ΚΑΤΑΣΚΕΥΑΣΤΕΣ – ΑΡΧΕΙΑ ΕΠΙΚΕΦΑΛΙΔΑΣ

ΚΑΤΑΣΚΕΥΑΣΤΕΣ (CONSTRUCTORS) – ΚΑΤΑΣΤΡΟΦΕΙΣ (DESTRUCTORS)

Όπως αρχικοποιούμε τις μεταβλητές μας με μία τιμή, έτσι πρέπει να αρχικοποιούμε και τα δεδομένα μέλη μιας κλάσης.

Αυτό επιτυγχάνεται με την χρήση μιας μεθόδου μέλους της κλάσης που ονομάζεται **μέθοδος εγκατάστασης** ή αλλιώς κατασκευαστής - **constructor**.

Επειδή είναι μέθοδος δηλώνεται στον χώρο δημόσιας ορατότητας public.

Ο κατασκευαστής ΔΕΝ καλείται παρόλο που είναι μεθοδος (συνάρτηση) αλλά τρέχει στο παρασκήνιο αυτόματα κατά την κατασκευή των αντικειμένων.

Constructor *(μέθοδος κατασκευής)*

- έχει το ίδιο όνομα με την κλάση,
- δηλώνεται μέσα ή και έξω από την κλάση ,
- μπορεί να έχει παραμέτρους που μπορεί να είναι προκαθορισμένες τιμές,
- δεν επιστρέφει τιμή, ούτε void.

Destructor *(μέθοδος καταστροφής)*
Κατά την κατασκευή του *constructor* δεσμεύεται χώρος στην μνήμη για να δεχθεί τον *constructor*.

Για να ''καθαρίσουμε'' την μνήμη από το αντικείμενο όταν αυτό τελειώσει την δουλειά του φτιάχνουμε μαζί του και μια μέθοδο καταστροφής ή αλλιώς **destructor.**

- ελευθερώνει την μνήμη που έχει δεσμευθεί για αντικείμενο.
- Δεν παίρνει ορίσματα δεν έχει εντολές και δεν επιστρέφει τιμή.
- Έχει και αυτός το ίδιο όνομα με την κλάση έχοντας μπροστά από το όνομά του μια περισπωμένη **(~)** .

Χρησιμοποιούμε δύο ειδών κατασκευαστές, τους κατασκευαστές **χωρίς ορίσματα** και τους κατασκευαστές **με ορίσματα**.

- Οι κατασκευαστές χωρίς ορίσματα αποδίδουν προκαθορισμένες αρχικές τιμές στα δεδομένα των αντικειμένων
- ενώ οι κατασκευαστές με ορίσματα αποδίδουν τις τιμές που τους δίνουν τα αντικείμενα κατά την κατασκευή τους. Οι τιμές αποδίδονται μέσω των ορισμάτων του κατασκευαστή.

Μπορούμε επίσης στην ίδια κλάση να χρησιμοποιούμε constructor με όρισμα και χωρίς.

Αυτό ονομάζεται ***υπερφόρτωση του constructor***.

Το ποιος constructor εκτελείται κατά την δημιουργία του αντικειμένου εξαρτάται από τα ορίσματα του constructor.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΔΗΛΩΣΗΣ ΚΑΤΑΣΚΕΥΑΣΤΩΝ

Παράδειγμα 1: ΟΡΙΣΜΟΣ ΚΑΙ ΧΡΗΣΗ ΚΑΤΑΣΚΕΥΑΣΤΗ - CONSTRUCTOR ΧΩΡΙΣ ΟΡΙΣΜΑΤΑ ΚΑΙ ΚΑΤΑΣΤΡΟΦΕΑ - DESTRUCTOR

Πιο κάτω φαίνεται ο ορισμός του κατασκευαστή χωρίς ορίσματα ο οποίος έχουμε επιλέξει να αρχικοποιεί και τα τέσσερα δεδομένα μέλη των αντικειμένων με προκαθορισμένες τιμές. Πιο συγκεκριμένα τα δεδομένα code, payrate και salary στην τιμή 0 ενώ το δεδομένο name στην τιμή +++. Όποιο αντικείμενο κατασκευάζεται θα παίρνει υποχρεωτικά αυτές τις τιμές.

```
#include<iostream>
#include<string.h>
using namespace std;
class Employee{
   private:
     int code;
     char name[30];
     double payrate;
     double salary;

   public:
    // ορισμός constructor χωρίς ορίσματα
    Employee(){
              code = 0;
              strcpy(name, "+++");
              payrate = 0;
              salary = 0;
      }

    //ορισμός destructor
    ~Employee(){
          cout << "\nDestructor!!!" << endl;
    }

    //Ορισμός μεθόδου εκτύπωσης τιμών
    void printData(){
        cout << "Employee Informations" << endl;
        cout << "Employee Code: "<< code<< endl;
        cout << "Employee Name: " << name << endl;
        cout << "Employee Payrate: " << payrate << endl;
        cout << "Employee Salary: " << salary << endl;
     }
};

int main(){
    Employee e1; //Το αντικείμενο παίρνει τις τιμές που του δίνει ο constructor
    e1.printData();

    return 0;
}
```

Παράδειγμα 2: ΟΡΙΣΜΟΣ ΚΑΙ ΧΡΗΣΗ ΚΑΤΑΣΚΕΥΑΣΤΗ - CONSTRUCTOR ΜΕ ΟΡΙΣΜΑΤΑ

Πιο κάτω φαίνεται ο ορισμός του κατασκευαστή με ορίσματα. Λειτουργεί όπως οι μέθοδοι απόδοσης τιμών μέσω παραμέτρων. Πιο συγκεκριμένα τα δεδομένα code, payrate και salary αρχικοποιούνται με τις τιμές των παραμέτρων _code, _payrate, _salary αντίστοιχα, ενώ το δεδομένο name στην τιμή της παραμέτρου _name, τις οποίες δίνει το αντικείμενο κατά την κατασκευή του.

```
#include<iostream>
#include<string.h>
using namespace std;
class Employee{ //Δήλωση της κλάσης
   private:
     int code;
     char name[30];
     double payrate;
     double salary;

   public:
    // ορισμός constructor με ορίσματα
    Employee(int _code, char _name[], double _payrate, double _salary){
             code = _code;
             strcpy(name, _name);
             payrate = _payrate;
             salary = _salary;
      }

      //ορισμός destructor
      ~Employee(){
          cout << "\nDestructor!!!" << endl;
      }

    //Ορισμός μεθόδου εκτύπωσης τιμών
    void printData(){
        cout << "Employee Informations" << endl;
        cout << "Employee Code: "<< code<< endl;
        cout << "Employee Name: " << name << endl;
        cout << "Employee Payrate: " << payrate << endl;
        cout << "Employee Salary: " << salary << endl;
     }

}; //Τέλος κλάσης

int main(){
    Employee e1(1, "Marios", 10, 80);//απόδοση τιμών από τον constructor στο αντικείμενο
    e1.printData();

    return 0;
}
```

Παράδειγμα 3: ΥΠΕΡΦΟΡΤΩΣΗ ΤΟΥ ΚΑΤΑΣΚΕΥΑΣΤΗ (Constructor Overloading)

Πιο κάτω φαίνεται η περίπτωση υπερφόρτωσης του κατασκευαστή. Ορίζουμε δύο αντικείμενα της κλάσης και το ποιος κατασκευαστής εκτελείται κατά την δημιουργία του αντικειμένου εξαρτάται από τα ορίσματα του κατασκευαστή. Ο καταστροφέας μόλις ολοκληρωθεί η εκτέλεση του προγράμματος θα εκτελεστεί δύο φορές για να αποδεσμεύσει τον χώρο που δέσμευσαν τα δύο αντικείμενα στη μνήμη.

```
#include<iostream>
#include<string.h>
using namespace std;
class Employee{ //Δήλωση της κλάσης
   private:
     int code;
     char name[30];
     double payrate;
     double salary;

   public:
    // ορισμός constructor χωρίς ορίσματα
    Employee(){
             code = 0;
             strcpy(name, "+++");
             payrate = 0;
             salary = 0;
      }

    // ορισμός constructor με ορίσματα
    Employee(int _code, char _name[], double _payrate, double _salary){
             code = _code;
             strcpy(name, _name);
             payrate = _payrate;
             salary = _salary;
      }

      //ορισμός destructor
      ~Employee(){
          cout << "\nDestructor!!!" << endl;
      }

    //Ορισμός μεθόδου εκτύπωσης τιμών
    void printData(){
        cout << "Employee Informations" << endl;
        cout << "Employee Code: "<< code<< endl;
        cout << "Employee Name: " << name << endl;
        cout << "Employee Payrate: " << payrate << endl;
        cout << "Employee Salary: " << salary << endl;
     }

}; //Τέλος κλάσης

int main(){
    /* Το πρώτο αντικείμενο (e1) μιλάει με τον constructor χωρίς ορίσματα
    ενώ το δεύτερο (e2) μιλάει με τον constructor με ορίσματα */
    Employee e1, e2(1, "Marios", 5, 40);
    e1.printData();
    cout << endl;
    e2.printData();

    return 0;
}
```

ΔΗΛΩΣΕΙΣ ΚΛΑΣΕΩΝ ΣΕ ΑΡΧΕΙΑ ΕΠΙΚΕΦΑΛΙΔΑΣ

Τις κλάσεις έχουμε μάθει να τις δηλώνουμε μέσα στο κυρίως πρόγραμμα.

Ένας άλλος τρόπος που υιοθετείται και είναι ο πιο σωστός είναι, τις δηλώσεις να τις κάνουμε σε ένα άλλο αρχείο το λεγόμενο **αρχείο επικεφαλίδας** και να το επισυνάπτουμε μέσα στο αρχείο του κυρίως προγράμματος μας.

Το αρχείο αυτό μπορεί να έχει οποιοδήποτε όνομα αλλά επέκταση **.hpp**. (π.χ. το πρόγραμμα μας είναι το **program.cpp** το αρχείο κεφαλίδας είναι το **class.hpp**).

Η επόμενη δουλειά είναι να εισάγουμε το αρχείο κεφαλίδας *class.hpp* μέσα στο αρχείο κώδικα *program.cpp*. Με αυτόν τον τρόπο χρησιμοποιούμε την κλάση μας σε κάθε διαφορετικό πρόγραμμα που φτιάχνουμε.

Έτσι οι πελάτες της κλάσης οι οποίοι δεν ενδιαφέρονται για τις λεπτομέρειες της κλάσης, μπορούν να την χρησιμοποιούν αφού όλες οι δηλώσεις υπάρχουν μέσα στο αρχείο επικεφαλίδας το οποίο διαβάζει ο compiler.

Αυτό είναι το πλεονέκτημα της επαναχρησιμοποίησης κώδικα.

Παράδειγμα 4: ΜΕΤΑΤΡΟΠΗ ΚΛΑΣΗΣ ΣΕ ΑΡΧΕΙΟ ΕΠΙΚΕΦΑΛΙΔΑΣ
Πιο κάτω φαίνεται το προηγούμενο πρόγραμμα μόνο που η κλάση τώρα έχει διαχωριστεί παπό το υπόλοιπο πρόγραμμα και έχει αποθηκευτεί σε ένα αρχείο με όνομα ***Employee.hpp.***

Την κλάση την ενσωματώνουμε μέσα στο πρόγραμμα χρησιμοποιώντας την εντολή συμπερίληψης include την οποία ακολουθεί το αρχείο Employee.hpp το οποίο βρίσκεται μέσα σε διπλά εισαγωγικά και όχι σε γωνιακές αγκύλες.

Το αρχείο που περιέχει την κλάση βρίσκεται μέσα στον φάκελο που είναι και το κυρίως πρόγραμμα.

Αρχείο επικεφαλίδας Employee.hpp

```
#ifndef EMPLOYEE_HPP_INCLUDED
#define EMPLOYEE_HPP_INCLUDED

#include<iostream>
#include<string.h> //συμπεριλαμβάνεται η κλάση string
using namespace std;
class Employee{ //Δήλωση της κλάσης
   private:
     int code;
     char name[30];
     double payrate;
     double salary;

   public:
    // ορισμός constructor χωρίς ορίσματα
    Employee(){
             code = 0;
             strcpy(name, "+++");
             payrate = 0;
             salary = 0;
      }

    // ορισμός constructor με ορίσματα
    Employee(int _code, char _name[], double _payrate, double _salary){
             code = _code;
             strcpy(name, _name);
             payrate = _payrate;
             salary = _salary;
      }
```

```
      //ορισμός destructor
      ~Employee(){
          cout << "\nDestructor!!!" << endl;
      }

    //Ορισμός μεθόδου εκτύπωσης τιμών
    void printData(){
        cout << "Employee Informations" << endl;
        cout << "Employee Code: "<< code<< endl;
        cout << "Employee Name: " << name << endl;
        cout << "Employee Payrate: " << payrate << endl;
        cout << "Employee Salary: " << salary << endl;
     }

}; //Τέλος κλάσης

#endif // EMPLOYEE_HPP_INCLUDED
```

Κυρίως πρόγραμμα

```
#include<iostream>
#include "Employee.hpp" //Συμπεριλαμβάνεται η κλάση Employee
using namespace std;


int main(){

    /* Το πρώτο αντικείμενο (e1) “μιλάει” με τον constructor χωρίς ορίσματα
    ενώ το δεύτερο (e2) “μιλάει” με τον constructor με ορίσματα */
    Employee e1, e2(1, "Marios", 5, 40);

    e1.printData();

    cout << endl;

    e2.printData();

    return 0;
}
```