ΥΠΟΛΟΓΙΣΤΕΣ ΙΙ

ΔΟΜΕΣ

πέρα απο απλές μεταβλητές...

- Μέχρι εδώ έχουμε δεί δύο τύπους μεταβλητών:
  - `int` (ακέραιος, 4 bytes)
  - `double` (ρητός διπλής ακρίβειας, 8 bytes)
- Σε πολλά προβλήματα χρειαζόμαστε πάνω απο μια μεταβλητή για να καθορίσουμε ένα μέγεθος. Π.χ.
  - διανυσμα (3 ρητοί: x, y, z)
  - μιγαδικός (2 ρητοί: real, imaginary)
  - άτομο (5 ρητοί: x, y, z, m, q)
- Πράξεις με αυτά τα μεγέθη σημαίνει
  - Πράξεις με πολλές μεταβλητές
  - το αποτέλεσμα εμπεριέχει πολλές μεταβλητές
- Είναι βολικό να δημιουργήσουμε νέες, πολυσύνθετες μεταβλητές

Ορισμός δομής

- Νέου τύπου μεταβλητή:
  - Ορίζεται απο τον προγραμματιστή
  - Εμπεριέχει οποιοδήποτε αριθμό και συνδιασμό άλλων μεταβλητών διαφόρων τύπων
- Στην δήλωση μιας δομής χρησιμοποιείται η δευσμευμένη λέξη της C++: `struct`
- Γενικός τρόπος δήλωσης

```
struct όνομα{
        τύπος μεταβλητή1 ;
        τύπος μεταβλητή2 ;
        .
        .
};
```


Παράδειγμα #1: διάνυσμα

```
struct vector{
        double x;
        double y;
        double z;
};
```

- Στο «όνομα» χρησιμοποιούμε κάτι που είναι περιγραφικό.
- Επιλέξαμε «vector». Στο εξής, το `vector` καθιστά νέο τύπο
  - δηλαδή, στο εξή υπάρχουν: `int, double, vector`
- Μέσα στο σώμα της δομής, ορίζουμε τα μέλη της δομής
  - τύπο και όνομα
  - χρησιμοποιούμε ονόματα που είναι περιγραφικά
  - για τις x,y,z συνιστώσες, χρησιμοποιούμε απλά `x, y, z.`
- Ο ορισμός δομής είναι εντολή, άρα τελειώνει με ερωτηματικό

Δήλωση μεταβλητών δομής

- Ο ορισμός της δομής γίνεται πρίν την `main`.
- Στο κυρίως πρόγραμμα δηλώνουμε μια μεταβλητή δομής όπως και στις απλές μεταβλητές
  - Π.χ. Για να δηλώσουμε δύο διανύσματα:

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

struct vector {
      double x, y, z;
};

int main(){
      vector v, u;
      .
      .
}
```


Προσπέλαση μελών δομής

- Πράξεις επιτρέπονται μόνο μεταξύ απλών μεταβλητών
  - δηλαδή μεταξύ των μελών της δομής
- Πως όμως φτάνουμε στα απλά μέλη της δομής;
  - χρησιμοποιούμε τον τελεστή της τελείας «.»
  - χρησιμοποιούμε τα ίδια σύμβολα με αυτά που ορίσαμε
- Π.χ. για ανάθεση τιμών σε διάνυσμα

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

struct vector {double x, y, z;};
int main(){
      vector v, u;
      v.x = 3.0;  v.y = 4.0;  v.z = 5.0;
      u.x = 11.0; u.y = 12.0; u.z = 13.0;
}
```


Είσοδος/έξοδος τιμών δομής

- Μπορούμε να εισάγουμε και κατευθείαν απο το πληκτρολόγιο
  Παράδειγμα:

```
cout<<"εισάγετε τις συνιστώσες διανύσματος"<< endl;
cin >> v.x >>  v.y >>  v.z;
```

- Μπορούμε να εξάγουμε τιμές στην οθόνη
  Παράδειγμα:

```
cout<<"οι συνιστώσες του διανύσματος είναι"<< endl;
cout << u.x <<" "<< u.y <<" "<<  u.z << endl;
```

ΠΡΟΣΟΧΗ: οι συναρτήσεις `cout` και `cin` δέχονται μόνο απλές μεταβλητές, ΟΧΙ δομές. Τα παρακάτω είναι ΛΑΘΟΣ:

```
cin >> v;
Cout << u;
```


Πράξεις μεταξύ δομών

- Μόνο η πράξη «=» επιτρέπεται μεταξύ μεταβλητών της ίδιας δομής.
  - γίνεται μια προς μια αντιγραφή των μελών της μιας στην άλλη

```
cin >> v.x >>  v.y >>  v.z;
u = v;
cout << u.x <<" "<< u.y <<" "<<  u.z << endl;
```

- Στην πράξη, αυτό που εκτελείται είναι

```
cin >> v.x >>  v.y >>  v.z;
u.x = v.x;
u.y = v.y;
u.z = v.z;
cout << u.x <<" "<< u.y <<" "<<  u.z << endl;
```

- Καμία άλλη πράξη, π.χ. «+, -, *, /» δεν επιτρέπεται

Παράδειγμα #1: πρόσθεση διανυσμάτων

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

struct vector {double x, y, z;};
int main(){
      vector v, u, w;
      cout <<"εισάγετε συνιστώσες 2 διανυσμάτων"<< endl;
      cin >> v.x >>  v.y >>  v.z;
      cin >> u.x >>  u.y >>  u.z;

      w.x = v.x + u.x;
      w.y = v.y + u.y;
      w.z = v.z + u.z;

      cout <<"οι συνιστώσες του αθροίσματος είναι"<< endl;
      cout << w.x <<" "<< w.y <<" "<<  w.z << endl;
}
```


Άλλα παραδείγματα δομών

- Σύμβολα και ονοματολογία είναι επιλογή του προγραμματιστή.
- Σημαντικό να περιγράφουν το μέγεθος στο οποίο αντιστοιχούν
  - Δομή για μιγαδικούς

```
struct complex{
      double real, imag;
};
```

  - Δομή για ιόντα

```
struct ion{
      double x, y, z, m, q;
};
```

  - Δομή για προϊόντα σε αποθήκη, που περιλαμβάνει κωδικό προϊόντος, έτος κατασκευής, τιμή πώλησης

```
struct item{
      int code, year;
      double price;
};
```


Πίνακες δομών

- Όταν ορίζουμε μια δομή, ορίζουμε μια νέου τύπου μεταβλητή
- Όπως και στις απλές μεταβλητές, μπορούμε να ορίσουμε πίνακα απο τις νέες μεταβητές
- Η προσπέλαση γίνεται πάλι με την τελεία, αμέσως μετά την δεικτοδότηση του πίνακα
- Παράδειγμα: δήλωση πίνακα με 1000 διανύσματα

```
int main(){
      vector v[1000];
      v[0].x = 3.0; v[0].y = 4.0; v[0].z = 5.0;
      v[1].x = 3.0; v[1].y = 4.0; v[1].z = 5.0;
      .
      .
}
```


Παράδειγμα #2: μέσος όρος του μέτρου διανυσμάτων (1/2)

Πρόγραμμα που διαβάζει n διανύσματα, και υπολογίζει τον μέσο όρο των μέτρων τους

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

struct vector {double x, y, z;};

int main(){
      int n; vector v[1000];
      cout << "εισάγετε τον αριθμό διανυσμάτων" << endl;
      cin >> n;

      if(n==0) cout << "ο μέσος όρος είναι 0" << endl;
      else if (n > 1000) cout << "πιό λίγα διανύσματα" <<endl;
      else {
            for (int i=0; i<n; ++i)
                  cin >> v[i].x >> v[i].y >> v[i].z;
```

Παράδειγμα #2: μέσος όρος του μέτρου διανυσμάτων (2/2)

```
      double s = 0;

      for (int i=0; i<n; ++i)
            s += sqrt(v[i].x * v[i].x +
                      v[i].y * v[i].y +
                      v[i].z * v[i].z);

      cout << "ο μέσος όρος είναι" << s/n << "\n";
      }
}
```


Δομές μέσα σε δομές

- Μια δομή συνιστά μια νέα μεταβλητή
- Με την σειρά της, μια δομή μπορεί να αποτελέσει μέλος μιας πιο σύνθετης δομής.
  - Παράδειγμα: ιόν με θέση, μάζα και φορτίο

```
struct vector {
      double x, y, z;
};

struct ion{
      vector r;
      double m, q;
};
```

- Διαδοχική προσπέλαση γίνεται με διαδοχικές τελείες

Παράδειγμα #3: Δομή για ιόντα (1/2)

Πρόγραμμα που διαβάζει n ιόντα, και υπολογίζει τη συνολική μάζα και φορτίο, και τη μέση απόσταση απο την αρχή των αξόνων

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

struct vector { double x, y, z; };

struct ion { vector r; double m, q; };


int main(){
      int n; ion a[1000];

      cout << "εισάγετε τον αριθμό των ιόντων" << endl;
      cin >> n;
```


Παράδειγμα #3: Δομή για ιόντα (2/2)

```
      if (n < 1 || n > 1000) return 1;

      for (int i=0; i<n; ++i)
            cin >> a[i].r.x >> a[i].r.y >> a[i].r.z >>
                  a[i].m >> a[i].q;

      double sm = 0; double sq = 0; double sd = 0;
      for (int i=0; i<n; ++i){
            sm += a[i].m;
            sq += a[i].q;
            sd += sqrt(v[i].r.x * v[i].r.x +
                       v[i].r.y * v[i].r.y +
                       v[i].r.z * v[i].r.z);}

      cout << "η συνολική μάζα είναι" << sm << endl;
      cout << "το συνολικό φορτίο είναι" << sq << endl;
      cout << "η μέση απόσταση είναι" << sd/n << endl;
}
```

Παράδειγμα #4: πράξεις με μιγαδικούς

Πρόγραμμα που διαβάζει δύο μιγαδικούς, και τυπώνει το άθροισμα, την διαφορά, και το γινόμενό τους

$$\tilde{z}_1 = z_{1r} + iz_{1i}$$
$$\tilde{z}_2 = z_{2r} + iz_{2i}$$

$$\tilde{z}_1 + \tilde{z}_2 = (z_{1r} + z_{2r}) + i(z_{1i} + iz_{2i})$$
$$\tilde{z}_1 - \tilde{z}_2 = (z_{1r} - z_{2r}) + i(z_{1i} - iz_{2i})$$

$$\tilde{z}_1 \cdot \tilde{z}_2 = (z_{1r} + iz_{1i}) \cdot (z_{2r} + iz_{2i}) =$$
$$(z_{1r}z_{2r} - z_{1i}z_{2i}) + i(z_{1r}z_{2i} + z_{1i}z_{2r})$$

Παράδειγμα #4: πράξεις με μιγαδικούς (1/2)

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

struct complex { double real, imag; };

int main(){
      complex z1, z2, zsum, zdif, zprod;

      cout << "εισάγετε δύο μιγαδικούς" << endl;
      cin >> z1.real >> z1.imag;
      cin >> z2.real >> z2.imag;

      zsum.real = z1.real + z2.real;
      zsum.imag = z1.imag + z2.imag;

      zdif.real = z1.real - z2.real;
      zdif.imag = z1.imag - z2.imag;
```


Παράδειγμα #4: πράξεις με μιγαδικούς (2/2)

```
      zprod.real = z1.real*z2.real - z1.imag*z2.imag;
      zprod.imag = z1.real*z2.imag + z1.imag*z2.real;

      cout << "το άθροισμα είναι" << endl;
      cout << zsum.real << " " << zsum.imag << endl;

      cout << "η διαφορά είναι" << endl;
      cout << zdif.real << " " << zdif.imag << endl;

      cout << "το γινόμενο είναι" << endl;
      cout << zprod.real << " " << zprod.imag <<
endl;

}
```


Γιατί δεν επιτρέπονται άλλες πράξεις;

- Ανάλογα το φυσικό πρόβλημα που επιλύουμε, μια πράξη μπορεί να έχει διαφορετικό νόημα
- Παράδειγμα, ο πολλαπλασιασμός:
  - Σε διανύσματα, το εσωτερικό γινόμενο είναι ρητός

$$\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y + a_z b_z = \text{ρητός αριθμός}$$

  - Σε μιγαδικούς, το γινόμενο είναι μιγαδικός

$$\tilde{z}_1 \cdot \tilde{z}_2 = (z_{1r} + iz_{1i}) \cdot (z_{2r} + iz_{2i}) =$$
$$(z_{1r}z_{2r} - z_{1i}z_{2i}) + i(z_{1r}z_{2i} + z_{1i}z_{2r}) = \text{μιγαδικός}$$

- Στην C++ ο χρήστης μπορεί να «προγραμματίσει» τις πράξεις μεταξύ μεταβλητών δομής.
  - Με χρήση συναρτήσεων
  - Με ολοκληρωμένο ορισμό αντικειμένων (τάξεις)

Συναρτήσεις με δομές

- Ορισμός, δήλωση και κλήση μιας συνάρτησης με δομές γίνεται όπως και με τις απλές μεταβλητές
  - Στην λίστα εισόδου μπορούμε να έχουμε δομές
  - Στην έξοδο μπορούμε να έχουμε δομή
  - Οι πράξεις μέσα στην συνάρτηση γίνονται όπως είδαμε πριν με τον τελεστή της τελείας «.»

- Παράδειγμα: συνάρτηση που υπολογίζει το μέτρο διανύσματος

```
double magnitude (vector v){
      double d = sqrt(v.x*v.x + v.y*v.y + v.z*v.z);
      return d;
}
```


Παράδειγμα #5: συναρτήσεις για πρόσθεση και αφαίρεση και γινόμενο διανυσμάτων

```
vector sum (vector v, vector u){
      vector w;
      w.x = v.x + u.x;
      w.y = v.y + u.y;
      w.z = v.z + u.z;
      return w;
}
```

```
vector dif (vector v, vector u){
      vector w;
      w.x = v.x - u.x;
      w.y = v.y - u.y;
      w.z = v.z - u.z;
      return w;
}
```

```
double product (vector v, vector u){
      return v.x*u.x + v.y*u.y + v.z*u.z;
}
```


Υπερφόρτωση συναρτήσεων: πολυμορφισμός

- Μπορούμε να γράψουμε μια βιβλιοθήκη με συναρτήσεις για κάθε είδους μεταβλητής
  - μια απλούς ρητούς

```
double sum (double x, double y){
      return x + y;
}
```

  - για μιγαδικούς

```
complex sum (complex z1, complex z2){
      complex z3;
      z3.real = z1.real + z2.real;
      z3.imag = z1.imag + z2.imag;
      return z3;
}
```

  - για διανύσματα

```
vector sum (vector v, vector u){
      vector w;
      w.x = v.x + u.x;
      w.y = v.y + u.y;
      w.z = v.z + u.z;
      return w;
}
```


Παράδειγμα #6: Πλήρες πρόγραμμα για πράξεις με διανύσματα (1/2)

Πρόγραμμα που διαβάζει δύο διανύσματα, και τυπώνει το μέτρο του καθενός, το μέτρο του αθροίσματος, το μέτρο της διαφοράς, καθώς και και το εσωτερικό τους γινόμενο.

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

struct vector {double x, y, z;};

vector sum (vector, vector);
vector dif (vector, vector);
double product (vector, vector);
double magnitude (vector);

int main(){
      vector v, u;
      cout <<"εισάγετε συνιστώσες 2 διανυσμάτων"<< endl;
      cin >> v.x >>  v.y >>  v.z;
      cin >> u.x >>  u.y >>  u.z;
```

Παράδειγμα #6: Πλήρες πρόγραμμα για πράξεις με διανύσματα (2/2)

```
        cout << "τα μέτρα των διανυσμάτων είναι" << endl;
        cout << magnitude(v) <<" " << magnitude(u) << endl;

        cout << "το μέτρο του αθροίσματος είναι" << endl;
        cout << magnitude( sum(v, u) );

        cout << "το μέτρο της διαφοράς είναι" << endl;
        cout << magnitude( dif(v, u) );

        cout << "το γινόμενο είναι" << endl;
        cout << product(v, u);

}
```


Μια συνάρτηση μπορεί να καλεί άλλη συνάρτηση

Παράδειγμα: συνάρτηση που υπολογίζει τη γωνία μεταξύ δύο διανυσμάτων

$$\vec{a}\cdot\vec{b} = |\vec{a}|\cdot|\vec{b}|\cdot\cos\theta \Rightarrow \theta = a\cos\left(\frac{\vec{a}\cdot\vec{b}}{|\vec{a}|\cdot|\vec{b}|}\right)$$

```
double angle (vector a, vector b){
      double theta, m1, m2;
      m1 = magnitude(a);
      m2 = magnitude(b);

      if (m1 == 0 || m2 == 0) theta = 0;
      else if (m1*m2 == 0) theta = 90;
      else theta = acos( product(a,b)/(m1*m2) )*
                   180/acos(-1.0);

      return theta;
}
```


Δείκτες σε δομές

- Όπως και στις απλές μεταβλητές, μπορούμε να δηλώσουμε δείκτη σε μεταβλητή δομής χρησιμοποιώντας τον τελεστή αστέρι «*».
  - Ο δείκτης είναι η διεύθυνση του πρώτου στοιχείου της δομής
  - παράδειγμα, δείκτης σε διάνυσμα

```
struct vector { double x, y, z; };

int main(){
      vector v, *p;

//    ανάθεση διεύθυνσης
      p = &v;
      .
      .
}
```


Εισαγωγή πίνακα δομής σε συνάρτηση

- Όπως και στην περίπτωση των πινάκων απλών μεταβλητών, ένας πίνακας δομής εισέρχεται ώς δείκτης
  - Παράδειγμα: μέσος όρος των μέτρων ενός πίνακα διανυσμάτων

```
double ave_magn(vector *v, int n){

      double s = 0;
      for (int i=0; i<n; ++i)
            s += sqrt(v[i].x * v[i].x +
                      v[i].y * v[i].y +
                      v[i].z * v[i].z);

      return s/n;
}
```

Παράδειγμα #7: μέσος όρος μέτρων διανυσμάτων

Πλήρες πρόγραμμα που υπολογίζει τον μέσο όρο των μέτρων ενός πίνακα διανυσμάτων

```
#include <iostream>
#include <cmath>
using namespace std;
struct vector {double x, y, z;};
double ave_magn(vector *, int);

int main(){
      int n; vector v[1000];
      cout << "εισάγετε τον αριθμό διανυσμάτων" << endl;
      cin >> n;

      if(n==0 || n>1000) return 1;
      for (int i=0; i<n; ++i) cin>>v[i].x>>v[i].y>>v[i].z;

      cout << ave_magn(v, n) <<endl;
}
```


Παράδειγμα #8: ιόντα

Πρόγραμμα που διαβάζει n ιόντα και υπολογίζει το κέντρο μάζας, την ολική ηλεκτροστατική ενέργεια, και την δύναμη που ασκείται σε κάθε ιόν

- Θα χρειαστούμε:
  - Την δομή για ιόντα
  - Συνάρτηση για κέντρο μάζας
  - Συνάρτηση για απόσταση δύο ιόντων
  - Συνάρτηση για ηλεκτροστατική ενέργεια ενός ιόντος
  - Συνάρτηση για την συνολική δύναμη που νοιώθει ένα ιόν

$$E_j = \sum_{\substack{i=0 \\ i\neq j}}^{n-1} \frac{q_i q_j}{|\vec{r}_j - \vec{r}_i|} \qquad F_j^x = \sum_{\substack{i=0 \\ i\neq j}}^{n-1} \frac{q_i q_j (x_j - x_i)}{|\vec{r}_j - \vec{r}_i|^3}$$

Παράδειγμα #8: συναρήσεις για ιόντα (1/5)

- Δομή για ιόντα

```
struct vector { double x, y, z; };
struct ion { vector r; double m, q; };
```

- Συνάρτηση για κέντρο μάζας
  - Το κέντρο μάζας έχει x, y, z συντεταγμένες ⇒ vector
  - Στην είσοδο δέχεται πίνακα με ιόντα
  - Υπολογίζει και τις τρεις συνιστώσες

$$\mu_x = \frac{\sum_i x_i m_i}{\sum_i m_i}, \ \ldots \text{και παρόμοια για } \mu_y, \mu_z$$

Παράδειγμα #8: συναρήσεις για ιόντα (2/5)

- Συνάρτηση για κέντρο μάζας

$$\mu_x = \frac{\sum_i x_i m_i}{\sum_i m_i}$$

```
vector centermass (ion *a, int n){
      vector cm; double m;

      cm.x = cm.y = cm.z = m = 0;
      for (int i=0; i<n; ++i){
            cm.x += a[i].r.x * a[i].m;
            cm.y += a[i].r.y * a[i].m;
            cm.z += a[i].r.z * a[i].m;
            m += a[i].m;
      }
      cm.x = cm.x / m;
      cm.y = cm.y / m;
      cm.z = cm.z / m;

      return cm;
}
```

Παράδειγμα #8: συναρήσεις για ιόντα (3/5)

- Συνάρτηση για απόσταση δύο διανυσμάτων

$$d = |\vec{v} - \vec{u}|$$

```
double dist (vector v, vector u){
      return sqrt( pow(v.x-u.x, 2) +
                   pow(v.y-u.y, 2) +
                   pow(v.z-u.z, 2));
}
```

- ή με τις συναρτήσεις `dif` και `magnitude`

```
double dist (vector v, vector u){
      return magnitude( dif(v, u) );
}
```


Παράδειγμα #8: συναρήσεις για ιόντα (4/5)

- Συνάρτηση για ηλεκτροστατική ενέργεια ενός ιόντος

$$E_j = \sum_{\substack{i=0 \\ i \neq j}}^{n-1} \frac{q_i q_j}{|\vec{r}_j - \vec{r}_i|}$$

```
double energy (ion *a, int n, int j){
      double e = 0;

      for (int i=0; i<n; ++i) if(i != j)
            e += a[i].q * a[j].q / dist(a[j].r, a[i].r);

      return e;
}
```


Παράδειγμα #8: συναρήσεις για ιόντα (5/5)

- Ηλεκτροστατική δύναμη πάνω σε ιόν

$$F_j^x = \sum_{\substack{i=0 \\ i \neq j}}^{n-1} \frac{q_i q_j (x_j - x_i)}{|\vec{r}_j - \vec{r}_i|^3}$$

```
vector force (ion *a, int n, int j){
      vector f, fi;
      f.x = f.y = f.z = 0;
      for (int i=0; i<n; ++i) if(i != j){
            fi.x = a[i].q*a[j].q * (a[j].r.x - a[i].r.x)/
                        pow( dist(a[j].r, a[i].r), 3);

            fi.y = a[i].q*a[j].q * (a[j].r.y - a[i].r.y)/
                        pow( dist(a[j].r, a[i].r), 3);

            fi.z = a[i].q*a[j].q * (a[j].r.z - a[i].r.z)/
                        pow( dist(a[j].r, a[i].r), 3);

            f = sum(f, fi);
      }
      return f;
}
```


Παράδειγμα #8: πρόγραμμα για ιόντα (1/3)

- Πλήρες πρόγραμμα για ιόντα

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

struct vector { double x, y, z; };
struct ion { vector r; double m, q; };

vector sum (vector, vector);
vector dif (vector, vector);
double magnitude (vector);
double dist (vector, vector);

vector centermass (ion *, int);
double energy (ion *, int, int);
vector force (ion *, int, int);
```

συνεχίζεται . . .

Παράδειγμα #8: πρόγραμμα για ιόντα (2/3)

```
int main(){
      int n; ion a[1000];
      vector f, m;

      cout << "εισάγετε τον αριθμό των ιόντων" << endl;
      cin >> n;

      if (n < 1 || n > 1000) return 1;

      for (int i=0; i<n; ++i)
            cin >> a[i].r.x >> a[i].r.y >> a[i].r.z >>
                  a[i].m >> a[i].q;

      m = centermass(a, n);
      cout << "το κέντρο μάζας έχει συντεταγμένες" <<endl;
      cout << m.x <<" "<< m.y <<" "<< m.z <<endl;
```

συνεχίζεται...

Παράδειγμα #8: πρόγραμμα για ιόντα (3/3)

```
      for (int i=0; i<n; ++i){
            cout << "το ιόν" << i << "έχει ενέργεια" <<endl;
            cout << energy(a, n, i);

            f = force(a, n, i);
            cout << "και υφίσταται δύναμη" << endl;
            cout << f.x <<" "<< f.y <<" "<< f.z << endl;
      }
}
```


Παράδειγμα #9: μοριακή δυναμική

- Έχουμε χτίσει όλη τη βάση ώστε να δημιουργήσουμε έναν κώδικα μοριακής δυναμικής:
  - ιόντα, λόγω των μεταξύ τους δυνάμεων εκτελούν τροχιές στον χώρο
- Χρειαζόμαστε για κάθε ιόν:
  - Θέση (διάνυσμα)
  - Μάζα (ρητός)
  - Φορτίο (ρητός)
  - Ταχύτητα (διάνυσμα)
  - Επιτάχυνση (διάνυσμα)
- Η πλήρης δομή για ένα ιόν

```
struct vector { double x, y, z; };
struct ion { vector r, vel, acc; double m, q; };
```


Παράδειγμα #9: μοριακή δυναμική: εξισώσεις κίνησης

- Ολοκλήρωση εξισώσεων κίνησης:
  - Ανάπτυγμα Taylor για την απόσταση

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{2}\vec{a}(t)(\Delta t)^2$$

  - Ανάπτυγμα για την ταχύτητα

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \vec{a}(t)\Delta t$$

  - **ΣΗΜΕΙΩΣΗ: εφαρμόζοντας αυτούς τους τύπους επιτυγχάνουμε λύση με ακρίβεια πρώτης τάξης, όχι ικανοποιητική. Με μικρή τροποποίηση μπορούμε να επιτύχουμε ακρίβεια δεύτερης τάξης, αλλά δεν είναι επι του παρόντως.**

Παράδειγμα #9: μοριακή δυναμική: συνάρτηση για πολλαπλασιασμό διανύσματος με ρητό

Θα χρειαστεί να πολλαπλασιάσουμε διάνυσμα με ρητό. Π.χ. $\vec{v} \cdot dt$
Χρειαζόμαστε μια συνάρτηση για να το κάνει αυτό

```
vector mult(vector r, double f){
      r.x *= f; r.y *= f; r.z *= f;
      return r;
}
```

$$\vec{r} \cdot f \Rightarrow \begin{cases} x \to x \cdot f \\ y \to y \cdot f \\ z \to z \cdot f \end{cases}$$

Υπάρχει πάντα η πιθανότητα κάποιος να την καλέσει λάθος, εισάγοντας πρώτα τον ρητό και μετά το διάνυσμα. Για να μην προκαλέσει λάθος αυτό, μπορούμε να υπερφορτώσουμε την συνάρτηση δηλώνοντάς την δεύτερη φορά.

```
vector mult(double f, vector r){
      r.x *= f; r.y *= f; r.z *= f;
      return r;
}
```

Τώρα όπως και να την καλέσει κανείς, είναι σωστό!

Παράδειγμα #9: μοριακή δυναμική: η στιγμιαία επιτάχυνση σε κάθε ιόν

Χρειαζόμαστε μια συνάρτηση για να υπολογίζει την στιγμιαία επιτάχυνση σε κάθε ιόν

$$\vec{a}_i = \vec{f}_i / m_i$$

```
void acceleration (ion *a, int n){

      for (int i=0; i<n; ++i)
            a[i].acc = mult( force(a,n,i), 1.0/a[i].m);

}
```


Παράδειγμα #9: μοριακή δυναμική: ανανέωση ταχύτητας κάθε ιόντος

Χρειαζόμαστε μια συνάρτηση για να υπολογίζει την νέα ταχύτητα του κάθε ιόντος

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \vec{a}(t)\Delta t$$

```
void velocity (ion *a, int n, double dt){

      for (int i=0; i<n; ++i){

            vector v1 = mult( a[i].acc, dt );

            a[i].vel = sum( a[i].vel, v1 );
      }
}
```


Παράδειγμα #9: μοριακή δυναμική: ανανέωση θέσης κάθε ιόντος

Χρειαζόμαστε μια συνάρτηση για να υπολογίζει την νέα θέση του κάθε ιόντος

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{2}\vec{a}(t)(\Delta t)^2$$

```
void position (ion *a, int n, double dt){

      for (int i=0; i<n; ++i){

            vector r1 = mult( a[i].vel , dt);
            vector r2 = mult( a[i].acc , 0.5*dt*dt);

            a[i].r = sum( a[i].r, sum(r1, r2));
      }
}
```

Παράδειγμα #9: μοριακή δυναμική

Έχοντας δώσει αρχικές τιμές σε θέση, μάζα και φορτίο, το κυρίως πρόγραμμα όσο αναφορά την δυναμική εξέλιξη του συστήματος των ιόντων είναι πολύ απλό:

```
int main(){
      int n, nsteps;
      ion a[1000];
      double dt;
.
.
      for (int j=0; j<nsteps; ++j){
            acceleration (a, n);
            position (a, n, dt);
            velocity (a, n, dt);
      }
.
.
}
```


Δείκτες σε δομές: Μια προσεκτικότερη ματιά: ανάθεση τιμών μέσω δείκτη

- Χρησιμοποιούμε και πάλι το αστέρι και την τελεία
  - Επειδή η τελεία έχει μεγαλύτερη προτεραιότητα, απαραίτητες είναι οι παρενθέσεις

```
vector v, *p;
p = &v;
cin >> (*p).x >>  (*p).y >>  (*p).z;
```

το `*(p.x)` όπως και το `p.x` δεν έχουν νόημα καθώς το `p` είναι δείκτης

- Πιο συμπαγής συμβολισμός
  - η C++ μας προσφέρει έναν πιο συμπαγή συμβολισμό με τον τελεστή του βέλους «`->`»

```
vector v, *p;
p = &v;
cin >> p->x >>  p->y >>  p->z;
```

  - Το βέλος «δείχνει» στα «συστατικά» της δομής

Δείκτες όταν έχουμε δομές μέσα σε δομές

- Για πρόσβαση σε αριθμητικές τιμής μέσω δείκτη, χρησιμοποιούμε το βέλος για το πρώτο «επίπεδο», και κατόπιν την τελεία, όσες φορές χρειαστεί.

```
struct vector { double x, y, z; };
struct ion { vector r; double m, q; };
int main(){
      ion a, *p;

      p = &a;
      p->r.x = 3.0;
      p->r.y = 4.0;
      p->r.z = 5.0;
      p->m   = 6.0;
      p->q   = 7.0;
      .
      .
}
```


Χρησιμότητα δεικτών σε δομές

- Πολλές δομές μπορεί να είναι μεγάλες. Τότε, εάν εισέρχονται σε συνάρτηση, η δημιουργία αντιγράφων μπορεί να είναι χρονοβόρα.
  - Λύση είναι να εισάγουμε έναν δείκτη προς την δομή
  - Παράδειγμα, η συνάρτηση `sum` για πρόσθεση δύο διανυσμάτων

```
vector sum (vector *v, vector *u){
      vector w;
      w.x = v->x + u->x;
      w.y = v->y + u->y;
      w.z = v->z + u->z;
      return w;
}
```

  - Στο κυρίως πρόγραμμα θα καλεστεί

```
int main(){
      vector v, u, w;
      w = sum (&v, &u);
```

Πίνακας δομής και δείκτες

- Όπως και στην περίπτωση των απλών μεταβλητών, ένας πίνακας έχει πολλά κοινά με έναν δείκτη
  - Το όνομα του πίνακα είναι δείκτης στο πρώτο του στοιχείο
  - Ισχύει η αριθμητική δεικτών

Παράδειγμα, πίνακας διανυσμάτων

```
struct vector { double x, y, z; };
int main(){
      vector v[1000]; vector *p;

      p = v;       } ταυτόσημα
      p = &v[0];   }

      v[2].x   = 3; }
      (p+2)->x = 3; } ταυτόσημα
      p[2].x   = 3; }
```