



ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΑΝΟΙΚΤΑ ΑΚΑΔΗΜΑΪΚΑ ΜΑΘΗΜΑΤΑ



ΥΠΟΛΟΓΙΣΤΕΣ II

Συναρτήσεις II

Διδάσκοντες: **Αν. Καθ. Δ. Παπαγεωργίου,**
Αν. Καθ. Ε. Λοιδωρικής



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



ΥΠΟΛΟΓΙΣΤΕΣ ΙΙ

ΣΥΝΑΡΤΗΣΕΙΣ ΙΙ

1

Κλήση συνάρτησης κατ' αξία

- Μέχρι τώρα είδαμε την κλήση συνάρτησης «κατ' αξία»
 - Στην λίστα εισόδου μεταφέρονται μόνο οι τιμές των μεταβλητών

Παράδειγμα: συνάρτηση που υπολογίζει το τετράγωνο αριθμού

```
double square(double x) {
    x = x * x;
    return x;
}
```

- Κατά την κλήση της συνάρτησης, δημιουργείται ένα αντίγραφο του x , το οποίο περνάει μέσα στην συνάρτηση. Όταν η συνάρτηση επιστρέφει στο κυρίως πρόγραμμα, το αντίγραφο καταστρέφεται.

2

Κλήση συνάρτησης κατ' αναφορά

- Εναλλακτικός τρόπος κλήσης συνάρτησης είναι «κατ' αναφορά»
 - Στην λίστα εισόδου μεταφέρονται οι διευθύνσεις των μεταβλητών (δηλαδή δείκτες)

Παράδειγμα: συνάρτηση που μετατρέπει έναν αριθμό στο τετράγωνό του

```
void square(double *x) {
    *x = (*x) * (*x);
}
```

- Κατά την κλήση της συνάρτησης, μεταφέρεται η διεύθυνση του x . Οποιαδήποτε αλλαγή στην τιμή του x παραμένει και όταν η συνάρτηση επιστρέφει στο κυρίως πρόγραμμα.

3

Δήλωση συνάρτησης

- Στην γενική περίπτωση στην λίστα εισόδου έχουμε μεταβλητές και δείκτες

```
τύπος όνομα(τύπος *δείκτης, τύπος μεταβλητή, ...) {
    εντολές;
    return ...;
}
```

- Το πρωτότυπο της συνάρτησης δηλώνεται όπως και πριν

```
τύπος όνομα(τύπος *δείκτης, τύπος μεταβλητή, ...);
```

4

Κλήση συνάρτησης

- Εάν καλούμε μια συνάρτηση από το κυρίως πρόγραμμα η οποία δέχεται δείκτες, τότε πρέπει στην είσοδο να δώσουμε διευθύνσεις

Παράδειγμα: συνάρτηση που μετατρέπει έναν αριθμό στο τετράγωνό του.

```
#include <iostream>
using namespace std;
int main(){

void square(double *x);

int main(){
    double x;
    cout << "δώσε έναν
    ρητό αριθμό" <<endl;
    cin >> x;
```

```
square (&x);
cout <<"στο τετράγωνο
είναι" << x <<endl;
}

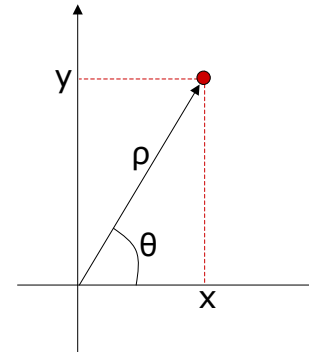
void square(double *x){
    *x = (*x) * (*x);
}
```

5

Παράδειγμα #1: μετατροπή συντεταγμένων

συνάρτηση που μετατρέπει πολικές συντεταγμένες σε καρτεσιανές

- Οι πολικές συντεταγμένες ορίζονται από την ακτίνα ρ και την γωνία θ



Οι καρτεσιανές συντεταγμένες δίνονται συναρτήσει των πολικών:

$$x = \rho \cos(\theta)$$

$$y = \rho \sin(\theta)$$

να γραφτεί η συνάρτηση η οποία να δέχεται δύο δείκτες (για την ακτίνα και την γωνία), και να τα μετατρέπει σε καρτεσιανές

6

Παράδειγμα #1: μετατροπή συντεταγμένων

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

void polar_to_cartesian(double *r, double *t);

int main(){
    double r, t;
    cout << "δώσε πολικές συντεταγμένες" << endl;
    cin >> r >> t;

    polar_to_cartesian(&r, &t);
    cout << "σε καρτεσιανές είναι" << r << t << endl;
}

void polar_to_cartesian(double *r, double *t){
    double x = (*r) * cos(*t);
    double y = (*r) * sin(*t);
    *r = x; *t = y;
}
```

7

Παράδειγμα #2: Ανταλλαγή της τιμής δύο μεταβλητών

Η συνάρτηση να δέχεται δύο ρητούς και να ανταλλάσσει τις τιμές τους

(θα μας χρειαστεί αργότερα για ταξινόμηση)

```
void swap(double *x, double *y){
    double z = *x ;
    *x = *y ;
    *y = z ;
}
```

8

Συνάρτηση που δέχεται πίνακες

- Για να περάσουμε έναν πίνακα σε μια συνάρτηση αρκεί να περάσουμε:
 - την διεύθυνση του πρώτου στοιχείου (δηλαδή δείκτης)
 - το πλήθος των στοιχείων του πίνακα (δηλαδή ακέραιος)
 - εάν είναι πολυδιάστατος, τότε περνάμε τα ανάλογα πλήθη
- Μέσα στην συνάρτηση χρησιμοποιούμε αριθμητική δεικτών για να προσπελάσουμε όλα τα στοιχεία του πίνακα
 - Υπενθυμίζεται ότι το $*(k+i)$ και το $k[i]$ είναι ταυτόσημα
- Για παράδειγμα, για να περάσουμε έναν πίνακα ρητών x με n στοιχεία, η συνάρτηση θα δηλωθεί ως

```
τύπος όνομα (double *x, int n){
    εντολές;
    return ...;
}
```

9

Παράδειγμα #3: Μέγιστο πίνακα

```
double xmax (double *x, int n){
    double m = x[0];
    for (int i = 1; i < n; ++i)
        if(x[i] > m) m = x[i];
    return m;
}
```

και ένα πρόγραμμα που το καλεί

```
#include <iostream>
using namespace std;
double xmax(double *x, int n);
int main(){
    int n; double x[1000];
    cout << "πόσα στοιχεία θα εισάγετε" << endl;
    cin >> n;
    for (int i=0; i<n; ++i) cin >> x[i];
    cout << "το μέγιστο είναι" << xmax(x, n); << endl;
}
```

0

Παράδειγμα #4: μέσος όρος στοιχείων πίνακα

Ορίστε συνάρτηση που γυρνάει τον μέσο όρο των στοιχείων ενός πίνακα

```
double average (double *x, int n){
    if (n == 0) return 0;

    double s = 0 ;
    for (int i = 0 ; i<n ; i++) s += x[i];

    return s/n ;
}
```

11

Παράδειγμα #5: εσωτερικό γινόμενο δύο πινάκων

Ορίστε συνάρτηση που δέχεται δύο πίνακες (και το πλήθος των στοιχείων τους) και επιστρέφει το εσωτερικό τους γινόμενο

έστω πίνακες $x[i]$ και $y[i]$, με $i=0, 1, 2, \dots, n-1$
Το εσωτερικό γινόμενο P :

$$P = \sum_{i=0}^{i < n} x[i] * y[i]$$

```
double array_product (double *x, double *y, int n){
    double p = 0;
    for (int i = 0; i < n; ++i) p += x[i] * y[i];

    return p;
}
```

12

Παράδειγμα #6: σύστημα n σωματιδίων

Έστω σύστημα n σωματιδίων, με τις συντεταγμένες τους αποθηκευμένες σε τρεις πίνακες: x , y , z . Γράψτε συνάρτηση που δέχεται τους τρεις πίνακες και έναν ακέραιο i , έτσι ώστε να υπολογίζει την απόσταση του σωματιδίου i από την αρχή των αξόνων

```
double dist (double *x, double *y, double *z, int i){
    double d = sqrt(x[i]*x[i] + y[i]*y[i] + z[i]*z[i]);
    return d;
}
```

13

Παράδειγμα #7: σύστημα n σωματιδίων

Έστω σύστημα n σωματιδίων, με τις συντεταγμένες τους αποθηκευμένες σε τρεις πίνακες: x , y , z . Γράψτε συνάρτηση που δέχεται τους τρεις πίνακες, το πλήθος των στοιχείων τους n , και έναν ακέραιο i , έτσι ώστε να υπολογίζει την μέση απόσταση του σωματιδίου i από τα άλλα σωματίδια.

```
double dist_ave (double *x, double *y, double *z, int n, int i){

    if (n < 2) return 0;

    double s = 0;
    for (int j = 0; j<n; ++j)
        if (j != i)
            s += sqrt( pow(x[i]-x[j], 2) +
                pow(y[i]-y[j], 2) + pow(z[i]-z[j], 2) );

    return s/(n-1);
}
```

Τέλος Ενότητας



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Ιωαννίνων**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «**Εκπαίδευση και Δια Βίου Μάθηση**» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
Περίοδος προγραμματισμού 2007-2013

ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



ΕΣΠΑ
2007-2013
Μίσθια για το μέλλον
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Έχουν προηγηθεί οι κάτωθι εκδόσεις:

- Έκδοση 1.0 διαθέσιμη εδώ.

<http://ecourse.uoi.gr/course/view.php?id=1227>.

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Ιωαννίνων, Διδάσκοντες: Αν.
Καθ. Δ. Παπαγεωργίου, Αν. Καθ. Ε. Λοιδωρίκης.
«ΥΠΟΛΟΓΙΣΤΕΣ ΙΙ. Συναρτήσεις ΙΙ». Έκδοση: 1.0.
Ιωάννινα 2014. Διαθέσιμο από τη δικτυακή
διεύθυνση:
<http://ecourse.uoi.gr/course/view.php?id=1227>.

Σημείωμα Αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά Δημιουργού - Παρόμοια Διανομή, Διεθνής Έκδοση 4.0 [1] ή μεταγενέστερη.



- [1] <https://creativecommons.org/licenses/by-sa/4.0/>.