Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Αντικειμενοστρεφής Προγραμματισμός

Εργαστήριο 4

Δείκτες (pointers)

Οι δείκτες στην C++ δουλεύουν όπως και στην C, με την εξαίρεση της διαδικασίας δέσμευσης και αποδέσμευσης μνήμης.

Για να δεσμεύσουμε δυναμικά μνήμη στην C++ χρησιμοποιούμε πλέον τον τελεστή new αντί για την συνάρτηση malloc:

```
char* str = new char[10];  //δέσμευση 10 bytes
```

Κάθε κομμάτι μνήμης που δεσμεύεται θα πρέπει πάντα να ελευθερώνεται όταν παύει να μας είναι χρήσιμο. Για την διαγραφή/απελευθέρωση της μνήμης αυτής χρησιμοποιούμε τον τελεστή delete (αντί για την συνάρτηση free):

```
delete []str;  //προσοχή  στις αγκύλες!
```

Μπορούμε να χρησιμοποιούμε pointers σε objects και να δεσμεύσουμε χώρο για ένα νέο αντικείμενο:

```
Point *obj;  //pointer σε MyClass
obj = new Point(2, 3); //δεσμεύουμε χώρο και καλούμε παράλληλα τον constructor
```

Για να αποκτήσουμε πρόσβαση στα περιεχόμενα του αντικειμένου (εννοείται μόνο στα public στοιχεία του) χρησιμοποιούμε τον τελεστή ->

```
obj->x = 2;
obj->y = 4;
```

Το αντικείμενο αυτό θα πρέπει να καταστραφεί ως εξής:

```
delete obj; //χωρίς αγκύλες
```

Αναφορές (references)

Όταν καλούμε συναρτήσεις, οι τιμές των ορισμάτων τους μεταβιβάζονται σ' αυτές ακολουθώντας τους κανόνες της κλήσης με τιμή (by value). Αυτό έχει ως συνέπεια οι τροποποιήσεις στα ορίσματα στο εσωτερικό των συναρτήσεων να αφορούν στα αντίγραφα των ορισμάτων και όχι στα αρχικά ορίσματα. Αν θέλουμε η συνάρτηση να τροποποιήσει τις τιμές των μεταβλητών που δίνονται ως ορίσματα θα πρέπει να δώσουμε ορίσματα δεικτών στις μεταβλητές και να συντάξουμε ανάλογα τον κώδικα.
Η C++ μας παρέχει τη δυνατότητα να έχουμε μεταβίβαση ορισμάτων σε συναρτήσεις με αναφορά (by reference) και όχι με τιμή. Αυτό μπορούμε να το επιτύχουμε αν στη διακήρυξη της συνάρτησης προτάξουμε τον τελεστή & στα ορίσματα που θέλουμε να καλέσουμε με αναφορά.

Παράδειγμα:
Δοκιμάστε να τρέξετε το παρακάτω πρόγραμμα. Η μεταβλητή a αλλάζει ή όχι, και γιατί;

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

void increase(int num) // num: copy of variable a
{
	num++;
}

int main()
{
	int a = 4;
	for(int i = 0; i<5; i++)
	{
		increase(a);
		cout << a << endl ;
	}
	system("pause");
	return 0;
}
```

Για να αλλάξει η μεταβλητή a έχουμε δύο τρόπους:

- Μέσω pointer
- Μέσω αναφοράς (reference)

Παράδειγμα με χρήση pointer:

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

void increase(int *num) // num: pointer of type int
{
	(*num)++;
}

int main()
{
	int a = 4;
	for(int i = 0; i<5; i++)
	{
		increase(&a); //we pass the address of variable a
		cout << a << endl ;
	}
	system("pause");
	return 0;
}
```

Παράδειγμα με χρήση αναφοράς:

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

void increase(int &num) // num: reference of variable a
```

```
{
	num++;
}

int main()
{
	int a = 4;
	for(int i = 0; i<5; i++)
	{
		increase(a);
		cout << a << endl ;
	}
	system("pause");
	return 0;
}
```

Η αναφορά είναι ουσιαστικά ένα ψευδώνυμο της αρχικής μεταβλητής. Αν αλλάξει η τιμή της αρχικής μεταβλητής τότε αλλάζει και η αναφορά. Αντίστοιχα, αν αλλάξει η τιμή της αναφοράς (όπως γίνεται στο παραπάνω παράδειγμα, τότε αλλάζει και η τιμή της αρχικής τιμής (αφού είναι το ίδιο πράγμα)

Άσκηση 1:
Θεωρήστε μια συνάρτηση swap, σκοπός της οποίας είναι η ανταλλαγή των τιμών δύο μεταβλητών. Αν η συνάρτηση αυτή κληθεί όπως παρακάτω, θα έχει επιτύχει τον σκοπό της και γιατί;

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

void swap(int a, int b)
{
	int temp = a;
	a = b;
	b = temp;
}

int main()
{
	int a = 3;
	int b = 4;
	swap(a, b);
	cout << a << ", " << b;
	system("pause");
	return 0;
}
```

Διορθώστε την συνάρτηση (**μια έκδοση με pointers και μια με references**) ώστε να εκτελεί τη προσδοκώμενη λειτουργία

Ο δείκτης this.

Οι συναρτήσεις-μέλη (member functions) μιας κλάσης έχουν αυτόματη πρόσβαση σε ένα δείκτη που ονομάζεται *this*, και δείχνει στο ίδιο το αντικείμενο. Με αυτό τον τρόπο, κάθε συνάρτηση αποκτά άμεση πρόσβαση στο ίδιο το αντικείμενο που την καλεί (π.χ. μπορεί να διαβάσει την διεύθυνσή του).

Δύο βασικές χρήσεις του δείκτη this:

1) Όταν θέλουμε να ξεχωρίσουμε μεταξύ μεταβλητών της κλάσης και άλλων μεταβλητών. Στο παρακάτω παράδειγμα η κλάση Counter έχει μια μεταβλητή i. Σην συνάρτηση setCount(int i) χρησιμοποιούμε μια μεταβλητή που επίσης ονομάζεται "i" ως παράμετρο στην συνάρτηση. Για να διακρίνουμε μεταξύ των δύο μεταβλητών χρησιμοποιούμε τον δείκτη this.
2) Όταν θέλουμε να υλοποιήσουμε μια συνάρτηση ώστε να επιστρέφει *το ίδιο το αντικείμενο*. Η παρακάτω συνάρτηση increase( ) δεν επιστρέφει κάτι ενώ η increaseBetter( ) επιστρέφει το ίδιο το αντικείμενο. Με αυτό τον τρόπο μπορούμε να το αντιγράψουμε σε ένα άλλο αντικείμενο.

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

class Counter
{
	int i;

public:
	Counter():i(0) { }

	void setCount(int i)
	{
		this->i = i;	//we use "this" to distinguish
					//between the 2 variables named "i"
	}

	void increase()
	{
		i++;
	}

	Counter& increaseBetter()
	{
		i++;
		//"this" is the pointer to the current object
		//"*this" is the current object itself
		return *this;
	}

	void display()
	{
		cout << "counter = " << i << endl;
	}
};

int main()
{
	Counter c;
	Counter d;
	c.increase();
	//Counter c is increased and at the same time is assigned to Counter d
	d = c.increaseBetter();

	c.display();
	d.display();

	system("PAUSE");
}
```

Άσκηση 2
Σε συνέχεια της Άσκησης 3 του Εργαστηρίου 3 να υλοποιήσετε τον τελεστή += για την προσθήκη κλασμάτων.
Παράδειγμα:

```
Fraction f1(3, 4);
Fraction f2(2, 3);
Fraction f3;
f3 = f1 += f2;
```

Προσοχή: Προσθέτουμε το κλάσμα f2 στο f1, άρα:

- Το κλάσμα d1 αλλάζει και γίνεται ίσο με f1 + f2
- Το αλλαγμένο κλάσμα f1 ανατίθεται στο κλάσμα f3, το οποίο είναι αρχικοποιημένο ίσο με μηδέν (χρησιμοποιήστε τον δείκτη this)

Υπερφόρτωση συντελεστή ανάθεσης

Στο προηγούμενο εργαστήριο μιλήσαμε για υπερφόρτωση τελεστών. Μια ειδική και πολύ σημαντική περίπτωση υπερφόρτωσης είναι η υπερφόρτωση του συντελεστή ανάθεσης (=).

Στις περιπτώσεις υπερφόρτωσης συντελεστών που είδαμε στο προηγούμενο εργαστήριο, η συνάρτηση υπερφόρτωσης επέστρεφε ένα νέο αντικείμενο, π.χ με την εντολή:

```
return Counter(++count);
```

Το νέο αυτό αντικείμενο μπορεί να ανατεθεί σε μια νέα μεταβλητή, πχ.:

```
c2 = ++c1;
```

Στην περίπτωση της υπερφόρτωσης των συντελεστών ανάθεσης, γενικώς δεν είναι καλό να δημιουργούνται τέτοια επιπλέον αντικείμενα. Για τον λόγο αυτό επιστρέφουμε το reference του ίδιου του αντικειμένου, αντί για ένα νέο αντίγραφό του. Για να το πετύχουμε αυτό χρησιμοποιούμε τον δείκτη this:

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

class alpha
   {
   private:
      int data;
   public:

      alpha()                        //no-arg constructor
      { }

      alpha(int d)                   //one-arg constructor
      { data = d; }

      void display()                 //display data
      { cout << data; }

      alpha& operator = (alpha& a) //overloaded = operator
      {
         data = a.data;              //not done automatically
         cout << "\nAssignment operator 1 invoked";
         return *this;               //return copy of this alpha
      }
```

```
      alpha& operator = (int number) //overloaded = operator
      {
         data = number;            //not done automatically
         cout << "\nAssignment operator 2 invoked";
         return *this;             //return copy of this alpha
      }
   };

int main()
{
   alpha a1(37);
   alpha a2, a3, a4;

   a3 = a2 = a1;                   //invoke overloaded =, twice
   a4 = 2;
   cout << "\na2="; a2.display();  //display a2
   cout << "\na3="; a3.display();  //display a3
   cout << "\na4="; a4.display();  //display a3
   cout << endl;
   return 0;
}
```