Προγραμματισμός σε C - Διανύσματα

Ιωάννης Γ. Τσούλος

Τμήμα Πληροφορικής και τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων

2023

Περίληψη

Ορισμός

- Τα διανύσματα είναι συνεχόμενες μεταβλητές ενός τύπου.
- Το πλήθος των στοιχείων είναι δεδομένο.
- Το πρώτο στοιχείο είναι **πάντα** στην θέση 0.
- Το μέγεθος ενός πίνακα δεν μπορεί να αλλάξει
- Δήλωση: type arrayName[SIZE];

Παράδειγμα χρήσης

```
#include <stdio.h>
int main()
{
        int data[5];
        int i;
        data[0]=100;data[1]=200;data[2]=5;
        data[3]=9;data[4]=2;
        for(i=0;i<5;i++)
           printf("data␣=%d\n",data[i]);
        return 0;
}
```

Παράδειγμα χρήσης πίνακα

- Ο πίνακας data είναι πίνακας με ακέραιες τιμές
- Κάθε στοιχείο του πίνακα αποτελεί και ανεξάρτητη μεταβλητή
- Το πρώτο στοιχείο του πίνακα έχει την τιμή 100
- Το τελευταίο στοιχείο του πίνακα είναι στην θέση 4 και **δεν είναι** στην θέση 5.

Αρχικοποίηση πίνακα με ανάθεση

```
# include <stdio.h>
int main()
{
        int data[5]={100,200,5,9,2};
        int i;
        for(i=0;i<5;i++)
                printf("data␣=%d\n",data[i]);
        return 0;
}
```

Αρχικοποίηση πίνακα

- Η αρχικοποίηση πίνακα μπορεί να γίνει στοιχείο προς στοιχείο
- Πολλές φορές διευκολύνει να γίνει αρχικοποίηση των στοιχείων κατά την δήλωση
- Δεν επιτρέπεται όμως η ανάθεση στοιχείων κατά την εκτέλεση του προγράμματος (οι πίνακες δεν είναι μία μεταβλητή).

Αρχικοποίηση με ανάγνωση

```
#include <stdio.h>
int main()
{
        int data[5];
        int i;
        for(i=0;i<5;i++)
        {
                printf("Dose␣stoixeio\n");
                scanf("%d",&data[i]);
        }
        for(i=0;i<5;i++)
                printf("data␣=%d\n",data[i]);
        return 0;
}
```

Ανάγνωση στοιχείων πίνακα

- Κάθε στοιχείο πίνακα θεωρείται ανεξάρτητη μεταβλητή
- Για την ανάγνωση ενός στοιχείου πίνακα ακολουθούνται οι ίδιοι κανόνες όπως και με την ανάγνωση απλών μεταβλητών
- Δεν επιτρέπεται η ανάγνωση ενός πίνακα με μια εντολή, πχ η εντολή scanf("%d",&data) είναι λανθασμένη.

Χρήση πινάκων σαν ορίσματα

- Οι πίνακες μπορούν να χρησιμοποιηθούν σαν ορίσματα σε συνάρτηση.
- Είναι καλή πρακτική σε μια συνάρτηση πέρα από τον πίνακα να δίνεται σαν όρισμα και το πλήθος των στοιχείων του πίνακα.

Εμφάνιση στοιχείων από συνάρτηση

```
#include <stdio.h>
void printVector(int x[],int n)
{
        int i;
        for(i=0;i<n;i++)
                printf("x=%d\n",x[i]);
}
int main()
{
        int data[5]={10,20,30,40,50};
        printVector(data,5);
        return 0;
}
```

Εμφάνιση στοιχείων σε συνάρτηση

1. Στην συνάρτηση printVector περάσαμε και το πλήθος των στοιχείων του πίνακα.
2. Η συνάρτηση μπορεί να χρησιμοποιηθεί για οποιοδήποτε πίνακα ακεραίων στοιχείων.
3. Αντί για την δήλωση int x[] σε πολλές περιπτώσεις χρησιμοποιείται η ισοδύναμη int *x.

Ανάγνωση στοιχείων πίνακα από συνάρτηση

```
void readVector(int x[],int n)
{
        int i;
        for(i=0;i<n;i++)
        {
                printf("Dose stoixeio\n");
                scanf("%d",&x[i]);
        }
}
```

Επιστροφή τιμής

- Μια συνάρτηση μπορεί να επιστρέφει και τιμές με μετρήσεις πάνω σε πίνακες.
- Παράδειγμα: μια συνάρτηση που να επιστρέφει το πλήθος των στοιχείων ενός πίνακα που είναι ζυγοί αριθμοί.

Καταμέτρηση ζυγών στοιχείων σε πίνακα

```
# include <stdio.h>
int countEven(int x[],int n){
        int count=0;
        int i;
        for(i=0;i<n;i++){
                if(x[i]%2==0) count++;
        }
        return count;
}
int main(){
        int data[5]={1,2,3,4,5},k;
        k=countEven(data,5);
        printf("k = %d \n",k);
        return 0;
}
```

Αναζητήσεις

- Σε πολλές εφαρμογές απαιτείται η αναζήτηση αριθμών ή του πλήθους τους.
- Σε άλλες περιπτώσεις απαιτείται ο υπολογισμός του μεγίστου ή του ελαχίστου ενός πίνακα

Εύρεση της πρώτης εμφάνισης στοιχείου σε πίνακα

```
int searchFirst(int value,int x[],int n)
{
        int i,pos=-1;
        for(i=0;i<n;i++)
        {
                if(x[i]==value)
                {
                        pos=i;
                        break;
                }
        }
        return pos;
}
```

Εύρεση στοιχείου σε πίνακα

- Η μεταβλητή pos αρχικοποιείται σε μια άκυρη θέση που δεν υπάρχει στον πίνακα (πχ -1).
- Αν το στοιχείο δεν βρεθεί θα επιστραφεί η τιμή -1.
- Με την εντολή break η αναζήτηση θα σταματήσει στην πρώτη εμφάνιση του στοιχείου στον πίνακα.
- Αν αφαιρεθεί η εντολή break, τότε θα γίνει αναζήτηση για την τελευταία εμφάνιση του στοιχείου value στον πίνακα.

Εύρεση μεγίστου στοιχείου σε πίνακα

```
int findMax(int x[],int n)
{
        int max= x[0];
        int i;
        for(i=0;i<n;i++)
        {
                if(x[i]>max) max=x[i];
        }
        return max;
}
```

Εύρεση μεγίστου

- Κατά την αναζήτηση πάντα γίνεται αρχικοποίηση του μέγιστου/ελαχίστου σε υπαρκτό στοιχείο στον πίνακα.
- Σε κάθε επανάληψη γίνεται ενημέρωση της αρχικής υπόθεσης
- Αν αλλάξει ο τελεστής της ανισότητας σε $<$, τότε γίνεται αναζήτηση για το ελάχιστο στοιχείο στον πίνακα.
- Στις περισσότερες εφαρμογές μεγαλύτερη σημασία δίνεται στην εύρεση της θέσης του μεγίστου, παρά στο ίδιο το μέγιστο

Ταυτόχρονη εύρεση ελαχίστου και μεγίστου

```
void findMaxAndMin(int x[],int n,int values[])
{
        int i;
        values[0]=x[0];
        values[1]=x[1];
        for(i=0;i<n;i++)
        {
                if(x[i]<values[0])
                        values[0]=x[i];
                if(x[i]>values[1])
                        values[1]=x[i];
        }
}
```

Ταυτόχρονη αναζήτηση ελαχίστου και μεγίστου

- Χρησιμοποιήθηκε πίνακας δύο στοιχείων για την αποθήκευση των τιμών.
- Στην πρώτη θέση του πίνακα είναι η ελάχιστη τιμή και στην δεύτερη θέση η μέγιστη τιμή.
- Η έμμεση επιστροφή πίνακα από συνάρτηση είναι μια αποδοτική τεχνική για την επιστροφή πολλαπλών τιμών.

Πράξεις

1. Αντικατάσταση στοιχείων σε διάνυσμα
2. Πρόσθεση διανυσμάτων
3. Γινόμενο διανυσμάτων
4. Ταξινόμηση διανυσμάτων
5. Ολίσθηση διανυσμάτων
6. Αντιστροφή διανυσμάτων

Αντικατάσταση αρνητικών τιμών

```
int replaceNegative(int x[],int n)
{
        int count=0,i;
        for(i=0;i<n;i++)
        {
                if(x[i]<0) x[i]=0;
                count++;
        }
        return count;
}
```

Αντικατάσταση στοιχείων πίνακα

- Οι συναρτήσεις που δέχονται σαν όρισμα πίνακες μπορούν να αλλάξουν τα στοιχεία των πινάκων.
- Η συνάρτηση replaceNegative() αλλάζει τα αρνητικά στοιχεία σε 0 και επιστρέφει το πλήθος των αλλαγών που έκανε.
- Συναρτήσεις σαν την replaceNegative() ονομάζονται **φίλτρα**.

Πρόσθεση διανυσμάτων

- Για να γίνει πρόσθεση διανυσμάτων, τα διανύσματα πρέπει να έχουν το ίδιο πλήθος στοιχείων.
- ```
  void addVectors(int x[],int y[],int z[],int n)
  {
          int i;
          for(i=0;i<n;i++)
          {
                  z[i]=x[i]+y[i];
          }
  }
  ```
  ```

Εσωτερικό γινόμενο

- Έστω δύο πίνακες $\overrightarrow{x}, \overrightarrow{y}$
- Εσωτερικό γινόμενο $\text{prod}\left(\overrightarrow{x}, \overrightarrow{y}\right) = x^T y$
- Είναι αριθμός και όχι διάνυσμα

Υλοποίηση εσωτερικού γινομένου

```
int prodVectors(int x[],int y[],int n)
{
        int i,sum=0;
        for(i=0;i<n;i++)
        {
                sum=sum+x[i]*y[i];
        }
        return sum;
}
```

Ταξινόμηση διανυσμάτων

- Αύξουσα σειρά: τα στοιχεία του πίνακα διατάσσονται από το μικρότερο προς το μεγαλύτερο.
- Φθίνουσα σειρά: τα στοιχεία του πίνακα διατάσσονται από το μεγαλύτερο προς το μικρότερο.
- Απαιτείται να γίνουν ανταλλαγές στοιχείων, προκειμένου να ταξινομηθεί ένας πίνακας.

Υλοποίηση ταξινόμησης διανυσμάτων

```
void sortVector(int x[],int n)
{
        int i,j;
        for(i=0;i<n;i++)
        {
                for(j=0;j<n-1;j++){
                        if(x[j]>x[j+1]){
                                int t=x[j];
                                x[j]=x[j+1];
                                x[j+1]=t;
                        }
                }
        }
}
```

Ταξινόμηση διανυσμάτων

- Χρειάζεται διπλή επανάληψη, καθώς με την πρώτη σειρά ανταλλαγών θα γίνουν ανταλλαγές μόνο γειτονικών στοιχείων
- Γίνεται ταξινόμηση σε αύξουσα σειρά
- Αν αντιστραφεί ο τελεστής της ανισότητας, τότε γίνεται ταξινόμηση σε φθίνουσα σειρά
- Αν ο πίνακας έχει διάσταση N, τότε θα γίνουν N^2 ανταλλαγές.

Ολίσθηση διανυσμάτων

- Αριστερή ολίσθηση: τα στοιχεία μετακινούνται από το μικρότερο προς το μεγαλύτερο.
- Δεξιά ολίσθηση: τα στοιχεία μετακινούνται από το μεγαλύτερο προς το μικρότερο.
- Επανατοποθέτηση: τα στοιχεία που χάνονται στην μετακίνηση τοποθετούνται στις θέσεις που αδειάζουν, πχ στην αριστερή ολίσθηση το τελευταίο στοιχείο που θα χαθεί μπορεί να μπει στην αρχή του διανύσματος.

Υλοποίηση ολίσθησης διανυσμάτων

```
void shiftLeft(int x[], int n)
{
        int last = x[n-1];
        int i;
        for(i=n-1;i>=1;i--)
        {
                x[i]=x[i-1];
        }
        x[0]=last;
}
```

Αντιστροφή διανυσμάτων

- Είναι η διαδικασία με την οποία τα στοιχεία ενός πίνακα αντιστρέφονται
- Το πρώτο στοιχείο πάει στην τελευταία θέση και αυτό που είναι στην τελευταία στη πρώτη κοκ
- Χρειάζεται να γίνουν ανταλλαγές στοιχείων

Υλοποίηση αντιστροφής διανυσμάτων

```
void reverseVector(int x[], int n)
{
        int i,t;
        for(i=0;i<n/2;i++)
        {
                int t=x[i];
                x[i]=x[n-i-1];
                x[n-i-1]=t;
        }
}
```

Σύνοψη

- Αρχικοποίηση διανυσμάτων.
- Συναρτήσεις και διανύσματα.
- Αναζήτηση στοιχείων σε διανύσματα.
- Πράξεις πινάκων.