ΤΕΙ ΣΕΡΡΩΝ

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Δείκτες

Ευάγγελος Γ. Ούτσιος
Θεόδωρος Γ. Λάντζος

Διάλεξη Νο7

Δείκτες

- Μια μεταβλητή που περιέχει τη διεύθυνση μιας άλλης μεταβλητής.
- Εργαλείο ευελιξίας και δύναμης στην υλοποίηση σύνθετων προβλημάτων
- Χρήσεις
  - Προσπέλαση στοιχείων πίνακα
  - Μεταβίβαση ορισμάτων σε συνάρτηση
  - Μεταβίβαση πινάκων και αλφαριθμητικών σε συναρτήσεις
  - Απόκτηση διεύθυνση μνήμης από το σύστημα
  - Δημιουργία δομών δεδομένων
- Έννοια της έμμεσης χρήσης

Δείκτες και τελεστές

- Τελεστές
  - Διεύθυνσης & (address operator)

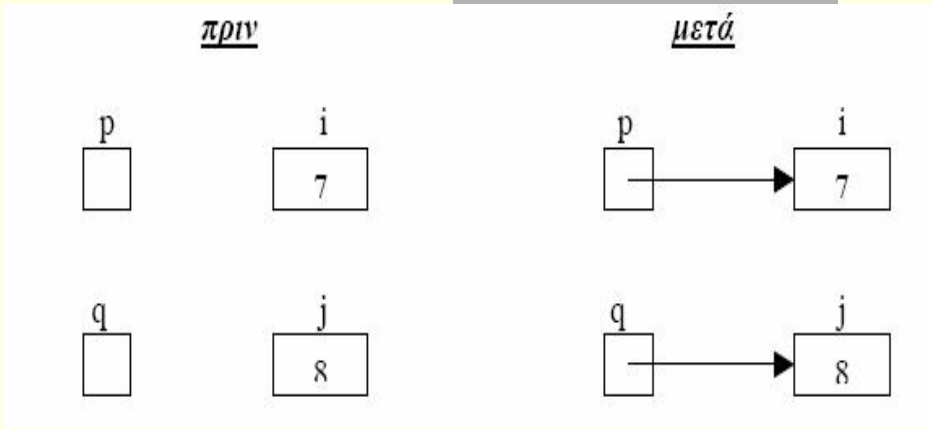
    Μοναδιαίος τελεστής και επιστρέφει ως τιμή τη διεύθυνση μνήμης της τιμής του τελεσταίου
  - Έμμεσης αναφοράς * (indirection operator)
    - Μοναδιαίος τελεστής ο οποίος επιστρέφει ως τιμή την τιμή της μεταβλητής, η οποία είναι τοποθετημένη στη διεύθυνση μνήμης την οποία έχει ως τιμή ο τελεσταίος
  - Δείκτης δομής -> (structure pointer operator)

Μεταβλητή	Δήλωση	Εκχώρηση	Τιμή	Διεύθυνση
i	int i;	i=7l	7	1000
				1001
p	int *p;;	p=&7;	1000	1002
				1003
j	int j;	j=*p;	7	1004

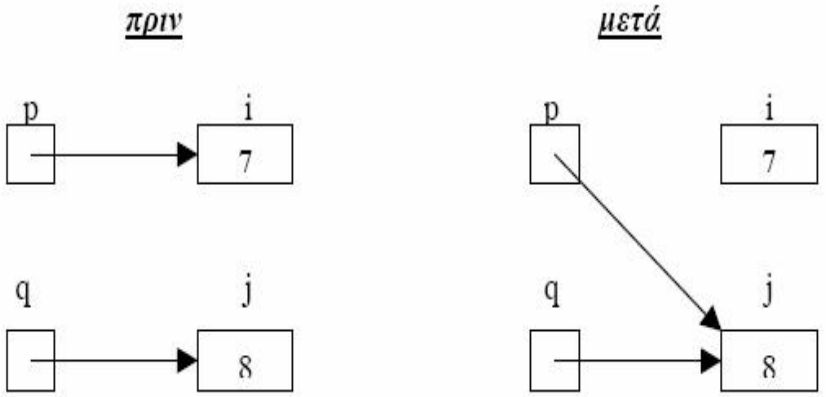
Δείκτες και χρήση με παραδείγματα

```
int *p, *q;
int i = 7, j = 8 ;
```

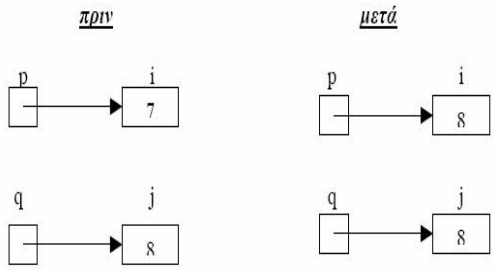
a) `p = &i;` *// εκχώρηση της διεύθυνσης της μεταβλητής i στον δείκτη p,*
// ή ο p δείχνει στην i .
`q = &j;` *// εκχώρηση της διεύθυνσης της μεταβλητής j στον δείκτη q,*
// ή ο q δείχνει στην j .



b) `p = q;` *// εκχώρηση της τιμής του δείκτη q στο δείκτη p,*
// ή εκεί που δείχνει ο q να δείχνει και ο p .



c) `*p = *q;` *// το περιεχόμενο της μεταβλητής όπου δείχνει ο q, εκχωρείται*
// ως περιεχόμενο στη μεταβλητή που δείχνει ο p.



Δείκτες και Δομές

Έστω η δομή

```
struct imerominia
{
    int mera;
    int minas;
    int etos;
};
```

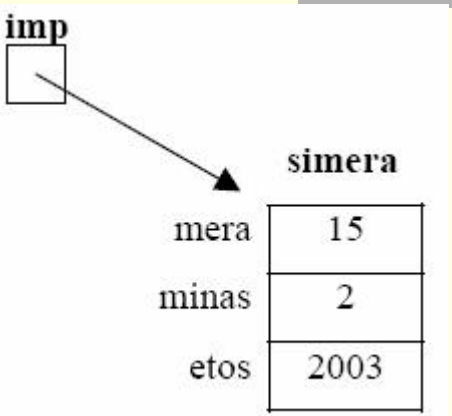
Δείκτες σε Δομές

Έστω η δομή

```
struct imerominia
{
      int mera;
      int minas;
      int etos;
};
```

Τότε μπορούμε να έχουμε δηλώσεις όπως

```
struct imerominia simera, *imp;
```



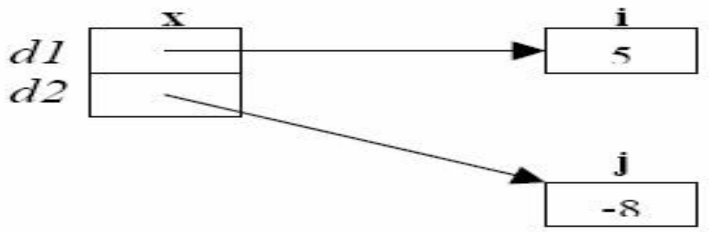

```
imp = &simera;
```

```
(*imp).mera = 15;
```

```
imp->mera = 15;      // αντί (*imp).mera = 15;
imp->minas = 2;      // αντί (*imp).minas = 2;
imp->etos = 2003;    // αντί (*imp).etos = 2003;
```

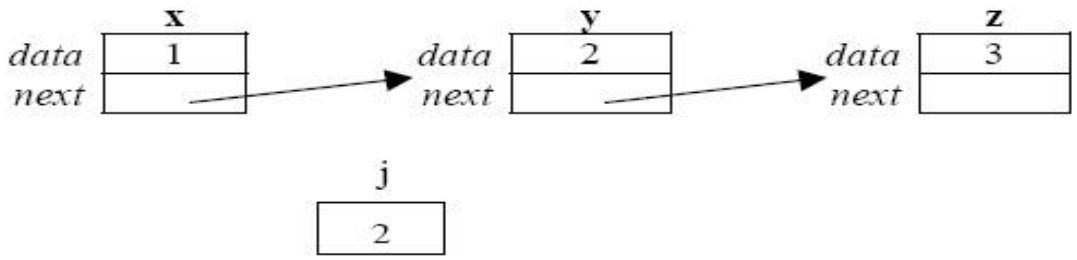
Δείκτες εντός δομών

```
struct deiktes
{
   int *d1;
   int *d2;
};

main()
{
   struct deiktes x;
   int i = 5, j;

   x.d1 = &i;    // η διεύθυνση της μεταβλητής i εκχωρείται στο δείκτη x.d1
   x.d2 = &j;    // η διεύθυνση της μεταβλητής j εκχωρείται στο δείκτη x.d2
   *x.d2 = -8;   // η τιμή –8 εκχωρείται ως περιεχόμενο στη μεταβλητή
                 // που δείχνει ο δείκτης x.d2
}
```




```
struct node
{
   int data;
   struct node *next;
};
```

```
struct node x, y, z;
int j;
x.data = 1 ;
y.data = 2 ;
z.data = 3;
x.next = &y;  // ο δείκτης x.next δείχνει στη δομή y
y.next = &z;  // ο δείκτης y.next δείχνει στη δομή z
j = x.next->data; // το περιεχόμενο του μέλους data της δομής όπου δείχνει
                  // ο δείκτης x.next εκχωρείται στη μεταβλητή j. Θα μπορούσαμε
                  // ισοδύναμα να πούμε  j = y.data
```



Δείκτης Γενικής Χρήσης void

```
// δείκτες για τύπο void
#include <iostream.h>
main()
{
   int intvar;
   float flovar;
   int *ptrint;          // ορισμός δείκτη για int
   float *ptrflo;        // ορισμός δείκτη για float
   void *ptrvoid;        // ορισμός δείκτη για void

   ptrint = &intvar;     // σωστό, απόδοση int* σε int*
//  ptrint = &flovar;     // λάθος, απόδοση float* σε int*

//  ptrflo = &intvar;     // λάθος, απόδοση int* σε float*
   ptrflo = &flovar;     // σωστό, απόδοση float* σε float*

   ptrvoid = &intvar;    // σωστό, απόδοση int* σε void*
   ptrvoid = &flovar;    // σωστό, απόδοση int* σε void*
}
```

Δείκτες και συναρτήσεις

Κλήση με τιμή

```
int  athroisma(int x, int y) ;
main()
{
   int  a, b;
   .
   sum = athroisma(a, b);    // μεταβιβάζονται οι τιμές των μεταβλητών
   .
   .
}
int  athroisma(int x, int y)
{
   return x+y ;
}
```

Κλήση με αναφορά

```
void  praxeis(int &s, int &d, int x, int y);
main()
{
   int  a, b, sum, diff;
   .
   praxeis(sum, diff, a, b);
   .
   .
}
void  praxeis(int &s, int &d, int x, int y)
{
   s = x+y;
   d = x-y;
}
```

Κλήση με δείκτες

```
void  praxeis(int *s, int *d, int x, int y);
main()
{
   int  a, b, sum, diff;
   .
   praxeis(&sum, &diff, a, b);
   .
   .
}
void  praxeis(int *s, int *d, int x, int y)
{
   *s = x+y;
   *d = x-y;
}
```

Διαχείριση μνήμης: new - delete

- Διαχείριση μνήμης με χρήση δύο τελεστών new και delete
- Παρόμοια χρήση με την συνάρτηση malloc()
- Η new επιστρέφει ένα δείκτη για το κατάλληλο τύπο δεδομένων
- Η delete αποδεσμεύει το χώρο μνήμης που είχε δεσμευτεί

```
#include <iostream.h>
#include <string.h>
struct  person
{
   int id;
   char name[20];
}
main()
{
    struct person *ptr;

   ptr = new struct person[1];
   cout << "Δώσε κωδικό:";
   cin >> ptr->id;
   cout << "Δώσε ένα όνομα:";
   cin >> ptr->name;
   delete ptr;
}
```

Δείκτες για αντικείμενα

```
class Person
{
   private:
         int id;
         char name[20];
   public:
         void readData()
         {
            cout << "Δώσε κωδικό:";
            cin >> id;
            cout << "Δώσε ένα όνομα:";
            cin >> name;
         }
         void printData()
         {
            cout << "Κωδικός: " << id << endl;
            cout << "Όνομα: " << name << endl;
         }
};
main()
{
   Person  *p;

   p = new Person;
   p->readData();
   p->printData();
}
```

Πίνακας Δεικτών προς Αντικείμενα

```
class Person
{
   private:
         int id;
         char name[20];
   public:
         void readData()
         {
            cout << "Δώσε κωδικό:";
            cin >> id;
            cout << "Δώσε ένα όνομα:";
            cin >> name;
         }
         void printData()
         {
            cout << "Κωδικός: " << id << endl;
            cout << "Ονομα: " << name << endl;
         }
};
main()
{
   Person  *perspin[100];
   int i = 0, j;
   char choice;
   do
   {
      perspin[i] = new Person;
      perspin[i]->readData();
      i++;
      cout << "More?";
      cin >> choice;
   }
   while (choice == 'y');
   for (j=0; j<i; j++)
         perspin[j]->printData();
}
```

Ο δείκτης this

```
#include <iostream.h>
#include <stdio.h>
class Employee
{
  private:
        float mikta;
        int categ;
  public:
        Employee()
        {
           this->mikta = 0;
           this->categ = 0;
        }
        Employee(float mikta0, int categ0)
        {
           this->mikta = mikta0;
           this->categ = categ0;
        }
        void showAddress()
        {
           cout << "The address of the object is " << this << endl;
        }
        float calcTax()
        {
           float foros;
           if (this->categ == 1)
              foros = this->mikta * 0.15;
           else
              foros = this->mikta * 0.3;
           return foros;
        }
        float calcSalary()
        {
           float misthos;
           misthos = this->mikta - this->calcTax();
           return misthos;
        }
};
main()
{
  Employee emp(1000,1);

  emp.showAddress();
  cout << "The salary is: " << emp.calcSalary() << endl;
}
```