Οντοκεντρικός Προγραμματισμός

Εισαγωγή

25/02/2021

Γιάννης Βασιλόπουλος
Ε.Δι.Π.

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Αντικειμενοστρεφής Προγραμματισμός

- Γιατί τον χρειαζόμαστε;
- Η C (διαδικαστικός προγραμματισμός) προσφέρει ότι χρειαζόμαστε;

Παράδειγμα σε C

```c
typedef struct data
{   double x;
    double y;
    float temp;
}myData;

myData pinakasMyData[1000];
.
.
void resetData ()
{ //epanaferei sta stoixeia tou pinakasMyData kapoies
sygkekrimenes times
for (i=1;i<N;i=i+1)
  {  pinakasMyData[i].x=f_x();
     pinakasMyData[i].y=f_y();
     pinakasMyData[i].temp=f_temp();}
}
```

```c
.
.
.
int superAlgorithm()
{//trexei ton algorithmo me vash ta dedomena tou
pinakaMyData
   coordinate_x= pinakasMyData[i].x;
   coordinate_y=pinakasMyData[i].y;
   coordinate_temp= pinakasMyData[i].temp;
    if (.....) resetData();
}
.
.
int main()
{
 resetData ();
 superAlgorithm();
}
```

Σενάριο

- Στον προηγούμενο κώδικα ζητούνται οι παρακάτω αλλαγές

1. Να μην υπάρχει απ’ ευθείας πρόσβαση στα στοιχεία της δομής MyData από όλες τις συναρτήσεις
2. Να καλείται *μια* συνάρτηση για την ανάθεση τιμών σε διαφορετικό πλήθος πεδίων της δομής MyData
3. Να δημιουργηθεί μια δομή παρόμοια με τη MyData - με ένα επιπλέον (ακέραιο) πεδίο - χωρίς να ξαναγραφούν τα πεδία που έχουν ήδη δηλωθεί

Πως ικανοποιούμε αυτές τις απαιτήσεις;

Σενάριο

- Στον προηγούμενο κώδικα ζητούνται οι παρακάτω αλλαγές

1. **Να μην υπάρχει απ' ευθείας πρόσβαση στα στοιχεία της δομής MyData από όλες τις συναρτήσεις**
2. Να καλείται *μια* συνάρτηση για την ανάθεση τιμών σε διαφορετικό πλήθος πεδίων της δομής MyData
3. Να δημιουργηθεί μια δομή παρόμοια με τη MyData - με ένα επιπλέον (ακέραιο) πεδίο - χωρίς να ξαναγραφούν τα πεδία που έχουν ήδη δηλωθεί

Απόκρυψη στοιχείων

```
newstruct myData
{  double x;
   double y;
   float temp;
```

Απαγορεύω την πρόσβαση σε αυτές τις μεταβλητές, εκτός από μέλη της δομής

```
double getX ()
  {return x;}

 void setX (double inx)
  {x =inx;}
```

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
double getY ()
  {return Y;}

 void setY (double iny)
  {y =iny;}
```

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
float getTemp ()
  {return temp;}

 void setTemp (float intemp)
  {temp =intemp;}
```

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
void resetData()
  { setX(f_x());
    setY(f_y());
    setTemp(f_temp());}
```

Συνάρτηση για το resetData(). Επιτρέπω την πρόσβαση.

```
void return2function (double *fx, double *fy, float *ftemp)
    {
     *fx=x;
     *fy=y;
     *ftemp=temp; }
};
```

Συνάρτηση για να επιστρέφει τις τιμές των x, y, temp στις παραμέτρους εισόδου. Επιτρέπω την πρόσβαση.

Συγχαρητήρια! Η πρώτη σας κλάση

```
class myData
{
  double x;
  double y;
  float temp;
```

Απαγορεύω την πρόσβαση σε αυτές τις μεταβλητές, εκτός από μέλη της κλάσης

```
double getX ()
  {return x;}

void setX (double inx)
  {x =inx;}
```

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
double getY ()
  {return Y;}

void setY (double iny)
  {y =iny;}
```

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
float getTemp ()
  {return temp;}

void setTemp (float intemp)
  {temp =intemp;}
```

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
void resetData()
  { setX(f_x());
    setY(f_y());
    setTemp(f_temp());}
```

Συνάρτηση για το resetData(). Επιτρέπω την πρόσβαση.

```
void return2function (double *fx, double *fy, float *ftemp)
    {
     *fx=x;
     *fy=y;
     *ftemp=temp; }
};
```

Συνάρτηση για να επιστρέφει τις τιμές των x, y, temp στις παραμέτρους εισόδου. Επιτρέπω την πρόσβαση.

Παράδειγμα σε C

```c
typedef struct data
{   double x;
    double y;
    float temp;
}myData;

myData pinakasMyData[1000];
.
.
void resetData ()
{ //epanaferei sta stoixeia tou pinakasMyData kapoies
sygkekrimenes times
for (i=1;i<N;i=i+1)
  {  pinakasMyData[i].x=f_x();
     pinakasMyData[i].y=f_y();
     pinakasMyData[i].temp=f_temp();}
}
```

```c
.
.
.
int superAlgorithm()
{//trexei ton algorithmo me vash ta dedomena tou
pinakaMyData
   coordinate_x= pinakasMyData[i].x;
   coordinate_y=pinakasMyData[i].y;
   coordinate_temp= pinakasMyData[i].temp;
    if (.....) resetData();
}
.
.
int main()
{
 resetData ();
 superAlgorithm();
}
```

Απόκρυψη στοιχείων

```
class myData
{
  double x;
  double y;
  float temp;
```

Απαγορεύω την πρόσβαση σε αυτές τις μεταβλητές, εκτός από μέλη της κλάσης

```
double getX ()
  {return x;}

 void setX (double inx)
  {x =inx;}
```

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
double getY ()
  {return Y;}

 void setY (double iny)
  {y =iny;}
```

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
float getTemp ()
  {return temp;}

 void setTemp (float intemp)
  {temp =intemp;}
```

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
void resetData()
  { setX(f_x());
   setY(f_y());
   setTemp(f_temp());}
```

Συνάρτηση για το resetData(). Επιτρέπω την πρόσβαση.

```
void return2function (double *fx, double *fy, float *ftemp)
   {
    *fx=x;
    *fy=y;
    *ftemp=temp; }
```

Συνάρτηση για να επιστρέφει τις τιμές των x, y, temp στις παραμέτρους εισόδου. Επιτρέπω την πρόσβαση.

```
};

myData pinakasMyData[1000];

int superAlgorithm ()
{
pinakasMyData[i].return2function ( &coordinate_x,
&coordinate_y,  &coordinate_temp);

if (.....)  pinakasMyData[i].resetData();
 }
```

Απόκρυψη στοιχείων

```
class myData
{
  double x;
  double y;
  float temp;

double getX ()
  {return x;}

 void setX (double inx)
  {x =inx;}

double getY ()
  {return Y;}

 void setY (double iny)
  {y =iny;}

float getTemp ()
  {return temp;}

 void setTemp (float intemp)
  {temp =intemp;}
```

Απαγορεύω την πρόσβαση σε αυτές τις μεταβλητές, εκτός από μέλη της κλάσης

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
void resetData()
  { setX(f_x());
   setY(f_y());
   setTemp(f_temp());}

void return2function (double *fx, double *fy, float *ftemp)
   {
    *fx=x;
    *fy=y;
    *ftemp=temp; }
};

myData pinakasMyData[1000];

main ()
{
for (i=1; i<N; i=i+1)
 pinakasMyData[i].resetData ();
 superAlgorithm();
}
```

Συνάρτηση για το resetData(). Επιτρέπω την πρόσβαση.

Συνάρτηση για να επιστρέφει τις τιμές των x, y, temp στις παραμέτρους εισόδου. Επιτρέπω την πρόσβαση.

Σενάριο

- Στον προηγούμενο κώδικα ζητούνται οι παρακάτω αλλαγές

1. **Να μην υπάρχει απ' ευθείας πρόσβαση στα στοιχεία της δομής MyData από όλες τις συναρτήσεις**
2. **Να καλείται *μια* συνάρτηση για την ανάθεση τιμών σε διαφορετικό πλήθος πεδίων της δομής MyData**
3. Να δημιουργηθεί μια δομή παρόμοια με τη MyData - με ένα επιπλέον (ακέραιο) πεδίο - χωρίς να ξαναγραφούν τα πεδία που έχουν ήδη δηλωθεί

Κλήση μιας συνάρτησης για ανάθεση

```
class myData
{  double x;
   double y;
   float temp;

double getX ()
  {return x;}

 void setX (double inx)
  {x =inx;}

double getY ()
  {return Y;}

 void setY (double iny)
  {y =iny;}

float getTemp ()
  {return temp;}

 void setTemp (float intemp)
  {temp =intemp;}
```

```
void resetData()
  { setX(f_x());
   setY(f_y());
   setTemp(f_temp());}

void return2function (double *fx, double *fy, float *ftemp)
   {
    *fx=x;
    *fy=y;
    *ftemp=temp; }

 void setAttrib (double inx, double iny)
{  x=inx;   y=iny;   }

 void setAttrib (double inx, double iny, float intemp)
{  x=inx;   y=iny;   temp=intemp;}

 void setAttrib (double inx, float intemp)
{   x=inx;    temp=intemp  };

};
```

Κλήση μιας συνάρτησης για ανάθεση

```
void resetData()
  { setX(f_x());
   setY(f_y());
   setTemp(f_temp());}
```

```
void return2function (double *fx, double *fy, float *ftemp)
   {
    *fx=x;
    *fy=y;
    *ftemp=temp; }

 void setAttrib (double inx, double iny)
{  x=inx;  y=iny;  }

 void setAttrib (double inx, double iny, float intemp)
{  x=inx;  y=iny;  temp=intemp;}

 void setAttrib (double inx, float intemp)
{   x=inx;   temp=intemp  };

};
```

```
myData pinakasMyData[1000];

int superAlgorithm ()
{
pinakasMyData[i].return2function (
&coordinate_x, &coordinate_y,
&coordinate_temp);

if (.....)
   pinakasMyData[i].setAttrib(2.23467, 3.12345678);
else
   pinakasMyData[i].setAttrib(2.2467, 3.5178, 24.6);
 }
```

Κλήση μιας συνάρτησης για ανάθεση

```
class myData
{   double x;
    double y;
    float temp;

 double getX ()
   {return x;}

  void setX (double inx)
   {x =inx;}

 double getY ()
   {return Y;}

  void setY (double iny)
   {y =iny;}

float getTemp ()
   {return temp;}

  void setTemp (float intemp)
   {temp =intemp;}
```

Που αλλού μπορούμε να χρησιμοποιήσουμε τη συνάρτηση setAttrib() ;

```
void resetData()
   { setX(f_x());
     setY(f_y());
     setTemp(f_temp());}

void return2function (double *fx, double *fy, float *ftemp)
     {
      *fx=x;
      *fy=y;
      *ftemp=temp; }

 void setAttrib (double inx, double iny)
{   x=inx;   y=iny;   }

 void setAttrib (double inx, double iny, float intemp)
{   x=inx;   y=iny;   temp=intemp;}

 void setAttrib (double inx, float intemp)
{    x=inx;    temp=intemp  };

};
```

Κλήση μιας συνάρτησης για ανάθεση

```
class myData
{   double x;
    double y;
    float temp;

 double getX ()
   {return x;}

  void setX (double inx)
   {x =inx;}

 double getY ()
   {return Y;}

  void setY (double iny)
   {y =iny;}

float getTemp ()
   {return temp;}

  void setTemp (float intemp)
   {temp =intemp;}
```

```
void resetData()
   { setX(f_x());
   setY(f_y());
   setTemp(f_temp());}
void return2function (double *fx, double *fy, float *ftemp)
    {
     *fx=x;
     *fy=y;
     *ftemp=temp; }

 void setAttrib (double inx, double iny)
{   x=inx;   y=iny;   }

 void setAttrib (double inx, double iny, float intemp)
{   x=inx;   y=iny;   temp=intemp;}

 void setAttrib (double inx, float intemp)
{    x=inx;    temp=intemp  };

};
```

Αντικαθιστούμε 3 συναρτήσεις με 1!

Κλήση μιας συνάρτησης για ανάθεση

```
class myData
{  double x;
   double y;
   float temp;

 double getX ()
   {return x;}

  void setX (double inx)
   {x =inx;}

 double getY ()
   {return Y;}

  void setY (double iny)
   {y =iny;}

float getTemp ()
   {return temp;}

  void setTemp (float intemp)
   {temp =intemp;}
```

```
void resetData()
  {
    setAttrib( f_x(),  (f_y(), f_temp() );
  }
void return2function (double *fx, double *fy, float *ftemp)
  {
   *fx=x;
   *fy=y;
   *ftemp=temp; }

 void setAttrib (double inx, double iny)
{  x=inx;  y=iny;  }

 void setAttrib (double inx, double iny, float intemp)
{  x=inx;  y=iny;  temp=intemp;}

 void setAttrib (double inx, float intemp)
{   x=inx;   temp=intemp  };

};
```

Σενάριο

- Στον προηγούμενο κώδικα ζητούνται οι παρακάτω αλλαγές

1. **Να μην υπάρχει απ' ευθείας πρόσβαση στα στοιχεία της δομής MyData από όλες τις συναρτήσεις**
2. **Να καλείται *μια* συνάρτηση για την ανάθεση τιμών σε διαφορετικό πλήθος πεδίων της δομής MyData**
3. **Να δημιουργηθεί μια δομή παρόμοια με τη MyData - με ένα επιπλέον (ακέραιο) πεδίο - χωρίς να ξαναγραφούν τα πεδία που έχουν ήδη δηλωθεί**

Νέα κλάση με επιπλέον πεδίο

```
class myData
{   double x;
    double y;
    float temp;

 double getX ()
   {return x;}

  void setX (double inx)
   {x =inx;}

 double getY ()
   {return Y;}

  void setY (double iny)
   {y =iny;}

float getTemp ()
   {return temp;}

  void setTemp (float intemp)
   {temp =intemp;}
```

```
void resetData()
   { setX(f_x());
     setY(f_y());
     setTemp(f_temp());}

void return2function (double *fx, double *fy, float *ftemp)
     {
      *fx=x;
      *fy=y;
      *ftemp=temp; }

 void setAttrib (double inx, double iny)
{   x=inx;   y=iny;   }

 void setAttrib (double inx, double iny, float intemp)
{   x=inx;   y=iny;   temp=intemp;}

 void setAttrib (double inx, float intemp)
{    x=inx;    temp=intemp  };

};
```

Νέα κλάση με επιπλέον πεδίο

```
class myNewData
extends myData
{
  int i;

  int getI ()
  {return i;}

  void setI (int ini)
  {i =ini;}

void return2functionNew ( double *fx,  double *fy,
float *ftemp,  int *fi)
   {*fx=x;      *fy=y;     *ftemp=temp;
    *fi=i;}
};
```

Κληρονομώ πεδία και συναρτήσεις από τη myData

Απαγορεύω την πρόσβαση σε αυτή την μεταβλητή, εκτός από μέλη της κλάσης

Επιτρέπω την πρόσβαση σε αυτές τις συναρτήσεις

```
myNewData  pinakasMyNewData[1000];

 int superAlgorithm ()
{
pinakasMyNewData[i].return2functionNew
(&coordinate_x,  &coordinate_y,
&coordinate_temp,  &coordinate_i);

if (.....)  pinakasMyNewData[i].resetData();
 }

main ()
{
for (j=1; j<N; j=j+1)
 pinakasMyNewData[j].getI();
 superAlgorithm();
}
```

Σενάριο

- Στον προηγούμενο κώδικα ζητούνται οι παρακάτω αλλαγές

1. **Να μην υπάρχει απ' ευθείας πρόσβαση στα στοιχεία της δομής MyData από όλες τις συναρτήσεις**
2. **Να καλείται *μια* συνάρτηση για την ανάθεση τιμών σε διαφορετικό πλήθος πεδίων της δομής MyData**
3. **Να δημιουργηθεί μια δομή παρόμοια με τη MyData - με ένα επιπλέον (ακέραιο) πεδίο - χωρίς να ξαναγράψουν τα πεδία που έχουν ήδη δηλωθεί**

Χαρακτηριστικά Αντικειμενοστρεφούς Προγραμματισμού

1. **Ενθυλάκωση**
2. **Πολυμορφισμός**
3. **Κληρονομικότητα**

Τι είναι αντικείμενο;

Υλοποίηση μια κλάσης στη μνήμη με δεδομένα (απόδοση τιμών στις μεταβλητές) και λειτουργίες (συναρτήσεις) -> **Στιγμιότυπο**

Μεταβλητές κλάσης -> **Ιδιότητες – Χαρακτηριστικά –Πεδία**

Συναρτήσεις κλάσης-> **Μέθοδοι (συμπεριφορά - επικοινωνία αντικειμένων)**

Ανακεφαλαίωση

- Στον Διαδικαστικό προγραμματισμό, ο κώδικας δομείται με βάση τις συναρτήσεις (διαδικασίες)
  - Αυτό έχει δομικούς περιορισμούς ειδικά σε μεγάλες εφαρμογές (πρόσβαση σε δεδομένα χωρίς περιορισμούς, δυσκολία στην επαναχρησιμοποίηση κώδικα) και στη συντήρηση κώδικα
- Στον Οντοκεντρικό (Αντικειμενοστρεφή) προγραμματισμό, ο κώδικας δομείται με βάση τα αντικείμενα
  - Χρησιμοποιούμε και διαδικαστικό προγραμματισμό

- Βασικές έννοιες του Οντοκεντρικού
  - Αντικείμενο (Object)
  - Κλάση (Class)
  - Ιδιότητες (Attributes)
  - Μέθοδοι (Methods)
  - Ενθυλάκωση (Encapsulation)
  - Πολυμορφισμός (Polymorphism)
  - Κληρονομικότητα (Inheritance)
- Δεν υπάρχει καλύτερη λύση, υπάρχει πιο λειτουργική με βάση το πρόβλημα

Ευχαριστώ για την προσοχή σας

- Απορίες
- Διευκρινήσεις