Προγραμματισμός Η/Υ Ι

03.
Εκφράσεις, Τελεστές, Είσοδος / Έξοδος

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2019-2020 | ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.

Χαρακτηριστικά γλώσσας
Εκφράσεις (expressions)

- οι εκφράσεις αποτελούνται από τελεστές και τελεσταίους
  - οι τελεστές καθορίζουν τι πρέπει να γίνει
  - οι τελεσταίοι ορίζουν τα δεδομένα στα οποία εφαρμόζονται οι τελεστές

  π.χ. `length * width` → πολλαπλασιασμός ακεραίων `length` και `width`

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) – Παραδείγματα

```
1.  // Υπολογισμός εμβαδού και μέσου όρου
2.
3.  int main (void)
4.  {
5.     int length = 20;                    // η πιο απλή έκφραση, μία σταθερά (20)
6.     int width = 40;
7.     int area = length * width;          // ένας πολλαπλασιασμός
8.     int average = (length + width) / 2; // μία πρόσθεση και μία διαίρεση
9.  }
```

- ισχύουν οι συνηθισμένοι κανονισμοί προτεραιότητας πράξεων
  - π.χ. `a*b+c/d` = `(a*b)+(c/d)` (και όχι `a*(b+c)/d`)
  - εάν αμφιβάλλετε ή εάν είναι αρκετά πολύπλοκο → χρησιμοποιήστε παρενθέσεις!
- μη γράφετε παράλογα περίπλοκες εκφράσεις
  - π.χ. `a*b+c/d*(e-f/g)/h+7` ← πολύπλοκο!
- επιλέγετε ονόματα που έχουν νόημα!

Σταθερές

- τα προγράμματα χρησιμοποιούν πολλές σταθερές, π.χ. **`π:3.14159`**
- δεν πρέπει να αλλάζουμε κατά λάθος αυτές τις σταθερές
  1. χρήση σταθερών μεταβλητών (**`const`**)
  2. χρήση σταθερών εκφράσεων

✓ αποφυγή μαγικών σταθερών → χρήση σταθερών μεταβλητών ή εκφράσεων

☞ καλύτερη διατηρησιμότητα προγράμματος, π.χ. αύξηση ακρίβειας του **π** σε **`3.14159265359`**

Χαρακτηριστικά γλώσσας
Σταθερή μεταβλητή

- καθορίζεται μια φορά και χρησιμοποιείται όσο συχνά επιθυμούμε

- η σταθερή μεταβλητή δηλώνεται με τη λέξη **const**
  - με τη δήλωση της σταθερής μεταβλητής, πρέπει να της εκχωρηθεί και μια αρχική τιμή

```
1.  const double pi = 3.14159;     // σταθερή μεταβλητή
2.  pi = 7;                        // σφάλμα: ανάθεση τιμής σε σταθερή μεταβλητή
3.  v = 2*pi/r;                    // σωστό: μόλις διάβασες το pi, μην προσπαθήσεις να το αλλάξεις
```

- αν επιχειρήσουμε να αλλάξουμε το περιεχόμενο μιας σταθερής μεταβλητής θα εμφανιστεί μήνυμα λάθους στη μεταγλώττιση

Χαρακτηριστικά γλώσσας
Σταθερή έκφραση

έκφραση με τιμή που αποτελείται αποκλειστικά από σταθερές

- στη C μπορούμε να δώσουμε όνομα σε σταθερές με τη μακροεντολή **#define** του προεπεξεργαστή, π.χ.

```
#define PI 3.14159
```

  - ο προεπεξεργαστής θα αντικαταστήσει το όνομα με την τιμή, πριν τη μεταγλώττιση

```
1.  #define PI 3.14159
2.  main () {
3.     const int max = 17;        // σταθερή μεταβλητή
4.     int val = 19;              // μεταβλητή
5.     val + 2;                   // μη σταθερή έκφραση (χρησιμοποιεί μια μεταβλητή)
6.     PI + 2;                    // σταθερή έκφραση
7.  }
```

- σε κάποια σημεία, η C απαιτεί μία σταθερή έκφραση, π.χ. **switch**

Χαρακτηριστικά γλώσσας
Μακροεντολή #define

- π.χ. `#define PI 3.14159`
  - ο προεπεξεργαστής θα αντικαταστήσει το όνομα με την τιμή, πριν τη μεταγλώττιση

προσοχή:

- η εντολή αυτή δεν τελειώνει με ερωτηματικό ';'
- υπάρχει κενό μεταξύ του ονόματος `PI` και της σταθεράς `3.14159`
- γράφουμε την εντολή έξω από οποιαδήποτε συνάρτηση, στην αρχή του προγράμματος
  - π.χ. μαζί με τις μακροεντολές `#include` του προεπεξεργαστή

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) - Αριθμητικοί τελεστές

χρησιμοποιούνται για την εκτέλεση αριθμητικών πράξεων

- όλοι οι αριθμητικοί τελεστές είναι δυαδικοί → για κάθε τελεστή απαιτούνται δύο τελεσταίοι
  - εξαίρεση αποτελούν τα σύμβολα της πρόσθεσης (+) και της αφαίρεσης (-) που μπορούν να είναι και μοναδιαίοι τελεστές
    - εκφράζουν το πρόσημο
  - οι τελεσταίοι μπορεί να είναι σταθερές ή μεταβλητές

τελεστής	όνομα/περιγραφή
+	πρόσθεση
-	αφαίρεση
*	πολλαπλασιαμός
/	διαίρεση
%	υπόλοιπο

```
1.  ilikia = etos1 - etos2;
2.  printf("%d", 95-55);
3.  celsious=(-25);
4.  a = 2*3+4;
5.  b = (5-c)/d;
6.  result = -(b+2)*a + (c +3*(a-b));
```

- μεταξύ τελεστών και τελεσταίων τα κενά διαστήματα δεν παίζουν ρόλο
- μπορείτε να χρησιμοποιήστε παρενθέσεις για μαθηματικές παραστάσεις
- ο τελεστής % μας δίνει το υπόλοιπο (καλείται «modulo») της ακέραιας διαίρεσης
  - εφαρμόζεται μόνο σε ακεραίους

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) - Αριθμητικοί τελεστές - Προτεραιότητα πράξεων

η C ακολουθεί την αλγεβρική προτεραιότητα στους αριθμητικούς τελεστές

1. μεγαλύτερη προτεραιότητα έχουν οι παρενθέσεις
2. μετά οι τελεστές προσήμου (με ίδια προτεραιότητα μεταξύ τους),
3. ακολουθούν οι *, / και % (με ίδια προτεραιότητα μεταξύ τους) και
4. τέλος οι + και − (με ίδια προτεραιότητα μεταξύ τους)

```
a = 2*3+4          // θα δώσει στην a την τιμή 10
```

- με τις παρενθέσεις μπορούμε να αλλάξουμε τον τρόπο υπολογισμού μιας έκφρασης

```
a = 2*(3+4)        // θα δώσει στην a την τιμή 14
```

- σε δυαδικούς τελεστές με ίδια προτεραιότητα, η φορά των πράξεων είναι από αριστερά προς δεξιά

```
b = 12/4*3         // θα δώσει στη b την τιμή 9 (και όχι 1)
```

✍ ο τελεστής εκχώρησης (=) έχει μικρότερη προτεραιότητα από τους αριθμητικούς τελεστές

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) - Αριθμητικοί τελεστές - Ο τελεστής υπόλοιπο % («modulo»)

```
a = 20 % 4          // δίνει στην alpha την τιμή 0
a = 20 % 3          // δίνει στην alpha την τιμή 2
a =  8 % 7          // δίνει στην alpha την τιμή 1
a =  8 % 8          // δίνει στην alpha την τιμή 0
a =  4 % 5          // δίνει στην alpha την τιμή 4
```

? χρησιμοποιώντας τον τελεστή % πώς μπορώ να καταλάβω εάν ένας ακέραιος αριθμός b είναι περιττός ή άρτιος;

- εκτελώντας b % 2 το αποτέλεσμα μπορεί να είναι είτε 0 είτε 1

  ? *γιατί;*

- εάν b % 2 έχεις ως αποτέλεσμα 0 → b είναι άρτιος
- εάν b % 2 έχεις ως αποτέλεσμα 1 → b είναι περιττός

```
1.  // Ισοδύναμος κώδικας σε C
2.  if (b % 2 == 0)
3.    printf("b is odd\n");
4.  else
5.    printf("b is even\n");
```

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) – «Συνοπτικοί» αριθμητικοί τελεστές

- για πολλούς αριθμητικούς τελεστές, υπάρχουν ισοδύναμοι πιο «συνοπτικοί »
  - `a += c` ⟷ `a = a+c`
  - `a *= scale` ⟷ `a = a*scale`
  - `++a` ⟷ `a += 1` ⟷ `a = a+1`

- γενικά, για ένα δυαδικό τελεστή op ισχύει: `a op= b` ⟷ `a = a op b`
  - ☞ `a+=b`, `a-=b`, `a*=b`, `a/=b`, `a%=b`

✍ προτιμούμε κατά κανόνα να χρησιμοποιούμε «συνοπτικούς» τελεστές → είναι πιο σαφής, εκφράζουν πιο άμεσα μία ιδέα

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) – Μετατροπές

```
1.  5/2                                  // 2, όχι 2.5
2.  5.0/2                                // 2.5, διότι σημαίνει 5.0/(double)2
3.
4.  double d = 2.5;
5.  int i = 2;
6.  double d2 = d/i;                     // d2 == 1.25
7.  int i2 = d/i;                        // i2 == 1
8.
9.  // μετατροπή βαθμών κελσίου σε φαρενάιτ
10. double dc = 10;
11. double df = 9/5 * dc + 32;           // df == 42
12. double df2 = 9.0/5 * dc + 32;        // df2 == 50
```

- μπορούμε να συνδυάσουμε διαφορετικούς τύπους σε εκφράσεις, όμως πρέπει να δίνουμε μεγάλη προσοχή στις μετατροπές τύπων → ώστε να γράφουμε σωστές εκφράσεις

Χαρακτηριστικά γλώσσας
Εντολή (statement)

1. μια έκφραση που ακολουθείται από ένα ερωτηματικό
   - `a = b;`
2. ένας ορισμός
   - `double d2 = 2.5;`
   - `int average = (length * width) / 2;`
3. μία "εντολή ελέγχου" που καθορίζει τη ροή του ελέγχου
   - `if (x == 2) y = 4;`
   - `while (i < 10) i++;`
   - `return x;`

❖ εάν δεν καταλαβαίνετε κάποια από τα παραπάνω, σύντομα θα τα καταλάβετε ...

Χαρακτηριστικά γλώσσας
Εντολή (statement) II

- από μία εντολή ζητάμε να έχει κάποιο αποτέλεσμα

```
1.   // παράδειγμα εντολών χωρίς αποτέλεσμα
2.
3.   1 + 2;        // εκτέλεσε μία πρόσθεση, αλλά μη χρησιμοποιήσεις το άθροισμα
4.   a * b;        // εκτέλεσε έναν πολλαπλασιασμό, αλλά μη χρησιμοποιήσεις το γινόμενο
```

  - εντολές χωρίς αποτέλεσμα είναι συνήθως λογικά σφάλματα και οι μεταγλωττιστές συχνά μας προειδοποιούν

- κενή εντολή

```
1.   // παράδειγμα (πιθανής) λανθασμένης χρήσης της κενής εντολής
2.
3.   int x = 0, y = 0;
4.   if (x == 5);
5.   { y = 3; }
                          // το y παίρνει την τιμή 3 ανεξαρτήτως της τιμής του x
```

  - πρέπει να προσέχουμε κατά τη χρήση (ή αποφυγή) των κενών εντολών

Έξοδος

στην οθόνη του τερματικού

Έξοδος
στην οθόνη του τερματικού - printf ()

- η γενική μορφή εκτέλεσης της συνάρτησης **printf()** είναι:

```
printf(ΣΕΙΡΑ_ΕΛΕΓΧΟΥ, στοιχείο-1, στοιχείο-2, …, στοιχείο-ν);
```

- η **ΣΕΙΡΑ_ΕΛΕΓΧΟΥ**
  - περικλείεται μέσα σε λατινικά διπλά εισαγωγικά (")
  - καθορίζει τον τρόπο εμφάνισης των δεδομένων
  - μπορεί να περιλαμβάνει
    1. επεξηγηματικό κείμενο, οτιδήποτε περιγράφει το αποτέλεσμα

```
printf("Hello World!");
```

    2. προσδιοριστές, μπαίνουν απαραίτητα μέσα στην **ΣΕΙΡΑ_ΕΛΕΓΧΟΥ**, μόνο όταν θέλουμε να εμφανίσουμε κάποιο στοιχείο, π.χ. μεταβλητή, ή σταθερά/τιμή, ή (σταθερή) έκφραση

```
printf("Hello World! My age is: %d", my_age);
```

    3. χαρακτήρες ελέγχου, που τους χρησιμοποιούμε για τη διαμόρφωση της εμφάνισης

```
printf("Hello World!\n");
```

Έξοδος
στην οθόνη του τερματικού - printf () II

- η γενική μορφή εκτέλεσης της συνάρτησης **printf()** είναι:

```
printf(ΣΕΙΡΑ_ΕΛΕΓΧΟΥ, στοιχείο-1, στοιχείο-2, …, στοιχείο-v);
```

  - κάθε στοιχείο-i, όπου i=1,2,…,v, μπορεί να είναι:
    1. όνομα μεταβλητής

```
printf("I am %d years old\n", my_age);
```

    2. σταθερά/τιμή

```
printf("I am %d years old\n", 25);
```

    3. (σταθερή) έκφραση

```
printf("I am %d years old\n", 2019 - 1994);
printf("I am %d years old\n", 2019 - birth_year);
```

- μετά την επιτυχή εκτέλεσή της, η **printf()** επιστρέφει τον αριθμό των χαρακτήρων που εμφάνισε στην οθόνη του τερματικού

Έξοδος

στην οθόνη του τερματικού - printf () - Περισσότερα παραδείγματα

```
1. printf("My name is: %s and I am %d years old\n", my_name, my_age);

2. printf("My name is: %s and I am %d years old\n", "NIKOS", 25);

3. printf("My name is: %s and I am %d years old\n", my_name, 2019 - 1994);

4. printf("My name is: %s and I am %d years old\n", my_name, 2019 - birth_year);

5. printf("My name is: %s.\nI am %d years old.\n", my_name, 2019 - birth_year);
```

Έξοδος
στην οθόνη του τερματικού - printf () - Προσδιοριστές

οι προσδιοριστές των στοιχείων

- έχουν σχέση με τον τύπο των δεδομένων που εμφανίζονται στην **printf()**
- αρχίζουν με το χαρακτήρα **%**

προσδιοριστής	τύπος στοιχείου	Παράδειγμα
%d ή %i	ακέραιος (int)	**392**
%c	χαρακτήρας (char)	**a**
%s	σειρά χαρακτήρων (συμβολοσειρά ή char *)	**Hello World**
%f ή %F	κινητής υποδιαστολής (float ή double)	**392.65 ή 392.65**
%e ή %E	κινητής υποδιαστολής (float ή double) σε εκθετική μορφή	**3.9265e+2 ή 3.9265E+2**
%g ή %G	κινητής υποδιαστολής όπως το %e ή %f (όποιο είναι μικρότερο)	**392.65 ή 392.65**
%u	Ακέραιος χωρίς πρόσημο (unsigned int)	**7235**
%p	Χρησιμοποιείται για την εμφάνιση διέθυνσης μνήμης	**b8000000**
%o	Ακέραιος χωρίς πρόσημο σε οκταδικό σύστημα	**610**
%x ή %X	Ακέραιος χωρίς πρόσημο σε δεκαεξαδικό σύστημα	**7fa ή 7FA**

Έξοδος

στην οθόνη του τερματικού - printf () - Προσδιοριστές II

- ο μεταγλωττιστής αντιστοιχίζει ένα-προς-ένα, από αριστερά προς τα δεξιά, τα στοιχεία με τους προσδιοριστές
  - αν οι προσδιοριστές είναι περισσότεροι από τα στοιχεία της **`printf()`**, τότε για τα επιπλέον προσδιοριστικά εμφανίζονται τυχαίες τιμές
  - αν οι προσδιοριστές λιγότεροι από τα στοιχεία της **`printf()`**, τότε δεν εμφανίζονται οι τιμές των επιπλέον στοιχείων

Έξοδος

στην οθόνη του τερματικού - printf () - Προσδιοριστές - Δυνατότητες Εμφάνισης

- στην εμφάνιση μιας τιμής μπορούμε να καθορίσουμε
  1. το συνολικό πλήθος των χαρακτήρων
  2. τα ψηφία ακρίβειας
  3. την υποδιαστολή

- **εξ' ορισμού (by default) εμφανίζονται 6 δεκαδικά ψηφία μετά την υποδιαστολή**
- **μπορούμε όμως να προσδιορίσουμε όσα θέλουμε**
- **μία πραγματική τιμή στρογγυλοποιείται (προς τα πάνω ή προς τα κάτω)**

παράδειγμα	τύπος στοιχείου
`%6d`	ακέραιος, όπου το **εύρος** του είναι 6 ψηφία (χαρακτήρες)
`%7.2f`	κινητής υποδιαστολής, όπου το **εύρος** του είναι 7 χαρακτήρες με 2 ψηφία **μετά** την υποδιαστολή (→ 4 χαρακτήρες για το **ακέραιο** μέρος, 1 χαρακτήρα για την **υποδιαστολή** και 2 χαρακτήρες για την **ακρίβεια**)
`%+d`	εμφανίζει το **πρόσημο**
`%-d`	**στοιχίζει** αριστερά το αποτέλεσμα
`%08d`	**γεμίζει** με μηδέν τα κενά που είναι άδεια μπροστά από το αποτέλεσμα
`%%`	**εμφανίζει** το χαρακτήρα '**%**'

Έξοδος

στην οθόνη του τερματικού - printf () - Χαρακτήρες ελέγχου

χαρακτήρας	περιγραφή
\a	ηχητικό σήμα (<BELL>)
\b	ο χαρακτήρας <BACKSPACE> (διάστημα πίσω)
\f	ο χαρακτήρας νέας σελίδας (<FORM FEED>)
\n	νέας γραμμής (<LINE FEED>)
\r	επιστροφής (<CR>)
\t	οριζοντίου προκαθορισμένου διαστήματος (<TAB>)
\v	κατακορύφου διαστήματος(<VTAB>)
\'	εμφάνιση του απλού εισαγωγικού
\"	εμφάνιση του διπλόυ εισαγωγικού
****	εμφάνιση της ανάποδης πλαγίας καθέτου
\?	εμφάνιση του λατινικού ερωτηματικού
\xhhh	εμφάνιση του χαρακτήρα **hhh** (όπου το **hhh** είναι σε δεκαεξαδική μορφή)
\ooo	εμφάνιση του χαρακτήρα **ooo** (όπου το **ooo** είναι σε οκταδική μορφή)

- όποιος χαρακτήρας ακολουθεί τον χαρακτήρα "\" τότε **επισημαίνεται** στο μεταγλωττιστή ότι (αυτός ο δεύτερος χαρακτήρας) έχει **ειδική** σημασία
- οι **συνδυασμοί** αυτών των δύο χαρακτήρων λαμβάνονται υπόψη σαν **ένας** χαρακτήρας και χρησιμοποιούνται για το **συμβολισμό** των **χαρακτήρων ελέγχου**

Έξοδος

στην οθόνη του τερματικού - printf () - Παραδείγματα

```
1. index = 5;
2. printf("%d", index);                                    // 5
```

```
1. index = 5;
2. printf("The value of variable index is: %d", index);    // The value of variable index is: 5
```

```
1. index = 5;
2. printf("The value of variable index is:\n%d", index);   // The value of variable index is:
3.                                                         // 5
```

Έξοδος

στην οθόνη του τερματικού - printf () - Παραδείγματα II

```
1. x = 5;
2. y = 3;
3. printf("%d + %d = %d", x, y, x+y );                    // 5 + 3 = 8
```

```
1. x = 5;
2. y = 3;
3. printf("x = %d\n y = %d\n x + y = %d", x, y, x+y );    // x = 5
4.                                                        // y = 3
5.                                                        // x + y = 8
```

Έξοδος

στην οθόνη του τερματικού - printf () - Παράδειγμα με προσδιοριστές

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int int_num;
5.      double float_num;
6.      int_num = 123;
7.      float_num = 123.456789;
8.      printf(“Integer format                                  : %d\n", int_num);
9.      printf("Filled with spaces                              : %8d\n", int_num);
10.     printf("Filled with zeros                               : %08d\n", int_num);
11.     printf(“Float format                                    : %f\n", float_num);
12.     printf("With precision specifier                        : %10.2f\n", float_num);
13.     printf("With precision specifier and left alignment: %-10.2f\n", float_num);
14. }
```

αποτέλεσμα εκτέλεσης

```
Integer format                                  : 123
Filled with spaces                              :      123
Filled with zeros                               : 00000123
Float format                                    : 123.456789
With precision specifier                        :     123.46
With precision specifier and left alignment: 123.46
```

Έξοδος

στην οθόνη του τερματικού - printf () - Παράδειγμα με προσδιοριστές II

```
1.  #include <stdio.h>
2.  main() {
3.      int i = 123;
4.      double f = 3.1415926535;
5.      printf("i = %i\n", i);
6.      printf("i = %o\n", i);
7.      printf("i = %x\n", i);
8.      printf("i = %X\n", i);
9.      printf("i = %+i\n", i);
10.     printf("i = %8i\n", i);
11.     printf("i = %08i\n", i);
12.     printf("i = %+08i\n", i);
13.     printf("f = %f\n", f);
14.     printf("f = %10.3f\n", f);
15.     printf("f = %+10.3f\n", f);
16.     printf("f = %g\n", f);
17.     printf("f = %10.6g\n", f);
18.     printf("f = %10.6e\n", f);
19. }
```

αποτέλεσμα εκτέλεσης

```
i = 123
i = 173
i = 7b
i = 7B
i = +123
i =      123
i = 00000123
i = +0000123
f = 3.141593
f =      3.142
f =     +3.142
f = 3.14159
f =    3.14159
f = 3.141593e+000
```

Έξοδος

στην οθόνη του τερματικού - printf () - Παράδειγμα με προσδιοριστές III

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int num1, num2, num3, num4, num5;
5.      num1 = 1;
6.      num2 = 12;
7.      num3 = 123;
8.      num4 = 1234;
9.      num5 = 12345;
10.     printf("%8d %-8d\n", num1, num1);
11.     printf("%8d %-8d\n", num2, num2);
12.     printf("%8d %-8d\n", num3, num3);
13.     printf("%8d %-8d\n", num4, num4);
14.     printf("%8d %-8d\n", num5, num5);
15. }
```

αποτέλεσμα εκτέλεσης

```
       1 1
      12 12
     123 123
    1234 1234
   12345 12345
```

Είσοδος

από το πληκτρολόγιο

Είσοδος
από το πληκτρολόγιο - scanf()

- διαβάζει από το πληκτρολόγιο (`stdin`) μορφοποιημένες τιμές μεταβλητών
- η γενική μορφή εκτέλεσης της συνάρτησης **`scanf()`** είναι:

```
scanf(ΣΕΙΡΑ_ΕΛΕΓΧΟΥ, διεύθυνση-1, διεύθυνση-2, …, διεύθυνση-ν);
```

- η **`scanf()`** είναι παρόμοια πολύ με την **`printf()`** με τις εξής διαφορές:
  1. η **`ΣΕΙΡΑ_ΕΛΕΓΧΟΥ`** στη **`scanf()`** περιέχει προσδιοριστές και κενά
     - ✗ όχι επεξηγηματικό κείμενο
     - ✓ μπορείτε να βάλετε χαρακτήρες ελέγχου
  2. οι προσδιοριστές αντιστοιχούν σε διευθύνσεις μεταβλητών
  3. για μεταβλητές τύπου `double`, ο προσδιοριστής είναι: **`%lf`**

✍ σημαντική διαφορά: η συνάρτηση **`scanf()`** δεν αναφέρεται σε ονομασίες μεταβλητών, αλλά σε **διευθύνσεις μεταβλητών**

Χώροι μνήμης
Μεταβλητές - Δέσμευση χώρου

διαφάνεια από παλαιότερη διάλεξη

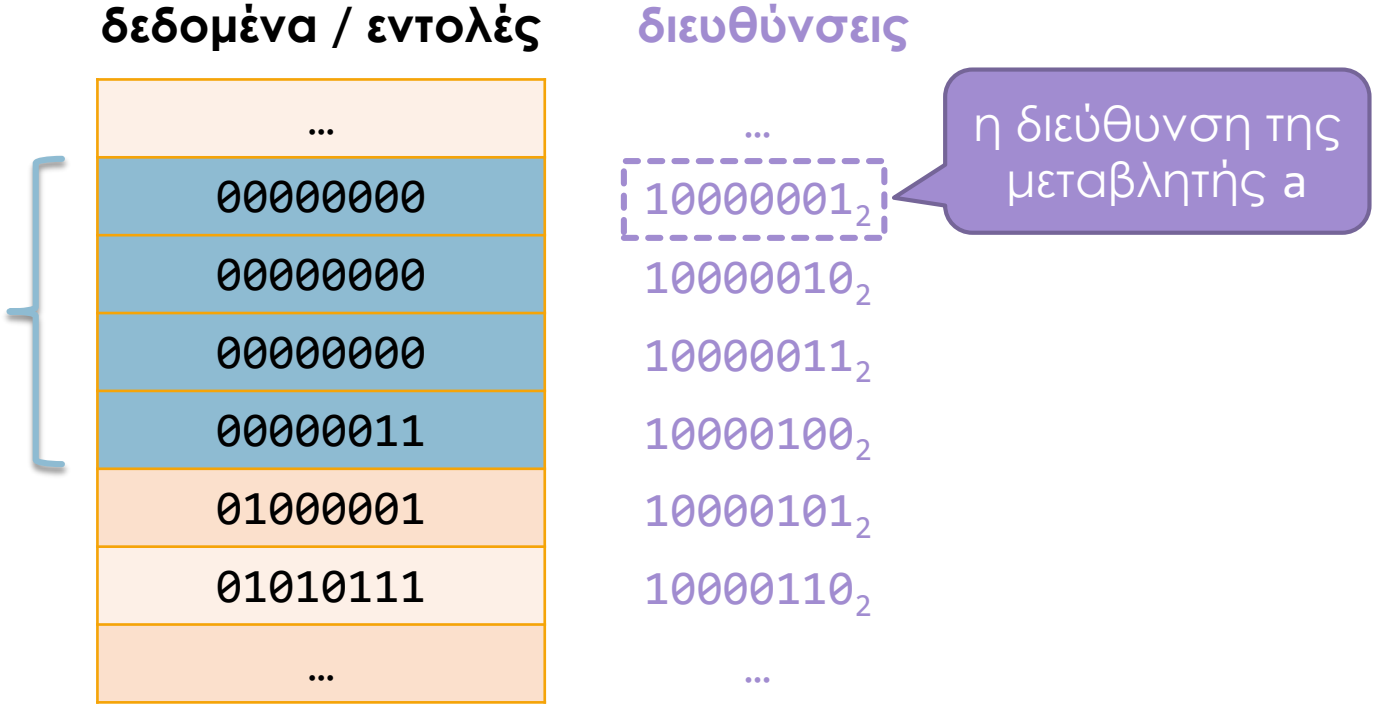
η δήλωση μιας μεταβλητής αποτελεί:

1. τη δέσμευση χώρου μνήμης για τη μεταβλητή
2. την απόδοση του ονόματος της μεταβλητής σε αυτό το χώρο

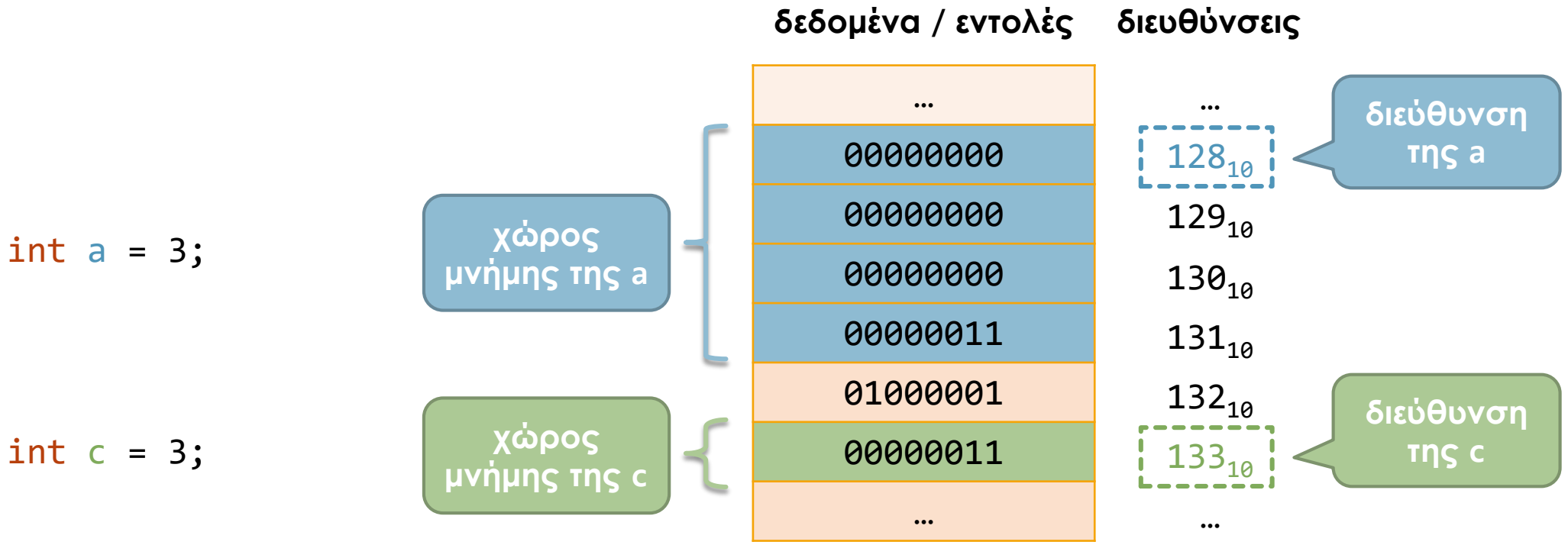
παράδειγμα:

```
int a = 3;
```

ο χώρος μνήμης της μεταβλητής a



Διευθύνσεις μεταβλητών



- ο τελεστής **&** επιστρέφει τη διεύθυνση μιας μεταβλητής
  - `&a` `// η διεύθυνση της μεταβλητής a → δηλαδή η τιμή` 128_{10}
  - `&c` `// η διεύθυνση της μεταβλητής c → δηλαδή η τιμή` 133_{10}

Είσοδος

από το πληκτρολόγιο - scanf() - Ανάγνωση ακεραίου

```
1. // ανάγνωση ακεραίου αριθμού από το πληκτρολόγιο και εκτύπωσή του στην οθόνη του τερματικού
2. int x;                                          // δήλωση ακέραιας μεταβλητής x
3. printf("Enter a number (and press ENTER): ");   // επεξηγηματικό μήνυμα
4. scanf("%d", &x);                                // ανάγνωση ακεραίου από το πληκτρολόγιο
5. printf("You entered number %d \n", x);          // εκτύπωση ακεραίου στην οθόνη
```

- είναι επιθυμητό πριν από κάθε κλήση της **scanf()** να προηγείται μία κλήση της **printf()**, με κατάλληλο επεξηγηματικό κείμενο → ώστε ο χρήστης γνωρίζει τι δεδομένο πρέπει να πληκτρολογήσει
- μετά την εισαγωγή των δεδομένων δίνουμε <ENTER>

☞ πολύ συχνό λάθος: απουσία **&**

- ✗ δηλαδή: `scanf("%d", x);` ← **ΛΑΘΟΣ!**
- ✓ αντί του: `scanf("%d", &x);` ← **ΣΩΣΤΟ!**

Είσοδος

από το πληκτρολόγιο - scanf() - Παραδείγματα

```
1. int x, y;
2. printf("Enter the first number (and press enter): ");
3. scanf("%d", &x);
4. printf("Enter the second number (and press enter): ");
5. scanf("%d", &y);
```

```
1. int x, y;
2. printf("Enter two numbers (with space in between and press enter): ");
3. scanf("%d %d", &x, &y);
```

όταν εισάγονται πολλαπλά δεδομένα αυτά θα πρέπει να έχουν μεταξύ τους τουλάχιστον 1 κενό (space)

Είσοδος

από το πληκτρολόγιο - scanf() - Παραδείγματα II

```
1. float a;                                          // δήλωση πραγματικής μεταβλητής a
2. printf("Enter a real number (and press enter): ");
3. scanf("%f", &a);                          // ανάγνωση πραγματικού αριθμού από το πληκτρολόγιο
4. printf(" You entered number %f \n", a);           // εκτύπωση αριθμού στην οθόνη
```

```
1. double d;                                         // δήλωση πραγματικής μεταβλητής d
2. printf("Enter a real number (and press enter): ");
3. scanf("%lf", &d);                         // ανάγνωση πραγματικού αριθμού από το πληκτρολόγιο
4. printf(" You entered number %f \n", d);               // εκτύπωση αριθμού στην οθόνη
```

Για μεταβλητές τύπου double, ο προσδιοριστής είναι: %lf

Είσοδος
από το πληκτρολόγιο - scanf() - επιστρεφόμενη τιμή

- η **scanf()** έχει κάποια τιμή επιστροφής που είναι, κατά περίπτωση:
  - εάν ολοκληρωθεί με επιτυχία, επιστρέφει το πλήθος των όρων που διάβασε
  - εάν αποτύχει να διαβάσει, επιστρέφει μηδέν (0)
  - εάν φτάσει απροσδόκητα στο τέλος αρχείου, επιτρέφει την τιμή του **EOF** (End Of File)
    - καθώς θα περίμενε να υπάρχουν περισσότερες τιμές, βάσει των καθορισμένων προσδιοριστών στη **ΣΕΙΡΑ_ΕΛΕΓΧΟΥ**

```
1. int i, n;
2. float x;
3. n = scanf("%d %f", &i, &x);
```

  - εάν από το πληκτρολόγιο εισάγουμε τις τιμές: **25 54.32**
    - η μεταβλητή **i** θα πάρει την τιμή **25**
    - η μεταβλητή **x** θα πάρει την τιμή **54.32**
    - η μεταβλητή **n** θα πάρει την τιμή **2**

Είσοδος

από το πληκτρολόγιο - scanf() - Ολοκληρωμένο παράδειγμα

```
1.  #include <stdio.h>
2.  main()
3.  {
4.      int index;
5.      float num;
6.      printf("Give me the integer index (and press enter): ");
7.      scanf("%d", &index);
8.      printf("Give me the float num (and press enter): ");
9.      scanf("%f", &num);
10.     printf("The value of index is : %d\n", index);
11.     printf("The value of num is : %f\n", num);
12.     num = 45.764;
13.     printf("The new value of num is : %f\n", num);
14. }
```

Είσοδος

από το πληκτρολόγιο - scanf() - Αγορά κινητού

```c
#include <stdio.h>
#define FPA 0.24
main()
{
    int quantity;
    float price, total;
    printf("Enter the price of cellular phone (and press enter): ");
    scanf("%f", &price);
    printf("Enter the quantity (and press enter): ");
    scanf("%d", &quantity);
    total = quantity*price;
    total += total*FPA;
    printf("Total price: %f\n", total);
}
```

Είσοδος/Έξοδος
Βιβλιοθήκη stdio.h - Χαρακτήρες

- γνωρίζουμε ήδη με χρήση των **scanf()** και **printf()**
- **getchar()**
  - διαβάζει έναν χαρακτήρα
  - δεν δέχεται παραμέτρους
- **putchar(<χαρακτήρας>)**
  - εμφανίζει στην οθόνη ένα μόνο χαρακτήρα
  - δέχεται το χαρακτήρα ως παράμετρο

```
1. char c;
2. c = getchar();
```

```
// ισοδύναμο με:
scanf("%c", &c);
```

```
3. putchar(c);
4. putchar(getchar());
```

εμφανίζει στην οθόνη το χαρακτήρα που πληκτρολογήθηκε

- π.χ. έστω ότι πληκτρολογούμε: **ABCDE**, τότε το παραπάνω πρόγραμμα θα εμφανίσει στην οθόνη:

  AB

  ✍ παρατηρήστε ότι στην είσοδο έχουν απομείνει μη διαβασμένοι χαρακτήρες (**CDE**), οι οποίοι θα μπορούσαν να διαβαστούν από επόμενες συναρτήσεις ανάγνωσης από το πληκτρολόγιο

Είσοδος/Έξοδος

Βιβλιοθήκη stdio.h - Συμβολοσειρές

- στη C η συμβολοσειρά είναι ένας πίνακας χαρακτήρων που τελειώνει με τον κενό χαρακτήρα ('\0'), δηλαδή ένα byte με τιμή 0_{10}
- **scanf()**
  - διαβάζει συμβολοσειρά με τον προδιοριστή %s
  - τοποθετεί το '\0' στο τέλος της συμβολοσειράς
  - το όνομα της μεταβλητής δίνεται χωρίς το &
- **printf()**
  - εμφανίζει συμβολοσειρά με τον προδιοριστή %s
  - δεν εμφανίζει το '\0'

το 20 είναι το μέγιστο πλήθος χαρακτήρων του πίνακα name συμπεριλαμβανομένου του '\0'

```
1. char name[20];
2. printf("Enter your name (& press enter): ");
3. scanf("%s", name);
```

```
4. printf("Hello %s!", name);
```

Είσοδος/Έξοδος

Βιβλιοθήκη stdio.h - Συμβολοσειρές II

- **gets(<μεταβλητή_συμβολοσειράς>)**
  - διαβάζει συμβολοσειρά από το πληκτρολόγιο μέχρι να συναντήσει το χαρακτήρα '**\n**' (αλλαγή γραμμής)
  - την αποθηκεύει στη μεταβλητή που δέχεται ως παράμετρο
  - τοποθετεί το '**\0**' στο τέλος της συμβολοσειράς
- **puts(<συμβολοσειρά>)**
  - εμφανίζει τη συμβολοσειρά που δέχετε ως παράμετρο

```
1. char name[20];
2. printf("Enter your name (& press enter): ");
3. gets(name);

4. puts("Hello ");
5. puts(name);
6. puts("!");
```

Τελεστές

Τελεστές
Σύγκρισης

- τους χρησιμοποιούμε για τη δημιουργία απλών λογικών εκφράσεων σύγκρισης

- το αποτέλεσμα μιας λογικής έκφρασης μπορεί να είναι:
  - είτε η τιμή **1** (αληθής)
  - είτε η τιμή **0** (ψευδής)

  π.χ. αν ισχύει **a=2** και **b=3**, τότε η λογική έκφραση **(a > b)** αποτιμάται στην τιμή **0** (ψευδής έκφραση)

προσέξτε ότι ο τελεστής για το λογικό ίσον (==) διαφέρει από τον τελεστή εκχώρησης (=)

Τελεστής	Περιγραφή
==	ίσο
!=	όχι ίσο
a < b	μικρότερο από
a <= b	μικρότερο από ή ίσο
a > b	μεγαλύτερο από
a >= b	μεγαλύτερο από ή ίσο

- οι τελεστές σύγκρισης χρησιμοποιούνται σαν συνθήκες κυρίως στις εντολές ελέγχου και επανάληψης

Τελεστές
Σύγκρισης - Παρατηρήσεις

η προτεραιότητα των τελεστών συσχετισμού είναι

- μικρότερη από αυτή των αριθμητικών τελεστών
- μεγαλύτερη από αυτή του τελεστή καταχώρησης

	Προτεραιότητα
μεγαλύτερη	αριθμητικοί τελεστές
	τελεστές συσχετισμού
μικρότερη	τελεστής εκχώρησης

π.χ.

```
int i, j = k = 13;
i = j == k;
```

1. επειδή ο τελεστής της ισότητας (`==`) έχει μεγαλύτερη προτεραιότητα από τον τελεστή καταχώρησης (`=`), θα εκτελεστεί πρώτα η ισότητα
2. δεδομένου ότι οι μεταβλητές `j` και `k` είναι ίσες, το αποτέλεσμα λογικής έκφρασης αυτών (`j == k`) θα είναι η τιμή `1` (αληθής)
3. έτσι, η μεταβλητή `i` θα πάρει την τιμή `1`

Τελεστές
Λογικοί

- μας επιτρέπουν να συνδυάσουμε απλές εκφράσεις συσχετισμού και να δημιουργήσουμε πιο πολύπλοκες λογικές προτάσεις

Τελεστής	Περιγραφή
&&	λογικό και (AND)
||	λογικό ή (OR)
!	λογική άρνηση (NOT)

- το αποτέλεσμα μιας πρότασης με λογικούς τελεστές είναι:
  - είτε η τιμή **0** (ψευδής)
  - είτε όχι η τιμή **0** (αληθής)

- οι τελεστές **&&** και **||** είναι δυαδικοί
  (απατείται η χρήση δύο όρων)

```
(x>10 && x<20)
((x>10 && x<20) || x>40)
```

- ο τελεστής **!** είναι μοναδιαίος

```
!(x>10 && x<20)  ⟺  (x<=10 || x>=20)
```

ισοδύναμες λογικές προτάσεις

Τελεστές
Λογικοί - Παρατηρήσεις

- η προτεραιότητα των τελεστών συσχετισμού είναι μεγαλύτερη από αυτή των λογικών τελεστών
- η προτεραιότητα των τελεστών συσχετισμού και των λογικών τελεστών είναι μικρότερη από αυτή των αριθμητικών τελεστών

	Προτεραιότητα
μεγαλύτερη	αριθμητικοί τελεστές
	τελεστές συσχετισμού
	λογικοί τελεστές
μικρότερη	τελεστής εκχώρησης

- π.χ. υπολογισμός δίσεκτου έτους:

```
(etos%4 == 0 && etos%100 != 0 || etos%400 == 0)
```

1ος τρόπος υπολογισμού

```
(!(etos%4) && (etos%100) || !(etos%400))
```

2ος τρόπος υπολογισμού

  - δίσεκτα έτη θεωρούνται όσα
    - διαιρούνται ακριβώς με το 4 αλλά όχι με το 100
    - διαιρούνται ακριβώς με το 400

Τελεστές

Χειρισμού bits - ολίσθηση

- με την εφαρμογή των τελεστών ολίσθησης τα bits της μεταβλητής μετατοπίζονται προς τα αριστερά ή δεξιά
  - π.χ. η έκφραση `a << 2;` (ή `a >> 2;`) μετατοπίζει τα bits της `a` κατά 2 θέσεις αριστερά (ή δεξιά)
  - στα κενά που δημιουργούνται τοποθετούνται μηδενικά

Τελεστής	Περιγραφή
<<	ολίσθηση προς τα αριστερά
>>	ολίσθηση προς τα δεξιά

```
1. unsigned short k, j;    // ολίσθηση αριστερά
2. j = 12;                 // j = 0000000000001100₂
3. k = j << 2;             // k = 0000000000110000₂ (48₁₀)
4.                         // ολίσθηση δεξιά
5. j = 12;                 // j = 0000000000001100₂
6. k = j >> 2;             // k = 0000000000000011₂ (3₁₀)
```

- η ολίσθηση μιας μεταβλητής κατά **n** θέσεις
  - προς τα αριστερά ισοδυναμεί με πολλαπλασιασμό της με 2^n
  - ενώ προς τα δεξιά με διαίρεση της με 2^n

Τελεστές
Χειρισμού bits

- δυαδική άρνηση: εναλλαγή 1 σε 0 και 0 σε 1
  - συμπλήρωμα ως προς 1
  - μοναδιαίος τελεστής
- δυαδική σύζευξη
  - δυαδικός τελεστής
  - να μη μπερδευτεί με το μοναδιαίο τελεστή & που όταν μπαίνει μπροστά σε μία μεταβλητή αναφέρεται στη διεύθυνση της
- δυαδική διάζευξη
  - δυαδικός τελεστής
- αποκλειστική διάζευξη
  - δυαδικός τελεστής

Τελεστής	Περιγραφή
<<	ολίσθηση προς τα αριστερά
>>	ολίσθηση προς τα δεξιά
~	δυαδική άρνηση
&	δυαδική σύζευξη ("και" ή AND)
|	δυαδικό διάζευξη ("ή" ή OR)
^	αποκλειστική διάζευξη (exclusive OR)

a	~a
0	1
1	0

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

Τελεστές

Χειρισμού bits - Παράδειγμα - Άρτιος ή περιττός

- έστω `i` ακέραια μεταβλητή χωρίς πρόσημο (`unsigned int i;`)
- θέλουμε να διαπιστώσουμε αν η τιμή της `i` είναι άρτιος ή περιττός αριθμός

λύση:

```
int k = i & 1;
```

- εάν η τιμή του `k` είναι `0` → `i` άρτιος
- εάν η τιμή του `k` είναι `1` → `i` περιττός

το k έχει την τιμή του λιγότερου σημαντικού bit του i

επεξήγηση:

- ένας ακέραιος είναι άρτιος (περιττός) όταν, στη δυαδική αναπαράστηση, το λιγότερο σημαντικό bit του είναι `0` (`1`)
- με την έκφραση `i & 1`, μηδενίζουμε όλα τα bits του `i` εκτός από το λιγότερο σημαντικό (που παραμένει ως έχει)

```
  00101011  (i = 43)
& 00000001  (1)
  00000001  (k = 1)
```

```
  00101010  (i = 42)
& 00000001  (1)
  00000000  (k = 0)
```

Τελεστές

Χειρισμού bits - Παράδειγμα - Ανάθεση σε bit

- έστω `i` μία ακέραια μεταβλητή (`int i;`)
- θέλουμε να κάνουμε **1** το 3ο λιγότερο σημαντικό bit της `i`

λύση:

```
i = i | 4;
```

επεξήγηση:

- το **4** στο δυαδικό είναι **00000100** (δηλαδή, το 3ο bit είναι **1** και τα υπόλοιπα **0**)
- εστώ ότι η `i` έχει την τιμή 120_{10} (01111000_2)
- με την έκφραση `i = i | 4;`, η `i` παίρνει τη δυαδική τιμή **01111100**
  - όπου το 3ο bit είναι **1**
  - και τα υπόλοιπα παραμένουν ως έχουν

```
  01111000  (i = 120)
| 00000100  (4)
  01111100  (i = 124)
```

Τελεστές

Συνδυαστικοί τελεστές εκχώρησης

Τελεστής	Περιγραφή	Παράδειγμα
`+=`	πρόσθεση σε μεταβλητή	`x += 1;` (ισοδύναμο με `x=x+1;`)
`-=`	αφαίρεση από μεταβλητή	`x -= 1;` (ισοδύναμο με `x=x-1;`)
`*=`	πολλαπλασιασμός μεταβλητής	`x *= 1;` (ισοδύναμο με `x=x*3;`)
`/=`	διαίρεση μεταβλητής	`x /= 1;` (ισοδύναμο με `x=x%2;`)
`%=`	υπόλοιπο ακέραιας διαίρεσης	`x %= 1;` (ισοδύναμο με `x=x%2;`)
`>>=`	ολίσθηση δεξιά	`x >>= 3;` (ισοδύναμο με `x=x>>3;`)
`<<=`	ολίσθηση αριστερά	`x <<= 3;` (ισοδύναμο με `x=x<<3;`)
`&=`	σύζευξη (AND)	`x &= 0177;` (ισοδύναμο με `x=x&0177;`)
`|=`	διάζευξη (OR)	`x |= 0177;` (ισοδύναμο με `x=x|0177;`)
`^=`	αποκλειστική διάζευξη (exclusive OR)	`x ^= 0177;` (ισοδύναμο με `x=x^0177;`)

Τελεστές

Προτεραιότητες

ιεραρχία	τελεστές	φορά
1	( ) [ ] . ->	→ (από αριστερά πρός δεξιά)
2	! ~ ++ -- &(διεύθυνση) *(indirection) (type) sizeof	← (από δεξιά πρός αριστερά)
3	*(πολλαπλασιασμός) / %	→ (από αριστερά πρός δεξιά)
4	+ -	→ (από αριστερά πρός δεξιά)
5	<< >>	→ (από αριστερά πρός δεξιά)
6	< <= > >=	→ (από αριστερά πρός δεξιά)
7	= = !=	→ (από αριστερά πρός δεξιά)
8	& (AND)	→ (από αριστερά πρός δεξιά)
9	^	→ (από αριστερά πρός δεξιά)
10	|	→ (από αριστερά πρός δεξιά)
11	&&	→ (από αριστερά πρός δεξιά)
12	||	→ (από αριστερά πρός δεξιά)
13	? :	← (από δεξιά πρός αριστερά)
14	= += -= *= /= %= &= ^= |= <<= >>=	← (από δεξιά πρός αριστερά)
15	,	→ (από αριστερά πρός δεξιά)

Σύνοψη

- Χαρακτηριστικά Γλώσσας
  - Σταθερή έκφραση
    - σταθερές
    - σταθερή μεταβλητή
    - εκφράσεις
  - Μετατροπές
  - Τελεστές
    - αριθμητικοί
    - σύγκρισης
    - λογικοί
    - χειρισμού bits (ολίσθηση, δυαδική άρνηση, δυαδική σύζευξη, δυαδική διάζευξη, αποκλειστική διάζευξη)
- Είσοδος/Έξοδος
  - Προσδιοριστές
  - Χαρακτήρες ελέγχου