ΠΕΜΠΤΗ ΔΙΑΛΕΞΗ

Σύνολο στοιχείων ίδιου τύπου (1/2)

Ένα σύνολο στοιχείων ίδιου τύπου διακρίνεται σε δύο κατηγορίες με βάση τη διάσταση:

Μονοδιάστατο Αν μπορούμε να θεωρούμε ότι τα στοιχεία είναι συνεχόμενα σε μία σειρά. Το σύνολο σε αυτή την περίπτωση ονομάζεται διάνυσμα.

Πολυδιάστατο Αν τα στοιχεία οργανώνονται σε δύο διαστάσεις (γραμμές και στήλες) ή περισσότερες. Το σύνολο ονομάζεται πίνακας.

Σύνολο στοιχείων ίδιου τύπου (2/2)

Ένα σύνολο στοιχείων ίδιου τύπου διακρίνεται σε δύο κατηγορίες με βάση το στάδιο (μεταγλώττιση ή εκτέλεση του προγράμματος) κατά το οποίο γίνεται η δημιουργία. Το σύνολο χαρακτηρίζεται ως:

Στατικό Αν

- το πλήθος των στοιχείων είναι γνωστό κατά τη μεταγλώττιση,
- το πλήθος των στοιχείων δεν πρόκειται να αλλάξει κατά την εκτέλεση του προγράμματος,
- το σύνολο δημιουργείται κατά τη μεταγλώττιση.

Δυναμικό Σε κάθε άλλη περίπτωση.

Στατικό διάνυσμα

Υλοποίηση στη C++11

Η C++ παρέχει στο `<array>` τη δομή `std::array<>` για την υλοποίηση στατικού διανύσματος.

Σύνταξη

`std::array`<τύπος,πλήθος> όνομα;

Παρατηρήσεις

- Το πλήθος στοιχείων πρέπει να είναι σταθερό και να μπορεί να υπολογιστεί κατά τη μεταγλώττιση.
- Η αρχική τιμή των στοιχείων είναι απροσδιόριστη.
- Πρόσβαση σε κάθε στοιχείο έχουμε με τον τελεστή `[]`.
- Η αρίθμηση των στοιχείων αρχίζει από το 0.
- Η ποσότητα που δηλώνεται αποτελεί μία, σύνθετη ποσότητα. Μπορούμε πάντα να υπολογίσουμε το πλήθος των στοιχείων μέσω αυτής, να αντιγραφεί ως σύνολο κλπ.

Στατικό διάνυσμα

Παράδειγμα στη C++11

Στατικό διάνυσμα 100 ακεραίων, με όνομα `a`, δηλώνεται ως εξής:

```
#include <array>

std::array<int,100> a;
```

- Το πρώτο στοιχείο είναι το `a[0]`, το δεύτερο το `a[1]`, ... το τελευταίο είναι το `a[99]`.
- Το πλήθος των στοιχείων είναι `a.size()`.

Στατικό διάνυσμα

Χρήση του `std::array<>` (1/2)

Δήλωση

Χωρίς συγκεκριμένη αρχική τιμή:

```
std::array<int,100> a;
```

Δήλωση με αρχική τιμή

```
std::array<double,10> a{};  // a[0]=a[1]=...=a[9] = 0.0
std::array<int,3> b{5,8,12};
auto c = b;
std::array<int,3> d{c};
std::array<int,4> e{3,6,9}; // e[3]=0;
```

Στατικό διάνυσμα

Χρήση του `std::array<>` (2/2)

Εκχώρηση τιμών

- Σε μεμονωμένα στοιχεία:

```
std::array<double,10> a;
a[4] = 2.31; a[8] = 1.2;
```

- Σε όλα μαζί τα στοιχεία:

```
std::array<double,10> a;
a.fill(2.3);
```

- Σε όλα μαζί τα στοιχεία με εκχώρηση άλλου array ίδιου πλήθους στοιχείων και κατάλληλου τύπου:

```
std::array<int,3> b{5,8,12};
std::array<int,3> c;
c = b;
```

Στατικό διάνυσμα

Υλοποίηση όπως κληρονομήθηκε από τη C (1/2)

Σύνταξη

τύπος όνομα[πλήθος];
τύπος όνομα[ ] = { τιμή1, τιμή2, ..., τιμήN };

Παρατηρήσεις

- Το πλήθος στοιχείων πρέπει να είναι σταθερό και να μπορεί να υπολογιστεί κατά τη μεταγλώττιση. Μπορεί να παραληφθεί αν δίνονται αρχικές τιμές.
- Η αρχική τιμή των στοιχείων είναι απροσδιόριστη στην πρώτη μορφή και συγκεκριμένη στη δεύτερη.
- Πρόσβαση σε κάθε στοιχείο έχουμε με τον τελεστή `[]`.
- Η αρίθμηση των στοιχείων αρχίζει από το 0.
- Η ποσότητα που δηλώνεται είναι ένας δείκτης στο πρώτο στοιχείο του συνόλου. Δεν μπορούμε πάντα να υπολογίσουμε το πλήθος των στοιχείων μέσω αυτού, τα στοιχεία δεν αντιγράφονται ως σύνολο, κλπ.

Στατικό διάνυσμα

Υλοποίηση όπως κληρονομήθηκε από τη C (2/2)

Παράδειγμα

Στατικό διάνυσμα 100 ακεραίων, με όνομα `a`, δηλώνεται ως εξής:

```
int a[100];
```

- Το πρώτο στοιχείο είναι το `a[0]`, το δεύτερο το `a[1]`, ...
  το τελευταίο είναι το `a[99]`.
- Το πλήθος των στοιχείων υπολογίζεται από την έκφραση

  ```
  sizeof(a) / sizeof(a[0])
  ```

  μόνο στη συνάρτηση στην οποία δηλώθηκε το `a`.

Δυναμικό διάνυσμα

Προτιμητέα Υλοποίηση

Η C++ παρέχει στο `<vector>` τη δομή `std::vector<>` για την υλοποίηση δυναμικού διανύσματος.

Σύνταξη

`std::vector<`τύπος`>` όνομα(πλήθος, αρχική_τιμή);

Παρατηρήσεις

- Το πλήθος στοιχείων μπορεί να είναι σταθερή ή μεταβλητή. Μπορεί να αλλάξει κατά την εκτέλεση του προγράμματος.
- Η αρχική τιμή των στοιχείων είναι συγκεκριμένη. Αν δεν προσδιοριστεί, δίνεται προεπιλεγμένη τιμή (0 για αριθμητικούς τύπους).
- Πρόσβαση σε κάθε στοιχείο έχουμε με τον τελεστή `[]`.
- Η αρίθμηση των στοιχείων αρχίζει από το 0.
- Η ποσότητα που δηλώνεται αποτελεί μία, σύνθετη ποσότητα. Μπορούμε πάντα να υπολογίσουμε το πλήθος των στοιχείων μέσω αυτής, να αντιγραφεί ως σύνολο κλπ.

Δυναμικό διάνυσμα

Παράδειγμα

Δυναμικό διάνυσμα πραγματικών, με όνομα a και πλήθος στοιχείων που διαβάζεται από το πληκτρολόγιο, δηλώνεται ως εξής:

```
#include <vector>
#include <iostream>
#include <cstddef>

std::size_t n;
std::cin >> n;
std::vector<double> a(n);
std::vector<double> b(2*n+3,1.5);
```

- Το πρώτο στοιχείο είναι το a[0], το δεύτερο το a[1], ...
  το τελευταίο είναι το a[n-1].
- Το πλήθος των στοιχείων είναι κάθε φορά a.size().
- Όλα τα στοιχεία του a είναι 0.0, όλα τα στοιχεία του b είναι 1.5.

Δυναμικό διάνυσμα

Εναλλακτικός μηχανισμός: **new/delete**[]

Δεσμεύεται μνήμη για ποσότητες κατάλληλου τύπου και πλήθους με την εντολή **new**. Η μνήμη αποδεσμεύεται με την εντολή **delete**[].

Παράδειγμα

```
#include <iostream>
#include <cstddef>

std::size_t n;
std::cin >> n;
double * a = new double[n];
... // χρήση του a
delete[] a;
```

- Το στοιχείο στη θέση `i` είναι το `a[i]`, με `i=0,1,...,n-1`.
- Το `a` είναι δείκτης στο πρώτο στοιχείο του συνόλου. Δεν μπορούμε πάντα να υπολογίσουμε το πλήθος των στοιχείων μέσω αυτού, τα στοιχεία δεν αντιγράφονται ως σύνολο, κλπ.

Χειρισμός διανύσματος

Για να διατρέξουμε ένα διάνυσμα χρησιμοποιούμε μία εντολή επανάληψης.

Παράδεγμα. Εισαγωγή και εκτύπωση τιμών

```
#include <array>
#include <iostream>
#include <cstddef>

std::array<int,10> a;
for (auto & x : a) {
   std::cin >> x;
}

for (std::size_t i{0}; i < a.size(); ++i) {
   std::cout << a[i] << '\n';
}
```

Στατικός πίνακας δύο διαστάσεων

Υλοποίηση όπως κληρονομήθηκε από τη C

Σύνταξη

τύπος όνομα[πλήθος1][πλήθος2];
τύπος όνομα[ ][πλήθος2] = { τιμή1, τιμή2, ..., τιμήN };

Παρατηρήσεις

- Τα πλήθη γραμμών και στηλών πρέπει να είναι σταθερά και να μπορούν να υπολογιστούν κατά τη μεταγλώττιση. Η πρώτη διάσταση μπορεί να παραληφθεί και να υπολογιστεί από το πλήθος των αρχικών τιμών (/ πλήθος2).
- Η αρχική τιμή των στοιχείων είναι απροσδιόριστη στην πρώτη μορφή, συγκεκριμένη στη δεύτερη.
- Οι αρχικές τιμές αποθηκεύονται κατά γραμμές.
- Η αρίθμηση των γραμμών και στηλών αρχίζει από το 0.
- Πρόσβαση σε κάθε στοιχείο έχουμε με τον τελεστή `[][]`: Π.χ. `a[i][j]`.
- Η ποσότητα που δηλώνεται είναι ένας δείκτης σε δείκτη στο πρώτο στοιχείο του συνόλου.

Δυναμικός πίνακας δύο διαστάσεων

Υλοποίηση ως διάνυσμα διανυσμάτων

Παράδειγμα

```
#include <iostream>
#include <cstddef>

std::size_t m, n;
std::cin >> m >> n;
double **a = new double*[m];
for (std::size_t i{0}; i < m; ++i) {
   a[i] = new double[n];
}
...//χρήση του a: a[i][j] το στοιχείο στη γραμμή i, στήλη j

for (std::size_t i{0}; i < m; ++i) {
   delete[] a[i];
}
delete[] a;
```

Πίνακας δύο διαστάσεων (1/3)

Σύνολο στοιχείων οργανωμένων σε $M = 3$ γραμμές και $N = 5$ στήλες:

	0	1	2	3	4
0	$(0,0)$ 0	$(0,1)$ 3	$(0,2)$ 6	$(0,3)$ 9	$(0,4)$ 12
1	$(1,0)$ 1	$(1,1)$ 4	$(1,2)$ 7	$(1,3)$ 10	$(1,4)$ 13
2	$(2,0)$ 2	$(2,1)$ 5	$(2,2)$ 8	$(2,3)$ 11	$(2,4)$ 14

Σχέσεις μεταξύ του δείκτη k, της γραμμής i και της στήλης j

```
k = i+3*j,      i = k%3,      j = k/3
```

Πίνακας δύο διαστάσεων (2/3)

Ένας διδιάστατος πίνακας A, με διαστάσεις $M \times N$ και στοιχεία A_{ij}, μπορεί να θεωρηθεί ως μονοδιάστατος (δηλαδή, διάνυσμα) με πλήθος στοιχείων $M * N$ και στοιχεία A_{i+M*j}.

Συνεπώς ένας διδιάστατος πίνακας που είναι

Στατικός Δηλώνεται ως `std::array<>` με $M * N$ στοιχεία.

Δυναμικός Δηλώνεται ως `std::vector<>` με $M * N$ στοιχεία.

Σε κάθε περίπτωση, το στοιχείο του διδιάστατου πίνακα στη θέση (i, j) είναι στη θέση $i + M * j$ του διανύσματος.

Πίνακας δύο διαστάσεων (3/3)

Πλεονεκτήματα

- Πολύ απλός τρόπος δημιουργίας δυναμικού διδιάστατου πίνακα.
- Τα στοιχεία είναι συνεχόμενα, αποθηκεύονται κατά στήλες και το διάνυσμα (που υλοποιεί πίνακα) μπορεί να περάσει ως όρισμα σε συναρτήσεις γραμμικής άλγεβρας της Fortran.

Μειονέκτημα

- Πρέπει να γνωρίζουμε (από ξεχωριστή ποσότητα) το πλήθος των γραμμών.

Χειρισμός πίνακα

Κάθε διάσταση χρειάζεται μία εντολή επανάληψης.
Σε πολυδιάστατο πίνακα οι εντολές είναι η μία μέσα στην άλλη.
Επιτρέπεται να τον χειριστούμε ως διάνυσμα και να έχουμε μία επανάληψη.

Παράδειγμα. Εισαγωγή και εκτύπωση τιμών

```
#include <vector>
#include <iostream>
#include <cstddef>

std::size_t m,n;
std::cin >> m >> n;
std::vector<double> b(m*n);
// Έστω ότι τα στοιχεία δίνονται κατά γραμμές
for (std::size_t i{0}; i < m; ++i) {
   for (std::size_t j{0}; j < n; ++j) {
      std::cin >> b[i+j*m];
   }
}
for (auto const & x : b) {
   std::cout << x << '\n'; // τα στοιχεία τυπώνονται κατά στήλες
}
```