

ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ

C++ για Μαθηματικούς

Μια εισαγωγή για σπουδαστές και καθηγητές



Περιέχει δωρεάν
CD-ROM

Edward Scheinerman



Επιστημονική επιμέλεια ελληνικής έκδοσης:

Γιώργος Στεφανίδης

Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπ. Μακεδονίας

Περιεχόμενα

Προγράμματα.....	15
Εικόνες.....	19
Πρόλογος.....	21
I Διαδικασίες	29
1 Τα βασικά.....	31
1.1 Τι είναι η C++.....	31
1.2 Hello C++	33
1.3 Ασκήσεις.....	38
2 Οι αριθμοί.....	41
2.1 Οι τύποι των ακεραίων	41
2.2 Οι τύποι των πραγματικών αριθμών.....	45
2.3 Οι τύποι bool και char	45
2.4 Έλεγχος του μεγέθους και της χωρητικότητας των διαφόρων τύπων	46
2.5 Οι καθιερωμένες πράξεις.....	50
2.6 Συγκρίσεις και λογικές (Boolean) πράξεις	56
2.7 Μιγαδικοί αριθμοί	57
2.8 Ονοματολογία μεταβλητών	63
2.9 Ασκήσεις.....	64
3 Μέγιστος κοινός διαιρέτης	69
3.1 Το πρόβλημα.....	69
3.2 Μια πρώτη προσέγγιση.....	69
3.3 Η μέθοδος του Ευκλείδη.....	77
3.4 Επανάληψη με for, while και do.....	83
3.5 Μια εξαντλητική προσέγγιση στο πρόβλημα του ΜΚΔ.....	85
3.6 Διευρυμένος Ευκλείδειος αλγόριθμος, κλήση κατ' αναφορά και υπερφόρτωση.....	88
3.7 Ασκήσεις.....	93

4	Τυχαίοι αριθμοί.....	97
4.1	Παραγωγή ψευδοτυχαίων αριθμών	97
4.2	Ομοιόμορφες τυχαίες τιμές.....	98
4.3	Περισσότερα για την παραγωγή ψευδοτυχαίων αριθμών.....	102
4.4	Ένα πρόγραμμα Monte Carlo για το πρόβλημα του ΜΚΔ.....	106
4.5	Κανονικές τυχαίες τιμές.....	108
4.6	Ασκήσεις.....	111
5	Συστοιχίες.....	115
5.1	Η συνάρτηση φ του Euler.....	115
5.2	Τα θεμελιώδη των συστοιχιών	118
5.3	Μια διαδικασία παραγοντοποίησης ακεραίων	121
5.4	Μια διαδικασία για τον υπολογισμό της συνάρτησης φ του Euler	127
5.5	Το κόσκινο του Ερατοσθένη: new και delete[].....	130
5.6	Μια ταχύτερη totient.....	137
5.7	Ο υπολογισμός της p_n για μεγάλα n	139
5.8	Η απάντηση	141
5.9	Ασκήσεις.....	143
II	Αντικείμενα	147
6	Σημεία στο επίπεδο.....	149
6.1	Δεδομένα και μέθοδοι.....	149
6.2	Η δήλωση της κλάσης Point	151
6.3	Απόκρυψη δεδομένων	155
6.4	Κατασκευαστριες μέθοδοι.....	157
6.5	Ανάθεση και μετατροπή	159
6.6	Μέθοδοι	160
6.7	Διαδικασίες που χρησιμοποιούν ορίσματα τύπου Point.....	163
6.8	Τελεστές.....	165
6.9	Ασκήσεις.....	174
7	Πυθαγόρειες τριάδες	179
7.1	Παραγωγή Πυθαγόρειων τριάδων	179
7.2	Σχεδίαση μιας κλάσης πρωτευουσών Πυθαγόρειων τριάδων	180
7.3	Υλοποίηση της κλάσης PTriple	182
7.4	Εύρεση και ταξινόμηση των τριάδων	188
7.5	Ασκήσεις.....	193

8	Υποδοχείς	195
8.1	Σύνολα	195
8.2	Επαναλήπτες συνόλων.....	199
8.3	Πολυσύνολα.....	204
8.4	Προσαρμόσιμες συστοιχίες μέσω της κλάσης vector	205
8.5	Διατεταγμένα ζεύγη.....	211
8.6	Απεικονίσεις	212
8.7	Λίστες, στοίβες και αταξινόμητες ουρές	219
8.7.1	Λίστες.....	219
8.7.2	Στοίβες	224
8.7.3	Ουρές	225
8.7.4	Deque.....	226
8.7.5	Ουρές προτεραιότητας	228
8.8	Ασκήσεις.....	229
9	Αριθμητική υπολοίπων	235
9.1	Σχεδίαση του τύπου Mod	235
9.2	Ο κώδικας	237
9.3	Το προκαθορισμένο modulus: Στατικές μεταβλητές και στατικές μέθοδοι κλάσης.....	243
9.4	Κατασκευάστριες μέθοδοι και μέθοδοι λήψης/ορισμού	248
9.5	Τελεστές σύγκρισης.....	249
9.6	Αριθμητικοί τελεστές.....	251
9.7	Γραφή αντικειμένων Mod σε ρεύματα εξόδου	255
9.8	Μια διαδικασία main που επιδεικνύει την κλάση Mod.....	255
9.9	Ασκήσεις.....	257
10	Το προβολικό επίπεδο	261
10.1	Εισαγωγή στο προβολικό επίπεδο, $\mathbb{RP}^2$	261
10.2	Σχεδίαση των κλάσεων PPoint και PLine	264
10.3	Κληρονομικότητα	266
10.4	Προστατευμένα μέλη κλάσης.....	270
10.5	Οργάνωση κλάσεων και αρχείων για τις κλάσεις PPoint και PLine.....	273
10.6	Η γονική κλάση PObject.....	275
10.7	Οι κλάσεις PPoint και PLine	285
10.8	Ανακάλυψη και διόρθωση ενός σφάλματος	292

10.9	Και πάλι ο Πάππος.....	300
10.10	Ασκήσεις.....	305
11	Μεταθέσεις.....	311
11.1	Το πρόβλημα του Ulam	311
11.2	Σχεδιασμός της κλάσης Permutation.....	313
11.2.1	Δεδομένα.....	314
11.2.2	Κατασκευάστριες μέθοδοι και μέθοδοι αποσύνδεσης	315
11.2.3	Αντιγραφή και ανάθεση	318
11.2.4	Βασικές μέθοδοι εξέτασης και τροποποίησης	323
11.2.5	Λειτουργίες μεταθέσεων.....	323
11.2.6	Τελεστές σύγκρισης.....	325
11.2.7	Έξοδος.....	326
11.2.8	Το αρχείο κώδικα Permutation.cc	326
11.3	Εύρεση μονότονων υπακολουθιών.....	330
11.4	Ασκήσεις.....	335
12	Πολυώνυμα.....	339
12.1	Υποδείγματα διαδικασιών	339
12.2	Υποδείγματα κλάσεων.....	343
12.2.1	Χρήση υποδειγμάτων κλάσεων	343
12.2.2	Δημιουργία υποδειγμάτων κλάσεων.....	345
12.3	Το υπόδειγμα κλάσης Polynomial	349
12.3.1	Δεδομένα.....	349
12.3.2	Κατασκευάστριες μέθοδοι	350
12.3.3	Μέθοδοι λήψης και ορισμού.....	352
12.3.4	Μέθοδοι συναρτήσεων.....	353
12.3.5	Ισότητα.....	354
12.3.6	Αριθμητική.....	354
12.3.7	Έξοδος στην οθόνη.....	355
12.3.8	Μέγιστος Κοινός Διαιρέτης	355
12.3.9	Ο κώδικας	356
12.4	Και πάλι το πρόβλημα του ΜΚΔ.....	365
12.5	Δουλεύοντας στο δυαδικό σύστημα	369
12.5.1	Προσημασμένοι και μη προσημασμένοι ακέραιοι.....	369
12.5.2	Πράξεις με bit	370

12.5.3	Το υπόδειγμα κλάσης bitset	373
12.5.4	Υποδείγματα κλάσεων με άτυπα ορίσματα	377
12.6	Ασκήσεις.....	378
III	Θέματα.....	381
13	Χρήση άλλων πακέτων.....	383
13.1	Αριθμητική αυθαίρετης ακρίβειας: το πακέτο GMP	383
13.2	Γραμμική άλγεβρα.....	388
13.2.1	Διδιάστατες συστοιχίες στη C++	388
13.2.2	Τα πακέτα TNT και JAMA.....	389
13.2.3	Το πακέτο newmat.....	400
13.3	Άλλα πακέτα	405
13.4	Ασκήσεις.....	407
14	Συμβολοσειρές, Είσοδος/Εξοδος, και οπτικοποίηση	409
14.1	Συστοιχίες χαρακτήρων	409
14.2	Η κλάση string.....	412
14.2.1	Αρχικοποίηση	412
14.2.2	Θεμελιώδεις λειτουργίες.....	413
14.2.3	Αναζήτηση	417
14.2.4	Μετατροπή μεταξύ των τύπων string και char*	420
14.3	Ορίσματα γραμμής διαταγών.....	421
14.4	Ανάγνωση και εγγραφή δεδομένων σε αρχεία	425
14.4.1	Άνοιγμα αρχείων για είσοδο/έξοδο.....	425
14.4.2	Ανάγνωση και εγγραφή	429
14.4.3	Ανίχνευση του τέλους ενός αρχείου εισόδου.....	430
14.4.4	Άλλες μέθοδοι για είσοδο	431
14.5	Ρεύματα συμβολοσειρών.....	434
14.6	Μορφοποίηση	436
14.6.1	Ορισμός ακρίβειας	436
14.6.2	Εμφάνιση όλων των ψηφίων.....	437
14.6.3	Ορισμός του πλάτους.....	437
14.6.4	Άλλοι χειριστές	439
14.7	Μια κλάση για τη συντακτική ανάλυση αρχείων	440

14.8	Οπτική αναπαράσταση	444
14.8.1	Εισαγωγή και εγκατάσταση του πακέτου plotutils	445
14.8.2	Σχεδίαση με το πακέτο plotutils — ένα πρώτο παράδειγμα.....	448
14.8.3	Το τρίγωνο του Pascal modulo 2	455
14.8.4	Παρακολούθηση της κίνησης ενός σημείου που κινείται τυχαία σε ένα τρίγωνο	457
14.8.5	Σχεδίαση γράφων Paley	460
14.9	Ασκήσεις.....	465
15	Παραλειπόμενα	469
15.1	Η εντολή switch	469
15.2	Ετικέτες και η εντολή goto	473
15.3	Χειρισμός εξαιρέσεων	475
15.3.1	Τα βασικά των εντολών try, throw, και catch	476
15.3.2	Άλλα χαρακτηριστικά του συστήματος χειρισμού εξαιρέσεων	480
15.4	Φίλιες	484
15.5	Άλλοι τρόποι δημιουργίας τύπων	487
15.5.1	Δομές.....	487
15.5.2	Απαριθμήσεις.....	488
15.5.3	Ενώσεις	489
15.5.4	Χρήση της typedef	490
15.6	Δείκτες	491
15.6.1	Βασικά στοιχεία για τους δείκτες	492
15.6.2	Αποαναφοροποίηση	493
15.6.3	Συστοιχίες και αριθμητική δεικτών	495
15.6.4	Πάλι οι τελεστές new και delete	498
15.6.5	Γιατί να χρησιμοποιούμε δείκτες;.....	500
15.7	Ασκήσεις.....	502
IV	Παραρτήματα	507
A	Το περιβάλλον προγραμματισμού C++.....	509
A.1	Προγραμματισμός με παράθυρο διαταγών και διορθωτή κειμένου.....	510
A.1.1	Τι χρειάζεστε και πώς θα το αποκτήσετε (δωρεάν)	510
A.1.2	Σύνταξη αρχείων προγραμμάτων.....	512

A.1.3	Μεταγλώττιση και εκτέλεση του προγράμματός σας	513
A.1.4	Επιλογές μεταγλωττιστή	517
A.1.5	Εισαγωγή στο πρόγραμμα make	519
A.2	Προγραμματισμός με ένα ολοκληρωμένο περιβάλλον ανάπτυξης.....	523
A.2.1	Visual C++ για Windows.....	524
A.2.2	Xcode για Macintosh OS X	528
A.3	Γενικές συμβουλές για την αποσφαλμάτωση	532
B	Τεκμηρίωση με το Doxygen.....	537
	Τεκμηριώστε τον κώδικά σας.....	537
B.1	Εντολές του Doxygen	538
B.1.1	Τεκμηρίωση αρχείων	538
B.1.2	Τεκμηρίωση διαδικασιών.....	538
B.1.3	Τεκμηρίωση κλάσεων, δεδομένων, και μεθόδων	540
B.2	Χρήση του Doxygen.....	544
B.2.1	Διευθέτηση του Doxygen.....	544
B.2.2	Εκτέλεση του Doxygen.....	547
B.2.3	Περισσότερες λειτουργίες.....	548
Γ	Αναφορά της C++	551
Γ.1	Μεταβλητές και τύποι.....	551
Γ.1.1	Θεμελιώδεις τύποι.....	551
Γ.1.2	Τυπικές κλάσεις/υποδείγματα	551
Γ.1.3	Δήλωση μεταβλητών.....	552
Γ.1.4	Στατικές μεταβλητές και εμβέλεια.....	553
Γ.1.5	Σταθερές και η δεσμευμένη λέξη const	553
Γ.1.6	Συστοιχίες	554
Γ.2	Λειτουργίες.....	554
Γ.2.1	Ανάθεση.....	555
Γ.2.2	Αριθμητική.....	555
Γ.2.3	Τελεστές σύγκρισης.....	555
Γ.2.4	Λογικοί τελεστές.....	555
Γ.2.5	Τελεστές bit.....	556
Γ.2.6	Διάφορα	556
Γ.3	Εντολές ελέγχου	557
Γ.3.1	if-else.....	557
Γ.3.2	Εκτέλεση βρόχων: for, while και do.....	558

Γ.3.3	switch	559
Γ.3.4	goto.....	560
Γ.3.5	Εξαιρέσεις.....	560
Γ.4	Διαδικασίες.....	561
Γ.4.1	Οργάνωση αρχείων.....	561
Γ.4.2	Κλήση κατά τιμή και κλήση κατ' αναφορά	561
Γ.4.3	Συστοιχίες (και δείκτες) ως ορίσματα.....	562
Γ.4.4	Προκαθορισμένες τιμές ορισμάτων	563
Γ.4.5	Πρότυπα.....	563
Γ.4.6	Διαδικασίες inline.....	564
Γ.5	Κλάσεις.....	564
Γ.5.1	Επισκόπηση και οργάνωση αρχείων.....	564
Γ.5.2	Κατασκευαστριες μέθοδοι και μέθοδοι αποσύνδεσης.....	565
Γ.5.3	Τελεστές.....	566
Γ.5.4	Αντιγραφή και ανάθεση.....	568
Γ.5.5	Δεδομένα και μέθοδοι static.....	570
Γ.5.6	this.....	571
Γ.5.7	Φίλιες.....	571
Γ.5.8	Υποδείγματα κλάσεων.....	572
Γ.5.9	Κληρονομικότητα	573
Γ.6	Καθιερωμένες συναρτήσεις.....	575
Γ.6.1	Μαθηματικές συναρτήσεις.....	575
Γ.6.2	Μαθηματικές σταθερές.....	578
Γ.6.3	Διαδικασίες χαρακτήρων	578
Γ.6.4	Άλλες χρήσιμες συναρτήσεις.....	580
Δ	Απαντήσεις.....	583
	Ευρετήριο.....	687

Κεφάλαιο 7

Πυθαγόρειες τριάδες

Πυθαγόρεια τριάδα είναι μια λίστα τριών ακεραίων (a, b, c) τέτοιων, ώστε $a^2 + b^2 = c^2$. Μια τέτοια τριάδα λέγεται *πρωτεύουσα* αν οι ακεραίοι είναι μη αρνητικοί και $\gcd(a, b, c) = 1$. Παραδείγματος χάριν, η $(3, 4, 5)$ είναι μια Πυθαγόρεια τριάδα επειδή $3^2 + 4^2 = 5^2$. Αν και οι $(30, 40, 50)$ και $(-3, 4, -5)$ είναι επίσης Πυθαγόρειες τριάδες, δεν είναι πρωτεύουσες.

Σε αυτό και το επόμενο κεφάλαιο αναπτύσσουμε το μηχανισμό της C++ που είναι απαραίτητος για να βρούμε όλες τις πρωτεύουσες Πυθαγόρειες τριάδες με $a, b, c \leq 100$ (ή οποιαδήποτε άλλη δεδομένη τιμή).

7.1 Παραγωγή Πυθαγόρειων τριάδων

Υπάρχει μια απλή μέθοδος παραγωγής Πυθαγόρειων τριάδων. Έστω z ένας ακεραίος του Gauss· δηλαδή, $z = m + ni$ όπου $m, n \in \mathbb{Z}$. Θεωρούμε το $|z^4|$. Είναι

$$|z^4| = (|z|^2)^2 = (m^2 + n^2)^2$$

αλλά

$$|z^4| = |z^2|^2 = |(m^2 - n^2) + (2mn)i|^2 = (m^2 - n^2)^2 + (2mn)^2.$$

Επομένως

$$(m^2 - n^2)^2 + (2mn)^2 = (m^2 + n^2)^2. \quad (*)$$

Επειδή $m, n \in \mathbb{Z}$, έπεται ότι το $(m^2 - n^2, 2mn, m^2 + n^2)$ είναι μια Πυθαγόρεια τριάδα. Αυτό μπορεί επίσης να ελεγχθεί απευθείας με ανάπτυξη των δύο μελών της ταυτότητας (*).

Παρότι κάθε τριάδα της μορφής $(m^2 - n^2, 2mn, m^2 + n^2)$ είναι μια Πυθαγόρεια τριάδα, δεν προκύπτουν όλες οι Πυθαγόρειες τριάδες με τον τρόπο αυτό.

Παραδείγματος χάριν, η (9, 12, 15) δεν μπορεί να εκφραστεί έτσι. [Απόδειξη. Αν η (9, 12, 15) ήταν αυτής της μορφής, τότε $2mn = 12$, οπότε $mn = 6$. Αυτό συνεπάγεται ότι $\{m, n\} = \{\pm 2, \pm 3\}$ ή $\{\pm 1, \pm 6\}$, κανένα από τα οποία δεν δίνει (9, 12, 15).]

Μπορούμε όμως να δείξουμε ότι κάθε *πρωτεύουσα* Πυθαγόρεια τριάδα είναι της μορφής $(m^2 - n^2, 2mn, m^2 + n^2)$. Χρησιμοποιούμε το γεγονός αυτό για να δημιουργήσουμε μια κλάση η οποία αντιπροσωπεύει πρωτεύουσες Πυθαγόρειες τριάδες.

7.2 Σχεδίαση μιας κλάσης πρωτευουσών Πυθαγόρειων τριάδων

Θέλουμε να σχεδιάσουμε μια κλάση της C++ που να αντιπροσωπεύει τις πρωτεύουσες Πυθαγόρειες τριάδες. Η πρώτη ερώτηση είναι: Πώς θα ονομάσουμε αυτή την κλάση; Το πλέον περιγραφικό όνομα είναι το φλύαρο `PrimitivePythagoreanTriple`, αλλά θα ήταν επώδυνο να το πληκτρολογούμε επανειλημμένα. Στο άλλο άκρο, θα μπορούσαμε να ονομάσουμε την κλάση `PPT`, αλλά θα αποκρύπταμε κάθε πληροφορία για το περιεχόμενό της. Καταλήγουμε σε μια συμβιβαστική λύση: `PTriple`.

Πώς θα διατηρεί μια `PTriple` τα δεδομένα της; Η ευκολότερη λύση είναι να αποθηκεύσουμε την τριάδα (a, b, c) . Αν και θα μπορούσαμε να παραλείψουμε το c επειδή μπορεί να υπολογιστεί από τα a και b , δεν αποτελούν πρόβλημα τα λίγα bit μνήμης που χρησιμοποιούνται για την αποθήκευση του c . Δεν επιθυμούμε να θεωρούμε την (3, 4, 5) διαφορετική από την οποιαδήποτε μετάθεση της τριάδας (3, 4, 5), οπότε ας συμφωνήσουμε ότι μια `PTriple` θα κρατά τους τρεις ακεραίους σε μη φθίνουσα σειρά. Οι τρεις ακέραιοι αποθηκεύονται σε μεταβλητές τύπου `long` με ονόματα a , b και c .

Κάθε κλάση χρειάζεται μια κατασκευάστρια μέθοδο. Σύμφωνα με όσα είπαμε, η κατασκευάστρια μέθοδος θα πρέπει να επεξεργάζεται ένα ζεύγος ακεραίων (m, n) για να παράγει την τριάδα $(m^2 - n^2, 2mn, m^2 + n^2)$. Δυστυχώς, η ζωή δεν είναι τόσο απλή όσο η ανάθεση

$$a \leftarrow m^2 - n^2, \quad b \leftarrow 2mn, \quad c \leftarrow m^2 + n^2.$$

Κατ' αρχήν, κάτι τέτοιο θα είχε ως ενδεχόμενο αποτέλεσμα το a ή το b να είναι αρνητικό. Ακόμα κι αν είναι και τα δύο θετικά, μπορεί να μην είναι στη σωστή σειρά (θέλουμε $0 \leq a \leq b$). Τέλος, οι τρεις τιμές μπορεί να μην είναι σχετικώς

πρώτοι αριθμοί. Το μόνο για το οποίο μπορούμε να είμαστε βέβαιοι είναι ότι το $m^2 + n^2$ είναι μη αρνητικός ακέραιος και ο μεγαλύτερος από τους τρεις.

Για να αντιμετωπίσει αυτά τα ζητήματα, η κατασκευάστρια μέθοδος χρειάζεται να διορθώσει τα πρόσημα των a και b , να τους βάλει στη σωστή σειρά, και να διαιρέσει με το μέγιστο κοινό διαιρέτη $\gcd(a, b)$.

Και κάτι τελευταίο. Σε περίπτωση που ισχύει $m = n = 0$, η τριάδα που κατασκευάζεται θα είναι η $(0, 0, 0)$. Τεχνικά, αυτή δεν είναι πρωτεύουσα επειδή οι καταχωρίσεις δεν είναι σχετικώς πρώτοι αριθμοί. Έχουμε δύο επιλογές: μπορούμε να επιτρέψουμε η $(0, 0, 0)$ να είναι μια έγκυρη `PTriple`, ή μπορούμε να το απαγορεύσουμε. Και στις δύο περιπτώσεις, η κατασκευάστρια μέθοδος πρέπει να χειριστεί αυτή την πιθανότητα ως μια ειδική περίπτωση. Στο βιβλίο αυτό αποφασίζουμε να θεωρήσουμε την $(0, 0, 0)$ ως μια έγκυρη `PTriple`.

Χρειάζεται επίσης να έχουμε μια κατασκευάστρια μέθοδο χωρίς ορίσματα. Είναι αυτή που καλείται μέσω μιας δήλωσης της μορφής `PTriple P`; Τι είδους Πυθαγόρεια τριάδα θα πρέπει να δημιουργεί αυτή; Με άλλα λόγια, ποια θα είναι η προκαθορισμένη πρωτεύουσα Πυθαγόρεια τριάδα; Υπάρχουν τρεις φυσικές επιλογές: $(0, 0, 0)$, $(0, 1, 1)$, ή $(3, 4, 5)$. Η τελευταία είναι η μικρότερη τριάδα στην οποία κανένα από τα στοιχεία δεν είναι μηδέν. Καλώς ή κακώς, αποφασίζουμε η προκαθορισμένη να είναι η $(0, 0, 0)$. Αν δεν είστε ικανοποιημένοι με οποιαδήποτε απ' αυτές τις επιλογές μπορείτε ασφαλώς να επιλέξετε κάτι διαφορετικό.

Τι μεθόδους χρειαζόμαστε; Χρειάζεται να ξέρουμε τις τιμές που είναι αποθηκευμένες στις a , b και c , οπότε προσανατολιζόμαστε για μεθόδους `getA()`, `getB()`, και `getC()` που θα αναφέρουν αυτές τις τιμές. Δεν θέλουμε μεθόδους οι οποίες να μπορούν να τροποποιούν αυτές τις τιμές, επειδή δεν θέλουμε να επιτρέψουμε στο χρήστη να θέσει μια `PTriple` σε μια μη έγκυρη κατάσταση. Αν είχαμε μια μέθοδο `setA`, τότε η `P.setA(2)`; θα είχε ως αποτέλεσμα μια `PTriple` η οποία θα μπορούσε να μην είναι μια Πυθαγόρεια τριάδα.

Θέλουμε επίσης τελεστές που να συγκρίνουν τριάδες `PTriple`. Εκτός από τους τελεστές `==` και `!=`, ορίζουμε έναν τελεστή `<`. Χρειαζόμαστε έναν τελεστή μικρότερο από, έτσι ώστε να μπορούμε να ταξινομούμε τριάδες `PTriple` σε μια συστοιχία και να διαγράφουμε διπλότυπα. (Να θυμόμαστε ότι ο σκοπός μας είναι να βρούμε όλες τις πρωτεύουσες Πυθαγόρειες τριάδες με $0 \leq a \leq b \leq c \leq 1000$.) Έχουμε κάποια επιλογή για το πώς να ορίσουμε τον τελεστή `<` για τριάδες `PTriple`: χρειάζεται απλώς να εξασφαλίσουμε ότι ο ορισμός του τελεστή `<` οδηγεί σε μια πλήρη διάταξη. Η απλή λύση μας είναι να διατάζουμε τις τριάδες λεξικογραφικά. Θέτουμε $(a, b, c) < (a', b', c')$ αν, $c < c'$, ή διαφορετικά,

αν $c = c'$ και $b < b'$. Φυσικά, αν $c = c'$ και $b = b'$, τότε $a = a'$ και οι τριάδες ταυτίζονται.

Τέλος είναι χρήσιμο να ορίσουμε τον τελεστή `<<` έτσι ώστε να μπορούμε να εμφανίζουμε τριάδες `PTriple` στην οθόνη του υπολογιστή.

Είμαστε τώρα σε θέση να υλοποιήσουμε την κλάση `PTriple`.

7.3 Υλοποίηση της κλάσης `PTriple`

Όπως σχεδιάστηκε, η κλάση `PTriple` έχει τρία στοιχεία δεδομένων τύπου `long` με ονόματα a , b , και c . Οι κατασκευάστριες μέθοδοι εξασφαλίζουν ότι ισχύουν πάντα οι ακόλουθες συνθήκες.

$$0 \leq a \leq b \leq c, \quad \gcd(a, b) = 1, \quad a^2 + b^2 = c^2. \quad (*)$$

Η μόνη εξαίρεση είναι ότι επιτρέπουμε να ισχύει $(a, b, c) = (0, 0, 0)$.

Δημιουργούμε την κλάση ώστε να έχει δύο κατασκευάστριες μεθόδους: μία που δεν δέχεται ορίσματα και μία που δέχεται δύο ορίσματα τύπου `long`. Η πρώτη κατασκευάζει την $(0, 0, 0)$ και η δεύτερη χρησιμοποιεί τη μέθοδο που περιγράψαμε στην αρχή του κεφαλαίου. Είναι αρμοδιότητα της κατασκευάστριας μεθόδου να διασφαλίσει ότι ισχύουν οι συνθήκες της (*). Αυτό το πα-
ραπέμπουμε σε μια βοηθητική μέθοδο η οποία ονομάζεται `reduce`. (Κυριολεκτώντας, δε χρειαζόμαστε μια ξεχωριστή διαδικασία γι' αυτά τα βήματα, όμως θέλουμε να δώσουμε την εικόνα ενός στιγμιότυπου μιας ιδιωτικής μεθόδου για μια κλάση.)

Οι δημόσιες μέθοδοι της `PTriple` είναι οι `getA`, `getB`, `getC` και οι τελεστές `==`, `!=` και `<`. Τέλος, έχουμε μια διαδικασία `operator<<` η οποία δεν είναι μέλος της κλάσης `PTriple`.

Παρουσιάζουμε τώρα το αρχείο κεφαλίδας `PTriple.h`. Σε αυτή την περίπτωση, δεν έχουμε διαγράψει τα σχόλια. Αυτό το αρχείο κεφαλίδας εισάγει κάποιες νέες ιδέες με τις οποίες θα ασχοληθούμε μετά την παρουσίαση του αρχείου.

Πρόγραμμα 7.1: Αρχείο κεφαλίδας για την κλάση `PTriple`.

```

1 #ifndef PTRIPLE_H
2 #define PTRIPLE_H
3 #include <iostream>
4 using namespace std;
5
```

```
6  /**
7   * Μια PTriples αντιπροσωπεύει μια ανηγμένη Πυθαγόρεια τριάδα. Δηλαδή μια
8   * ακολουθία τριών μη αρνητικών ακεραίων (a,b,c) σε μη φθίνουσα
9   * διάταξη, τέτοια ώστε  $a^2+b^2=c^2$  και οι (a,b,c) είναι σχετικώς
10  * πρώτοι. Η απαίτηση "σχετικώς πρώτοι" σημαίνει ότι ασχολούμαστε μόνο
11  * με πρωτεύουσες Πυθαγόρειες τριάδες. Επιτρέπουμε την τριάδα (0,0,0).
12  */
13  class PTriples {
14  private:
15      /// Η μικρότερη πλευρά της τριάδας
16      long a;
17      /// Η μεγαλύτερη πλευρά της τριάδας
18      long b;
19      /// Η υποτείνουσα
20      long c;
21
22  /**
23   * Αυτή η ιδιωτική μέθοδος διασφαλίζει ότι τα στοιχεία της τριάδας
24   * είναι μη αρνητικοί, σε μη φθίνουσα διάταξη, και σχετικώς πρώτοι.
25   */
26   void reduce();
27
28  public:
29      /**
30       * Προκαθορισμένη κατασκευάστρια μέθοδος. Δημιουργεί την τριάδα (0,0,0).
31       */
32       PTriples() {
33           a = b = c = 0;
34       }
35
36       /**
37        * Κατασκευή από ένα ζεύγος ακεραίων. Δοθέντων των ακεραίων m και n,
38        * δημιουργούμε μια Πυθαγόρεια τριάδα παίρνοντας τις πλευρές να είναι 2mn
39        * και  $m^2-n^2$  και την υποτείνουσα να είναι  $m^2+n^2$ . Μετά
40        * σιγουρευόμαστε ότι οι τρεις αριθμοί είναι μη αρνητικοί, σε μη
41        * φθίνουσα διάταξη, και μετά διαιρούμε με τον ΜΚΔ τους.
42        * Π.χ., η PTriples(2,1) δημιουργεί τη γνωστή τριάδα (3,4,5).
43        */
44       PTriples(long m, long n);
45
46       /// Ποια είναι η μικρότερη πλευρά αυτής της τριάδας;
47       long getA() const {
48           return a;
49       }
50
51       /// Ποια είναι η μεγαλύτερη πλευρά αυτής της τριάδας;
52       long getB() const {
```

```

53     return b;
54 }
55
56 /// Ποια είναι η υποτείνουσα αυτής της τριάδας;
57 long getC() const {
58     return c;
59 }
60
61 /**
62  * Ελέγχουμε αν αυτή η τριάδα PTriples είναι μικρότερη από μια άλλη. Η διάταξη
63  * είναι λεξικογραφική αρχίζοντας με το c, και μετά το b. Δηλαδή πρώτα
64  * συγκρίνουμε υποτείνουσες. Αν είναι ίσες συγκρίνουμε τη μεγαλύτερη πλευρά.
65  * @param that Μια άλλη PTriples
66  * @return true αν αυτή η PTriples είναι λεξικογραφικά μικρότερη
67  */
68 bool operator<(const PTriples& that) const;
69
70 /**
71  * Ελέγχουμε αν αυτή η PTriples είναι ίση με μια άλλη.
72  */
73 bool operator==(const PTriples& that) const {
74     return ( a==that.a) && (b==that.b) && (c==that.c) );
75 }
76
77 /**
78  * Ελέγχουμε αν αυτή η PTriples δεν είναι ίση με μια άλλη.
79  */
80 bool operator!=(const PTriples& that) const {
81     return !( (*this) == that );
82 }
83 };
84
85 /// Στέλνουμε μια PTriples σε ένα ρεύμα εξόδου
86 ostream& operator<<(ostream& os, const PTriples& PT);
87
88 #endif

```

Η δήλωση κλάσης αρχίζει στη γραμμή 13. Εντός της ενότητας `private` βλέπουμε τις τρεις μεταβλητές τύπου `long` που κρατούν τα δεδομένα της τριάδας. Έχοντας στο μυαλό μας ένα ορθογώνιο τρίγωνο με μήκη πλευρών a , b , c , αποκαλούμε το a μικρότερη πλευρά, το b μεγαλύτερη πλευρά και το c υποτείνουσα.

Μέσα επίσης στην ενότητα `private` υπάρχει η μέθοδος `reduce()`. Αυτή είναι μια μέθοδος που καλείται από την κατασκευάστρια μέθοδο των δύο ορισμάτων `PTriple(m,n)`. Θα αναφερθούμε σ' αυτή με περισσότερες λεπτομέρειες παρακάτω.

Η ενότητα `public` αρχίζει στη γραμμή 28 με τις δύο κατασκευάστριες μεθόδους. Η άνευ ορίσματος κατασκευάστρια μέθοδος `PTriple()` δεν δηλώνεται όπως θα περιμέναμε. Κανονικά, μια δήλωση διαδικασίας δεν περιλαμβάνει τον κώδικα της διαδικασίας. Όλο κι όλο που περιμένουμε είναι αυτό:

```
PTriple();
```

Αντί γι' αυτό, βλέπουμε εκείνα που θα περιμέναμε κανονικά να βρούμε στο αρχείο `.cc`.

```
PTriple() {
    a = b = c = 0;
}
```

Πρόκειται για ένα ιδιαίτερο χαρακτηριστικό της C++. Μπορούν να ορίζονται διαδικασίες σε αρχεία κεφαλίδας, και αυτές είναι γνωστές ως *ενδογραμμικές* (*inline*) διαδικασίες. Η ενδογραμμική διαδικασία συνδυάζει τη δήλωση και τον ορισμό μιας διαδικασίας σε μία μόνο θέση στο αρχείο κεφαλίδας.

Μπορούμε να μετατρέψουμε οποιαδήποτε διαδικασία σε ενδογραμμική διαδικασία. Η δεσμευμένη λέξη `inline` χρησιμοποιείται για να ανακοινώσει αυτό το γεγονός. Για διαδικασίες όμως που είναι μέλη κλάσεων (δηλαδή μεθόδους) η δεσμευμένη λέξη είναι προαιρετική. Για όλες τις άλλες διαδικασίες (δηλαδή συνηθισμένες μεθόδους που δεν είναι μέλη κλάσεων) η δεσμευμένη λέξη `inline` είναι υποχρεωτική.

Πότε θα πρέπει να χρησιμοποιείτε ενδογραμμικές διαδικασίες, αντί της συνηθούς τεχνικής μιας δήλωσης στο αρχείο `.h` και ενός ορισμού στο αρχείο `.cc`; Αν ο κώδικας της μεθόδου είναι μόνο μία γραμμή ή δύο, τότε είναι καλή ιδέα να τη γράφετε ως ενδογραμμική διαδικασία. Αρκετές από τις μεθόδους της `PTriple` είναι ιδιαίτερα σύντομες και τις γράφουμε όντως ως ενδογραμμικές διαδικασίες. Άλλες (όπως η κατασκευάστρια μέθοδος με τα δύο ορίσματα ή η `operator<`) είναι πιο περίπλοκες, οπότε γι' αυτές χρησιμοποιούμε τη συνηθισμένη τεχνική.

Υπάρχουν τα υπέρ και τα κατά στο να γράφουμε ενδογραμμικό κώδικα. Είναι βολικό να γράφουμε τον κώδικα για μια μέθοδο στο ίδιο αρχείο όπου ορίζεται. Ο μεταγλωττιστής της C++ παράγει αποδοτικότερο κώδικα για διαδικασίες που είναι γραμμένες ενδογραμμικά. (Πάντως, οι έξυπνοι μεταγλωττιστές

της C++ μπορούν να καταλάβουν πότε αξίζει τον κόπο να μετατρέπουν μια συνηθισμένη διαδικασία σε ενδογραμμική διαδικασία.) Όμως, ο αντικειμενικός κώδικας για προγράμματα που χρησιμοποιούν ενδογραμμικές διαδικασίες πιάνει κατά τι περισσότερο χώρο στο δίσκο. Επίσης, παίρνει λίγο περισσότερο χρόνο η μεταγλώττιση προγραμμάτων με ενδογραμμικές διαδικασίες.

Μερικές φορές είναι υποχρεωτικό να γράψουμε μεθόδους ως ενδογραμμικές διαδικασίες. Όταν έρθει η κατάλληλη στιγμή θα το εξηγήσουμε.

Οι διαδικασίες που δεν καθορίζονται από ενδογραμμικό κώδικα στο αρχείο κεφαλίδας ορίζονται στο αρχείο `PTriple.cc` που παρουσιάζουμε στη συνέχεια.

Πρόγραμμα 7.2: Αρχείο προγράμματος για την κλάση `PTriple`.

```
1 #include "PTriple.h"
2 #include "gcd.h"
3
4 PTriple::PTriple(long m, long n) {
5     a = 2*m*n;
6     b = m*m - n*n;
7     c = m*m + n*n;
8
9     reduce();
10 }
11
12 void PTriple::reduce() {
13     // Δεν γίνεται τίποτα αν a=b=c=0
14     if ((a==0) && (b==0) && (c==0)) return;
15
16     // Διασφαλίζουμε ότι a,b είναι μη αρνητικοί (ο c πρέπει να είναι)
17     if (a<0) a = -a;
18     if (b<0) b = -b;
19
20     // Διασφαλίζουμε ότι a <= b
21     if (a>b) {
22         long tmp = a;
23         a = b;
24         b = tmp;
25     }
26
27     // Διασφαλίζουμε ότι οι a,b,c είναι σχετικώς πρώτοι
28     long d = gcd(a,b);
29     a /= d;
30     b /= d;
```

```

31   c /= d;
32 }
33
34 bool PTriples::operator<(const PTriples& that) const {
35     if (c < that.c) return true;
36     if (c > that.c) return false;
37
38     if (b < that.b) return true;
39     return false;
40 }
41
42 ostream& operator<<(ostream& os, const PTriples& PT) {
43     os << "(" << PT.getA() << "," << PT.getB() << ","
44         << PT.getC() << ")";
45     return os;
46 }

```

Ο κώδικας για την κατασκευάστρια μέθοδο `PTriples(m,n)` διασπάται σε δύο μέρη. Δοθέντων των ακεραίων εισόδου m, n θέτουμε

$$a \leftarrow 2mn, \quad b \leftarrow m^2 - n^2, \quad \text{και} \quad c \leftarrow m^2 + n^2.$$

Μετά χρειάζεται να ρυθμίσουμε ορισμένα πράγματα. Χρειάζεται να σιγουρέψουμε ότι οι a και b είναι μη αρνητικοί, ότι $a \leq b$, και ότι $\gcd(a, b, c) = 1$. Αυτές τις δουλειές τις παραπέμπουμε στην ιδιωτική μέθοδο `reduce` (η οποία καλείται από την κατασκευάστρια μέθοδο `PTriples` στη γραμμή 9).

Ακολουθεί ο κώδικας για τον τελεστή `<`. Ο τελεστής αυτός χρησιμοποιείται για να συγκρίνει μια δεδομένη `PTriples` με μια άλλη. Αν R και P είναι μεταβλητές τύπου `PTriples`, η παράσταση $R < P$ έχει ως αποτέλεσμα να εκτελεστεί η διαδικασία `operator<` που ανήκει στο R .

Πρώτα συγκρίνουμε τις τιμές του c στα R και P : αυτό συμβαίνει στις γραμμές 35–36. Η πρώτη σύγκριση είναι $c < \text{that}.c$. Το πρώτο c είναι η υποτείνουσα του ορίσματος του αριστερού μέλους του `<` (δηλαδή R). Το "ακέφαλο" c αναφέρεται στο c του αντικειμένου επί του οποίου κλήθηκε η μέθοδος. Το c του ορίσματος του δεξιού μέλους χρειάζεται να καθοριστεί, οπότε πρέπει να αναφερόμαστε σ' αυτό ως `that.c` επειδή το όρισμα του δεξιού μέλους ονομάζεται `that`. Επειδή το P μεταβιβάζεται κατ' αναφορά στον τελεστή `<` (απαιτείται για τους τελεστές), η `that.c` είναι ακριβώς η ίδια μεταβλητή με την $P.c$.

Εν συντομία: όταν εμπλέκεται η $R < P$, το "ακέφαλο" c στη γραμμή 35 είναι το μέλος δεδομένων c του R και το `that.c` στη γραμμή 35 είναι το μέλος δεδομένων c του P .

7.4 Εύρεση και ταξινόμηση των τριάδων

Ο σκοπός μας είναι να βρούμε όλες τις Πυθαγόρειες τριάδες (a, b, c) με $0 \leq a \leq b \leq c \leq 1000$. Η στρατηγική μας είναι η εξής.

- Πρώτον, δημιουργούμε μια συστοιχία τριάδων `PTriple` καλώντας την `PTriple(m, n)` για ένα κατάλληλα μεγάλο διάστημα τιμών των m και n . Επειδή κάθε πρωτεύουσα Πυθαγόρεια τριάδα μπορεί να προκύψει από την $(m^2 - n^2, 2mn, m^2 + n^2)$ δεν χρειάζεται να θεωρήσουμε τιμές του m ή του n μεγαλύτερες του $\sqrt{1000}$.

Το άνω φράγμα 1000 είναι φυσικά αυθαίρετο. Η σχεδίασή μας επιτρέπει οποιοδήποτε άνω φράγμα N .

- Καθώς παράγονται οι Πυθαγόρειες τριάδες, τις αποθηκεύουμε σε μια συστοιχία. Επειδή το άνω φράγμα των a, b, c είναι αυθαίρετο (το N καθορίζεται από το χρήστη), η συστοιχία που δημιουργούμε χρειάζεται να δεσμεύεται δυναμικά.

Πόσο μεγάλη θα πρέπει να είναι αυτή η συστοιχία; Στο σενάριο χειρότερης περίπτωσης, όλες οι Πυθαγόρειες τριάδες που παράγουμε μπορεί να είναι πρωτεύουσες και διαφορετικές η μία με την άλλη. (Αυτή είναι στην πραγματικότητα μια χονδροειδής υπερεκτίμηση, αλλά εξυπηρετεί τους σκοπούς μας.) Επειδή κάνουμε επαναλήψεις επί όλων των m και n με $1 \leq m, n \leq \sqrt{N}$, μια συστοιχία που μπορεί να φιλοξενήσει N τιμές είναι αρκετά μεγάλη.

- Άπαξ και συμπληρωθεί η συστοιχία, την ταξινομούμε. Στην εκ των προτέρων ανάγκη της ταξινόμησης οφείλεται το ότι ορίσαμε τον τελεστή `<`.
- Τελικώς, διαβάζουμε όλη τη συστοιχία εμφανίζοντας στην οθόνη τα μοναδικά στοιχεία.

Παρουσιάζουμε το πρόγραμμα που εκτελεί όλα αυτά τα βήματα: μετά την παρουσίαση του κώδικα εξηγούμε τα κύρια χαρακτηριστικά του.

Πρόγραμμα 7.3: Ένα πρόγραμμα για την εύρεση Πυθαγόρειων τριάδων.

```
1 #include "PTriple.h"
2 #include <iostream>
3 #include <cmath>
4 using namespace std;
5 /**
6  * Βρίσκει όλες τις πρωτεύουσες Πυθαγόρειες τριάδες (a,b,c) με
7  *  $0 \leq a \leq b \leq c \leq N$  όπου το N καθορίζεται από το χρήστη.
8  */
9
10 int main() {
11     PTriple* table;    // πίνακας για να αποθήκευση των τριάδων
12     long N;            // άνω φράγμα στις τριάδες.
13
14     // Ζητά από το χρήστη το N
15     cout << "Please enter N (upper bound on triples) --> ";
16     cin >> N;
17     if (N <= 0) return 0; // δεν κάνει τίποτα όταν το N δεν είναι θετικό
18
19     // Δεσμεύει χώρο για τον πίνακα
20     table = new PTriple[N];
21
22     // Συμπληρώνει τον πίνακα με όλες τις πιθανές τριάδες PTriple
23     long idx = 0;      // δείκτης μέσα στον πίνακα
24     long rootN = long( sqrt(double(N)) );
25
26     for (long m=1; m<=rootN; m++) {
27         for (long n=1; n<=rootN; n++) {
28             PTriple P = PTriple(m,n);
29             if (P.getC() <= N) {
30                 table[idx] = P;
31                 idx++;
32             }
33         }
34     }
35
36     // Ταξινομεί τον πίνακα
37     sort(table, table+idx);
38
39     // Εμφανίζει στην οθόνη (όχι διπλότυπα) στοιχεία του πίνακα
40     cout << table[0] << endl;
41     for (int k=1; k<idx; k++) {
42         if (table[k] != table[k-1]) {
43             cout << table[k] << endl;
44         }
45     }
46 }
```

```

47 // Ελευθερώνει μνήμη που καταλαμβάνόταν από τον πίνακα
48 delete[] table;
49 return 0;
50 }

```

Σημείωση: Ορισμένοι μεταγλωττιστές μπορεί να απαιτούν την οδηγία `#include <algorithm>` προκειμένου να χρησιμοποιήσουν τη διαδικασία `sort` (γραμμή 37).

Για την ανάλυση τώρα του κώδικα.

- Χρειαζόμαστε έναν πίνακα για τις τριάδες `PTriple`, οπότε δηλώνουμε μια μεταβλητή `table` τύπου `PTriple*` (δείκτης σε ένα `PTriple`, δηλαδή την κεφαλή μιας συστοιχίας). Επειδή δεν γνωρίζουμε πόσο μεγάλη χρειάζεται να είναι αυτή η συστοιχία, δεν μπορούμε να τη δηλώσουμε με μια εντολή της μορφής `PTriple table[100];`.
- Οι γραμμές 15–17 ζητούν από το χρήστη να μας δώσει έναν ακέραιο N . Αν ο χρήστης μας δώσει έναν ακέραιο ο οποίος είναι μικρότερος του 1, τερματίζουμε το πρόγραμμα.
- Η γραμμή 20 δεσμεύει χώρο για τη συστοιχία `table`.
- Η μεταβλητή `idx`, που δηλώνεται στη γραμμή 23, χρησιμοποιείται για προσπέλαση στοιχείων της συστοιχίας `table`.
- Η μεταβλητή `rootN` ορίζεται ίση με $\lfloor \sqrt{N} \rfloor$. Η σύνταξη, δυστυχώς, είναι άβολη. Η διαδικασία `sqrt` της C++ (η οποία μπορεί να απαιτεί την οδηγία `#include <cmath>`) δέχεται και επιστρέφει τιμές τύπου `double`. Έτσι, χρειάζεται πρώτα να μετατρέψουμε ρητά (`cast`) την τιμή της N σε `double`, να υπολογίσουμε την τετραγωνική της ρίζα, και μετά να τη μετατρέψουμε πίσω σε τύπο `long`. Όταν μια μεταβλητή τύπου `double` μετατρέπεται σε τύπο `long`, τα ψηφία δεξιά της υποδιαστολής παραβλέπονται. Επομένως, η γραμμή 24 υπολογίζει το επιθυμητό.
- Οι γραμμές 26–34 διατρέχουν όλες τις δυνατές τιμές των m, n με $1 \leq m, n \leq \sqrt{N}$. Για κάθε ζεύγος, παράγουμε μια πρωτεύουσα Πυθαγόρεια τριάδα. Αν η c -τιμή της δεν είναι μεγαλύτερη του N (γραμμή 29) εισάγουμε την τιμή της στον πίνακα και αυξάνουμε κατά 1 την `idx`. Επομένως, αφού τελειώσουμε αυτόν το διπλό βρόχο `for`, η μεταβλητή `idx` έχει το πλήθος των καταχωρίσεων που κάναμε στη συστοιχία `table`.

- Η ταξινόμηση του πίνακα λαμβάνει χώρα στη γραμμή 37. Η διαδικασία `sort` της C++ μπορεί να χρησιμοποιηθεί σε οποιαδήποτε συστοιχία στοιχείων, αρκεί ο τύπος αυτών των στοιχείων να επιδέχεται σύγκριση με έναν τελεστή `<`. Έτσι, η `sort` μπορεί να χρησιμοποιηθεί για την ταξινόμηση μιας συστοιχίας ακεραίων τύπου `long` ή `double`. Για να ταξινομήσουμε μια συστοιχία τριάδων `PTriple`, χρειάζεται απλώς να εξασφαλίσουμε έναν τελεστή `<`.

Η διαδικασία `sort` μπορεί να απαιτεί την κεφαλίδα `algorithm`.

Η επίκληση της ταξινόμησης γίνεται με μια ανορθόδοξη σύνταξη: `sort(table, table+idx);`. Το πρώτο όρισμα, `table`, έχει νόημα. Η διαδικασία `sort` χρειάζεται να γνωρίζει τι συστοιχία έχει να ταξινομήσει. Το δεύτερο όρισμα είναι που δυσκολευόμαστε να καταλάβουμε.

Το δεύτερο όρισμα στη `sort` χρειάζεται να είναι ένας δείκτης προς τη θέση του πρώτου στοιχείου *πέρα από το τέλος της συστοιχίας*. Με άλλα λόγια, αν η συστοιχία έχει πέντε στοιχεία (δεικτοδοτημένα από 0 έως 4), τότε το δεύτερο όρισμα στη `sort` χρειάζεται να είναι ένας δείκτης προς το ανύπαρκτο *έκτο* στοιχείο της συστοιχίας (αριθμοδείκτης 5).

Γενικότερα, η `sort` μπορεί να χρησιμοποιηθεί για την ταξινόμηση ενός συνεχόμενου μπλοκ στοιχείων μιας συστοιχίας. Το πρώτο όρισμα στη `sort` θα πρέπει να είναι ένας δείκτης στην αρχή του μπλοκ και το δεύτερο όρισμα θα πρέπει να είναι ένας δείκτης *μετά* το τέλος του μπλοκ.

Ξέρουμε ότι το `table` είναι ένας δείκτης προς το πρώτο στοιχείο της συστοιχίας `table`. Δηλαδή, το `table` έχει τη διεύθυνση μνήμης του `table[0]`. Στη C++, το `table+1` παίρνει ως τιμή τη διεύθυνση μνήμης που έχει το `table[1]`. Έτσι το `table+idx` είναι η θέση μνήμης που έχει το `table[idx]`. Όμως, σε αυτό το σημείο του προγράμματός μας, το `idx` έχει έναν αριθμό μεγαλύτερο από τη θέση όπου τοποθετήσαμε τελευταία μια `PTriple`. Αυτό είναι που θέλει η `sort` και έτσι αυτό είναι που κά-νουμε.

Τελευταία γραμμή: Για την ταξινόμηση μιας συστοιχίας, με όνομα έστω `table`, η οποία περιέχει έστω 25 στοιχεία, η εντολή που χρησιμοποιούμε είναι αυτή: `sort(table, table+25);`. Δεν υπάρχει κανένα πρόβλημα να ξεχάσουμε ότι είπαμε γύρω από τους δείκτες και απλώς να θυμόμαστε το εξής:

```
sort(table_name, table_name + number_of_elements);
```

- Οι γραμμές 40–45 διατρέχουν τα στοιχεία της `table`. Αν το τρέχον στοιχείο είναι διαφορετικό από το προηγούμενο, το εμφανίζουμε στην οθόνη.
- Στο τέλος, στη γραμμή 48, ελευθερώνουμε τη μνήμη που έχει δεσμευτεί για την `table`. Μιλώντας αυστηρά, αυτό δεν είναι αναγκαίο επειδή βρίσκμαστε στο τέλος του προγράμματος και θα γινόταν αυτόματα η ανάκτηση της μνήμης. Παρόλα αυτά, είναι μια καλή συνήθεια να διασφαλίζουμε ότι κάθε `new` εντολή αντιστοιχίζεται με μια εντολή `delete[]`.

Αυτό που βλέπουμε όταν εκτελείται το πρόγραμμα (για $N = 100$) είναι το εξής.

```
Please enter N (upper bound on triples) --> 100
(0,1,1)
(3,4,5)
(5,12,13)
(8,15,17)
(7,24,25)
(20,21,29)
(12,35,37)
(9,40,41)
(28,45,53)
(11,60,61)
(33,56,65)
(16,63,65)
(48,55,73)
(36,77,85)
(13,84,85)
(39,80,89)
(65,72,97)
```

Βλέπουμε ότι υπάρχουν 17 πρωτεύουσες Πυθαγόρειες τριάδες με $0 \leq a \leq b \leq c \leq 100$. Αυτό συνεπάγεται ότι η συστοιχία που δημιουργήσαμε, η `table`, χρησιμοποίησε περισσότερη μνήμη απ' ότι στην πραγματικότητα χρειαζόμασταν.

Ας πούμε και τις ενστάσεις μας για αυτό το πρόγραμμα.

- Σπαταλάει μνήμη. Αυτό δεν αποτελεί τρομερό πρόβλημα επειδή η αποθήκευση εκατοντάδων, ή ακόμα και εκατοντάδων χιλιάδων Πυθαγόρειων τριάδων, είναι στις δυνατότητες ακόμα και των πιο μέτριων υπολογιστών. Ωστόσο, μπορεί να συναντήσουμε άλλες καταστάσεις στις οποίες χρειάζεται να είμαστε προσεκτικοί ως προς την ποσότητα μνήμης που χρησιμοποιούμε.

- Ήταν ενοχλητικό ότι χρειαζόταν να υπολογίζουμε πόση μνήμη πρέπει να θέτουμε κατά μέρος στην `table`. Θα ήταν πολύ ευκολότερο αν ο πίνακας μπορούσε να προσαρμόζει το μέγεθός του για να ικανοποιεί τις ανάγκες μας, αντί να απαιτείται από μας να υπολογίζουμε πόσο μεγάλο να τον κάνουμε.
- Η κλήση της `sort` με μπερδεύει ακόμα. Η προσθήκη ενός αντικειμένου τύπου `PTriple*` και ενός αντικειμένου τύπου `long` φαίνεται απλώς εσφαλμένη. (Δεν είναι λάθος, αλλά προκαλεί σύγχυση.)

Θα ασχοληθούμε με αυτές τις ενστάσεις στο επόμενο κεφάλαιο.

7.5 Ασκήσεις

- 7.1 Δημιουργήστε μια κλάση `Interval` που αντιπροσωπεύει κλειστά διαστήματα στην ευθεία των πραγματικών αριθμών: $[a, b] = \{x \in \mathbb{R} : a \leq x \leq b\}$. Υλοποιήστε πλήρως αυτή την κλάση σε ένα αρχείο `Interval.h` χωρίς κανένα αρχείο κώδικα. Για να το πετύχετε αυτό θα χρειαστεί να κάνετε όλες τις μεθόδους και τις διαδικασίες ενδογραμμαμικές.

Η κλάση θα πρέπει να περιλαμβάνει τα ακόλουθα χαρακτηριστικά.

- (α) Δύο κατασκευάστριες μεθόδους: μία κατασκευάστρια μέθοδο χωρίς όρισμα η οποία κατασκευάζει ένα προκαθορισμένο διάστημα (έστω το $[0, 1]$) και μία κατασκευάστρια μέθοδο δύο ορισμάτων που κατασκευάζει το διάστημα με τα καθορισμένα άκρα. Ως απόκριση είτε στην `Interval(3, 4)` είτε στην `Interval(4, 3)` θα πρέπει να κατασκευάζεται το διάστημα $[3, 4]$.
 - (β) Μέθοδοι `get` για να αποκαλύπτουν τα άκρα του διαστήματος.
 - (γ) Τελεστές σύγκρισης για ισότητα (`==`) και ανισότητα (`!=`).
 - (δ) Μια διαδικασία `operator<` για λεξικογραφική ταξινόμηση των διαστημάτων. Δηλαδή, $[a, b] < [c, d]$, αρκεί $a < c$ ή ($a = c$ και $b < d$).
 - (ε) Μια διαδικασία `operator<` για έξοδο στην οθόνη.
- 7.2 Στην Άσκηση 7.1 δημιουργήσατε μια κλάση `Interval`. Χρησιμοποιήστε αυτή την κλάση για να εξετάσετε το ακόλουθο πρόβλημα. Έστω $I_1, I_2, \dots, I_n$ μια συλλογή τυχαίων διαστημάτων. Ποια είναι η πιθανότητα ότι ένα από αυτά τα διαστήματα τέμνει όλα τα άλλα;

C++ για Μαθηματικούς

Μια εισαγωγή για σπουδαστές και καθηγητές

Στις περιπτώσεις προβλημάτων που απαιτούν εκτεταμένους υπολογισμούς, ένα πρόγραμμα C++ μπορεί να διατρέξει δεκάτομμυρια παραδειγμάτων ταχύτερα από τις περισσότερες άλλες επιλογές υπολογισμού. Η C++ δίνει τη δυνατότητα στους μαθηματικούς να δημιουργούν προγράμματα πολύ γρήγορα, ενώ είναι διαθέσιμη στα περισσότερα συστήματα υπολογιστών χωρίς επιπλέον κόστος. Το βιβλίο **C++ για Μαθηματικούς: Μια εισαγωγή για σπουδαστές και καθηγητές** δίνει έμφαση σε έννοιες της C++ που είναι πολύτιμες τόσο για τη θεωρητική όσο και για την εφαρμοσμένη μαθηματική έρευνα.

Αυτό το βιβλίο είναι το πρώτο που κυκλοφορεί για τον προγραμματισμό σε C++ και απευθύνεται ειδικά σε μαθηματικό ακροατήριο: παραλείπει τα «σκοτεινότερα» χαρακτηριστικά της γλώσσας προκειμένου να αναδείξει εκείνες τις πλευρές της που έχουν μεγαλύτερη χρηστική αξία για μαθηματικές εργασίες. Ο συγγραφέας εξηγεί πώς να χρησιμοποιείτε τη C++ για να διατυπώνετε εικασίες, πώς να δημιουργείτε εικόνες και διαγράμματα, πώς να επαληθεύετε αποδείξεις, πώς να κατασκευάζετε μαθηματικές δομές, και πώς να εξερευνάτε μυριάδες παραδειγμάτων. Επειδή ακριβώς δίνει έμφαση στον ουσιαστικό ρόλο της *πρακτικής εξάσκησης* κατά τη διαδικασία εκμάθησης, το βιβλίο είναι ιδανικό για προπτυχιακούς φοιτητές — αλλά και για προσωπική μελέτη. Σε κάθε κεφάλαιο περιλαμβάνονται πολλά προβλήματα και λύσεις, έτσι ώστε ο αναγνώστης να θεμελιώσει καλύτερα τις γνώσεις του, ενώ το συνοδευτικό CD-ROM περιέχει όλα τα αριθμημένα προγράμματα ώστε ο αναγνώστης να μπορεί να τα χρησιμοποιήσει αμέσως ή να τα προσαρμόσει στις ανάγκες του.

Χαρακτηριστικά του βιβλίου

- Παρουσιάζει τη γλώσσα προγραμματισμού C++ μέσω παραδειγμάτων που αναδεικνύουν τον τρόπο με τον οποίο χρησιμοποιείται η C++ στη μαθηματική έρευνα
- Περιγράφει πώς μπορεί να χρησιμοποιηθεί η C++ σε διαφορετικά συστήματα υπολογιστών, έτσι ώστε ο αναγνώστης να είναι σε θέση να αρχίσει τον προγραμματισμό αμέσως
- Παρέχει μια σαφή και περιεκτική εισαγωγή της αντικειμενοστρεφούς μεθόδου προγραμματισμού
- Εξετάζει διάφορα σχετικά θέματα — όπως την είσοδο/έξοδο, την οπτική αναπαράσταση, και την τεκμηρίωση — καθώς και επιλεγμένα ειδικά χαρακτηριστικά της γλώσσας C++
- Περιλαμβάνει έναν οδηγό γρήγορης αναφοράς στη γλώσσα C++

Παρουσιάζοντας σαφείς εξηγήσεις και παραδείγματα από τον κόσμο των μαθηματικών, και αναπτύσσοντας τις έννοιες από τη βάση τους, το βιβλίο **C++ για Μαθηματικούς** μπορεί να χρησιμοποιηθεί ανά πάσα στιγμή ως πηγή πληροφοριών για την εφαρμογή της C++ σε κάθε είδους προβλήματα — από τα πλέον βασικά έως τα πλέον πολύπλοκα.

Επισκεφθείτε μας στο Internet
www.klidarithmos.gr



ΕΚΔΟΣΕΙΣ
ΚΛΕΙΔΑΡΙΘΜΟΣ

Διοικητικό 4, Σταθμός Αττικής, 10440 Αθήνα, Τηλ. 210-5237605

ISBN 978-960-461-322-9



9 789604 613229