Αντικειμενοστραφής Προγραμματισμός

Χαρακτηριστικά ΑΠ

- ✓ Διαθέτει μηχανισμούς:
  - απόκρυψης πληροφορίας
  - περάσματος μηνυμάτων
  - δημιουργίας και καταστροφής στιγμιοτύπων
  - αναπαράστασης ιεραρχίας
  - κληρονομικότητας κ.α.

Χαρακτηριστικά ΑΠ

- ✓ Οι μηχανισμοί αυτοί βελτιώνουν την ποιότητα του λογισμικού ως προς τα χαρακτηριστικά του:
  - επαναχρησιμοποίηση (reusability)
  - επεκτασιμότητα (extendibility)
  - ορθότητα (correctness)
  - ευρωστία (robustness)
  - συντηρησιμότητα (maintainability)

C++ Class

```
class Classic_Example {
public:
// Δεδομένα και διαδικασίες προσβάσιμα από παντού
protected:
//Δεδομένα και διαδικασίες προσβάσιμα από την κλάση
//  τις παραγόμενες κλάσεις και  friends κλάσεις
private:
// Δεδομένα και διαδικασίες προσβάσιμα από την κλάση
// και  friends κλάσεις
};
```

C++ Class - Συμβουλή

- ✓ Δηλώστε τις συναρτήσεις στο ***PUBLIC***
- ✓ Δηλώστε τα πεδία δεδομένων στο ***PRIVATE***

C++ Class - Παράδειγμα

```
class Date{
    private:
        int  _day;
        int _month;
        int _year;
    public:
    // . . .
  };
```

C++ Classes

```
class Mailbox
{
private:
        //…
public:
        void add(Message);
        Message get_current();
        void delete_current();
        // . . .
};
```

Classes vs structs

- Η κλάση Date είναι παρόμοια με τον ορισμό της στη C ως structure

```
struct date
{int day;
 int month;
 int year;
};
```

- Τα structs δεν παρέχουν τη δυνατότητα ελέγχου πρόσβασης στα δεδομένα.

Πρόσβαση σε συναρτήσεις κλάσεων

- ✓ Εκτός κλάσης, public συναρτήσεις μπορούν να κληθούν μόνο με αναφορά σε αντικείμενο της κλάσης.

- ✓ Μέσα στην κλάση, οι διάφορες συναρτήσεις μπορούν να κληθούν από άλλες συναρτήσεις ως απλές συναρτήσεις.

C++ Class

- ✓ Τα δεδομένα μιας κλάσης καλούνται ΔΕΔΟΜΕΝΑ ΜΕΛΗ ενώ οι διαδικασίες λέγονται ΣΥΝΑΡΤΗΣΕΙΣ ΜΕΛΗ.
- ✓ Ο τελεστής '.' χρησιμοποιείται για πρόσβαση και στα δεδομένα και στις συναρτήσεις.
- ✓ Η συνάρτηση αλλάζει τις τιμές που ανήκουν στο αντικείμενο που καλεί τη διαδικασία

  b.advance(30);
- ✓ Μια συνάρτηση κλάσης μπορεί να έχει και ανεξάρτητες με την κλάση παραμέτρους

Εφαρμογή 1

Να δημιουργήσετε μια κλάση, η οποία να αντιπροσωπεύει γενικώς έναν κύκλο. Συγκεκριμένα:

i. να περιέχει μεταβλητές-μέλη (member-variables) ή ιδιότητες (attributes) που να τον περιγράφουν (η τιμή της ακτίνας αρκεί στην περίπτωσή μας – είναι κινητής υποδιαστολής διπλής ακρίβειας),
ii. να περιέχει 2 συναρτήσεις-μέλη (member-functions) ή μεθόδους (methods) που να ενεργούν πάνω στις μεταβλητές του:
    i. μια που δέχεται σαν «είσοδό της» μια τιμή, η οποία να καθορίζει την ακτίνα. Αυτή να ονομαστεί set_aktina.
    ii. μια που να υπολογίζει και να επιστρέφει το εμβαδόν του κύκλου (κινητής υποδιαστολής διπλής ακρίβειας). Αυτή να ονομαστεί get_embadon.

Οι μεταβλητές να είναι private ενώ οι συναρτήσεις public.

```
#include <iostream>
using namespace std;
class circle{
      private:
         double aktina;

      public:
         void set_aktina(double a);
         double get_emvadon();
 };

void circle::set_aktina(double a){
            aktina=a;
}

double circle::get_emvadon(){
            return 3.14*aktina*aktina;
}
```

Έναρξη κλάσης circle

Δήλωση μεταβλητών

Δήλωση συναρτήσεων

Τέλος ορισμού κλάσης

Συνάρτηση καθορισμού ακτίνας

Συνάρτηση υπολογισμού εμβαδόν

Στη συνέχεια, να γράψετε ένα πρόγραμμα, στο οποίο:

i. Να δηλώσετε **ένα αντικείμενο** της παραπάνω κλάσης, το οποίο να αντιπροσωπεύει έναν συγκεκριμένο κύκλο, π.χ. τον kyklos1.

ii. Το πρόγραμμά σας να ζητά από τον χρήστη την τιμή της ακτίνας, την οποία θα διαβάζει από το πληκτρολόγιο και θα αποθηκεύει σε κάποια (απλή) μεταβλητή.

iii. Κατόπιν, χρησιμοποιώντας τη μία από τις συναρτήσεις-μέλη, να δίνει την παραπάνω τιμή της ακτίνας στο αντικείμενο kyklos1.

iv. ‘Επειτα, να υπολογίζει το εμβαδόν του kyklos1, χρησιμοποιώντας τη δεύτερη συνάρτηση-μέλος.

v. Τέλος, θα εμφανίζει το εμβαδόν αυτό με ανάλογο μήνυμα.

Εφαρμογή 2

Να δημιουργήσετε μια κλάση, η οποία να αντιπροσωπεύει ένα τετράγωνο. Συγκεκριμένα:

i. να περιέχει μεταβλητές-μέλη (member-variables) ή ιδιότητες (attributes) που να τον περιγράφουν (η τιμή της πλευράς αρκεί στην περίπτωσή μας – είναι κινητής υποδιαστολής διπλής ακρίβειας),
ii. να περιέχει 2 συναρτήσεις-μέλη (member-functions) ή μεθόδους (methods) που να ενεργούν πάνω στις μεταβλητές του:
    i. μια που δέχεται σαν «είσοδό της» μια τιμή, η οποία να καθορίζει την πλευρά. Αυτή να ονομαστεί set_pleura.
    ii. μια που να υπολογίζει και να επιστρέφει το εμβαδόν του τετραγώνου (κινητής υποδιαστολής διπλής ακρίβειας). Αυτή να ονομαστεί get_embadon.

Οι μεταβλητές να είναι private ενώ οι συναρτήσεις public.

Στη συνέχεια, να γράψετε ένα πρόγραμμα, στο οποίο:

i. Να δηλώσετε **ένα αντικείμενο** της παραπάνω κλάσης, το οποίο να αντιπροσωπεύει ένα συγκεκριμένο τετράγωνο, π.χ. το tetr1.

ii. Το πρόγραμμά σας να ζητά από τον χρήστη την τιμή της πλευράς, την οποία θα διαβάζει από το πληκτρολόγιο και θα αποθηκεύει σε κάποια (απλή) μεταβλητή.

iii. Κατόπιν, χρησιμοποιώντας τη μία από τις συναρτήσεις-μέλη, να δίνει την παραπάνω τιμή της πλευράς **στο αντικείμενο** tetr1.

iv. ‘Επειτα, να υπολογίζει το εμβαδόν του tetr1, χρησιμοποιώντας τη δεύτερη συνάρτηση-μέλος.

v. Τέλος, θα εμφανίζει το εμβαδόν αυτό με ανάλογο μήνυμα.

Εφαρμογή 3

Να δημιουργήσετε μια κλάση, η οποία να αντιπροσωπεύει γενικώς ένα άτομο. Συγκεκριμένα:

i. να περιέχει μεταβλητές-μέλη (member-variables) ή ιδιότητες (attributes) που να τον περιγράφουν (το όνομα του και η ηλικία του αρκεί στην περίπτωσή μας),
ii. να περιέχει 2 συναρτήσεις-μέλη (member-functions) ή μεθόδους (methods) που να ενεργούν πάνω στις μεταβλητές του:
    i. μια που δέχεται σαν «είσοδό της» από το πληκτρολόγιο το όνομα και την ηλικία του ατόμου. Να χρησιμοποιήσετε σχετικά μηνύματα προς τον χρήστη. Αυτή να ονομαστεί read_data.
    ii. μια που να εμφανίζει στον χρήστη το όνομα και την ηλικία του ατόμου. Αυτή να ονομαστεί print_data.

Στη συνέχεια, να γράψετε ένα πρόγραμμα, στο οποίο:

i. Να δηλώσετε **δύο αντικείμενα** της παραπάνω κλάσης, τα οποία να αντιπροσωπεύουν δύο συγκεκριμένα άτομα, π.χ. τα p1 και p2.

ii. Κατόπιν, χρησιμοποιώντας τη μία από τις συναρτήσεις-μέλη, να δίνει την τιμές στα ονόματα και στις ηλικίες **των αντικειμένων p1 και p2**.

iii. Έπειτα, χρησιμοποιώντας τη δεύτερη συνάρτηση-μέλος, να τυπώνει τις τιμές των αντικειμένων p1 και p2.

```
#include <iostream>
using namespace std;
class Person
{
        private:
                char name[30];
                int age;
        public:
                void readData();
                void printData();
};
void Person::readData()
{
        cout <<"Enter name:";
        cin >>name;
        cout << "Enter age:";
        cin >> age;
}
void Person::printData()
{
        cout << "The name of the person is " << name << endl;
        cout << "The age of the person is " << age << endl;
}
```

```
int main()
{
        Person  p1,  p2;
        p1.readData();
        p1.printData();
        p2.readData();
        p2.printData();
}
```

Ορισμός Αντικειμένων

Η εντολή:

Person p1, p2; *Ορίζει δύο αντικείμενα την κλάσης Person*

Όπως σε μια δομή, όταν την ορίζουμε δεν δημιουργούμε καμία μεταβλητή δομής, έτσι και με την κλάση, όταν την ορίζουμε δεν δημιουργούμε αντικείμενα, απλά περιγράφουμε πως θα είναι όταν δημιουργηθούν. Τα αντικείμενα δημιουργούνται όταν ορίζονται, σύμφωνα με την παρακάτω εντολή.

1. Όνομα κλάσης **2.** Όνομα/τα αντικειμένου/ων **3.** ;

Ιδιωτικά VS δημόσια μέλη

- Βασικό χαρακτηριστικό του ΑΠ είναι η απόκρυψη δεδομένων.
- Ή ένδειξη private (ιδιωτικά) σημαίνει ότι τα δεδομένα είναι ***κρυμμένα*** μέσα σε μία κλάση, έτσι ώστε να μην είναι προσπελάσιμα από συναρτήσεις που είναι έξω από την κλάση.
- Η ένδειξη public (δημόσια δεδομένα) επεξηγεί ότι τα δεδομένα προσπελαύνονται και έξω από τις κλάσεις.
- Προτείνεται τα δεδομένα μιας κλάσης να είναι ιδιωτικά και οι συναρτήσεις να είναι δημόσιες.

Κλήση συναρτήσεων-μελών

- ✓ Για να καλέσουμε μια συνάρτηση-μέλος μίας κλάσης, τη συνδέουμε μέσω του τελεστή ***τελεία*** (.) με κάποιο αντικείμενο:

Αντικείμενο. Όνομα συνάρτησης μέλους

- ✓ Η σύνταξη είναι παρόμοια με τον τρόπο που αναφερόμαστε στα μέλη μιας δομής, με τις παρενθέσεις να υποδηλώνουν όμως ότι αναφερόμαστε σε συνάρτηση-μέλος και όχι σε στοιχείο δεδομένων.

p1.readData();

Συναρτήσεις-μέλη οριζόμενες έξω από την κλάση

- ✓ Μπορούμε να διατηρήσουμε μέσα στην κλάση τις δηλώσεις όλων των συναρτήσεων μελών, αλλά να τις ορίσουμε έξω από την κλάση.
- ✓ Εφαρμόζεται σε μια εφαρμογή που ο αριθμός των συναρτήσεων-μελών είναι μεγάλος, ώστε να μας βοηθήσει να ελέγξουμε καλύτερα την κλάση.
- ✓ Στον ορισμό της συνάρτηση ορίζουμε πάντα την κλάση στην οποία ανήκει. Σε αυτή την περίπτωση παρεμβάλεται το όνομα της κλάσης και το σύμβολο **::** **που ονομάζεται τελεστής διάκρισης ή αναγωγής εμβέλειας**

```
void Person::readData()
```

Συναρτήσεις-μέλη οριζόμενες μέσα στην κλάση

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

{
        private:
                char name[30];
                int age;
        public:
                void readData();
                {
                    cout <<"Enter name:";
                    cin >>name;
                    cout << "Enter age:";
                    cin >> age;
                }

                void printData();
                {
                    cout << "The name of the person is " << name << endl;
                    cout << "The age of the person is " << age << endl;
                }
};
 int main()
 {
        Person p1, p2;
        p1.readData();
        p1.printData();
        p2.readData();
        p2.printData();
 }
```

Απόδοση τιμών μέσω παραμέτρων (1/3)

```
#include <iostream>
#include <stdio.h>
#include <string.h>
using namespace std;

class Person
{
        private:
                char name[30];
                int age;
        public:
                void SetData(char name1[],int age1);
                void printData();
};

void Person::SetData(char name1[],int age1)
{
        strcpy(name,name1);
        age=age1;
}
```

Αντιγράφει το name1 στο name

Απόδοση τιμών μέσω παραμέτρων (2/3)

```
void Person::printData()
{
   cout << "The name of the person is " << name << endl;
   cout << "The age of the person is " << age << endl;
}

int main()
{
   Person P1;
   P1.SetData("Oikonomakos",58);
   P1.printData();
}
```

Απόδοση τιμών μέσω παραμέτρων (3/3)

- ✓ Ορίζεται μόνο ένα αντικείμενο (P1).
- ✓ Με την εντολή:

```
P1.SetData("Oikonomakos",58);
```

Καλείται η συνάρτηση-μέλος setData(), στην οποία μεταβιβάζονται οι τιμές "***Oikonomakos***" και ***58*** στις παραμέτρους ***name1*** και ***age1***, αντίστοιχα.

- ✓ Όταν εκτελείται ο κώδικας της συνάρτησης, οι τιμές αυτών των παραμέτρων αποδίδονται στα δεδομένα του αντικειμένου.

Συναρτήσεις κατασκευής

- ✓ Μερικές φορές είναι προτιμότερο για ένα αντικείμενο να λαμβάνει άμεσα αρχικές τιμές, την πρώτη φορά που δημιουργείται, χωρίς να απαιτείται να καλέσουμε μια ξεχωριστή συνάρτηση-μέλος.
- ✓ Η αυτόματη απόδοση αρχικών τιμών πραγματοποιείται με χρήση μίας ειδικής συνάρτησης-μέλους που ονομάζεται ***συνάρτηση κατασκευής (constructor)***.
- ✓ Η συνάρτηση κατασκευής είναι μια συνάρτηση-μέλος της κλάσης, η οποία υποχρεωτικά έχει ίδιο όνομα με την κλάση.
- ✓ Καλείται αυτόματα όταν δηλώνεται ένα αντικείμενο της κλάσης.
- ✓ Μια συνάρτηση κατασκευής δεν επιστρέφει τιμή ούτε δηλώνεται ως void.

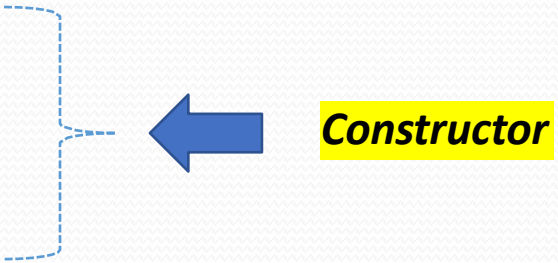
Συναρτήσεις κατασκευής

Εφαρμογή

Να δημιουργήσετε μια κλάση, η οποία να αντιπροσωπεύει ένα τραπεζικό λογαριασμό. Συγκεκριμένα:

i. να περιέχει μεταβλητές-μέλη (member-variables) ή ιδιότητες (attributes) που να τον περιγράφουν (το ποσό ανάληψης και το υπόλοιπο του λογαριασμού αρκούν),
ii. να περιέχει σχετική συνάρτηση κατασκευής ώστε να αρχικοποιείται ο λογαριασμός (=0),
iii. να περιέχει 3 συναρτήσεις-μέλη (member-functions) ή μεθόδους (methods) που να ενεργούν πάνω στις μεταβλητές του:
    i. μια που δέχεται σαν «είσοδό της» ένα ποσό, το οποίο να καθορίζει το επιθυμητό ποσό ανάληψης. Αυτή να ονομαστεί set_data.
    ii. μια που να διαβάζει το υπόλοιπο του λογαριασμού του καταθέτη. Αυτή να ονομαστεί read_balance.
    iii. μια που να εμφανίζει το υπόλοιπο του λογαριασμού του καταθέτη μετά την επιθυμητή ανάληψη. Σε περίπτωση που δεν υπάρχει υπόλοιπο να εμφανίζει σχετικό μήνυμα. Αυτή να ονομαστεί withdraw.

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

class Account
{
    private:
            double money;
            double balance;

    public:
            void set_data(double a);
            float read_balance();
            float withdraw();

            Account()
            {
                balance=0;
            }

};

void Account::set_data(double a)
{
        money=a;
}
```



Constructor

```
float Account::read_balance()
{
        cout<<"Dwse upoloipo logariasmou: ";
        cin>>balance;
        return balance;
}

float Account::withdraw()
{
        if (money<=balance)
        {
                balance=(balance-money);
                cout<<"To ypoloipo tou logariasmou einai: "<<balance;
        }
        else
                cout<<"To ypoloipo den eparkei gia to epithymito poso analipsis";
}
```

Εφαρμογή (συνέχεια)

- Στη συνέχεια, να γράψετε ένα πρόγραμμα, στο οποίο:

i. Να δηλώσετε **ένα αντικείμενο** της παραπάνω κλάσης, το οποίο να αντιπροσωπεύει ένα συγκεκριμένο λογαριασμό, π.χ. ac1.

ii. Κατόπιν, χρησιμοποιώντας τις κατάλληλες συναρτήσεις, να τυπώνει το υπόλοιπο του λογαριασμού.

```
int main()
{
        Account ac1;
        cout<<"Dwse epithumito poso analipsis: ";
        double poso;
        cin>>poso;
        ac1.set_data(poso);
        ac1.read_balance();
        ac1.withdraw();
}
```

Συναρτήσεις κατασκευής

- ✓ Στο πρόγραμμα ορίζεται μια κλάση Account, με ένα μέλος δεδομένων balance, που αναφέρεται στο υπόλοιπο ενός λογαριασμού.
- ✓ Ορίζονται επίσης τρείς συναρτήσεις μέλη για την ανάγνωση του υπολοίπου, την ανάγνωση του ποσού ανάληψης και μια συνάρτηση-μέλος που επιστρέφει, κατά την κλήση της, το τρέχον ποσό του λογαριασμού.
- ✓ Το τρέχον ποσό του λογαριασμού αρχικοποιείται στο 0. Αυτό επιτυγχάνεται κατά την δημιουργία ενός αντικειμένου, με τη συνάρτηση κατασκευής **Account( ), η οποία εκτελείται αυτόματα.**

Συναρτήσεις καταστροφής

- ✓ Μία συνάρτηση δόμησης καλείται αυτόματα όταν δημιουργείται ένα αντικείμενο.
- ✓ Μπορούμε αντίστοιχα να ορίσουμε μια συνάρτηση ***καταστροφής*** (destructor), η οποία θα καλείται αυτόματα όταν ένα αντικείμενο καταστρέφεται.
- ✓ Η συνάρτηση αποδόμησης έχει το ίδιο όνομα με την κλάση, έχοντας μπροστά **μια περισπωμένη ~** .
- ✓ Η πιο συνηθισμένη χρήση των συναρτήσεων καταστροφής είναι η αποδέσμευση της μνήμης που έχει δεσμευτεί για ένα αντικείμενο από τη συνάρτηση δόμησης.

Συναρτήσεις καταστροφής

```
#include <iostream>
using namespace std;
class Account
{
    private:
            double money;
            double balance;
    public:
            void set_data(double a);
            float read_balance();
            float withdraw();
            Account()
            {
                balance=0;
            }
            ~Account()
            {
                cout<<“The account has been deleted”;
            }
};
```

Destructor

Συναρτήσεις κατασκευής με υπερφόρτωση

```
#include <iostream>
using namespace std;
class Account
{
    private:
            double money;
            double balance;
    public:
            void set_data(double a);
            float read_balance();
            float withdraw();
            Account()
            {
                balance=0;
            }
            Account(float balance1)
            {
                balance=balance1;
            }
};
```

Συνάρτηση κατασκευής χωρίς ορίσματα

Συνάρτηση κατασκευής με όρισμα

Συναρτήσεις κατασκευής με υπερφόρτωση

- ✓ Δύο συναρτήσεις κατασκευής, μια με ορίσματα και μια χωρίς.
- ✓ Και οι δύο συναρτήσεις έχουν το ίδιο όνομα/ το όνομα της κλάσης.
- ✓ Στο συγκεκριμένο παράδειγμα θεωρούμε ότι η συνάρτηση κατασκευής έχει υποστεί ***υπερφόρτωση***.
- ✓ Το ποια συνάρτηση κατασκευής εκτελείται όταν δημιουργηθεί ένα αντικείμενο, εξαρτάται από τον αριθμό των ορισμάτων που χρησιμοποιούνται στον ορισμό του αντικειμένου.

Συναρτήσεις κατασκευής εναλλακτικός ορισμός

- ✓ Η συνάρτηση κατασκευής μπορεί εναλλακτικά να γραφτεί:

```
Account:: Account(float balance1):balance(balance1)
{
    ...
}
```

- ✓ Οι τιμές με τις οποίες θα αρχικοποιηθούν οι μεταβλητές-μέλη μπαίνουν σε παρενθέσεις μετά τα ονόματα των μεταβλητών μελών, ενώ αριστερά αυτών τίθεται ο προσδιοριστής **:**

Συναρτήσεις κατασκευής εναλλακτικός ορισμός

- ✓ Εάν στην κλάση υπάρχουν δύο ή περισσότερες μεταβλητές-μέλη (π.χ. float balance, float epitokio), οι οποίες θα λάβουν τις αρχικές τιμές balance1 και epitokio1, ο εναλλακτικός τρόπος της συνάρτησης κατασκευής θα γινόταν:

```
Account:: Account(float balance1):balance(balance1),epitokio(epitokio1)
{
    ...
}
```

Συναρτήσεις κατασκευής εναλλακτικός ορισμός

- ✓ Το σώμα της συνάρτησης κατασκευής μπορεί να περιέχει κώδικα π.χ. το επιτόκιο να είναι θετικό:

```
Account:: Account(float balance1):balance(balance1),
epitokio (epitokio1)
{
    if (epitokio<0)
        cout>>"Λάθος επιτόκιο. Δώστε τιμή"
}
```