Αντικειμενοστραφής Προγραμματισμός

Διδάσκων: Χριστόφορος Οικονομάκος
Γραφείο: B213B
e-mail: ceconomakos@uoa.gr

Δομή του μαθήματος

- Διαλέξεις θεωρίας, κοινές.
  - Πέμπτη 09:00-13:00, Μεγάλο αμφιθέατρο
  - Θα τελειώνουμε γύρω στις 12:15 για να ξεκινάει το εργαστήριο.
- Εργαστήριο
  - Τμήμα Ψηφιακής Βιομηχανίας:
    - Πέμπτη μετά τη θεωρία, στην αίθουσα Β118, σε δίωρα τμήματα.
  - Τμήμα Αεροδιαστημικής
    - Μια ώρα, μετά το πέρας της διάλεξης της θεωρίας.
  - Περισσότερες λεπτομέρειες για τον τρόπο που θα οργανωθούν και θα λειτουργήσουν τα εργαστήρια θα ανακοινωθούν μέχρι την άλλη εβδομάδα.

Ιστορικό των γλωσσών C, C++

Ιστορικό της C++

- Επέκταση της C
- 1983: Bjarne Stroustrup (Εργαστήρια Bell)
- "Περιποιημένη" C
- Παρέχει τη δυνατότητες αντικειμενοστραφούς προγραμματισμού

  Αντικείμενα: επαναχρησιμοποιήσιμα συστατικά λογισμικού

  Αντικειμενοστραφή προγράμματα

  Εύκολα στην κατανόηση, διόρθωση και τροποποίηση
- Υβριδική γλώσσα

  Περιέχει στοιχεία και φορμαλισμό από τη C

  Έχει την τεχνοτροπία του αντικειμενοστραφούς προγραμματισμού

Ιστορικό των γλωσσών C, C++

- Η C++ προέρχεται από τη γλώσσα C. Για την ακρίβεια είναι ένα υπερσύνολο της C. Κάθε σωστή πρόταση της C, είναι και πρόταση της C++. Τα επιπλέον στοιχεία που προστέθηκαν στην C για να προκύψει η C++, είναι οι κλάσεις και τα αντικείμενα και γενικά ο αντικειμενοστραφής προγραμματισμός. Επιπρόσθετα, η C++ έχει πολλά νέα χαρακτηριστικά που περιλαμβάνουν, κυρίως, μία βελτιωμένη προσέγγιση της εισόδου/εξόδου δεδομένων.

- Πρότυπα C++:

  ANSI/ISO: 1998

  ISO: 2011

Πρότυπη Βιβλιοθήκη C++

Ο αντικειμενοστραφής προγραμματισμός παριστάνει ένα σύστημα ως μία συλλογή από αντικείμενα τα οποία αλληλεπιδρούν και αλλάζουν με το χρόνο.

Ένα αντικειμενοστραφές πρόγραμμα αποτελείται από κλάσεις και αντικείμενα.

Τα αντικείμενα έχουν ιδιότητες (χαρακτηριστικά) και ενέργειες (συμπεριφορά) που συνδέονται μ'αυτά.

Η πρότυπη βιβλιοθήκη περιέχει μία ευρύτατη συλλογή από κλάσεις και συναρτήσεις.

Java

- 1991: Sun Microsystems

  «Πράσινο πρόγραμμα» - Green project (Patrick Naughton, James Gosling, Mike Sheridan).
  Στόχος ο προγραμματισμός μικροσυσκευών, χωρίς την πολυπλοκότητα της C++.

- 1995: Sun Microsystems

  Επίσημη ονοματοδοσία της νέας γλώσσας ως Java

- Ιστοσελίδες με δυναμικό και διαδραστικό περιεχόμενο

- Ανάπτυξη εφαρμογών μεγάλης κλίμακας

- Διευρυμένη λειτουργικότητα web servers

- Εφαρμογές για καταναλωτικές συσκευές

  Κινητά τηλέφωνα, βομβητές, PDA, κ.λ.π.

Λόγοι εκμάθησης C++

- Γλώσσα αντικειμενοστραφούς προγραμματισμού υψηλού επιπέδου, με στοιχεία από διαδικαστικές γλώσσες (C, Pascal κ.λ.π.)
- Απαιτούμενο προσόν για πολλές θέσεις εργασίας.
- Παρέχεται πλήρης έλεγχος στον προγραμματιστή:
  - Υποκαθιστά σε πολλές περιπτώσεις τη χρήση γλώσσας μηχανής (όπως και η γλώσσα C).
  - Επιτυγχάνεται ακριβής έλεγχος παντού, κυρίως δε στη διαχείριση της μνήμης.
- Υψηλή ταχύτητα εκτέλεσης.
- Μειονεκτήματα:
  - Περίπλοκη γλώσσα
  - Απαιτεί προσοχή και σχολαστικότητα στη συγγραφή κώδικα, καθώς και ιδιαίτερη εξάσκηση

C++ vs JAVA

•Η κάθε γλώσσα προορίζεται για άλλες εφαρμογές και διαφορετικές απαιτήσεις. Παρόλο που η Java μοιάζει με τη C++, πρόκειται για διαφορετικές γλώσσες. Π.χ:

- Η C++ επιτρέπει συναρτήσεις που δεν είναι μέθοδοι τάξεων.

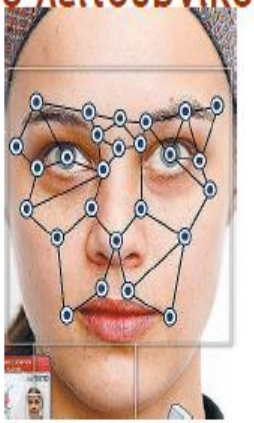
- Η C++ επιτρέπει τη χρήση δεικτών και δεν παρέχει αυτόματη αποκομιδή απορριμμάτων (garbage collection). Βέβαια το πρότυπο C++11 καθορίζει μηχανισμούς αποκομιδής απορριμάτων, αλλά η υποστήριξή τους από τους μεταγλωττιστές είναι προαιρετική.

- Ο κάθε μεταγλωττιστής της C++ παράγει εκτελέσιμο κώδικα για συγκεκριμένο επεξεργαστή.

C++ vs JAVA

Σε πολλές εφαρμογές που απαιτείται ταχύτητα και μεγάλη προβλεψιμότητα [illegible] έλεγχος, η Java δε μπορεί να ανταπεξέλθει:

•Προγραμματισμός σε επίπεδο λειτουργικού συστήματος



• Επιστημονικοί υπολογισμοί

• Μαζικά δεδομένα (π.χ. συστήματα διαχείρισης δεδομένων)

•Αλληλεπιδραστικά συστήματα (π.χ. γραφικά Η/Υ, GUIs)

C++ vs JAVA

Προτιμάται η C++ όταν η ταχύτητα του προγράμματος είναι κρίσιμος παράγοντας:

- Παράγεται κώδικας μηχανής από την αρχή, όχι byte code και Just in-time (JIT) παραγωγή κώδικα
- Ολική βελτιστοποίηση του κώδικα από τον compiler, κάτι που δεν μπορεί να γίνει στη Java.
- Δε μεσολαβεί κάποια Virtual Machine (π.χ. JVM) που σπαταλά πόρους για τη δική της λειτουργία.
- Μπορούν να χρησιμοποιηθούν βελτιστοποιημένες υλοποιήσεις και ειδικά CPU Instruction Sets (π.χ. SSE4).
- Ο προγραμματιστής μπορεί να παρέμβει σε πολύ χαμηλό επίπεδο στη βελτιστοποίηση της απόδοσης.

C++ vs JAVA

Προτιμάται η C++ όταν η κατανάλωση μνήμης του προγράμματος είναι κρίσιμος παράγοντας:

- Δεν παρέχεται άυτόματη αποκομιδή απορριμάτων. Ο προγραμματιστής αποφασίζει πώς και πότε θα αποδεσμεύσει μνήμη.
- Γίνεται ακριβής έλεγχος των δεδομένων που αποθηκεύονται.
- Παράγεται ιδιαίτερα συμπαγής κώδικας.
- Μπορούν να υλοποιηθούν δικοί μας μηχανισμοί διαχείρισης μνήμης (caches, memory pooling, lazy allocators).

Επίσης προτιμάται η C++ όταν απαιτείται χαμηλή κατανάλωση ενέργειας:

- Δε μεσολαβεί Virtual Machine. Η VM είναι και η ίδια ένα εκτελέσιμο πρόγραμμα, οπότε όταν εκτελείται ο κώδικά, χρησιμοποιούνται παράλληλα επιπρόσθετοι πόροι του συστήματος (κυρίως κύκλους CPU) που καταναλώνουν ενέργεια.
- Είναι σημαντικός παράγοντας σε αρχιτεκτονικές κινητών συσκευών (κινητά τηλάφωνα, tablets, ενσωματωμένα συστήματα κ.λ.π.).

Διαδικαστικός προγραμματισμός σε C++

Είσοδος/Έξοδος δεδομένων

Αφού κάνουμε **#include** το <iostream>:

- Το

std::cin >> a;

διαβάζει τιμή στη μεταβλητή a.

- Το

std::cout << a;

τυπώνει την τιμή της ποσότητας a.

- Το

std::cout << u8"Ελληνικό κείμενο";

τυπώνει το κείμενο που γράφουμε εντός διπλών εισαγωγικών.
Αν το κείμενο έχει ελληνικούς χαρακτήρες, βάζουμε u8 πριν τα αρχικά εισαγωγικά.

Τύποι ακέραιων ποσοτήτων (1/2)

- Βασικός τύπος δεδομένων: int
- Μπορεί να δέχεται τροποποιητές (modifiers)
  - Σε σχέση με το πρόσημο
    - signed (εμπρόσημος τύπος)
    - unsigned (απρόσημος τύπος)
  - Σε σχέση με το μέγεθος
    - short (τουλάχιστον 16 bit)
    - long (τουλάχιστον 32 bit)
    - long long (τουλάχιστον 64 bit)
- Όταν δεν υπάρχει τροποποιητής προσήμου εννοείται ο signed.
- Όταν υπάρχουν τροποποιητές η λέξη int μπορεί και να παραλείπεται.
- Για μικρές τιμές που μπορούν να παρασταθούν με 8 bit μπορεί να χρησιμοποιηθεί και ο τύπος char που δέχεται τους τροποποιητές signed και unsigned.

Τύποι ακέραιων ποσοτήτων (2/2)

Παρατηρήσεις:

- Εμπρόσημος ακέραιος τύπος για γενική χρήση: **int**
- Συνήθης μέγιστη τιμή int: 2147483647 (2^31-1)
- Συνήθης ελάχιστη τιμή int: -2147483648 (-2^31)
- Απρόσημος ακέραιος τύπος για μετρητή ή δείκτη σε array: std::size_t (από το cstddef)
- Συνήθης μέγιστη τιμή size_t: 2^64-1
- Τα ψηφία μπορούν να χωρίζονται με απόστροφο: 12’345’678
- Οι αριθμοί μπορούν να γραφούν εναλλακτικά στο δεκαεξαδικό (0x), στο οκταδικό (0) ή στο δυαδικό σύστημα (0b).

Τύποι πραγματικών ποσοτήτων (1/2)

- **float** (απλής ακρίβειας, 6 σημαντικά ψηφία),
- **double** (διπλής ακρίβειας, 15 σημαντικά ψηφία),
- **long double** (εκτεταμένης ακρίβειας, 18 σημαντικά ψηφία).

Τιμές

Σειρά αριθμητικών ψηφίων χωρίς κενά, με πιθανό πρόσημο (+,-), και

- ή τελεία (αντί για υποδιαστολή) που χωρίζει το ακέραιο από το δεκαδικό μέρος: Π.χ. 12.345, -1.02.
- ή το χαρακτήρα e (ή E) που χωρίζει τον αριθμό από τη δύναμη του 10 με την οποία πολλαπλασιάζεται.

Π.χ. 123E2

(≡ 123 × 102 ≡ 12300.0), -12e-1 (≡ −1.2),

- ή συνδυασμό των παραπάνω: Π.χ. -1.2E-2 (≡ −0.012),

και πιθανή κατάληξη: Π.χ. 3E4f, 3E4L.

Τύποι πραγματικών ποσοτήτων (2/2)

Παρατηρήσεις

- Ποιον τύπο πρέπει να χρησιμοποιούμε; **double**
- Συνήθης μέγιστη απόλυτη τιμή **double**: ≈ 10^308
- Συνήθης ελάχιστη απόλυτη τιμή **double**: ≈ 10^−308.
- Πώς γράφουμε σε **double** το 10^−6; 1E-6

Αριθμητικοί τελεστές μεταξύ ακέραιων ποσοτήτων

Άθροισμα: + , Π.χ. 5 + 7
Διαφορά: - , Π.χ. 5 - 7
Γινόμενο: *, Π.χ. 2 * 3
Πηλίκο: / , Π.χ. 7/2 → 3
Υπόλοιπο: %, Π.χ. 7%2 → 1
Γενικά ισχύει (για ακέραια a,b με b> 0)

- 0 ≤ a%b < b

Ύψωση σε δύναμη

Με συνάρτηση: std::pow(a,b) → *ab* από το <cmath>.
Τα *a, b* είναι ακέραια.
Το αποτέλεσμα είναι **double** (γιατί οι τιμές τους μετατρέπονται σε **double** κατά την κλήση).

Αριθμητικοί τελεστές μεταξύ ακέραιων ποσοτήτων

Άθροισμα: 5.0 + 7.0
Διαφορά: 5.0f - 7.0f
Γινόμενο: 2.0 * 3e2
Λόγος: 7.0/2.0

Ύψωση σε δύναμη:

Με συνάρτηση: std::pow(a,b) → *a^b* από το <cmath>.
Τα *a, b* είναι πραγματικοί και το αποτέλεσμα έχει τον ίδιο τύπο με αυτούς.

Γενικές παρατηρήσεις στους αριθμητικούς τελεστές

Κανόνες

- Τελεστές που δρουν μεταξύ ποσοτήτων ίδιου τύπου δίνουν αποτέλεσμα αυτού του τύπου.
- Τελεστές που δρουν μεταξύ ποσοτήτων διαφορετικού τύπου προκαλούν μετατροπή της τιμής της ποσότητας «χαμηλότερης» ακρίβειας στον τύπο με την «υψηλότερη» ακρίβεια πριν από την πράξη.

Παράδειγμα: **int** σε **double**: 2*3.0→ 2.0*3.0 → 6.0.

Εξαίρεση: Ποσότητες τύπου **short int** γίνονται **int** πριν την πράξη.

Συντμήσεις αριθμητικών τελεστών

x = x + a; → x += a;
x = x * a; → x *= a;
x = x % a; → x %= a;
x = x - a; → x -= a;
x = x / a; → x /= a;

Ειδικές περιπτώσεις

x = x + 1; → x += 1; → ++x; ή x++;
x = x - 1; → x -= 1; → --x; ή x--;

Παρατηρήσεις

- Οι συντμήσεις μπορεί να εμφανιστούν σε σύνθετο κώδικα, π.χ.

b = a+= 5; → a=a+5; b = a;

- Οι σύνθετες εντολές b = ++a; και b = a++; διαφέρουν:

b = ++a; → a = a + 1; b = a;

Σχετικές προτεραιότητες αριθμητικών τελεστών

Πολύ Υψηλή: Παρενθέσεις ()

Υψηλή: ++,--

Μεσαία: *,/,%

Χαμηλή: +,-

Πολύ Χαμηλή: +=,-=,*=,/=,%=

Παρατήρηση

Τελεστές με ίδια προτεραιότητα σε μια έκφραση εκτελούνται από αριστερά προς τα δεξιά.

Μεταβλητή (1/3)

Κανόνας

Κάθε μεταβλητή προτού χρησιμοποιηθεί πρέπει να δηλωθεί, κατά προτίμηση λίγο πριν χρησιμοποιηθεί. Αν γνωρίζουμε την αρχική της τιμή, καλό είναι να κάνουμε δήλωση με απόδοση αρχικής τιμής.

Δήλωση

τύπος όνομα_μεταβλητής;

Παραδείγματα

int a;
double b;

Είναι πιο ευανάγνωστος ο κώδικας αν δηλώνουμε κάθε ποσότητα σε ξεχωριστή γραμμή.

Μεταβλητή (2/3)

Δήλωση με απόδοση αρχικής τιμής

τύπος όνομα_μεταβλητής{αρχική τιμή};

Παραδείγματα:

int a{3};
double b{4.5};

Παρατηρήσεις:

- ΟΧΙ **int** a{4.3};
- Επιτρέπεται το **int** a = 4.3; ;
- Κενά άγκιστρα → default τιμή (για αριθμούς, 0).
- Αν δεν δώσουμε αρχική τιμή, οι μεταβλητές των ενσωματωμένων τύπων αποκτούν απροσδιόριστη τιμή.

Μεταβλητή (3/3)

Αυτόματη αναγνώριση τύπου από την αρχική τιμή

auto *όνομα_μεταβλητής = αρχική τιμή;*

Παραδείγματα

auto a = 3;

auto b = 3.6;

Το a είναι **int** με τιμή 3, το b **double** με τιμή 3.6.

Παρατήρηση

ΟΧΙ **auto** a{4.3};

Σταθερή ποσότητα

- ✓ Μια ποσότητα που επιθυμούμε να πάρει τιμή που να μην μπορεί να αλλάξει κατά τη διάρκεια εκτέλεσης του προγράμματος, δηλώνεται με τη χρήση της λέξης **const**.

- ✓ Αν *είναι γνωστή η τιμή κατά τη μεταγλώττιση*, μπορούμε να χρησιμοποιήσουμε τη λέξη **constexpr**. Είναι απαραίτητο να της δώσουμε αρχική (και μόνιμη) τιμή κατά τον ορισμό της.

Παράδειγμα

```
double const x{std::pow(2.3, 1.2)};
double constexpr pi{3.14159265358};
double constexpr twopi{2.0*pi};
```

Κανόνες σχηματισμού ονόματος

- Επιτρεπτοί χαρακτήρες: a–z, A–Z, 0–9, και _.
- Μήκος: οποιοδήποτε.
- Δεν επιτρέπεται να αρχίζει από αριθμητικό ψηφίο.
- Δεν επιτρέπεται να αποτελεί ενσωματωμένη λέξη της C++ (πίνακας 2.1 σημειώσεων).
- Κεφαλαία και πεζά γράμματα είναι διαφορετικά.

Ρητή μετατροπή από ένα τύπο σε άλλο

Τι θα τυπωθεί στον παρακάτω κώδικα;

```
int a{9}, b{2};
std::cout << a/b << '\n';
```

Χωρίς να αλλάξουμε τον τύπο των a,b, πώς θα υπολογίσουμε το λόγο τους (και όχι το πηλίκο);

Ρητή (σε αντιδιαστολή με την αυτόματη) μετατροπή τιμής από ένα τύπο σε άλλο γίνεται με το static_cast<>:
static_cast<double>(a)/static_cast<double>(b);

Η μετατροπή αφορά την τιμή που έχει η μεταβλητή στο όρισμα. Ο τύπος της δεν αλλάζει.

Σύνταξη
static_cast<νέος_τύπος>(έκφραση);

Εντολή εκχώρησης τιμής

μεταβλητή = [γενική έκφραση];

- ✓ Πρώτα εκτελούνται όλες οι πράξεις, κλήσεις συναρτήσεων κλπ. που εμφανίζονται στο δεξί μέλος.
- ✓ Κατόπιν, το αποτέλεσμα μετατρέπεται (αν χρειάζεται) στον τύπο της (υποχρεωτικά) μεταβλητής του αριστερού μέλους και η τιμή που προκύπτει εκχωρείται σε αυτή.

Παραδείγματα:

```
int a;
float b;
a = 2.5 * 3; // a ← 7
b = 1.00000001; // b ← 1.0f
```

Τύπος χαρακτήρα (char)

Ο τύπος **char** είναι κατάλληλος για την αναπαράσταση ποσοτήτων που οι δυνατές τιμές τους είναι χαρακτήρες.

Τιμές:
Απλοί ή ειδικοί χαρακτήρες εντός απλών (') εισαγωγικών.
Δεν επιτρέπονται ελληνικοί χαρακτήρες.

Δήλωση:
char c;
char d{'a'};

Εκχώρηση:
c = 'e';
c = '\n';

Ισοδυναμία με ακέραιους:
Κάθε χαρακτήρας *είναι* ακέραιος με τιμή που ορίζεται στο σύνολο χαρακτήρων της υλοποίησης (συνήθως το ASCII).

Ειδικοί Χαρακτήρες της C++

Ειδικός Χαρακτήρας	Περιγραφή
\'	Απόστροφος
\"	Εισαγωγικά
\?	Ερωτηματικό
\\	Ανάποδη κάθετος
\a	Κουδούνι
\b	Διαγραφή προηγούμενου χαρακτήρα
\f	Αλλαγή σελίδας
\n	Αλλαγή γραμμής
\r	Μετακίνηση στην αρχή της γραμμής
\t	Οριζόντιο tab
\v	Κατακόρυφο tab
\ooo	Χαρακτήρας με οκταδική αναπαράσταση ooo
\xhhh	Χαρακτήρας με δεκαεξαδική αναπαράσταση hhh
\unnnn	Ο χαρακτήρας unicode U+nnnn
\Unnnnnnnn	Ο χαρακτήρας unicode U+nnnnnnnn

Εμβέλεια

Μια ποσότητα (μεταβλητή, σταθερή, συνάρτηση, κλπ.) μπορεί:

- ✓ να χρησιμοποιηθεί από το σημείο της δήλωσής της έως το } που κλείνει την περιοχή στην οποία δηλώθηκε.
- ✓ να οριστεί ξανά (ίσως με άλλο τύπο) σε «εσωτερικό» τμήμα.

```
{
   int a;
   {
      a = ...    // Σωστό
      int b = ...    // Σωστό
      double a =  ...    // Σωστό, νέο a
   }
   a = ...      // Σωστό, είναι το αρχικό a
   b = ...     // Λάθος
}
a = ...     // Λάθος
b = ...     // Λάθος
```

Μαθηματικές συναρτήσεις

Συνάρτηση	Επιστρεφόμενη τιμή
`double cos(double x)`	Συνημίτονο του x.
`double sin(double x)`	Ημίτονο του x.
`double tan(double x)`	Εφαπτομένη του x.
`double acos(double x)`	Τόξο συνημιτόνου του x.
`double asin(double x)`	Τόξο ημιτόνου του x.
`double atan(double x)`	Τόξο εφαπτομένης του x.
`double atan2(double x, double y)`	Τόξο εφαπτομένης $\tan^{-1}(x/y)$.
`double pow(double x, double a)`	Ύψωση σε δύναμη, x^a.
`double sqrt(double x)`	Η τετραγωνική ρίζα του x.
`double cbrt(double x)`	Η κυβική ρίζα του x.
`double hypot(double x, double y)`	$\sqrt{x^2+y^2}$.
`double exp(double x)`	Εκθετικό του x (e^x).
`double log(double x)`	Φυσικός λογάριθμος του x ($\ln x$).
`double log2(double x)`	Δυαδικός λογάριθμος του x ($\log_2 x$).
`double log10(double x)`	Δεκαδικός λογάριθμος του x ($\log_{10} x$).
`double abs(double x)`	Απόλυτη τιμή του x.

Όλες χρειάζονται πρόθεμα std::. Προϋποθέτουν το #include <cmath>.

Οι τριγωνομετρικές συναρτήσεις δέχονται/επιστρέφουν τις γωνίες σε rad.

Παράδειγμα

Το e^−x cos(y) γράφεται std::exp(-x) * std::cos(y).

Λογικός τύπος (bool)

Ο τύπος **bool** είναι κατάλληλος για την αναπαράσταση ποσοτήτων που μπορούν να πάρουν δύο μόνο τιμές (π.χ. ναι/όχι, αληθές/ψευδές,…)

Τιμές:
true ή **false**

Δήλωση:
bool a;
bool b{**true**};

Εκχώρηση:
a = **false**;

Ισοδυναμία με ακέραιους:
Το **true** ισοδυναμεί με 1, το **false** με 0.
Αντίστροφα: μη μηδενικός ακέραιος μετατρέπεται σε **true**, το 0 ισοδυναμεί με **false**.

Απλή λογική έκφραση

Σύγκριση δύο ποσοτήτων με τους τελεστές:

Τελεστής	Σύγκριση	Τελεστής	Σύγκριση
==	ίσο	!=	άνισο
>	μεγαλύτερο	>=	μεγαλύτερο ή ίσο
<	μικρότερο	<=	μικρότερο ή ίσο

Παραδείγματα:

x > 10, y != 3, k == 5

Προτεραιότητα:

Οι τελεστές σύγκρισης έχουν χαμηλότερη προτεραιότητα από τους αριθμητικούς τελεστές (επομένως, το k>3+2 κάνει αυτό που αναμένουμε, τη σύγκριση του k με το 5)

Σύνθετη λογική έκφραση

Προκύπτει από ένωση απλών συνθηκών με τους τελεστές && (AND) και || (OR), ή αλλαγή της τιμής μίας λογικής ποσότητας με τον τελεστή ! (NOT).

Ισοδύναμες λέξεις: **and**, **or** και **not** αντίστοιχα.

Παραδείγματα

- $10 \leq x \leq 20 \Rightarrow$ 10.0 <= x && x <= 20.0
- *k* είναι 3 ή 5 $\Rightarrow$ k == 3 || k == 5
- !(j==3)

Προτεραιότητες τελεστών

Πολύ Υψηλή Τελεστής !,

Υψηλή Αριθμητικοί τελεστές,

Μεσαία Τελεστές σύγκρισης,

Χαμηλή Τελεστής &&,

Πολύ Χαμηλή Τελεστής ||.

Για σιγουριά βάζουμε παρενθέσεις!

1η άσκηση

```
// Your First C++ Program
#include <iostream>
int main()
{
    std::cout << "Hello World!";
    return 0;
}
```

2η άσκηση

```
#include <iostream>
/*
main:
Den pairnei orismata.
Zhta ena pragmatiko kai typwnei to tetragwno tou.
Epistrefei 0.
*/
int main()
{
std::cout << "DWSE ARITHMO"; // Mhnyma sthn othoni
double a; // Dhlwsh pragmatikhs metavlhths.
std::cin >> a; // Eisagwgi timhs apo plhktrologio
double b; // Dhlwsh allhs metablhths
b = a * a;
std::cout <<"TO TETRAGWNO EINAI"; // Mhnyma sthn othoni
std::cout << b << '\n'; // Ektypwsi apotelesmatos kai allagh grammhs
return 0; // Epistrofh me epityxia.
}
```

Επεξήγηση

#include <iostream> :
προκαλεί την εισαγωγή του *header* <iostream> και δίνει τη δυνατότητα στον κώδικά μας να χρησιμοποιήσει, ανάμεσα σε άλλα, το πληκτρολόγιο και την οθόνη για είσοδο και έξοδο δεδομένων.

/*…*/:
Ο μεταγλωττιστής (compiler) αγνοεί τους χαρακτήρες που περιλαμβάνονται μεταξύ:
• του // και του τέλους της γραμμής στην οποία εμφανίζεται αυτός ο συνδυασμός χαρακτήρων,
• των /* και */ ανεξάρτητα από το πλήθος γραμμών που περιλαμβάνουν.

Οι χαρακτήρες αυτοί αποτελούν τα *σχόλια* και πρέπει να είναι μόνο λατινικοί (σχεδόν πάντα από το σύνολο χαρακτήρων ASCII).

int main():

Η δήλωση **int** main() {...} ορίζει τη βασική συνάρτηση σε κάθε πρόγραμμα C++: το όνομά της είναι main, επιστρέφει ένα ακέραιο αριθμό, ενώ, στο συγκεκριμένο ορισμό, δε δέχεται ορίσματα. Δεν υπάρχουν ποσότητες μεταξύ των παρενθέσεων που ακολουθούν το όνομα. Οι εντολές (αν υπάρχουν) μεταξύ των αγκίστρων '{}' που ακολουθούν την κενή λίστα ορισμάτων είναι ο κώδικας που εκτελείται με την κλήση της.

Εφαρμογές (1/2)

Άσκηση 1. Να γράψετε πρόγραμμα που να διαβάζει έναν αριθμό και να τον εμφανίζει στην οθόνη.

Άσκηση 2. Να γράψετε πρόγραμμα που θα διαβάζει 2 αριθμούς και θα εμφανίζει

α. Το άθροισμα τους

β. Το γινόμενό τους.

γ. Το μέσο όρο τους

Άσκηση 3. Να γράψετε πρόγραμμα που θα διαβάζει 2 αριθμούς στις μεταβλητές α και β. Στη συνέχεια να αντιμεταθέτει τις τιμές των μεταβλητών και να εμφανίζει την τελική τιμή τους.

Άσκηση 4. Ο μέσος όρος ενός μαθητή στον προγραμματισμό προκύπτει από το μέσο όρο των εργασιών του και του γραπτού βαθμού στην εξεταστική. Ο βαθμός των εργασιών είναι ο μέσος όρος των 2 επιμέρους εργασιών. Να γράψετε πρόγραμμα που θα εμφανίζει τον τελικό βαθμό ενός φοιτητή.

Εφαρμογές (2/2)

Άσκηση 5. Να μετατρέψετε το παρακάτω διάγραμμα ροής σε πρόγραμμα C++