ΥΠΟΛΟΓΙΣΤΕΣ ΙΙ

ΔΕΙΚΤΕΣ

Τι περιλαμβάνει μια μεταβλητή;

- Πρώτα να δούμε τι ακριβώς συμπεριλαμβάνει μια μεταβλητή
  - τύπος
    - Καθορίζει το μέγεθος στην μνήμη σε Bytes
  - τιμή
    - Η αριθμητική τιμή που αποθηκεύεται στην μνήμη
- Τι ξεχάσαμε; Το πού αποθηκεύεται στην μνήμη...
  - διεύθυνση
- Κάθε μεταβλητή χαρακτηρίζεται από 3 ιδιότητες:
  - τύπος, τιμή, διεύθυνση

Διεύθυνση μεταβλητής

- Η ανάθεση της διεύθυνσης μιας μεταβλητής γίνεται κατα την εκτέλεση. Ο χρήστης δεν γνωρίζει που αποθηκεύεται.
- Μήπως όμως θα 'πρεπε;
- Η C++ μας δίνει την δυνατότητα να γνωρίζουμε την διεύθυνση μιας μεταβλητής, και να χρησιμοποιήσουμε αυτή την γνώση για καλύτερο προγραμματισμό.
  - *Λίστες, βάσεις δεδομένων, γραφικά, λειτουργικά συστήματα*
- **Δείκτης:** μια νέα μεταβλητή, που αντί για αριθμητικές τιμές, αποθηκεύει διευθύνσεις μνήμης

Δείκτης

- Μεταβλητή που «δείχνει» σε κάποια θέση μνήμης
- Συστατικά δείκτη
  - Τύπος
    - Καθορίζει τι τύπου μεταβλητή έχει αποθηκευθεί εκεί που δείχνει (π.χ. `int` ή `double`)
  - Τιμή
    - Μια διεύθυνση μνήμης
- Για να έχει νόημα ένας δείκτης θα πρέπει να «δείχνει» σε κάποια υπάρχουσα μεταβλητή

Δήλωση δείκτη

- Χρησιμοποιείται ο τελεστής αστέρι «*»
  - Δυστυχώς είναι το ίδιο με τον τελεστή πολλαπλασιασμού, οπότε θέλει προσοχή
- Δηλώνεται όπως και οι απλές μεταβλητές αλλά με το αστέρι μπροστά

Παράδειγμα: δηλώστε δύο ακέραιους και δύο δείκτες σε ακέραιους

```
int main(){
      int n, m;
      int *k, *j;
.
.
}
```


Ανάθεση τιμών σε δείκτη

- Αναθέτουμε σε δείκτη την διεύθυνση μιας μεταβλητής. Η μεταβλητή αυτή πρέπει:
  - να υπάρχει
  - να είναι του ίδιου τύπου
- Η ανάθεση γίνεται με τον τελεστή «&»

Παράδειγμα: δηλώστε δύο ακέραιους και δύο δείκτες σε ακέραιους
Κατόπιν αναθέστε στους δείκτες τις διευθύνσεις των ακεραίων

```
int main(){
      int n, m;
      int *k, *j;

      k = &n;
      j = &m;
}
```


Πράξεις με δείκτες

- Μια βασική λειτουργία/πράξη με δείκτες είναι:
  - ποια η τιμή της μεταβλητής εκεί πού δείχνει ένας δείκτης;
- Ο τελεστής «*» χρησιμοποιείται για να μας δώσει την αριθμητική τιμή στην θέση που δείχνει ο δείκτης

```
int main(){
      int n, m;
      int *k, *j;
      n = 7; m = 8;

      k = &n; j = &m;

      cout << *k << *j <<endl;
}
```


Πράξεις με δείκτες

- Επιτρέπεται η απευθείας ανάθεση της διεύθυνσης ενός δείκτη σε άλλον δείκτη
  - ...δηλαδή επιτρέπεται το «=»

```
int main(){
      int n, m;
      int *k, *j;
      n = 7; m = 8;

      k = &n; j = k;

      cout << *k << *j <<endl;
}
```

Τι αποτέλεσμα θα δώσει το παραπάνω;

Ανάθεση με δείκτες

- Μπορούμε να αναθέτουμε αριθμητική τιμή σε μεταβλητές
  - Απευθείας (όπως κάναμε μέχρι τώρα)
  - Μέσω δεικτών προς τις μεταβλητές αυτές

Τα δύο παρακάτω είναι ουσιαστικά ταυτόσημα

```
int main(){
      int n;
      int *k;

      k = &n;
      n = 7;

      cout <<n<<*k<<endl;
}
```

```
int main(){
      int n;
      int *k;

      k = &n;
      *k =7;

      cout <<n<<*k<<endl;
}
```


ΠΡΟΣΟΧΗ στην ανάθεση με δείκτες

Τα δύο παρακάτω μοιάζουν. Το ένα όμως είναι τραγικά λάθος.
Ποιο και γιατί;

```
int main(){
      int n;
      int *k;

      k = &n;
      *k =7;

      cout <<n<<*k<<endl;
}
```

```
int main(){
      int n;
      int *k;

      *k =7;
      k = &n;

      cout <<n<<*k<<endl;
}
```

Το δεύτερο: αναθέτει την τιμή 7 στην διεύθυνση όπου δείχνει ο δείκτης k, πριν όμως καθοριστεί το πού δείχνει ο k. Άρα κατά την εκτέλεση του `*k =7`, το 7 θα αποθηκευτεί σε κάποια τυχαία θέση στην μνήμη... Πολύ κακό..!!

Ανακεφαλαίωση: Τελεστές δεικτών

- Εισάγαμε δύο νέους τελεστές για δείκτες
  - Τελεστής «*»
    - Κατά την δήλωση του τύπου επισημαίνει ότι είναι δείκτης
    - Μέσα στο πρόγραμμα σημαίνει: «δώσε την τιμή στην διεύθυνση που δείχνει ο δείκτης»
  - Τελεστής «&»
    - «δώσε την διεύθυνση της μεταβλητής»

`*k = m;`	Βάλε την τιμή της `m` εκεί όπου δείχνει ο `k`
`m = *k;`	Δώσε στην `m` την τιμή στην οποία δείχνει ο `k`

`k = &m;`	Δώσε στον `k` την διεύθυνση του `m`

- Προτεραιότητα: Οι τελεστές * και & έχουν την υψηλότερη προτεραιότητα, μετά τους `++`, `--`, `-`
  - εάν δεν είμαστε σίγουροι, χρησιμοποιούμε παρενθέσεις

Παράδειγμα #1: Διάταξη μεταβλητών και δεικτών στην μνήμη (1/3)

```
int main(){
      int n1, n2, n3;
      int *k1, *k2;

      k1 = &n1;
      k2 = &n2;
      n1 = 5;
      *k2 = 10;
      n3 = 15;
      .
      .
}
```

θέση	A1	A2	A3	A4	A5
όνομα	n1	n2	n3	k1	k2
τιμή	5	10	15	A1	A2

Παράδειγμα #1: Διάταξη μεταβλητών και δεικτών στην μνήμη (2/3)

```
int main(){
      .
      .
      *K1 = 1;
      *K2 = 2;
      .
      .
}
```

θέση	A1	A2	A3	A4	A5
όνομα	n1	n2	n3	k1	k2
τιμή	1	2	15	A1	A2

Παράδειγμα #1: Διάταξη μεταβλητών και δεικτών στην μνήμη (3/3)

```
int main(){
      .
      .
      k1 = &n3;
      *K1 = 11;
      .
      .
}
```

θέση	A1	A2	A3	A4	A5
όνομα	n1	n2	n3	k1	k2
τιμή	1	2	11	A3	A2

Παράδειγμα #2: βρείτε τις σωστές και λάθος εκφράσεις

```
#include <iostream>
using namespace std;
int main(){
      double a1, a2;
      double *b1, *b2;
```

			a1,a2
`b1 = &a1;`	σωστή		- -
`b2 = &a2;`	σωστή		- -
`*a1 = 33;`	λάθος:	ο α1 δεν είναι δείκτης	
`a1 = 22;`	σωστή		22 -
`a2 = 20;`	σωστή		22 20
`b2 = *a1;`	λάθος:	το *α1 δεν έχει νόημα	
`b2 = a1;`	λάθος:	ο b2 είναι δείκτης, ο a1 μεταβλητή	
`b2 = 40;`	λάθος:	ο b2 είναι δείκτης, οχι μεταβλητή	
`b2 = b1;`	σωστή		22 20
`*b2 = 40;`	σωστή		40 20
`*b1 = 30;`	σωστή		30 20
`a2 = *b1;`	σωστή		30 30

Δείκτες και πίνακες

- Πολύ στενή σχέση
- Τι είναι πίνακας;
  - Ομαδοποίηση μεταβλητών που καταλαμβάνουν διαδοχικές θέσεις στην μνήμη
- Στην C++ το όνομα του πίνακα (χωρίς τις αγκύλες) είναι ένας δείκτης προς την διεύθυνση του πρώτου στοιχείου

```
int main(){
      int n[10];

      n[0] = 100;
      .
      .
}
```

```
int main(){
      int n[10]; int *k;

      k = n;   (ή k = &n[0];)
      *k = 100;
      .
}
```

Προσπέλαση πινάκων με δείκτες

- Αφού γνωρίζουμε την διεύθυνση του πρώτου στοιχείου του πίνακα, μπορούμε να χρησιμοποιήσουμε μόνο δείκτες για την προσπέλασή του;

- Στην C++ επιτρέπεται η αριθμητική δεικτών
  - το k+1 «δείχνει» στην δεύτερη θέση (4 bytes παρακάτω εαν δείχνει σε ακέραιο), το k+2 στην τρίτη, κοκ

```
int main(){
      int n[10];

      n[0] = 100;
      n[1] = 200;
      n[2] = 300;
      .
}
```

```
int main(){
      int n[10]; int *k;

      k = n;
      *k = 100;
      *(k+1) = 200;
      *(k+2) = 300;
}
```

Δηλαδή, το `k+1 ≡ &n[1]; k+2 ≡ &n[2];` κτλ

Ποιά η διαφορά δεικτών και πινάκων;

- Πολύ μικρή. Στην πράξη, επιτρέπεται να χρησιμοποιούμε αγκύλες και στους δείκτες, σαν να ήταν πίνακες

```
int main(){
      int n[10]; int m, *k;

      k=n;
      n[2] = 300;

      m = n[2];
      m = *(k+2);
      m = k[2];
}
```

ισοδύναμες εκφράσεις

- Η μόνη ουσιαστική διαφορά δείκτη και πίνακα;
  - Στον δείκτη επιτρέπεται να αλλάξει τιμή: π.χ. `k=k+1;`
  - Στον πίνακα δεν επιτρέπεται: το `n=n+1;` είναι λάθος

Παράδειγμα #3: ανάθεση τιμών σε πίνακα χρησιμοποιώντας δείκτες

Γράψτε κώδικα που εκτελεί το παρακάτω χρησιμοποιώντας αποκλειστικά αριθμητική δεικτών

```
int main(){
      int n[100];
      for (int i=0; i<100; ++i) n[i] = 10 * i;
}
```

Λύση

```
int main(){
      int n[100]; int *k;
      k = n;
      for (int i=0; i<100; ++i){
            *k = 10 * i;
            ++k;
      }
}
```