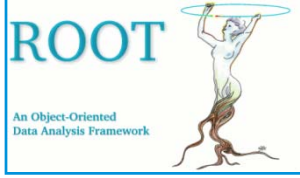
Αναφορές, Δείκτες και Αλφαριθμητικά

- Ο τελεστής αναφοροποίησης
- Αναφορές
- Δείκτες
- Πίνακες και δείκτες
- Ο τελεστής new και delete
- Δυναμικοί πίνακες
- Δείκτες προς συναρτήσεις
- Αλφαριθμητικά της C
- Πίνακες Αλφαριθμητικών
- Ο τύπος string της καθιερωμένης C++
- Αρχεία

Ο τελεστής αναφοροποίησης



- Η δήλωση μιας μεταβλητής συσχετίζει τρία θεμελιώδη χαρακτηριστικά: το *όνομα*, τον *τύπο* και την *διεύθυνση μνήμης*.
- Αναπτύξτε το παράδειγμα και εκτελέστε το μερικές φορές:

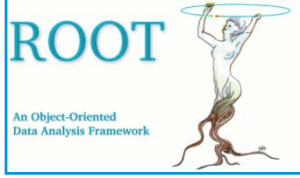
```
#include <iostream>
using namespace std;

int main()
{ // prints a variable's value and its address:
    int n=44;
    cout << "n = " << n << endl;       // prints the value of n
    cout << "&n = " << &n << endl;  // prints the address of n
}
```

- Ο τελεστής & ονομάζεται τελεστής αναφοροποίησης (reference operator) ή τελεστής διεύθυνσης (address operator).
- Εκτυπώνεται η τιμή της μεταβλητής n και η διεύθυνσή της (του πρώτου byte) σε δεκαεξαδική μορφή.
- Κάθε φορά που εκτελούμε το πρόγραμμα η διεύθυνση διαφοροποιείται.

n ⟶

Διευθύνσεις Μνήμης	Περιεχόμενο Μνήμης RAM
bfff98dc	
bfff98e0	44
bfff98e4	
bfff98e8	
.....	

Αναφορές



- Η αναφορά (reference) είναι ένα ψευδώνυμο ή συνώνυμο μιας άλλης μεταβλητής.

  τύπος& όνομα-αναφοράς = όνομα-μεταβλητής

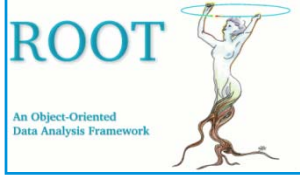
- Παράδειγμα:

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

int main(){
  int n=44;
  int& rn=n;  // rn is a reference of n
  cout << "n=" << n << ",  rn=" << rn << endl;

  n++;
  cout << "n=" << n << ",  rn=" << rn << endl;

  rn=rn*2;
  cout << "n=" << n << ",  rn=" << rn << endl;

  cout << "&n=" << &n << ", &rn=" << &rn << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ anafores.cpp
[panos@pc-247 Cpp]$ a.out
n=44,   rn=44
n=45,   rn=45
n=90,   rn=90
&n=0xbfff8b54, &rn=0xbfff8b54
[panos@pc-247 Cpp]$
```

Τα η και rn είναι διαφορετικά ονόματα της ίδιας μεταβλητής.

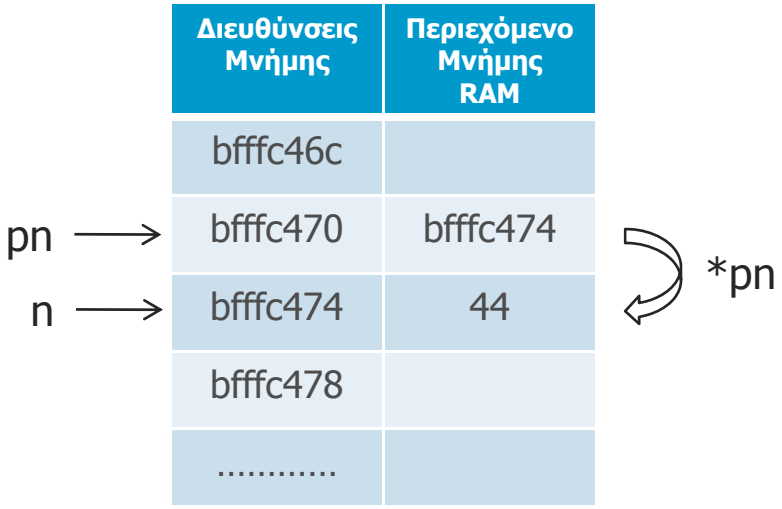
Δείκτες



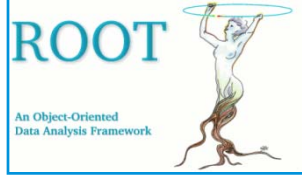
- Ο τύπος μεταβλητής ο οποίος δύναται να αποθηκεύει μια διεύθυνση ονομάζεται δείκτης (pointer).
- Παράδειγμα:

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

int main(){
  int n = 44;     // Orismos metavlitis n
  int* pn=&n;     // Deiktis sti metavliti n

  cout << "  n=" << n << endl;
  cout << " &n=" << &n << endl;
  cout << " pn=" << pn << endl;
  cout << "&pn=" << &pn << endl;
  cout << "*pn=" << *pn << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ diktes.cpp
[panos@pc-247 Cpp]$ a.out
  n=44
 &n=0xbfffc474
 pn=0xbfffc474
&pn=0xbfffc470
*pn=44
[panos@pc-247 Cpp]$
```



Πίνακες και δείκτες



- Με την βοήθεια των δεικτών μπορούμε να χειριστούμε τους πίνακες. Το όνομα του πίνακα είναι στην πραγματικότητα ένας *σταθερός δείκτης* προς το πρώτο στοιχείο. Ορισμοί:

  int a[10];

  int* pa = &a[0] ή int* pa = a;

- Ο χειρισμός των στοιχείων του πίνακα μπορεί να γίνει ως:

  a[0] ⇔ *a ⇔ *pa

  a[1] ⇔ *(a+1) ⇔ *(pa+1)

  a[2] ⇔ *(a+2) ⇔ *(pa+2)

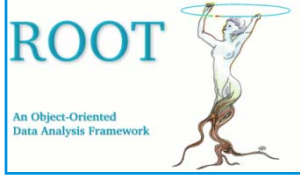
  .....

- Παράδειγμα:

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

int main(){
  int a[5] = {3, 7, 8, 1, 9};     // Orismos pinaka a
  int* pa=a;     // Deiktis ston pinaka a

  for(int i=0; i<5; ++i){
    cout << "a[" << i <<"] = " << a[i] << "  " << *(a+i) << "  " << *(pa+i) << endl;
  }
}
```

```
[panos@pc-247 Cpp]$ c++ diktes_pinakes.cpp
[panos@pc-247 Cpp]$ a.out
a[0] = 3  3  3
a[1] = 7  7  7
a[2] = 8  8  8
a[3] = 1  1  1
a[4] = 9  9  9
[panos@pc-247 Cpp]$
```

Ο τελεστής new και delete



- Για να αποφεύγονται σφάλματα πρέπει να δίνετε αρχική τιμή στους δείκτες διαφορετικά ο δείκτης είναι αιωρούμενος (dangling pointer):

```
float x = 3.14159;    // Ο x περιέχει την τιμή 3.14159
float* p = &x;        // Ο p περιέχει την διεύθυνση του x
cout  << *p;          // Ok: ο *q έχει κατανεμηθεί
```

- Ένας άλλος τρόπος για να κατανέμουμε ρητά μνήμη σε έναν δείκτη είναι μέσω του τελεστή new:

```
float * q;
q = new float;     // κατανέμει μνήμη για 1 float
*q= 3.14159;       // Ok: ο *q έχει κατανεμηθεί
```

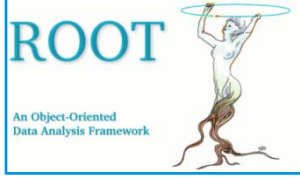
ή όλα σε μία εντολή:

```
float* q = new float(3.14159);
```

- Ο τελεστής delete αντιστρέφει την ενέργεια του τελεστή new, επιστρέφοντας την κατανεμημένη (δεσμευμένη) μνήμη στην ελεύθερη μνήμη. Πρέπει να χρησιμοποιείται μόνο σε δείκτες για τους οποίους έχει ρητά δεσμευθεί μνήμη μέσω του τελεστή new:

```
float* q = new float(3.14159);
delete q;
```

Δυναμικοί πίνακες



- Το όνομα ενός πίνακα είναι απλά ένας σταθερός δείκτης ο οποίος κατανέμεται κατά τον χρόνο μεταγλώττισης.

```
float a[20];     //Στατικός πίνακας
```

Η δήλωση του a ονομάζεται *στατική δέσμευση* (static binding) γιατί η μνήμη του κατανέμεται κατά τον χρόνο μεταγλώττισης και παραμένει κατανεμημένη σε όλη την διάρκεια της εκτέλεσης του προγράμματος είτε ο πίνακας χρησιμοποιείται είτε όχι.

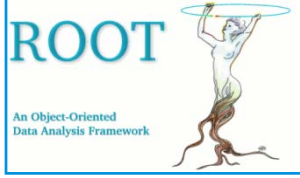
- Αντί του παραπάνω, μπορούμε να χρησιμοποιήσουμε ένα μη σταθερό δείκτη για να καθυστερήσουμε την κατανομή μνήμης μέχρι την εκτέλεση του προγράμματος.

```
float* p = new float[20];     //Δυναμικός πίνακας
```

Αυτό γενικά ονομάζεται *δέσμευση κατά τον χρόνο εκτέλεσης* (run time binding) ή *δυναμική δέσμευση* (dynamic binding). Ο δυναμικός πίνακας p δημιουργείται κατά τον χρόνο εκτέλεσης (η μνήμη κατανέμεται μόνον όταν εκτελεστεί η δήλωσή του). Η μνήμη αυτή αποδεσμεύεται αμέσως όταν εφαρμοστεί σε αυτό ο τελεστή delete:

```
delete [] p;     // Αποδεσμεύει τον πίνακα p
```

Δυναμικοί πίνακες



Το παρακάτω παράδειγμα δείχνει την χρήση των δυναμικών πινάκων

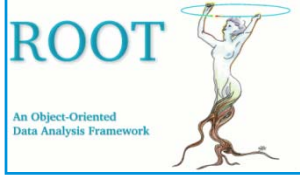
```
//  Χρήση δυναμικών πινάκων
#include <iostream>
using namespace std;
void get(double*&,int&);
void print(double*,int);

int main(){
   double* a;    // a is simply an unallocated pointer
   int n;
   get(a,n);     // now a is an array of n doubles
   print(a,n);
   delete [] a;  // now a is simply an unallocated pointer again
   get(a,n);     // now a is an array of n doubles
   print(a,n);
}

void get(double*& a, int& n){
   cout << "Enter number of items: ";
   cin >> n;
   a = new double[n];
   cout << "Enter " << n << " items, one per line:\n";
   for (int i = 0; i < n; i++) {
      cout << "\t" << i+1 << ": ";
      cin >> a[i];
   }
}
```

```
void print(double* a, int n) {
   for (int i = 0; i < n; i++) {
      cout << a[i] << " ";
   }
   cout << endl;
}
```

!!! Στη συνάρτηση get η παράμετρος πίνακα a είναι ένας δείκτης που μεταβιβάζεται κατ' αναφορά:

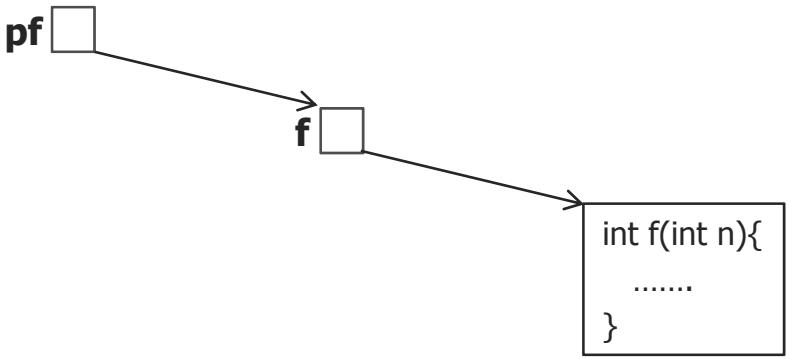
void get(double*& a, int& n)

Δείκτες προς συναρτήσεις



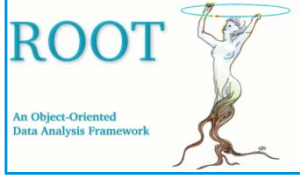
- Όπως ένα όνομα πίνακα, έτσι και ένα όνομα συνάρτησης είναι ένας σταθερός δείκτης. Μπορούμε να θεωρήσουμε ότι η τιμή του είναι η διεύθυνση του κώδικα που υλοποιεί τη συνάρτηση.
- Ένας δείκτης προς συνάρτηση είναι απλώς ένας δείκτης του οποίου η τιμή είναι η διεύθυνση του ονόματος της συνάρτησης.

```
int f(int);     //Δήλωση της συνάρτησης f
int (*pf) (int);   // δηλώνει τον δείκτη pf προς τη συνάρτηση
pf = &f;      //Εκχωρεί τη διεύθυνση της f στον pf
```



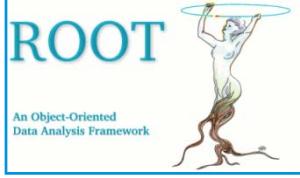
- Η χρησιμότητα των δεικτών προς συναρτήσεις είναι ότι επιτρέπουν να ορίσουμε συναρτήσεις συναρτήσεων. Ακολουθεί παράδειγμα:

Δείκτες προς συναρτήσεις




```
//  Το άθροισμα μιας συνάρτησης
#include <iostream>  // Ορίζει το αντικείμενο cout
using namespace std;
int sum(int (*)(int), int);
int square(int);
int cube(int);

int main(){
   cout << sum(square,4) << endl;    // 1 + 4 + 9 + 16
   cout << sum(cube,4) << endl;      // 1 + 8 + 27 + 64
}

int sum(int (*pf)(int k), int n){
   // returns the sum f(0) + f(1) + f(2) + . . . + f(n-1):
   int s = 0;
   for (int i = 1; i <= n; i++) {
      s += (*pf)(i); }
   return s;
}

int square(int k){  return k*k;}     //Συνάρτηση square

int cube(int k){  return k*k*k;}    //Συνάρτηση cube
```

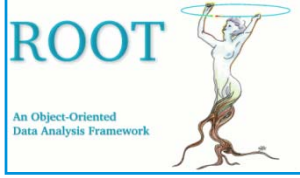
Παράδειγμα



Το ακόλουθο παράδειγμα δείχνει έναν τρόπο για να σαρώσουμε έναν πίνακα χρησιμοποιώντας δείκτες.

```
//  Σάρωση πίνακα με δείκτη
#include <iostream>
using namespace std;

int main(){
   const int SIZE = 3;
   short a[SIZE] = {22, 33, 44};
   cout << "a = " << a << endl;
   cout << "sizeof(short) = " << sizeof(short) << endl;
   short* end = a + SIZE;       // converts SIZE to offset 6
   short sum = 0;
   for (short* p = a; p < end; p++){
      sum += *p;
      cout << "\t p = " << p;
      cout << "\t *p = " << *p;
      cout << "\t sum = " << sum << endl;
   }
   cout << "end = " << end << endl;
}
```

Αλφαριθμητικά της C



- Στη C++ ένα αλφαριθμητικό της C είναι ένας πίνακας χαρακτήρων με τελευταίο χαρακτήρα το NUL, '\0'.
- Ορισμός:

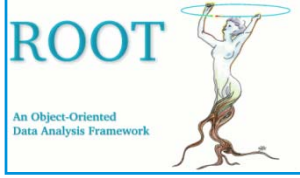
  char str[6]="Hello";
  char str[]="Hello";
  char* str = "Hello";

- Παράδειγμα εισόδου-εξόδου αλφαριθμητικών της C

```
//IO of C strings

#include <iostream>
using namespace std;

int main(){
  char word[80];
  cin >> word;
  while(*word){
    cout << "\t\"" << word << "\"\n";
    cin >> word;
  }
}
```

```
[panos@pc-247 Cpp]$ c++ alpharithmitika.cpp
[panos@pc-247 Cpp]$ a.out
Physics Department University of Ioannina
        "Physics"
        "Department"
        "University"
        "of"
        "Ioannina"

[panos@pc-247 Cpp]$
```

Αλφαριθμητικά της C



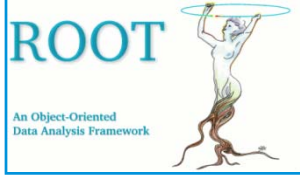
- Τα αντικείμενα του ρεύματος εισόδου-εξόδου cin-cout περιλαμβάνουν τις ακόλουθες συναρτήσεις-μέλη:

Συνάρτηση-μέλος	
cin.getline(str,n)	Εισαγωγή γραμμής κειμένου
cin.get(char)	Εισαγωγή ενός χαρακτήρα
cout.put(char)	Εκτύπωση ενός χαρακτήρα
cin.putback(char)	Επαναφορά στο ρεύμα εισόδου του τελευταίου χαρακτήρα που διαβάστηκε μέσω του cin.get()
cin.ignore(n)	Διάβασμα ενός ή περισσοτέρων χαρακτήρων χωρίς επεξεργασία
cin.peek()	Αντιγράφει έναν χαρακτήρα χωρίς να τον αφαιρέσει από το ρεύμα εισόδου

Παραδείγματα στο βιβλίο σελίδες 197-200.

Περισσότερες καθιερωμένες συναρτήσεις χαρακτήρων της C στις σελ. 200 - 203.

Πίνακες Αλφαριθμητικών



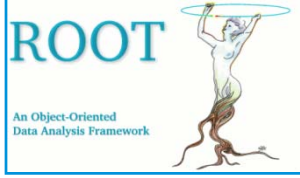
- Ένας δισδιάστατος πίνακας χαρακτήρων είναι στην πραγματικότητα ένας πίνακας αλφαριθμητικών.

  char name[5][20];

```
// Example on alpharithmetic matrix and cin.getline()
// Give Control-D to terminate!!!
#include <iostream>
using namespace std;

int main()
{ char name[5][20];
  int count=0;
  cout << "Enter at most 4 names with at most 19 characters:\n";
  while (cin.getline(name[count++], 20)){
                ;
  }
   --count;
  cout << "The names are:\n";
  for (int i=0; i<count; i++)
    cout << "\t" << i << ". [" << name[i] << "]" << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ alphar_matrix.cpp
[panos@pc-247 Cpp]$ a.out
Enter at most 4 names with at most 19 characters:
John Adams
George Washington
Nikolaos Petrou
The names are:
        0. [John Adams]
        1. [George Washington]
        2. [Nikolaos Petrou]
[panos@pc-247 Cpp]$
```

Ο τύπος string της καθιερωμένης C++



- Η καθιερωμένη C++ ορίζει τον τύπο string στο αρχείο-κεφαλίδα <string>.

```
string s1;                  //το s1 περιέχει 0 χαρακτήρες
string s2="New York";       //το s2 περιέχει 8 χαρακτήρες
string s3(60, '*');         //το s3 περιέχει 60 αστερίσκους
string s4=s3;               //το s4 περιέχει 60 αστερίσκους
string s5(s2, 4, 2);        //το s5 περιέχει τους χαρακτήρες "Yo"
```

- Συνάρτηση getline():

```
string s="ABCDEFG";
getline(cin, s);            // εισάγει στο s όλη τη γραμμή χαρακτήρων
```

- Μήκος του string:

```
cout << s.length() << endl;   // εκτυπώνει τον αριθμό χαρακτήρων του s
```

- Πρόσβαση των χαρακτήρων του string

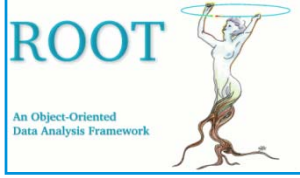
```
char c = s[2];              // εκχωρεί το 'C' στο c
s[4] = '*';                 // αλλάζει το s σε "ABCD*FG"
```

- Σύγκριση των strings:

```
if(s1 < s2)  cout << "Το s1 προηγείται λεξικογραφικά του s2\n";
```

- Περισσότερα σελ. 226 βιβλίου.

Αρχεία

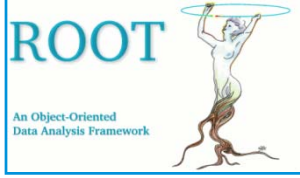


- Στη C++ η διαχείριση της εισόδου-εξόδου από αρχείο γίνεται από τα αντικείμενα ifstream–ofstream αντίστοιχα.
- Ακολουθεί παράδειγμα εγγραφής δεδομένων σε αρχείο.

```
// Example write to file
#include <fstream>   //To read-write to files
#include <iostream>
using namespace std;

int main(){
  ofstream outfile("my_data.txt");  //Create aoutput file my_data.txt

  for(int i=0; i<10; ++i){
    outfile << i << "\n";           // Write to file
  }

}
```

```
[panos@pc-247 Cpp]$ c++ write_file.cpp
[panos@pc-247 Cpp]$ a.out
[panos@pc-247 Cpp]$ more my_data.txt
0
1
2
3
4
5
6
7
8
9
[panos@pc-247 Cpp]$
```

Αρχεία



- Ακολουθεί παράδειγμα ανάγνωσης του αρχείου που δημιουργήθηκε από το προηγούμενο πρόγραμμα.

```
// Example to read from file
#include <fstream>   //To read-write to files
#include <iostream>
using namespace std;

int main(){
  ifstream infile("my_data.txt");  //Link to existing file my_data.txt
  int a;

  cout << "Data from file my_data.txt" << endl;
  for(int i=0; i<10; ++i){
    infile >> a;                   // Read data
    cout << a << endl;
  }

}
```

```
[panos@pc-247 Cpp]$ c++ read_file.cpp
[panos@pc-247 Cpp]$ a.out
Data from file my_data.txt
0
1
2
3
4
5
6
7
8
9
[panos@pc-247 Cpp]$
```