Εργαστήριο 3:

3.1 Αριθμητικοί και Λογικοί Τελεστές, Μετατροπές Τύπου (Casting)

Η C++, όπως όλες οι γλώσσες προγραμματισμού, χρησιμοποιεί τελεστές για να εκτελέσει τις αριθμητικές και λογικές λειτουργίες. Συνοπτικά οι βασικότεροι είναι οι εξής:

- Ανάθεσης (=)
- Αριθμητικοί (+, -, *, /, % → ακέραιο υπόλοιπο)
- Λογικοί (**&&** → AND, || → OR, **!** →NOT)
- Σύγκρισης ή Συσχετιστικοί (>, >=, ==, !=, <, <=)
  == → τελεστής ισότητας (όχι το μονό '=', αφού σημαίνει ανάθεση)
  != → τελεστής ανισότητας (≠)
- Μοναδιαίας αύξησης και μείωσης (++, --)

Ανάθεση ή καταχώρηση ή εκχώρηση τιμής: Το = είναι ένας τελεστής καταχώρηση τιμής. Δεν μπορούμε να καταχωρήσουμε τιμή σε μια σταθερά, ούτε σε παράσταση, άρα στα αριστερά του συμβόλου '=' πρέπει να είναι το όνομα μιας μόνο μεταβλητής.

Εφαρμογές της εντολής αυτής είναι οι εξής :

year =2012;

i = i+1;

t1=t2=68;

Στην τελευταία εντολή οι καταχωρήσεις γίνονται από τα δεξιά προς τα αριστερά. Στην πραγματικότητα μια εντολή ανάθεσης είναι επίσης μια παράσταση, η οποία παίρνει τιμή την τιμή που ανατέθηκε στη μεταβλητή! Έτσι το t2 γίνεται 68 και το t1 γίνεται ίσο με την τιμή της παράστασης t2=68, δηλαδή με 68.

Αριθμητικοί Τελεστές:

πρόσθεσης: +, αφαίρεσης: -, πολλαπλασιασμού: *, διαίρεσης: /, ακέραιο υπόλοιπο διαίρεσης: %.

Ο τελεστής ακεραίου υπολοίπου (%) χρησιμοποιείται στην αριθμητική των ακεραίων και επιστρέφει το υπόλοιπο της ακέραιας διαίρεσης του ακεραίου στα αριστερά με τον ακέραιο στα δεξιά. Για παράδειγμα, η πράξη 13 % 5 δίνει σαν αποτέλεσμα την τιμή 3, αφού το 5 χωράει δύο φορές στο 13 και έχει υπόλοιπο 3.

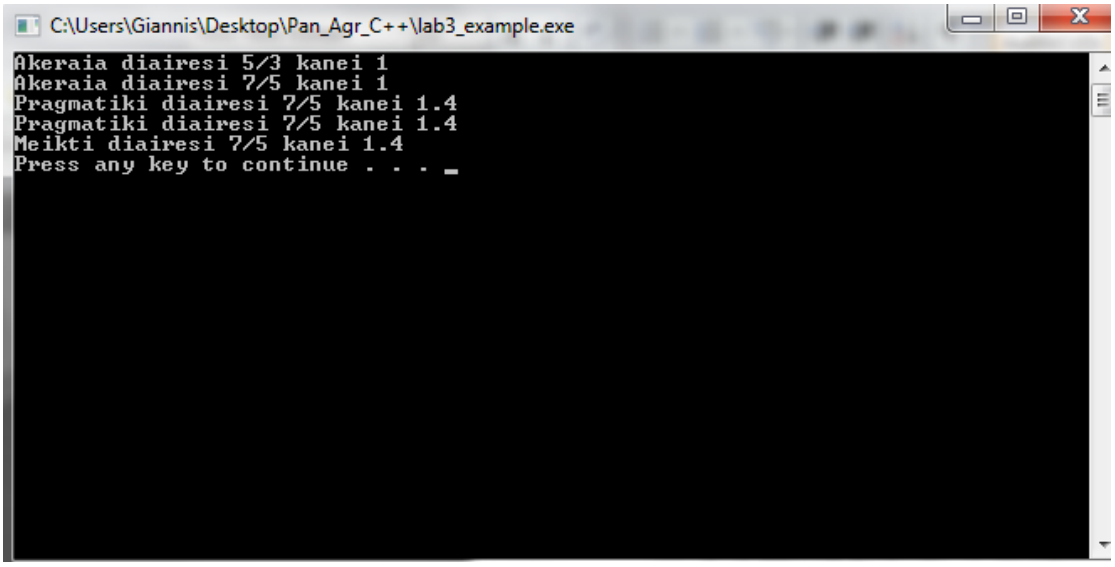
ΠΡΟΣΟΧΗ! Στη διαίρεση πρέπει να έχουμε υπόψη μας ότι η διαίρεση μεταξύ δύο ακεραίων δίνει ακέραιο, δηλαδή το δεκαδικό μέρος αποκόπτεται!

Διαίρεση μεταξύ ακεραίου και πραγματικού (float ή double) δίνει πραγματικό.

Παράδειγμα 1: Γράψτε ένα πρόγραμμα που να εμφανίζει αποτελέσματα πράξεων

```
#include <iostream>
using namespace std;

/*****************************************************************
25/11/2012
Example
Telestes, Arithmitikes Prakseis, Proteraiotita Telestwn
Author: Giannis
*****************************************************************/

int main()
{

    cout << "Akeraia diairesi 5/3 kanei " << 5/3 <<endl;
    cout << "Akeraia diairesi 7/5 kanei " << 7/5 <<endl;
    cout << "Pragmatiki diairesi 7/5 kanei " << 7.0/5.0 <<endl;
    cout << "Pragmatiki diairesi 7/5 kanei " << 7./5. <<endl;
    cout << "Meikti diairesi 7/5 kanei " << 7./5 <<endl;

    system("pause");
    return 0;
}
```

Σχήμα 3.1: Παράδειγμα υπολογισμού εμβαδου και περιμέτρου κύκλου



Σχήμα 3.2: Έξοδος κονσόλας

Λογικοί Τελεστές και Πράξεις

Το λογικό ΚΑΙ (AND) παριστάνεται με "&&". Η παράσταση A&&B δίνει αποτέλεσμα αληθές αν και μόνο αν και το Α και το Β είναι αληθή.
Το λογικό Ή (OR) παριστάνεται με "||". Η παράσταση A||B δίνει αποτέλεσμα αληθές αν είτε το Α είτε το Β είναι αληθές, ή και τα δύο. Αλλιώς δίνει ψευδές.
Το λογικό ΟΧΙ (NOT) παριστάνεται με "!". Η παράσταση !A δίνει αποτέλεσμα αληθές αν και μόνο αν και το Α είναι ψευδές.
Για τη C++, το ψευδές συμβολίζεται με μηδέν (0), ενώ το αληθές με τη μονάδα (1). Συνεπώς η παράσταση 5>2 έχει τιμή (αποτέλεσμα) 1. Επίσης οποιαδήποτε μη μηδενική τιμή θεωρείται αληθής, π.χ. 6 && 7 δίνει 1, ενώ 6 && 0 δίνει 0.

Οι Τελεστές Μοναδιαίας Αύξησης ++ και Μείωσης –

Μπορεί να εμφανιστούν μέσα σε μια παράσταση και να συνοδεύουν μια μεταβλητή. Ο τελεστής αύξησης (++) αυξάνει κατά ένα την τιμή της μεταβλητής την οποία συνοδεύει, ενώ ο -- τη μειώνει κατά μια μονάδα. Οι τελεστές αυτοί υπάρχουν σε δύο μορφές : Το ++ ή το -- μπορεί να βρίσκεται:
a) Πρίν από την επηρεαζόμενη μεταβλητή, π.χ. ++n, (προθεματικός), οπότε αλλάζει την τιμή της μεταβλητής πριν να γίνει ο υπολογισμός της παράστασης,
b) Μετά από αυτήν, π.χ. n++, οπότε ονομάζεται επιθεματικός τελεστής, οπότε υπολογίζεται πρώτα η παράσταση και μετά γίνεται η αλλαγή τιμής της μεταβλητής. Η διαφορά φαίνεται παρακάτω:

```
int p=5, n=10; // αρχικοποίηση

y = p++ * --n;
```

πρώτα αλλάζει η τιμή του n (γιατί έχει προθεματικό --) και γίνεται 9, μετά γίνεται η πράξη y=p*n=5*9=45,
και μετά γίνεται η p++ δηλαδή p=p+1=6.

Παρατηρήσεις: Οι τελεστές αύξησης και μείωσης έχουν υψηλή προτεραιότητα και μόνο οι παρενθέσεις έχουν υψηλότερη.

ΠΡΟΣΟΧΗ: η έκφραση n++ είναι ισοδύναμη με την n=n+1 και όχι με την n+1. Είναι ένα λάθος που κάνουν συχνότατα οι αρχάριοι προγραμματιστές της C++.

Δεν πρέπει να χρησιμοποιούμε τελεστές αύξησης ή μείωσης σε μεταβλητή που εμφανίζεται περισσότερες από μία φορές σε μια έκφραση, π.χ. y = ++n + 2*n γιατί δεν μπορούμε να είμαστε σίγουροι για το αποτέλεσμα.

Οι Προτεραιότητες των Τελεστών

Στα μαθηματικά οι αριθμητικοί τελεστές έχουν διαφορετική προτεραιότητα μεταξύ τους όταν βρίσκονται πολλοί σε μια παράσταση χωρίς παρενθέσεις. Για παράδειγμα, ο πολλαπλασιασμός και η διαίρεση έχουν υψηλότερη προτεραιότητα από την πρόσθεση και την αφαίρεση. Αν 2 τελεστές διαίρεσης ή πολλαπλασιασμού βρίσκονται στην ίδια έκφραση, αφού έχουν ίδια προτεραιότητα, ο κανόνας λέει ότι εκτελούνται οι πράξεις από αριστερά προς τα δεξιά. Έτσι και η C++ τοποθετεί κάθε τελεστή σε κάποιο επίπεδο προτεραιότητας. Τα επίπεδα προτεραιότητας των τελεστών φαίνονται παρακάτω:

( )	→
++ -- -	←
* / %	→
+ -	→
< <= > >=	→
== !=	→
&&	→
||	→
=	←

Σχήμα 3.3: Προτεραιότητες Τελεστών

Ψηλότερα στον πίνακα σημαίνει υψηλότερη προτεραιότητα. Προφανώς οι παρενθέσεις έχουν την υψηλότερη από όλους, οπότε τις χρησιμοποιούμε αν δεν είμαστε βέβαιοι για τη σειρά εκτέλεσης των πράξεων. Επίσης έτσι ο κώδικας γίνεται πιο ευανάγνωστος. Το βέλος δείχνει την σειρά εκτέλεσης μεταξύ τελεστών με ίδια προτεραιότητα.

Αν υπάρχουν μέσα στην έκφραση τελεστές στο ίδιο επίπεδο, ισχύει γενικά προτεραιότητα από αριστερά προς τα δεξιά (εκτός του τελεστή ανάθεσης: πχ. x = y = 1 . πρώτα y=1 και μετά x=1) και των τελεστών μοναδιαίας μεταβολής (++, --) και του προσήμου (-)

Παράδειγμα: x = 5 > 3 && 9 = = 8; σημαίνει: x= ((5 μεγαλύτερο 3) ΚΑΙ (9 ίσο με 8)) άρα ψευδές,

Άρα το x θα γίνει μηδέν.

Υπονοούμενες Μετατροπές Τύπου (σε εκφράσεις)

Στη C++ μπορούμε να έχουμε μια αριθμητική παράσταση με μεταβλητές και σταθερές διαφορετικών τύπων δεδομένων. Η C++ χρησιμοποιεί ορισμένους κανόνες για να μετατρέψει αυτόματα τους τύπους και να δώσει αποτέλεσμα:

1. Όταν εμφανίζονται σε εκφράσεις ο τύπος char και ο τύπος short, με πρόσημο ή χωρίς, αυτόματα μετατρέπονται σε τύπο int.

2. Σε κάθε πράξη όπου εμπλέκονται δύο τύποι, οι δύο τιμές μετατρέπονται στον τύπο αυτής με τον "υψηλότερο" βαθμό.

3. Η ιεραρχία των τύπων από τους υψηλότερους προς τους χαμηλότερους είναι η εξής : long double, double, float, unsigned long, long, unsigned int και int.

4. Σε μια εντολή ανάθεσης το τελικό αποτέλεσμα των υπολογισμών μετατρέπεται στον τύπο της μεταβλητής στην οποία καταχωρήθηκε η τιμή. Έτσι, όμως, μπορεί μια τιμή να μετατραπεί σε τύπο χαμηλότερου βαθμού, όταν π.χ. καταχωρούμε τύπο float σε τύπο int και γίνεται, όπως είδαμε, αποκοπή του δεκαδικού μέρους.

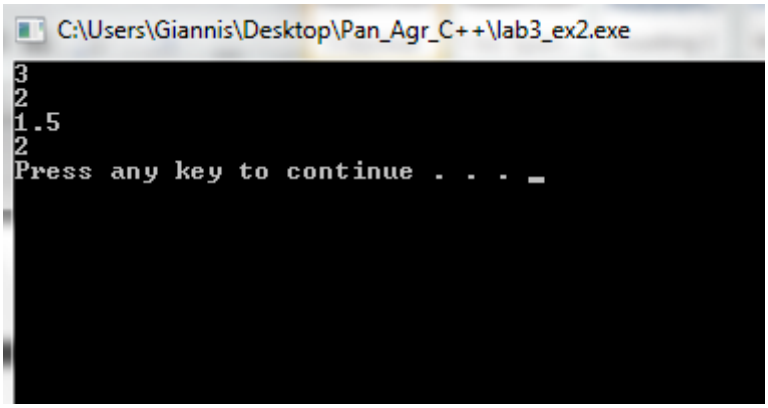
Ρητές μετατροπές τύπου (Casting)

Όλες οι μετατροπές τύπων που αναφέραμε μέχρι τώρα γίνονται αυτόματα. Όμως, είναι πιθανό να θέλουμε να δώσουμε εμείς τις οδηγίες για την ακριβή μετατροπή του τύπου που θέλουμε. Η μέθοδος αυτή λέγεται ρητή μετατροπή (casting) και συνίσταται στην τοποθέτηση μπροστά από την μεταβλητή (ή μια σταθερά, ή μια παράσταση μέσα σε παρενθέσεις), του επιθυμούμενου τύπου μέσα σε παρενθέσεις. Οι παρενθέσεις μαζί με το όνομα του τύπου αποτελούν τον τελεστή-casting.

Παράδειγμα 2: Κατανόηση του casting

```
#include <iostream>
using namespace std;
/*****************************************************************
25/11/2012
Katanoisi tis ritis metatropis - casting
Author: Giannis
*****************************************************************/
int main()
{
 int m;
 float k=1.5;
 float x;
 m = 1.6 + 1.7; // to m ginetai 3, diladi to akeraio meros toy 3.3
 cout << m <<endl;
 m = (int) 1.6 + (int) k; // t0 m ginetai 1+1=2
 cout << m <<endl;
 x = (int) k*k; // 1*1.5 = 1.5, ara to x ginetai 1.5
 cout << x <<endl;
 x = (int) (k*k); // 1.5*1.5=2.25, ara to x ginetai 2.0
 cout << m <<endl;
 system("pause");
 return 0;
}
```

Σχήμα 3.4: Παράδειγμα κατανόησης casting



Σχήμα 3.5: Έξοδος κονσόλας

3.2 Εργαστηριακές Ασκήσεις

Άσκηση 3.1. Να ορίσετε τους ακεραίους i, j, k και τους πραγματικούς x,y. Δώστε τιμές 1, 2 και 3 στα i, j, k αντίστοιχα και 3.0 και 5.0 στα x και y.

Ορίστε νέες μεταβλητές κατάλληλου τύπου και δώστες τους τιμές τα αποτελέσματα των παρακάτω πράξεων. Τα αποτελέσματα λογικών πράξεων καταχωρήστε τα σε bool μεταβλητές. Καταγράψτε τα αποτελέσματα που υπολογίσατε πριν τρέξετε την άσκηση και συγκρίνετε τα με τα αποτελέσματα του υπολογιστή.

a) i+j/x*y
b) k/j*++y
c) j>i
d) j>i&&!x>=--y
e) !x==0

Άσκηση 3.2 Υλοποιήστε πρόγραμμα σε C++ το οποίο:

α) Να ζητά από το χρήστη δύο πραγματικούς αριθμούς στο διάστημα [10,100), πρώτα το μεγαλύτερο και μετά το μικρότερο. Αφού τους δώσει ο χρήστης (θεωρήστε, για τις εκτυπώσεις, ότι οι 2 αριθμοί έχουν διψήφιο ακέραιο μέρος και μονοψήφιο δεκαδικό μέρος), το πρόγραμμα:

β) Να απαντά πόσο είναι το ακέραιο μέρος και ποιο το δεκαδικό μέρος για καθέναν αναλυτικά.

γ) Να εκτελεί και να εκτυπώνει τις 4 αριθμητικές πράξεις μεταξύ τους, καθώς και το ακέραιο πηλίκο και το υπόλοιπο της διαίρεσης των ακέραιων μερών τους.

Ακολουθεί παράδειγμα επιθυμητής συμπεριφοράς της αλληλεπίδρασης προγράμματος – χρήστη:.

```
Please enter the bigger of the two numbers:
40.7
Please enter the smaller of the two numbers:
10.3
The integer part of  40.7   is 40 and its fractional part is 0.7.
The integer part of   10.3   is  10 and its fractional part is 0.3.
40.7 + 10.3 =  51.00
40.7 - 10.3 =  30.40
40.7 x 10.3 =  419.21
40.7 / 10.3 =    3.95
40 / 10 = 4 and the remainder is 0
```