Αντικειμενοστρεφής Προγραμματισμός I - C++
4ο Φυλλάδιο εργαστηρίου
Τετάρτη 23/03/2022

Σκοπός: Εξοικείωση με const συναρτήσεις μέλη και const αντικείμενα στην C++, overload συναρτήσεις, εμβέλεια μεταβλητών, συναρτήσεις με προκαθορισμένες τιμές παραμέτρων και inline συναρτήσεις.

Άσκηση 1η

Περιγράψτε την λειτουργία του παρακάτω προγράμματος. Εντοπίστε τυχόν λάθη που υπάρχουν.

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

class Point {
private :
    int x, y;
public :
    Point (int tmp_x, int tmp_y) : x(tmp_x), y(tmp_y) { }
    int getX() const { return x; }
    int getY() const { return y; }
    void addVal() const {
        x += 2;
        y += 2;
    }
};

int main() {
Point Point1(4, 5);
Point1.addVal();

Point Point2;
Point2.addVal();

cout << Point1.getX() << " " << Point1.getY() << "\n";
cout << Point2.getX() << " " << Point2.getY() << "\n";

return 0;
}
```

Απάντηση

Η μέθοδος addVal() δεν μπορεί να ορισθεί ως const διότι μεταβάλει τις τιμές των ιδιοτήτων της κλάσης (προσθέτει την τιμή 2). Επίσης για να δημιουργηθεί το αντικείμενο Point2 σύμφωνα με την 3η εντολή του προγράμματος θα πρέπει απαραίτητα να δημιουργήσουμε έναν default constructor. Ακολουθεί η διορθωμένη έκδοση του προγράμματος:

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

class Point {
private :
    int x, y;
```

```
public :
    // Λύση με προκαθορισμένες τιμές στον constructor
    Point (int tmp_x=0, int tmp_y=0) : x(tmp_x), y(tmp_y) { }

    // ή εναλλακτικά λύση με default constructor
    // Point () {x=0; y=0;}

    int getX() const { return x; }
    int getY() const { return y; }

    void addVal() {
        x += 2;
        y += 2;
    }
};

int main () {
Point Point1(4, 5);
Point1.addVal();

Point Point2;
Point2.addVal();

cout << Point1.getX() << " " << Point1.getY() << "\n";
cout << Point2.getX() << " " << Point2.getY() << "\n";

return 0;
}
```

Άσκηση 2η

Εντοπίστε τυχόν λάθη που υπάρχουν στο παρακάτω πρόγραμμα και διορθώστε τα ώστε να εκτελείται.

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

class TestConst {
    int number;
public:
    TestConst(int n = 0) {
        number = n;
    }

    int getNumber() {
        return number;
    }

    void displayNumber() const {
        cout << "Number is: " << number << endl;
    }

    void displayNewNumber() {
            cout << "New number is: " << 2*number << endl;
    }
};

int main() {
```

```
    const TestConst constObject(10);

    cout << constObject.getNumber() << endl;
    constObject.displayNumber();
    constObject.displayNewNumber();

    return 0;
}
```

Απάντηση

Όπως οι συναρτήσεις μέλη μιας κλάσης αλλά και οι παράμετροι των συναρτήσεων μπορούν να δηλωθούν ως const, και τα αντικείμενα μιας κλάσης που δημιουργούμε στην εφαρμογή μας μπορούμε επίσης να τα δηλώσουμε ως const. Στο παράδειγμα μας ένα τέτοιο αντικείμενο είναι το constObject. Ένα αντικείμενο που ορίζεται ως const δεν μπορεί να τροποποιηθεί και ως εκ τούτου μπορούμε να καλέσουμε με αυτό το αντικείμενο μόνο const συναρτήσεις. Οποιαδήποτε προσπάθεια για μεταβολή στοιχείων ενός const αντικειμένου θα δημιουργήσει λάθος μεταγλώττισης. Ένα const αντικείμενο μπορεί να αρχικοποιηθεί μόνο την στιγμή της δήλωσης του με την χρήση constructor. Ακολουθεί ο κώδικας διορθωμένος:

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

class TestConst {
    int number;
public:
    TestConst(int n = 0) {
        number = n;
    }

    int getNumber() const {
        return number;
    }

    void displayNumber() const {
        cout << "Number is: " << number << endl;
    }

    void displayNewNumber() const {
            cout << "New number is: " << 2*number << endl;
    }
};

int main() {
    const TestConst constObject(10);

    cout << constObject.getNumber() << endl;
    constObject.displayNumber();
    constObject.displayNewNumber();

    return 0;
}
```

Άσκηση 3η

Υλοποιήστε δύο διαφορετικές εκδόσεις της συνάρτησης με όνομα **average** (overloaded συναρτήσεις) στην C++ που θα επιστρέφουν τον μέσο όρο των παραμέτρων τους. Δηλαδή

όταν θα καλείται η συνάρτηση average με δύο πραγματικές παραμέτρους θα επιστρέφει τον μέσο όρο των πραγματικών αριθμών και όταν θα δέχεται ως παράμετρο έναν πίνακα πραγματικών αριθμών και το μέγεθος του θα επιστρέφει τον μέσο όρο του περιεχομένου του πίνακα. Υλοποιήστε την main συνάρτηση στην οποία θα πρέπει να ορίσετε έναν πραγματικό αριθμό και έναν ακέραιο αριθμό και καλέστε την συνάρτηση average. Τέλος ορίστε έναν πίνακα πραγματικών αριθμών και καλέστε ακόμα μια φορά τη συνάρτηση. Παρατηρήστε τη λειτουργία του προγράμματος.

Απάντηση

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

float average(float, float);
float average (float [], int);

int main() {
float a = 3.0;
int b = 2;
float c[5] = {20, 15.5, 30, 25.5, 40};

cout << "O mesos oros twn 2 arithmwn einai :" << average(a, b) << endl;
cout << "O mesos oros twn stoixeiwn tou pinaka einai :" << average(c, 5) << endl;
}


// Μέσος όρος πραγματικών αριθμών
float average(float x, float y) {
    return (x + y)/2.0;
}


// Μέσος όρος στοιχείων πίνακα
float average (float x[], int n) {
float sum = 0;
int i;

for (i = 0; i < n; i++)
    sum += x[i];

return sum/n;
}
```

Άσκηση 4η

Υπάρχει λάθος στο παρακάτω πρόγραμμα ή πρόκειται να εκτελεσθεί κανονικά; Σε περίπτωση που υπάρχει λάθος, διορθώστε τον κώδικα.

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

float f(float);
double f(double);

int main() {
float x = 15.05;
double y = 12.10;
```

```
cout << f(x) << endl;
cout << f(y) << endl;
cout << f(20);

return 0;
}

float f(float i) {
    return i/3.0;
}

double f(double i) {
    return i/2.0;
}
```

Απάντηση

Υπάρχει ασάφεια γιατί στην περίπτωση κλήσης της συνάρτησης με παράμετρο ακέραιο (cout << f(20);) ο compiler μπορεί να καλέσει και την πρώτη συνάρτηση με παράμετρο float αλλά και την δεύτερη με παράμετρο double.

Θα πρέπει να κάνουμε κατάλληλη μετατροπή στην παράμετρο για να δηλώσουμε στον compiler τι ακριβώς θέλουμε. π.χ. cout << f((double) 20) .

Αν καλέσουμε την συνάρτηση π.χ. με f(20.0) θα κληθεί εξ ορισμού η συνάρτηση με παράμετρο double. Εναλλακτικά μπορούμε να την καλέσουμε ως f(20.0d) για κλήση της double εκδοχής ή με f(20.0f) για κλήση της float εκδοχής.

Άσκηση 5η

Περιγράψτε τον τρόπο λειτουργίας του παρακάτω C++ προγράμματος. Υπάρχουν κάποια λάθη; Σε περίπτωση που εντοπίσατε λάθη, διορθώστε τον κώδικα.

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

char k = 'a';

int main()
{
    int k = 1;
    {
        int i = 2;
        {
            float k = 2.0;
            i = 3;

            cout << k << endl;
        }
    }

float k;
i = 4;

cout << ::k << " " << k << endl;

::k = 'b';

cout << ::k << " " << k << endl;
```

```
return 0;
}
```

Απάντηση

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

char k = 'a'; //το k είναι καθολική μεταβλητή

int main()
{
int k = 1;   //το συγκεκριμένο k είναι τοπικό στην main
             //αυτός ο ορισμός υπερκαλύπτει την καθολική μεταβλητή

    { // αρχή 1ου block
    int i = 2; //η i μεταβλητή είναι τοπική του 1ου block
        { // αρχή 2ου block
        float k = 2.0; // το k είναι τοπική μεταβλητή του 2ου block
        i = 3;

        cout << k << endl;
        } // τέλος 2ου block
    } // τέλος 1ου block

float k; // Λάθος, δεν μπορεί να γίνει επανα-ορισμός του k
i = 4; // Λάθος, η μεταβλητή i δεν έχει οριστεί στο συγκεκριμένο block

cout << ::k << " " << k << endl;

::k = 'b';   // Χρησιμοποιείται ο συγκεκριμένος τελεστής
             // για προσπέλαση στη καθολική μεταβλητή k

cout << ::k << " " << k << endl;
return 0;
}
```

Άσκηση 6η

Έστω ότι σε ένα πρόγραμμα ορίζουμε την ακόλουθη συνάρτηση:

```
void functionDefaultParam(double x = 7.3, int y = 4, char z = '*')
```

Ποιες από τις παρακάτω κλήσεις της συνάρτησης, θεωρείτε ότι είναι σωστές:

```
1. functionDefaultParam();
2. functionDefaultParam(2.8);
3. functionDefaultParam(3.2, 0, 'h');
4. functionDefaultParam(9.2, '!');
5. functionDefaultParam(7, 3);
```

Απάντηση

Σωστές κλήσεις της συνάρτησης **functionDefaultParam** είναι οι: 1, 2, 3 και 5.
Στην 4η κλήση θέλουμε να ορίσουμε τιμές για την 1η και 3η παράμετρο (για αυτό περνάμε ως παραμέτρους έναν πραγματικό αριθμό και έναν χαρακτήρα) αλλά στην ουσία με αυτό τον τρόπο ορίζουμε τιμή για την 1η και 2η παράμετρο.
Η 2η παράμετρος (ακέραιος) θα πάρει ως τιμή, την τιμή του χαρακτήρα '*' με βάση τον πίνακα Ascii.

Άσκηση 7η

Υλοποιήστε πρόγραμμα στην C++ που θα υπολογίζει το εμβαδόν ενός τραπεζίου. Υπενθυμίζεται ότι το εμβαδόν ενός τραπεζίου είναι ίσο με το γινόμενο του ημιαθροίσματος των βάσεων με το ύψος του ($E = \frac{(B+\beta)\cdot\upsilon}{2}$). Ο υπολογισμός του εμβαδού θα γίνεται με τη χρήση κατάλληλης συνάρτησης στην οποία θα ορίζονται ως προκαθορισμένες τιμές στην μικρή και στην μεγάλη βάση η τιμή 2 και στο ύψος η τιμή 4. Στο κύριο πρόγραμμα να κληθεί η συνάρτηση με τους εξής εναλλακτικούς τρόπους :

- χωρίς να δίνονται παράμετροι
- να δίνεται μόνο η μεγάλη βάση
- να δίνεται μόνο η μεγάλη βάση και η μικρή βάση
- να δίνονται και οι τρεις παράμετροι

Απάντηση

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

double emvadon_trapeziou(int vasi_megali=2, int vasi_mikri=2, int ipsos=4) {
    return ((vasi_megali+vasi_mikri)*ipsos/2.0);
}

int main() {
double e;

e = emvadon_trapeziou();
cout << "To emvadon tou trapeziou stin 1h periptwsi einai: " << e << endl;

e = emvadon_trapeziou(5);
cout << "To emvadon tou trapeziou stin 2h periptwsi einai: " << e << endl;

e = emvadon_trapeziou(5, 3);
cout << "To emvadon tou trapeziou stin 3h periptwsi einai: " << e << endl;

e = emvadon_trapeziou(5, 3, 3);
cout << "To emvadon tou trapeziou stin 4h periptwsi einai: " << e << endl;

return 0;
}
```

Άσκηση 8η

Να υλοποιήσετε συνάρτηση σε C++ η οποία εντοπίζει και επιστρέφει τον μέγιστο από δύο ακέραιους αριθμούς. Ορίστε τη συνάρτηση ως εμβόλιμη (inline).

Απάντηση

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

inline int greatest(int x, int y) {
int max;

max = (x > y) ?  x : y;

return max;
}
```

```
int main( ) {
int num1, num2;

cout<<"enter two values: ";
cin >> num1 >> num2;

cout << "greatest of two numbers: " << greatest(num1, num2);
}
```

Inline συναρτήσεις: Γνωρίζουμε ότι ο κώδικας μιας συνάρτησης υπάρχει μόνο μια φορά γραμμένος μέσα στο μεταγλωττισμένο πρόγραμμα. Κάθε φορά που καλείται μια συνάρτηση ο έλεγχος του προγράμματος μεταφέρεται σε αυτό το τμήμα κώδικα, και με την ολοκλήρωση του επιστρέφει πίσω αμέσως μετά από το σημείο κλήσης. Όλη αυτή η διαδικασία επιφέρει καθυστερήσεις. Στις περιπτώσεις μικρών σε μέγεθος συναρτήσεων, συμβαίνει ο κώδικας που απαιτείται για την κλήση και την επιστροφή να είναι περισσότερος από τον κώδικα της ίδιας της συνάρτησης.

Για αυτές τις περιπτώσεις η C++ έχει προβλέψει τις **εμβόλιμες** (**inline συναρτήσεις**). Όταν ο μεταγλωττιστής συναντήσει τη κλήση μιας εμβόλιμης συνάρτησης, τότε παρεμβάλει τον κώδικα της συνάρτησης στο σημείο της κλήσης (όσες φορές και αν κληθεί). Έχει νόημα για μικρού μεγέθους συναρτήσεις. Η δήλωση inline αποτελεί *προτροπή* προς τον μεταγλωττιστή που μπορεί να αποφασίσει να την μεταγλωττίσει και ως μια κανονική συνάρτηση (π.χ. λόγω μεγάλου μεγέθους). Για να μπορέσει ο μεταγλωττιστής να κάνει μια συνάρτηση εμβόλιμη, θα πρέπει να έχει συναντήσει τον ορισμό της (όχι τη δήλωση της) πριν από την πρώτη κλήση της.