Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Αγρονόμων Τοπογράφων Μηχανικών

Προγραμματιστικές τεχνικές

Βασίλειος Βεσκούκης
Δρ. Ηλεκτρολόγος Μηχανικός &
Μηχανικός Υπολογιστών ΕΜΠ
v.vescoukis@cs.ntua.gr

Δομές δεδομένων στη C++
Εισαγωγή στις κλάσεις

Δομές δεδομένων

Μέχρι σήμερα, μοναδικός μηχανισμός αποθήκευσης πληροφοριών σε ένα πρόγραμμα την ώρα που αυτό εκτελείται, ήταν οι μεταβλητές μνήμης οι οποίες:

- **είναι ατομικές, δηλαδή περιέχει μία τιμή η κάθε μία**
- **δηλώνονται πριν τη χρήση τους**
- **βασιζόμαστε στην τήρηση ορισμένων παραδοχών κατά την ονομασία και τη χρήση τους**

Αυτά είναι καλά, όμως...

- **Ο πραγματικός κόσμος είναι πιο σύνθετος**
- **Πολλές μεταβλητές μνήμης χαρακτηρίζουν μία έννοια του πραγματικού κόσμου**
  - **Σημείο: x, y**
  - **Γραμμή: x1, y1, x2, y2**
  - **Κύκλος: x, y, R, area, perimeter**

Αν πρέπει να χρησιμοποιούμε αποκλειστικά απλές μεταβλητές μνήμης, τότε πρέπει να προσέχουμε ιδιαίτερα κατά τη χρήση τους

Η ανάγκη για σύνθετες μεταβλητές μνήμης, είναι φανερή

Δομές (structures)

Ορισμός:
Ενα struct είναι ένας τύπος που ορίζεται από τον προγραμματιστή και περιέχει μέσα του περισσότερες από μία απλές μεταβλητές μνήμης, οι οποίες λέγονται πεδία (fields) ή μέλη (members)

Κάθε structure

- έχει όνομα που μετά τη δήλωσή του χρησιμοποιείται ως *όνομα τύπου*
- δηλαδή ορίζουμε μεταβλητές ενός *τύπου* τον οποίο εμείς δημιουργήσαμε
- περιέχει απλές μεταβλητές μνήμης με διαφορετικά ονόματα
- μπορεί να περιέχει αναφορά (δείκτη) στον ίδιο τύπο
- η δήλωσή του ολοκληρώνεται με ";"

Παράδειγμα

```
struct Line {
        float x1, y1, x2, y2;
        int line_id;
        };                          // προσοχή εδώ! βάζουμε και ";"
Line L1, L2, L3;                    // όπως θα ορίζαμε "int i,j,k;"
float a,b,x1;
```


Χρήση των structures

Για ποιο λόγο είναι χρήσιμα τα structures;

- Για την παράσταση σύνθετων οντοτήτων στα προγράμματά μας
  - Γραμμή
  - Κύκλος
  - Φυσικό πρόσωπο
  - Αντικείμενο
  - Πράξη
- Αν και φαίνονται "σύνθετα" είναι ισχυροί μηχανισμοί οργάνωσης των δεδομένων

Παραδείγματα

```
struct Person {
  char[20] firstname, lastname;
  int birthdate, birthmonth, birthyear;
  char[10] height;
  float weight;
  char[20] ColorOfEyes;
};
```

```
struct Polygon {
  char[30] description;
  int koryfes;
  float x[max_koryfes];
  float y[max_koryfes];
  float area;
};
```

Χρήση των structures

Αναφορά στα πεδία/μέλη:

- **Δήλωση του struct και μίας μεταβλητής τέτοιου τύπου**
- **όνομα της μεταβλητής τύπου struct +**
- **μια τελεία +**
- **όνομα του πεδίου μέσα στο stuct**

```
struct Polygon {
  char[30] description;
  int koryfes;
  float x[max_koryfes];
  float y[max_koryfes];
  float area;
};

Polygon p1;
p1.koryfes=4;
p1.x[0]=0;
p1.y[0]=0;
p1.x[1]=1;
...
p1.area=calc_area(p1.x[], p1.y[], koryfes];
```


Χρήση των structures

Αναφορά στα πεδία/μέλη:

- **Δήλωση του struct και τουλάχιστον μίας μεταβλητής τύπου *δείκτη* στο struct**
- **όνομα της μεταβλητής τύπου δείκτη +**
- **το σύμβολο "->" +**
- **όνομα του πεδίου μέσα στο stuct**

```
struct Polygon {
  char[30] description;
  int koryfes;
  float x[max_koryfes];
  float y[max_koryfes];
  float area;
};

Polygon *P1;
P1->koryfes=4;
P1->x[0]=0;
P1->y[0]=0;
P1->x[1]=1;
...
P1->area=calc_area(p1->x[], p1->y[], koryfes);
```

Παράδειγμα χρήσης structures (1)

```
struct Line {
    float x1, y1, x2, y2;
};

float linelength(Line L)
{
    float mikos;
    mikos=sqrt( pow((L.x1-L.x2),2)+pow((L.y1-L.y2),2) );
    return mikos;
}

int main()
{
    Line et1,et2;
    float len1, len2;
        cout<<"Dose tis syntetagmenes tou et1"<<endl;
    cout<<"x1="; cin>>et1.x1;
    cout<<"y1="; cin>>et1.y1;
    cout<<"x2="; cin>>et1.x2;
    cout<<"y2="; cin>>et1.y2;

    cout<<endl<<"Tmhma et1: ("<<et1.x1<<", "<<et1.y1<<") - ("<<et1.x2<<",
"<<et1.y2<<")"<<endl;

    len1=linelength(et1);
        cout<<"Length="<<len1;
    return 0;
}
```


Παράδειγμα χρήσης structures (2)

```
struct Line {
    float x1, y1, x2, y2;
};

float linelength(Line L)
{
    float mikos;
    mikos=sqrt( pow((L.x1-L.x2),2)+pow((L.y1-L.y2),2) );
    return mikos; }

int main()
{
    Line *et1,*et2;
    float len1, len2;

    cout<<"Dose tis syntetagmenes tou et1"<<endl;
    cout<<"x1=";    cin>>et1->x1;
    cout<<"y1=";    cin>>et1->y1;
    cout<<"x2=";    cin>>et1->x2;
    cout<<"y2=";    cin>>et1->y2;

    cout<<endl<<"Tmhma et1: ("<<et1->x1<<", "<<et1->y1<<") - ("<<et1->x2<<", "<<et1-
>y2<<")"<<endl;

    len1=linelength(*et1);
    cout<<"Length="<<len1;
    return 0;
}
```

Γενικά περί structures

Γενικές δυσκολίες

- Πότε τα χρησιμοποιούμε;
- Τι βάζουμε μέσα;
- Τι σημαίνει ότι ορίζουμε ένα δικό μας τύπο;

Τεχνικές δυσκολίες

- Σωστή αναφορά στα πεδία ενός struct
- Σωστή χρήση των pointers και των συμβόλων `*`, `.` , `->`, `&`
- Συντακτικά σφάλματα κατά τη χρήση των παραπάνω

```
struct Polygon {
  char[30] description;
  int koryfes;
  float x[max_koryfes];
  float y[max_koryfes];
  float area;
};
```

Θέματα προς συζήτηση

- Τι είναι τα πεδία του struct;
- Μπορούμε να έχουμε κρυφά πεδία;
- Μπορούμε να υπολογίζουμε εξαρτημένα πεδία εσωτερικά στο struct;

Δομές δεδομένων και μεταβλητές μνήμης

Απλές (ατομικές) δομές δεδομένων

- Αποθηκεύουν μία τιμή κάθε φορά
- Ανήκουν σε κάποιους *προκαθορισμένους τύπους* της γλώσσας προγραμματισμού που χρησιμοποιούμε (int, float, char, κλπ)
- Δεν είναι κατάλληλες για την παράσταση σύνθετων εννοιών και των ιδιοτήτων τους, οι οποίες, γενικά ΔΕΝ είναι ατομικές

Σύνθετες δομές δεδομένων

- Αποθηκεύουν πολλές τιμές μαζί
- *Ορίζονται ως νέοι τύποι από το χρήστη* και μετά τον ορισμό τους χρησιμοποιούνται κανονικά ως τύποι της γλώσσας
- Είναι εργαλεία για την παράσταση σύνθετων εννοιών και οντοτήτων

Υπενθύμιση

Τύπος που υπάρχει ενσωματωμένος στη C++

```
float x,y;
```

Ονόματα μεταβλητών μνήμης (τύπου float)

Ορισμός ενός νέου τύπου από τον χρήστη

```
struct point {
  float x,y;
  int pid;
};
```

Ονομα του νέου τύπου

Ονόματα μεταβλητών μνήμης (τύπου point)

```
point p1, p2, p3;
```

Σύνθετες δομές δεδομένων

Ιδιότητες ή πεδία

- Ολες οι μεταβλητές μνήμης που είναι ενός σύνθετου τύπου (struct) ο οποίος έχει οριστεί από τον χρήστη, έχουν τις ίδιες ιδιότητες
- Διαφέρουν μεταξύ τους από το όνομά τους, όπως ακριβώς και οι απλές μεταβλητές μνήμης

```
struct point {
  float x,y;
  int pid;
};

point p1, p2, p3;
```

- Στο παράδειγμά μας:
  - όλα τα σημεία (p1, p2, p3) έχουν τις ίδιες ιδιότητες, δηλαδή τα x, y και pid.

Αναφορά στις ιδιότητες

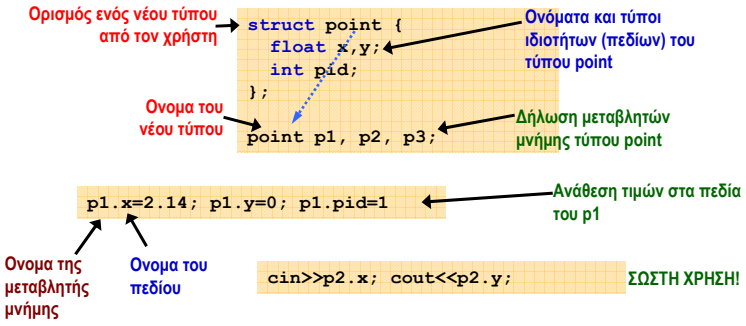
- Το όνομα της ιδιότητας (x, y, pid) ΔΕΝ έχει νόημα από μόνο του διότι πρέπει να προσδιορίσουμε ΣΕ ΠΟΙΟΥ ΜΕΛΟΥΣ ΤΙΣ ΙΔΙΟΤΗΤΕΣ ΑΝΑΦΕΡΟΜΑΣΤΕ. Ετσι χρησιμοποιούμε τον συμβολισμό:

  όνομα_μεταβλητής_μνήμης + τελεία + όνομα_πεδίου

Αναφορά στα πεδία των struct

Ονομα_μεταβλητής_μνήμης +
τελεία +
όνομα_πεδίου

Ετσι καθορίζουμε σε ποιο πεδίο ποιας σύνθετης οντότητας θα αναφερθούμε



Το struct ως εργαλείο μοντελοποίησης

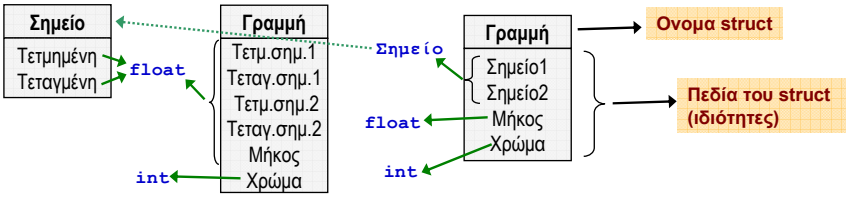
Μια από τις πρώτες διατυπώσεις σχετικά με την πληροφορική:

- **Από τον πραγματικό κόσμο...**
- **...σε ένα *μοντέλο* κατάλληλο για Η/Υ...**
- **...στον υπολογιστή με χρήση μιας γλώσσας προγραμματισμού**

Τα μοντέλα δεδομένων χρησιμοποιούνται για να αντιστοιχίσουν

- **Εννοιες του πραγματικού κόσμου (πχ μήκος, συντεταγμένες, εμβαδόν)**
- **Σε έννοιες που χρησιμοποιεί ο Η/Υ (float length, x, y, area;)**

Σύνθετα αντικείμενα του πραγματικού κόσμου δεν παριστάνονται απλά και κατανοητά με απλές (ατομικές) μεταβλητές μνήμης



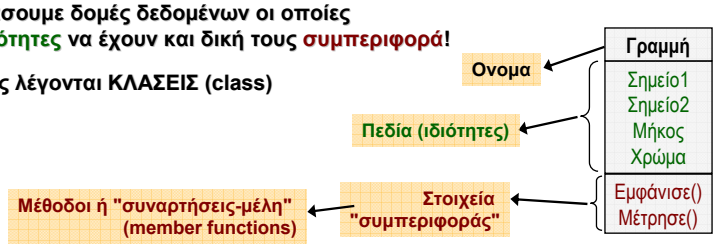
Νέες ιδέες μοντελοποίησης δεδομένων

Ενα struct Α περιέχει

- **Πεδία, δηλαδή απλές (ατομικές) μεταβλητές μνήμης**
- **Τα πεδία αυτά αποτελούν τις ιδιότητες του struct ή καλύτερα καθορίζουν ποιες θα είναι οι ιδιότητες των μεταβλητών που θα οριστούν ως τύπου Α**
- **Προσοχή: Καθορίζουν ποιες θα είναι οι ιδιότητες και όχι τι τιμές θα έχουν**
  - **Ποιες θα είναι: (ύψος, βάρος, ημ.γέννησης)**
  - **Τι τιμές θα έχουν: (1.70, 60, 1.1.1980)**

Μια νέα ιδέα:
να κατασκευάσουμε δομές δεδομένων οι οποίες εκτός από ιδιότητες να έχουν και δική τους συμπεριφορά!

Αυτές οι δομές λέγονται ΚΛΑΣΕΙΣ (class)



Κλάσεις

Σύνθετες, οριζόμενες από το χρήστη δομές, οι οποίες περιέχουν

- **Ιδιότητες (πεδία, fields, attributes) και**
- **Συμπεριφορά (μεθόδους, methods, member functions)**

Οι κλάσεις μπορούν να ιδωθούν:

- **Ως προγραμματιστικό εργαλείο**
- **Ως εργαλείο μοντελοποίησης δεδομένων**

Ορολογία

- **Απλοί τύποι: τύπος δεδομένων -> μεταβλητή**
  - **int i (Μεταβλητή τύπου int)**
  - **Point p1 (μεταβλητή τύπου Point (struct που ορίστηκε από εμάς)**
- **Κλάσεις: Κλάση -> αντικείμενο (object)**
  - **Οπως ορίζουμε μεταβλητές ενός τύπου, δημιουργούμε και Αντικείμενα (objects) μιας κλάσης**
  - **Εξ' ου και "object-oriented" (αντικειμενοστρεφής)**

Ενα πρώτο παράδειγμα

```
01 #include <iostream>
02 using namespace std;
03
04 class line {
05
06 public:
07     float x1, y1, x2, y2;
08     float len;
09
10     float linelen() {
11         float x;
12         x=sqrt(pow((x1-x2),2)+pow((y1-y2),2));
13         len=x;
14         return x;
15     }
16
17 };//end of class declaration
18
19 int main()
20 {
21     line l1;
22
23     cout<<"Welcome to oo programming"<<endl;
24
25     l1.x1=0; l1.y1=0;
26     l1.x2=1; l1.y2=1;
27
28     cout<<l1.linelen();
29
30     return 0;
31 }
```

Αρχή δήλωσης της κλάσης

Τέλος δήλωσης της κλάσης

Ενα πρώτο παράδειγμα

```
01 #include <iostream>
02 using namespace std;
03
04 class line {
05
06 public:
07     float x1, y1, x2, y2;
08     float len;
09
10     float linelen() {
11         float x;
12         x=sqrt(pow((x1-x2),2)+pow((y1-y2),2));
13         len=x;
14         return x;
15     }
16
17 }; //end of class declaration
18
19 int main()
20 {
21     line l1;
22
23     cout<<"Welcome to oo programming"<<endl;
24
25     l1.x1=0; l1.y1=0;
26     l1.x2=1; l1.y2=1;
27
28     cout<<l1.linelen();
29
30     return 0;
31 }
```

ΠΡΟΣΔΙΟΡΙΣΤΗΣ ΟΡΑΤΟΤΗΤΑΣ

ΠΕΔΙΑ

ΜΕΘΟΔΟΙ

Ενα πρώτο παράδειγμα

```
01 #include <iostream>
02 using namespace std;
03
04 class line {
05
06 public:
07     float x1, y1, x2, y2;
08     float len;
09
10     float linelen() {
11         float x;
12         x=sqrt(pow((x1-x2),2)+pow((y1-y2),2));
13         len=x;
14         return x;
15     }
16
17 }; //end of class declaration
18
19 int main()
20 {
21     line l1;
22
23     cout<<"Welcome to oo programming"<<endl;
24
25     l1.x1=0; l1.y1=0;
26     l1.x2=1; l1.y2=1;
27
28     cout<<l1.linelen();
29
30     return 0;
31 }
```

ΔΗΜΙΟΥΡΓΙΑ ΑΝΤΙΚΕΙΜΕΝΟΥ

ΠΡΟΣΠΕΛΑΣΗ ΠΕΔΙΩΝ

ΚΛΗΣΗ ΜΕΘΟΔΟΥ