Προχωρημενος Προγραμματισμος I : Εισαγωγη στη C++

Θέματα Εξετάσεων Ιανουαρίου 2011

1. Υλοποιείστε τον αλγόριθμο `std::merge`. Γράψτε δηλαδή μια συνάρτηση που να συγχωνεύει δύο ήδη ταξινομημένες συλλογές (containers) στοιχείων δημιουργώντας άλλη, επίσης ταξινομημένη. Τα στοιχεία θα είναι οποιουδήποτε τύπου (αρκεί να έχει νόημα η σύγκρισή τους με τον τελεστή <). Οι δύο συλλογές εισόδου θα είναι ταξινομημένες με αύξουσα σειρά. Τέτοια ταξινόμηση θα πρέπει να έχει και η συλλογή εξόδου. 4/10

   Η συνάρτησή σας θα δέχεται πέντε iterators. Οι τέσσερις πρώτοι θα "δείχνουν" στην αρχή και στο "τέλος" της πρώτης συλλογής εισόδου και στην αρχή και στο "τέλος" της δεύτερης συλλογής εισόδου. Ο πέμπτος θα "δείχνει" στην αρχή της συλλογής εξόδου. Υποθέτουμε ότι η συλλογή εξόδου έχει αρκετές θέσεις διαθέσιμες ώστε να χωρέσουν όλα τα στοιχεία. Η συνάρτηση θα επιστρέφει ένα iterator που θα "δείχνει" στη θέση μετά το τελευταίο στοιχείο που γράφτηκε στη συλλογή εξόδου.

   Λάβετε υπόψη ότι οι δύο συλλογές εισόδου έχουν γενικά διαφορετικό πλήθος στοιχείων και διαφορετικό τύπο (και επομένως και διαφορετικούς iterators).

   Υπόδειξη: Συμπληρώστε το σώμα της συνάρτησης

```
template<typename iteratorA, typename iteratorB,
         typename iteratorC>
iteratorC
merge(iteratorA Abeg, iteratorA Aend,
      iteratorB Bbeg, iteratorB Bend,
      iteratorC Cbeg);
```

2. Υλοποιήστε τον αλγόριθμο ταξινόμησης mergesort εφαρμόζοντας την παρακάτω διαδικασία. Η συνάρτηση που θα γράψετε θα ταξινομεί από το μικρότερο στο μεγαλύτερο ένα `std::vector` με στοιχεία οποιουδήποτε τύπου (αρκεί να έχει νόημα η σύγκριση των στοιχείων με τον τελεστή <). 3/10

   Η διαδικασία έχει ως εξής:

   (α΄) Αν η λίστα εισόδου έχει 0 ή 1 στοιχείο, είναι ταξινομημένη. Αλλιώς:

   (β΄) Χωρίζουμε τη λίστα σε δύο περίπου ίσα μέρη.

   (γ΄) Ταξινομούμε κάθε νέο τμήμα με ξεχωριστή εφαρμογή της mergesort.

   (δ΄) Συγχωνεύουμε τις δύο ταξινομημένες λίστες με τέτοιο τρόπο ώστε η τελική να είναι επίσης ταξινομημένη. Σε αυτό το στάδιο θα χρειαστεί προσωρινή αποθήκευση είτε πριν είτε μετά τη συγχώνευση. Θα σας διευκολύνει αν αποθηκεύσετε τις προς ταξινόμηση λίστες σε προσωρινά `std::vector`. Κατόπιν, συγχωνεύετε τα προσωρινά `std::vector` στο αρχικό.

Προσέξτε ότι βασικό στοιχείο του αλγορίθμου είναι η διαδικασία της συγχώνευσης ταξινομημένων τμημάτων. Για αυτή χρησιμοποιήστε τη συνάρτηση που γράψατε στην προηγούμενη άσκηση. Αν δεν την κάνατε, χρησιμοποιήστε τον αλγόριθμο `std::merge`.

3. Ένας τρόπος να σχεδιάσουμε ένα διδιάστατο fractal είναι ο εξής: ξεκινάμε από ένα σημείο του επιπέδου, έστω το ($x = 0$, $y = 0$), και το μετακινούμε στη θέση (x', y') όπου 3/10

$$\begin{aligned} x' &= a \cdot x + b \cdot y + e \\ y' &= c \cdot x + d \cdot y + f \end{aligned}$$

και a, b, c, d, e, f σταθερές.

Το νέο σημείο το μεταφέρουμε με τον ίδιο μετασχηματισμό στο επόμενο σημείο του fractal (δηλαδή, θέτουμε $x' \to x$ και $y' \to y$ και παράγουμε το νέο (x', y')). Τη διαδικασία αυτή την επαναλαμβάνουμε επ' άπειρο. Η ακολουθία των σημείων (x, y) που παράγονται, αποτελεί το fractal.

Γράψτε ένα πρόγραμμα το οποίο:

(α') Θα διαβάζει από το αρχείο "in.dat" 4 γραμμές. Σε κάθε γραμμή θα υπάρχουν 7 πραγματικοί αριθμοί: οι 6 πρώτοι αντιστοιχούν στους συντελεστές a, b, c, d, e, f και ο τελευταίος στην πιθανότητα p να γίνει ο συγκεκριμένος μετασχηματισμός. Κάθε γραμμή αντιστοιχεί σε άλλο μετασχηματισμό. Το άθροισμα των πιθανοτήτων, $\sum p_i$, όλων των μετασχηματισμών είναι 1.

(β') Θα επιλέγει ένα τυχαίο πραγματικό αριθμό r στο διάστημα $[0, 1)$. Ανάλογα με την τιμή του θα εφαρμόζεται διαφορετικός μετασχηματισμός. Δηλαδή, αν ισχύει $0 \leq r < p_1$ θα εκτελείται ο πρώτος μετασχηματισμός, αν ισχύει $p_1 \leq r < p_1 + p_2$ θα εκτελείται ο δεύτερος κλπ.

(γ') Θα επαναλαμβάνει το προηγούμενο βήμα 1000 φορές σώζοντας κάθε φορά το σημείο που προκύπτει στο αρχείο "fractal.dat".

Δοκιμάστε το πρόγραμμά σας με τις εξής παραμέτρους στο "in.dat"

$$\begin{array}{rrrrrrr} 0 & 0 & 0 & 0.16 & 0 & 0 & 0.01 \\ 0.85 & 0.04 & -0.04 & 0.85 & 0 & 1.6 & 0.85 \\ 0.2 & -0.26 & 0.23 & 0.22 & 0 & 1.6 & 0.07 \\ -0.15 & 0.28 & 0.26 & 0.24 & 0 & 0.44 & 0.07 \end{array}$$

και

$$\begin{array}{rrrrrrr} 0 & 0 & 0 & 0.25 & 0 & -0.4 & 0.02 \\ 0.95 & 0.005 & -0.005 & 0.93 & -0.002 & 0.5 & 0.84 \\ 0.035 & -0.2 & 0.16 & 0.04 & -0.09 & 0.02 & 0.07 \\ -0.04 & 0.2 & 0.16 & 0.04 & 0.083 & 0.12 & 0.07 \end{array}$$

Αν θέλετε, μπορείτε να σχεδιάσετε τα "fractal.dat" που προκύπτουν.