator ? :

- σύντομος τρόπος έκφρασης της απλής μορφής πρότασης if else
- αποτελεί το μόνο ternary operator

εκφρ1 **?** εκφρ2 **:** εκφρ3

```
if (a>b)
   max = a;
else
   max = b;
```

```
max = (a>b) ? a : b;
```


Relational operators (Συσχετιστικοί τελεστές)

< > <= >= == !=

```
x += (y>=n)
```

```
((++j) == m) !=
   (y * 2)
```

```
++j == m != y * 2
```

Προσοχή στην έκφραση

```
if(count = MAX_COUNT){
```

Logical operators (Λογικοί τελεστές)

&&	AND
||	OR
!	NOT

Οι τελεστέοι μπορεί να είναι int και float

Τιμή έκφρασης = ?

```
x < y < z
```

```
if (a<b)
   if (b<c)
      πρόταση
```

```
if ((a<b) && (b<c))
 πρόταση
```


sizeof, cast και , (comma)

επιστρέφει τον αριθμό των bytes που η τιμή της **έκφρασης** ή ο **τύπος δεδομένων** καταλαμβάνει στην μνήμη

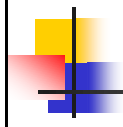
sizeof

Τελεστής μετατροπής τύπου ή cast operator

(float)num1

Τελεστής κόμμα (,)

```
for(i=0 , j=10 ; j - i >0 ; i++ , j++)
```



Τελεστής

- **είναι** ένα **σύμβολο** ή μια **λέξη** της γλώσσας προγραμματισμού που
- **αναπαριστά** συγκεκριμένη διεργασία,

- **Εφαρμόζεται** πάνω σε ένα ή περισσότερα δεδομένα (τελεστέους -operands) σχηματίζοντας απλές ή σύνθετες εκφράσεις (expressions).