Προγραμματισμός Ι (ΗΥ120)

Διάλεξη 12:
Συναρτήσεις & Δείκτες

Χρήστος Δ. Αντωνόπουλος
12/12/2018

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Αλλαγή του «εξωτερικού» περιβάλλοντος

- Αν σαν παράμετρος μιας συνάρτησης δοθεί μια μεταβλητή, σαν **πραγματική** παράμετρος θα περαστεί η **τιμή** της.
- Το πέρασμα παραμέτρων είναι **καθ' αποτίμηση**.
- Οι πραγματικές παράμετροι (τιμές) αποθηκεύονται σε τοπική μνήμη – προσωρινές μεταβλητές με τα ονόματα των αντίστοιχων τυπικών παραμέτρων.
- Αν ο κώδικας της συνάρτησης αλλάξει την τιμή μιας τυπικής παραμέτρου, στην πραγματικότητα αλλάζει την τιμή της αντίστοιχης τοπικής μεταβλητής, **όχι** της μεταβλητής που «περάστηκε» σαν παράμετρος.


```
#include <stdio.h>

void inc0(int a) {
  a=a+1;
}

int main(int argc, char *argv[]) {
  int a;

  a=5;

  inc0(a);
  printf("a=%d\n", a);

  inc0(a);
  printf("a=%d\n", a);

  return(0);
}
```

οποιαδήποτε αλλαγή στην `b` δεν επηρεάζει το «εξωτερικό» περιβάλλον της κλήσης

σαν **πραγματική** παράμετρος (για την τυπική παράμετρο `a`) περνιέται η **τιμή** `5`

Δείκτες ως παράμετροι συναρτήσεων

- Αν σαν παράμετρος μιας συνάρτησης δοθεί μια **διεύθυνση**, τότε η συνάρτηση **μπορεί** (προφανώς) να **αλλάξει** τα **περιεχόμενα** σε αυτή την θέση μνήμης.
- Η αντίστοιχη τυπική παράμετρος πρέπει να δηλωθεί ως **δείκτης-σε-Τ**, και ο κώδικας της συνάρτησης πρέπει να χρησιμοποιεί αυτή την τιμή αντίστοιχα, όπως ακριβώς απαιτείται σε μια μεταβλητή δείκτη.
- Αντίστοιχα, όταν καλείται η συνάρτηση, σαν παράμετρος πρέπει να δίνεται μια **διεύθυνση** που αντιστοιχεί σε ένα αντικείμενο (μεταβλητή) τύπου Τ.
  - Προσοχή: ένα κλασικό λάθος είναι το πέρασμα της τιμής αντί της διεύθυνσης της μεταβλητής.

```
#include <stdio.h>

void inc0(int a) {
  a = a + 1;
}

void inc1(int *b_ptr) {
  *b_ptr = *b_ptr + 1;
}

int main(int argc, char *argv[]) {
  int a1, a2;

  printf("enter int value: ");
  scanf("%d", &a1);
  a2=a1;
  inc0(a1);
  inc1(&a2);
  printf("a1=%d, a2=%d\n", a1, a2);
  inc0(a1);
  inc1(&a2);
  printf("a1=%d, a2=%d\n", a1, a2);

  return(0);
}
```

δείκτης!

διεύθυνση!

```
#include <stdio.h>

void swap(int *a_ptr, int *b_ptr) {
  int tmp;

  tmp=*a_ptr; *a_ptr=*b_ptr; *b_ptr=tmp;

}

int main(int argc, char *argv[]) {
  int i,j;

  printf("enter 2 int values: ");
  scanf("%d %d", &i, &j);
  printf("i=%d, j=%d\n", i, j);

  swap(&i,&j);

  printf("i=%d, j=%d\n", i, j);

  return(0);

}
```

Πίνακες ως παράμετροι συναρτήσεων

- Η τυπική παράμετρος δηλώνεται ως πίνακας.
- Δεν είναι υποχρεωτικό να προσδιοριστεί το μέγεθος της “μεγαλύτερης” διάστασης του πίνακα.
- Κατά την κλήση, σαν πραγματική παράμετρος περνιέται (στην στοίβα) η **διεύθυνση** (του πρώτου στοιχείου) του πίνακα, **όχι τα περιεχόμενα του**.
- Ο κώδικας της συνάρτησης μπορεί να αλλάξει τις τιμές των στοιχείων του πίνακα, και αυτές οι αλλαγές θα «**παραμείνουν**» αφού επιστρέψει η συνάρτηση.
  - **Παρατήρηση:** μια τυπική παράμετρος τύπου δείκτη-σε-Τ μπορεί να ερμηνευτεί ως η αρχή ενός πίνακα από Τ.


```
/* αλλαγή σε κεφαλαία */
#include <stdio.h>
#define N 16

void smallToCapitals(char s[]) {
  int i;

  for (i=0; s[i] != '\0'; i++) {
    if ( (s[i] >= 'a') && (s[i] <= 'z') ) {
      s[i] = 'A' + s[i] - 'a';
    }
  }
}

int main(int argc, char *argv[]) {
  char str[N];

  scanf("%15s",str);
  printf("%s\n",str);
  smallToCapitals(str);
  printf("%s\n",str);

  return(0);
}
```

```
/* αλλαγή σε κεφαλαία */
#include <stdio.h>
#define N 16

void smallToCapitals(char *s) {
  int i;

  for (i=0; s[i] != '\0'; i++) {
    if ( (s[i] >= 'a') && (s[i] <= 'z') ) {
      s[i] = 'A' + s[i] - 'a';
    }
  }
}

int main(int argc, char *argv[]) {
  char str[N];

  scanf("%15s",str);
  printf("%s\n",str);
  smallToCapitals(str);
  printf("%s\n",str);

  return(0);
}
```

Ας δουλέψουμε λίγο στο πάνω μισό...

```
/* αλλαγή σε κεφαλαία */
#include <stdio.h>
#define N 16

void smallToCapitals(char s[]) {
  int i;

  for (i=0; s[i] != '\0'; i++) {
    if ( (s[i] >= 'a') && (s[i] <= 'z') ) {
      s[i] = 'A' + s[i] - 'a';
    }
  }
}

int main(int argc, char *argv[]) {
  char str[N];

  scanf("%7s",&str[8]);
  printf("%s\n",str);
  smallToCapitals(&str[8]);
  printf("%s\n",&str[8]);

  return(0);
}
```

```
/* ανάγνωση και ταξινόμηση ακεραίων */
#include <stdio.h>
#define N 10

void swap(int *a_ptr, int *b_ptr) { … }

void sort(int t[], int len) {
  int i, j;
  for (i=0; i<len; i++)
    for (j=i; j<len; j++)
      if (t[i]>t[j])
         swap(&t[i],&t[j]);
}

int main(int argc, char *argv[]) {
  int buf[N], i;
  for (i=0; i<N; i++)
    scanf("%d", &buf[i]);
  sort(buf,N);
  for (i=0; i<N; i++)
    printf("%d ", buf[i]);
  printf("\n");
}
```


```c
/* ανάγνωση και ταξινόμηση ακεραίων */
#include <stdio.h>
#define N 10

void swap(int *a_ptr, int *b_ptr) { … }

void sort(int *t, int len) {
  int i, j;
  for (i=0; i<len; i++)
    for (j=i; j<len; j++)
      if (t[i]>t[j])
        swap(&t[i],&t[j]);
}

int main(int argc, char *argv[]) {
  int buf[N], i;
  for (i=0; i<N; i++)
    scanf("%d", &buf[i]);
  sort(buf,N);
  for (i=0; i<N; i++)
    printf("%d ", buf[i]);
  printf("\n");
}
```

Δεν είναι σωστή λύση. Απλά ταξινομείται το κάτω μισό και το πάνω μισό του πίνακα

```
/* ανάγνωση και ταξινόμηση ακεραίων */
#include <stdio.h>
#define N 10

void swap(int *a_ptr, int *b_ptr) { … }

void sort(int t[], int len) {
  int i, j;
  for (i=0; i<len; i++)
    for (j=i; j<len; j++)
      if (t[i]>t[j])
        swap(&t[i],&t[j]);
}

int main(int argc, char *argv[]) {
  int buf[N],i;
  for (i=0; i<N; i++)
    scanf("%d", &buf[i]);
  sort(buf,N/2);
  sort(&buf[N/2],N/2);
  for (i=0; i<N; i++)
    printf("%d ", buf[i]);
  printf("\n");
  return(0);
}
```

Σχόλιο

- Η αλλαγή της τιμής μιας εξωτερικής μεταβλητής μέσα από συνάρτηση, μέσω δείκτη, μπορεί να θεωρηθεί ως μια μορφή παρενέργειας.
- Η διαφορά σε σχέση με την αλλαγή καθολικών μεταβλητών είναι ότι για να γίνει αυτό πρέπει να υπάρχει **τυπική παράμετρος** που να έχει **δηλωθεί** σαν **δείκτης**, και όταν γίνεται η κλήση να δίνεται σαν παράμετρος η **διεύθυνση** της μεταβλητής.
- Αυτό είναι ορατό κατά την ανάγνωση του κώδικα, συνεπώς δεν αποτελεί «πραγματική» παρενέργεια.
- Το πρόθεμα `const` σε μια τυπική παράμετρο δείκτη, δηλώνει ότι η συνάρτηση **δεν αλλάζει** τα περιεχόμενα που βρίσκονται σε αυτή τη διεύθυνση.

Δείκτες ως αποτέλεσμα συναρτήσεων

- Μια συνάρτηση μπορεί να δηλωθεί έτσι ώστε να επιστρέφει σαν αποτέλεσμα ένα δείκτη-σε-Τ.
- Χρειάζεται προσοχή ώστε η τιμή που επιστρέφεται να αντιστοιχεί σε μεταβλητή (μνήμη) που είναι **μόνιμη**.
  - Η επιστροφή της διεύθυνσης μιας (συμβατικής) προσωρινής τοπικής μεταβλητής μιας συνάρτησης είναι **προγραμματιστικό λάθος**, καθώς αυτή μπορεί να **μην** υφίσταται μετά την κλήση της συνάρτησης (καταστρέφεται το πλαίσιο εκτέλεσης).
  - Αυτό δεν εντοπίζεται από το μεταφραστή αλλά οδηγεί (αν είμαστε τυχεροί, και όχι πάντα) σε τερματισμό της εκτέλεσης του προγράμματος.


```c
#include <stdio.h>

int *add(int a, int b) {
  int c;
  c=a+b;
  return(&c); /* αυτό είναι λάθος ! */
}

int f(int a, int b) {
  int c = 0;
}

int main(int argc, char *argv[]) {
  int a,b,*c;

  printf("enter 2 int values: ");
  scanf("%d %d", &a, &b);
  c=add(a,b);
  f(a,b);
  printf("the result is %d\n", *c);

  return(0);
}
```

Χρήση συναρτήσεων

Χρησιμοποιούμε συναρτήσεις:

- Όταν το πρόγραμμα αποτελείται από μια ομάδα εντολών που επαναλαμβάνεται πολλές φορές, και η οποία μπορεί να **παραμετροποιηθεί** έτσι ώστε να γράψουμε τον κώδικα μια μοναδική φορά.
- Όταν επιθυμούμε, για λόγους καλύτερης δόμησης, να σπάσουμε ένα μεγάλο τμήμα κώδικα σε περισσότερα, νοηματικά ανεξάρτητα, κομμάτια.
- Όταν επιθυμούμε να έχουμε ανεξάρτητα τμήματα κώδικα σε διαφορετικά αρχεία ή/και με δυνατότητα ξεχωριστής μετάφρασης (π.χ. βιβλιοθήκες).

Χρήση μεταβλητών

- Κάθε μεταβλητή εξυπηρετεί ένα συγκεκριμένο σκοπό και ονομάζεται αντίστοιχα (χωρίς υπερβολές).
- Μεταβλητές με «ειδικό» ρόλο σχολιάζονται ώστε να διευκολύνουν την ανάγνωση του κώδικα.
- Ιδανικά, η λειτουργία κάθε συνάρτησης πρέπει να είναι κατανοητή «από μόνη της», χωρίς να γνωρίζουμε το τι (ακριβώς) κάνει ο υπόλοιπος κώδικας του προγράμματος.
- Οι καθολικές μεταβλητές πρέπει να χρησιμοποιούνται με σύνεση, μόνο όταν απλουστεύουν τον κώδικα.

Συναρτήσεις με άγνωστο αριθμό παραμέτρων

- Μπορεί να υλοποιηθούν συναρτήσεις με άγνωστο (μεταβλητό) αριθμό παραμέτρων, ορίζοντας ως τελευταία (αλλά όχι πρώτη) παράμετρο το "...".
- Αυτό είναι βολικό σε περιπτώσεις που ο αριθμός των παραμέτρων δεν μπορεί να προσδιοριστεί εκ των προτέρων ή είναι επιθυμητό η συνάρτηση να μπορεί να καλείται με μεταβλητό αριθμό παραμέτρων.
- Ο συνολικός αριθμός των παραμέτρων πρέπει να μπορεί να υπολογίζεται με βάση τις τιμές των (γνωστών) παραμέτρων της συνάρτησης.

- Κλασικό παράδειγμα: `printf` και `scanf`.

Βασικές λειτουργίες βιβλιοθήκης

- Για την πρόσβαση στις παραμέτρους που δόθηκαν κατά την κλήση, χρησιμοποιούνται οι μάκρο-εντολές (από τη βιβλιοθήκη `stdarg`):
  - **`va_list`**: ο τύπος της λίστας των παραμέτρων
  - **`void va_start(va_list ap, last)`**: αρχικοποιεί την μεταβλητή `ap` ώστε να δείχνει στην πρώτη άγνωστη παράμετρο μετά την τελευταία γνωστή παράμετρο της συνάρτησης `last`
  - **`type va_arg(va_list ap, type)`**: επιστρέφει την τιμή της παραμέτρου στην οποία δείχνει η `ap` ερμηνεύοντας την σύμφωνα με τον τύπο `type` και μεταθέτει το `ap` στην επόμενη θέση
    - **`void va_end(va_list ap)`**: καλείται πριν τον τερματισμό της συνάρτησης


```
#include<stdio.h>
#include<stdarg.h>

void sum(int nof_args, ...) {
  int i,s;
  va_list ap;

  va_start(ap,nof_args);
  for(i=0,s=0; i<nof_args; i++) {
    s=s+va_arg(ap,int);
  }
  va_end(ap);
}

int main(int argc, char *argv[]) {
  printf("sum of 1..7 is %d\n",sum(7,1,2,3,4,5,6,7));
  printf("sum of 1..5 is %d\n",sum(5,5,7,9,11,13));

  return(0);
}
```