Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

5α

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #5

Παρασκευή, 2 Απριλίου 2021

Παναγιώτης Παύλου

c-programming@allos.gr

Έλεγχος ροής εντός βρόχου

Έλεγχος με τις break και continue – Ατέρμονες βρόχοι

break & continue

Οι εντολές break και continue μας επιτρέπουν να ελέγχουμε την εκτέλεση ενός βρόχου, από το σώμα του, ανεξάρτητα από τη συνθήκη του. Δείτε πως συντάσσονται και πως λειτουργούν:

```
for (int i=0; i<10; i++) {
    if (i==5) {
        break;
    }
    printf("%d ", i);
}
```

0 1 2 3 4 5 6 7 8 9

```
for (int i=0; i<10; i++) {
    if (i==5) {
        continue;
    }
    printf("%d ", i);
}
```

0 1 2 3 4 5 **6 7 8 9**

Προφανώς αυτές, εντός των βρόχων, θα τις συναντάμε πάντα μέσα σε κάποιο if. Επίσης σε εμφωλευμένους βρόχους, το break και το continue αφορά το loop στο οποίο γράφεται.

Ατέρμονες βρόχοι

Τώρα που υπάρχει η δυνατότητα να σταματά η εκτέλεση ενός βρόχου μέσα από το σώμα του, έχει πλέον νόημα η χρήση ενός βρόχου με συνθήκη που είναι πάντα αληθής. Δηλαδή ενός ατέρμονος βρόχου.

Οπότε πλέον η εκτέλεσή του μπορεί να τερματιστεί από διαφορετικές συνθήκες και σε διαφορετικά σημεία.

```
while (true) {
    ...
    if (...) {
        break;
    }
    ...
    if (...) {
        break;
    }
    ...
}
```

Εφαρμογή – Μενού

```
int sum=0;
printf("Kalws hr8ate sto menu mas.\nKante thn prwth sas epilogh:\n");
while(true) {
    printf(" 1. Random int\n");
    printf(" 2. Sum of ints\n");
    printf(" 3. Show options\n");
    printf(" 4. Exit\n");
    int item = smReadInt("");
    if( item == 1) {
        int r = rand();
        sum += r;
        printf("New num: %d\n", r);
    } else if (item == 2) {
        printf("Sum so far: %d\n", sum);
    } else if (item == 3) {
        printf("Deite 3ana tis epiloges sas:\n");
        continue;
    } else if (item == 4) {
        printf("Eyxaristoume gia th synergasia.\n");
        break;
    }
    printf("Kante mia nea epilogh\n");
}
```

Σε κάθε επανάληψη

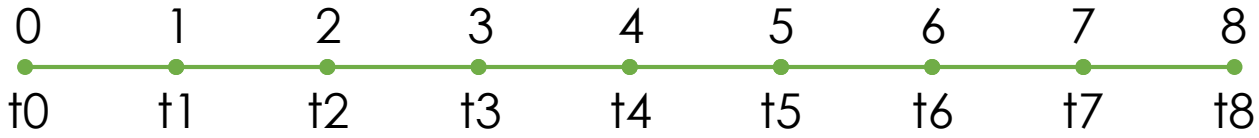
- Αρχικά τυπώνεται το μήνυμα
- Διαβάζεται η επιλογή του χρήστη από το πληκτρολόγιο
- Εάν είναι 1 ή 2 εκτελείται η λογική του μενού
- Εάν είναι 3, τότε αλλάζει η κεφαλίδα του μηνύματος και με την continue εκτυπώνεται το υπόλοιπο μενού
- Εάν είναι 4, τότε τερματίζεται ο βρόχος, δηλαδή το μενού
- Στο τέλος της κάθε επανάληψης εκτυπώνεται η κανονική κεφαλίδα του μενού

Πίνακες

Πίνακες δεδομένων μίας και περισσοτέρων διαστάσεων

Μονοδιάστατοι πίνακες

Αν υποθέσουμε ότι θέλουμε – με όσα γνωρίζουμε μέχρι τώρα – να αποθηκεύσουμε τις θερμοκρασίες ενός ψυγείου μήκους 2μ ανά 25εκ.

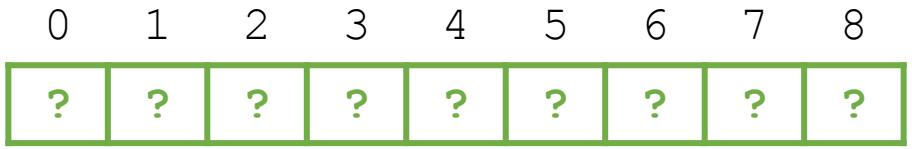


Θα χρειαστούν 9 μετρήσεις γι'αυτό θα πρέπει να χρησιμοποιήσουμε 9 μεταβλητές, τις t0, t1, t2, …, t7, t8 οπότε κάθε τι που θα γράφουμε θα πρέπει να τις αναφέρει ρητά π.χ.

```
double average = (t0 + t1 + t2 + t3 + t4 + t5 + t6 + t7 + t8)/9;
```

αφού δεν μπορούμε να δημιουργήσουμε παραμετρικές παραστάσεις. Λύση δίνουν οι πίνακες!

Οι **πίνακες (arrays)** είναι μία ομάδα μεταβλητών του **ίδιου τύπου δεδομένων**, κάτω από **κοινό όνομα** και κάθε μία από αυτές αναφέρεται μέσω ενός **δείκτη (index)**. Κάθε μία από τις μεταβλητές ονομάζεται **στοιχείο** του πίνακα και είναι αριθμημένη βάσει του δείκτη.



Η αρίθμηση **ξεκινά από το 0 μέχρι το Ν-1** (όπου Ν το πλήθος των στοιχείων του πίνακα).

Μονοδιάστατοι πίνακες

Το Ν το αποκαλούμε και **μέγεθος** ή **μήκος** του πίνακα. Το μήκος του πίνακα είναι προκαθορισμένο και ορίζεται κατά τη δήλωση του πίνακα, ενώ στη συνέχεια δεν μπορεί να μεταβληθεί.

Η δήλωση ενός πίνακα π.χ. ακεραίων (int) με όνομα x με μήκος Ν γράφεται ως ακολούθως:

```
int x[N];
```

και το στοιχείο του πίνακα x με δείκτη i γράφεται ως:

```
x[i]
```

και λειτουργεί όπως οποιαδήποτε μεταβλητή. Αυτό σημαίνει ότι οι τιμές των στοιχείων είναι ακαθόριστες κατά τη δήλωσή του πίνακα και πρέπει να αρχικοποιηθούν.

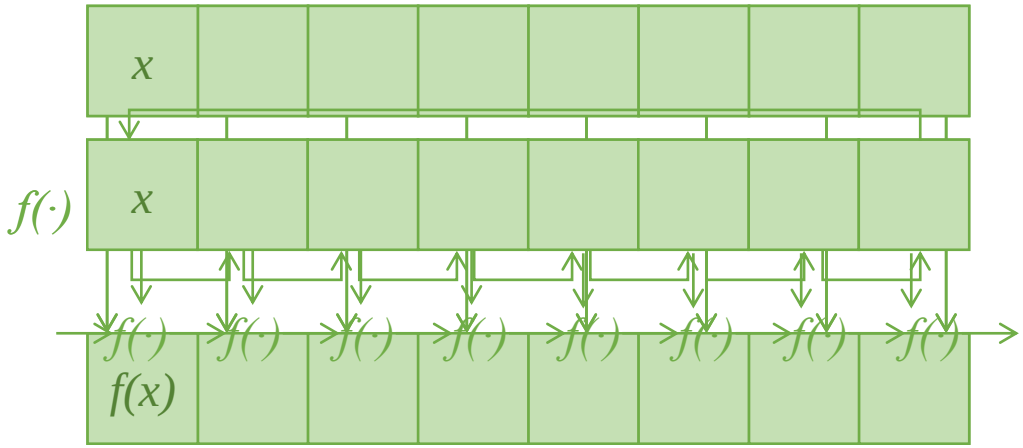
Η αρχικοποίηση του πίνακα μπορεί να γίνει κατά τη δήλωσή του με τον ακόλουθο τρόπο.

```
int x[N] = { 1,12,58,44,20,-3,0 };
```

Εάν τα στοιχεία είναι λιγότερα από το μέγεθος του πίνακα, τότε οι τιμές αποδίδονται στα πρώτα στοιχεία του και τα υπόλοιπα τα θεωρούμε μη αρχικοποιημένα. Εάν τα στοιχεία μέσα στα άγκιστρα είναι όλα τα στοιχεία του πίνακα, τότε το μέγεθος του πίνακα μπορεί να παραληφθεί.

Μονοδιάστατοι πίνακες

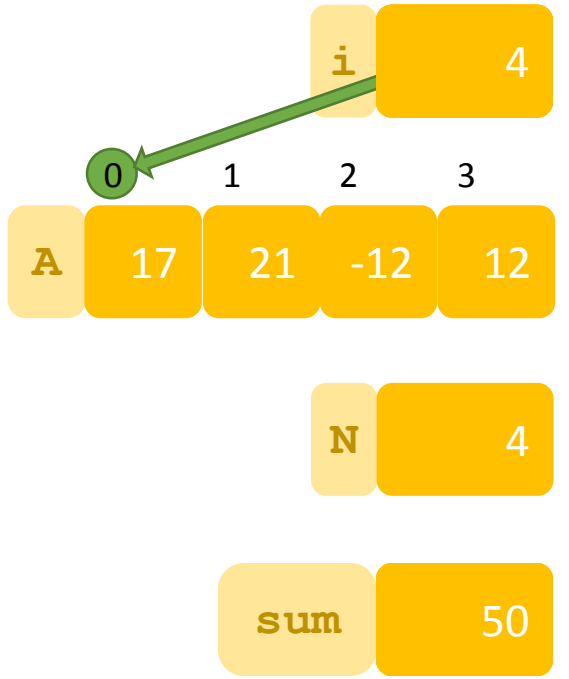
- **Ενδεικτικά** οι πίνακες μπορούν να χρησιμοποιηθούν για να παραστήσουν :
  - Διανύσματα
  - Ακολουθίες (π.χ. χωρικές ή χρονικές μεταβολές)
  - Σύνολα
- Η χρήση των πινάκων είναι ευρύτατη και είναι η βασικότερη δομή δεδομένων πέρα από τις μεταβλητές, η οποία υποστηρίζεται και από τον ίδιο τον επεξεργαστή.
- Σχήματα επεξεργασίας των πινάκων :
  - Filter – Φιλτράρισμα
  - Map – Αντιστοίχιση
  - Reduce – Αναγωγή
  - Rotation – Περιστροφή



Εφαρμογή – Άθροισμα θετικών

Ο παρακάτω κώδικας υπολογίζει το άθροισμα των θετικών στοιχείων του πίνακα A με μήκος N.

```
int sumPositives(int A[], int N) {
    int sum = 0;
    for (int i=0; i<N; i++) {
        if (A[i] > 0) {
            sum += A[i];
        }
    }
    return sum;
}
```



Συναρτήσεις που επιστρέφουν 1D πίνακες

Κάποιες φορές χρειάζεται να δημιουργούμε συναρτήσεις που επιστρέφουν πίνακες ως αποτέλεσμα. Επειδή ακόμα όμως δεν έχουμε όλη την τεχνική γνώση, εάν θέλουμε κάτι τέτοιο θα πρέπει να χρησιμοποιήσουμε από την smLib τους δύο εικονικούς τύπους δεδομένων:

`smIntArray` και `smDoubleArray`

για πίνακες `int` και `double` αντίστοιχα.

Για να δημιουργήσουμε τέτοιους πίνακες μπορούμε να χρησιμοποιούμε αντίστοιχα τις:

`smNewIntArray` και `smNewDoubleArray`

οι οποίες και οι δύο παίρνουν ένα όρισμα, το μήκος του πίνακα.

Το αποτέλεσμά τους είναι είτε ο ζητούμενος πίνακας, είτε η ειδική τιμή `smFailure` που υποδεικνύει ότι απέτυχε.

Η συνάρτηση επιστρέφει έναν πίνακα που τα στοιχεία του έχουν τιμές από 1 έως και N:

```
smIntArray createRangeToN(int N) {
    smIntArray A = smNewIntArray(N);
    if (A == smFailure) {
        return smFailure;
    }
    for (int i=0; i<N; i++) {
        A[i] = i+1;
    }
    return A;
}
```

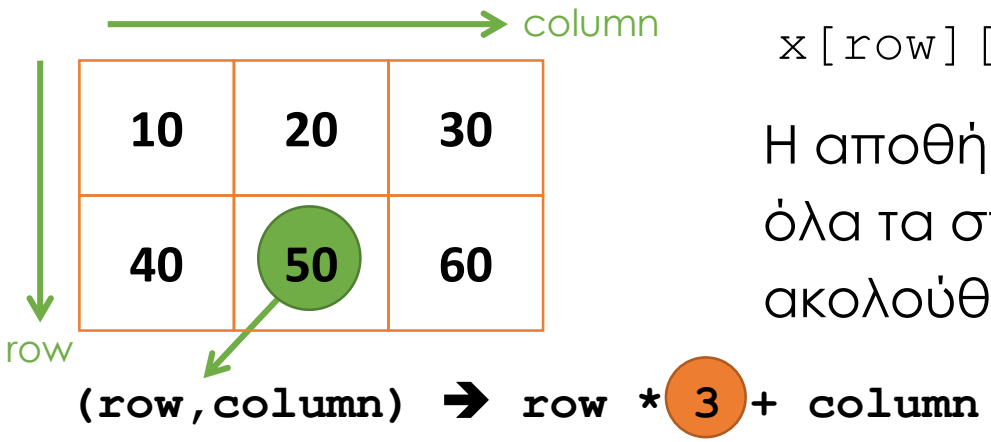
Πολυδιάστατοι πίνακες

Αφού κάθε στοιχείο ενός πίνακα μπορεί να είναι οποιουδήποτε τύπου δεδομένων, τότε δεν μας απαγορεύει τίποτα να δημιουργήσουμε έναν πίνακα από πίνακες. Αυτό ισοδυναμεί με τη δημιουργία ενός δισδιάστατου πίνακα.

Η δήλωσή του γίνεται παρόμοια με τους απλούς μονοδιάστατους πίνακες ως εξής:

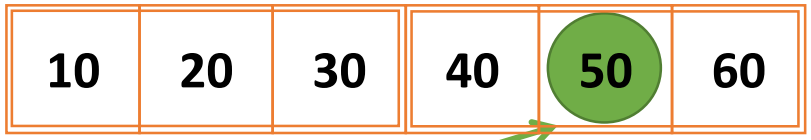
```
int x[2][3];
```

Που αντιστοιχεί σε έναν πίνακα 2 γραμμών και 3 στηλών, ενώ η αναφορά στο στοιχείο του στη γραμμή `row` και στη στήλη `column` γίνεται ως:




```
x[row][column]
```

Η αποθήκευσή του στη μνήμη είναι συνεχόμενη για όλα τα στοιχεία του και γίνεται όπως φαίνεται ακολούθως.



Πολυδιάστατοι πίνακες

Τα στοιχεία του δισδιάστατου αρχικοποιουνται ως εξής:

```
int x[2][4] = { { 1,2,3,4 } , { 5,6,7,8 } };
```

Εδώ δεν μας επιτρέπεται να παραλείψουμε τις διαστάσεις του πίνακα όταν δίνονται όλα τα στοιχεία του.

Αντίστοιχα με την ίδια λογική μπορούμε να ορίσουμε και πίνακες περισσότερων διαστάσεων. Π.χ.

```
int y[10][10][10];
```

Εφαρμογή printArray 2D

```
void print2DArray(int R, int C, int A[R][C]) {
    for (int r = 0; r < R; ++r) {
        for (int c = 0; c < C; ++c) {
            printf("%d ", A[r][c]);
        }
        printf("\n");
    }
}
```

Εφαρμογές πινάκων

Αναζήτηση και ταξινόμηση

Αναζήτηση σε αταξινόμητο πίνακα

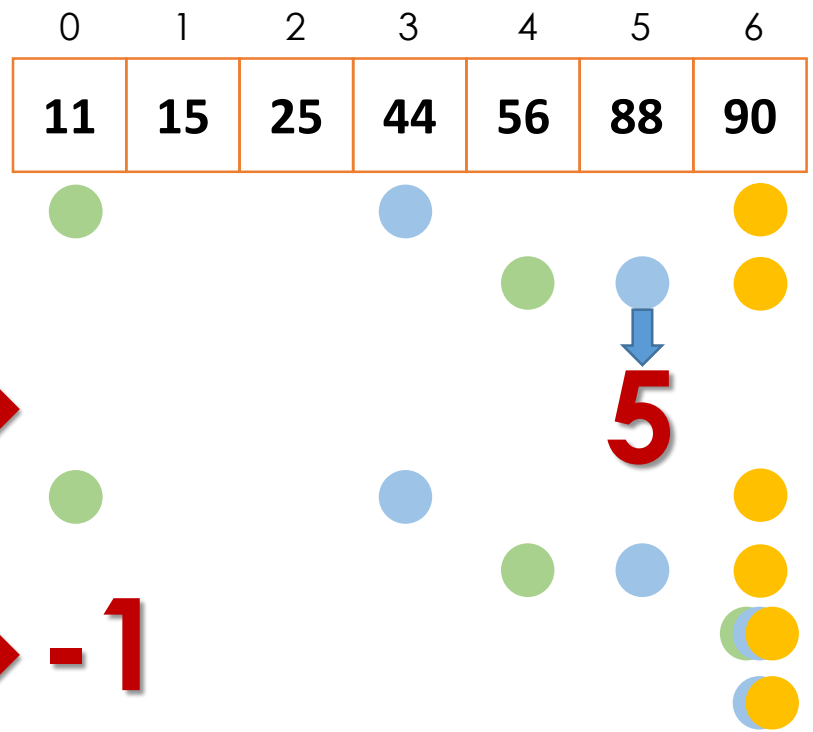
Η αναζήτηση τιμής σε αταξινόμητο πίνακα είναι αρκετά απλή, όμως δεν είναι αποδοτική. Ας δούμε τον κώδικα:

```
int findInArray(int value, int a[], int N) {
    for (int i = 0; i < N; ++i) {
        if (a[i] == value) {
            return i;
        }
    }
    return -1;
}
```

Αναζήτηση σε ταξινομημένο πίνακα

Η αναζήτηση σε ταξινομημένο πίνακα όμως δίνει περισσότερες δυνατότητες στο να γραφτεί πιο αποτελεσματικός κώδικας. Η συνάρτηση φυσικά καλείται με τις ίδιες παραμέτρους, αλλά (προ)υποθέτει ότι τα στοιχεία του πίνακα είναι ταξινομημένα σε αύξουσα σειρά.

```
int findInSortedArray(int value, int a[], int N) {
    int from = 0, to = N-1;
    do {
        int middle = (to + from)/2;
        int Am = a[middle];
        if (value < Am) {
            to = middle-1;
        } else if (Am < value) {
            from = middle+1;
        } else {
            return middle;
        }
    } while (to >= from) ;
    return -1;
}
```



88? ➔

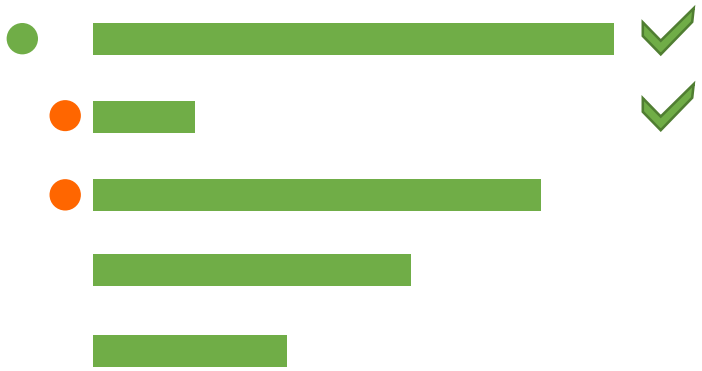
95? ➔ -1

Ταξινόμηση μονοδιάστατου πίνακα

Υπάρχουν αρκετοί αλγόριθμοι ταξινόμησης και μπορείτε να δείτε μερικούς εν δράσει στη σελίδα αυτή.

Εδώ θα δούμε τον απλούστερο, που ονομάζεται selection sort.

```
void selectionSort(int a[], int N) {
    for (int i = 0; i < N; ++i) {
        int indexOfMin = i;
        int minValue = a[i];
        for (int j = i+1; j < N; ++j) {
            if (a[j] < minValue) {
                indexOfMin = j;
                minValue = a[j];
            }
        }
        if (indexOfMin > i) {
            int Ai = a[i];
            a[i] = a[indexOfMin];
            a[indexOfMin] = Ai;
        }
    }
}
```



Σημαντικά σημεία

Μετά από τη σημερινή διάλεξη θα πρέπει να γνωρίζετε:

- Τις εντολές break και continue με τις οποίες μπορείτε να ελέγξετε τη ροή εκτέλεσης εντός των βρόχων
- Τους ατέρμονες βρόχους
- Τους μονοδιάστατους πίνακες
- Τους πολυδιάστατους πίνακες
- Εφαρμογές των πινάκων:
  - Αναζήτηση
  - Ταξινόμηση

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!