ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

ΑΡΧΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Κεφάλαιο 5

Επιμέλεια:
Βασίλης Παλιουράς , Αναπληρωτής Καθηγητής
Ευάγγελος Δερματάς , Αναπληρωτής Καθηγητής
Σταύρος Νούσιας , Βοηθός Ερευνητή
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών

Άδειες Χρήσης

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

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου των διδασκόντων καθηγητών.
- Το έργο «Ανοικτά Ακαδήμαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών» έχει χρηματοδοτηθεί μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκη Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικους πόρους.

Ανάπτυξη

Το παρόν εκπαιδευτικό υλικό αναπτύχθηκε στο τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Τύποι long, unsigned, long long

```c
#include <stdio.h>
#include <stdint.h>
#include <inttypes.h>

int main(void) {
    long long unsigned m = 1ULL << 63ULL;
    long unsigned n = 1ULL << 31ULL ;
    unsigned l = 1ULL << 31ULL ;

    printf("long long number: %"PRIu64" with a size of %d bytes\n", m, sizeof(m));
    printf("long number: %lu with a size of %d bytes\n", n, sizeof(n));
    printf("number: %u with a size of %d bytes\n", l, sizeof(l));

    system("pause");
    return 0;

}
```

Εργαστήριο

```c
#include <stdio.h>
#include <stdlib.h>
int getuserresponse(void);
void square(void);
void triangle(void);

int main(void) {
    int answer;

    while((answer=getuserresponse())!=3){
           switch (answer) {
           case 1: square (); break;
           case 2: triangle(); break;
           }
           }

    return 0;
}
```

```c
int getuserresponse(void) {
    int a;

    printf("Enter choice:\n");
    scanf("%d", &a);

    return a;
}

void square(void) {
    printf("square\n");
}

void triangle(void) {
    printf("triangle\n");
}
```

Πίνακες στη C

- Δεσμεύουν ***συνεχή χώρο*** στη μνήμη

- Οι ακόλουθες δηλώσεις οδηγούν τον compiler να δ ημιουργήσει διαφορετική assembly.

```
int a[32];

int s = 32;
int a[s];
```

Πίνακες

- Συλλογή μεταβλητών **ίδιου τύπου**, οι οποίες απ οθηκεύονται σε διαδοχικές θέσεις μνήμης.
- **float** temperature[31];
  - *δήλωση* πίνακα μεταβλητών **float,** 31 στοιχείων
  - temperature[**0**] είναι το **πρώτο** στοιχείο,
  - temperature[**1**] είναι το **δεύτερο** στοιχείο,
  - ...
  - temperature[**30**] είναι το **τριακοστό πρώτο** στοιχείο,
  - temperature είναι **η διεύθυνση του πρώτου στοιχείου**
    - temperature είναι το ίδιο με &temperature[0]

Πίνακες δύο (ή περισσότερων) διαστάσεων

- `int a[3][3] ;`
- `int a[3][3] = {{1,2,3}, {3,2,1}, {1,1,1}};`

1	2	3
3	2	1
1	1	1

Παράδειγμα

```
#include <stdio.h>
#define N 3

int main ( ) {
int i, j;

int a[N][N] = {{1,2,3}, {3,2,1}, {1,1,1}};

for (i=0; i<N; i++) {
    for (j=0; j<N; j++)
            printf("%d ",a[i][j]);
    printf("\n");
}

system("pause");
return 0;
}
```

Αποθήκευση στη μνήμη

a[1] σημαίνει δεύτερη γραμμή
a[1][2] σημαίνει τρίτο στοιχείο δεύτερης γραμμής

1	2	3
3	2	1
1	1	1

Διεύθυνση	Στοιχείο
1000	1
1004	2
1008	3
100C	3
1010	2
1014	1
1018	1
101C	1
1020	1

```c
#include <stdio.h>
#define N 3
int main ( ) {
int i, j;
int a[N][N] = {{1,2,3}, {3,2,1}, {1,1,1}};
int *b = &a[0][0];
for (i=0; i< N; i++) {
    for (j=0; j< N; j++)
            printf("%d ",a[i][j]);
    printf("\n");
    }
for (i=0; i< N*N; i++)
    printf("%d ", *(b+i));

system("pause");
return 0;
}
```

Συνάρτηση της βασικής βιβλιοθήκης scanf ( )

```
int number;
char ch;

scanf("%d", &number);

scanf("%c", &ch);
```

%d θα διαβάσει ακέραιο

%c θα διαβάσει χαρακτήρα

τελεστής διεύθυνσης (&): Επιστρέφει τη διεύθυνση της θέσης μνήμης η οποία αντιστοιχεί στη μεταβλητή που ακολουθεί

Καλύτερα!

```
#define N 2
#include <stdio.h>
void readdata(int [N][N]);
void writedata(int [N][N]);

main ( ) {
int data[N][N] ;

readdata(data) ;
writedata(data);

}
```

```
void readdata(int a[N][N]) {
int i,j;
for (i =0 ; i < N ; i++)
   for ( j = 0 ; j < N ; j ++) {
    printf ("element (%d,%d)?\t",
   i,j);
     scanf("%d", &a[i][j]);
     }
}

void writedata(int b[N][N]) {
int i,j;
for (i =0 ; i < N ; i++) {
   for ( j = 0 ; j < N ; j ++)
        printf ("%d\t", b[i][j]);
   printf("\n");
 }
}
```

Παράδειγμα

```
#define N 2
#include <stdio.h>
main ( ) {
int data[N][N] ;
int i, j ;

for (i =0 ; i < N ; i++)
   for ( j = 0 ; j < N ; j ++) {
     printf ("element (%d,%d)?\t", i, j);
     scanf("%d", &data[i][j]);
     }

for (i =0 ; i < N ; i++) {
   for ( j = 0 ; j < N ; j ++)
        printf ("%d\t", data[i][j]);
   printf("\n");
 }
}
```

```c
#define N 2
#include <stdio.h>
void readdata(int [N][N]);
void writedata(int [N][N]);
int sumdata(int [N][N]);

main ( ) {
int data[N][N] ;

readdata(data) ;
writedata(data);

printf("The sum is: %d\n",
   sumdata(data));

}
```

```c
int sumdata(int x[N][N]) {
int i, j;
int sum = 0;

for (i=0; i<N; i++)
   for (j=0; j<N; j++)
        sum += x[i][j];

return sum;
}
```

Παράδειγμα

- Ζητήθηκε από 40 φοιτητές να βαθμολογήσουν το φαγητό στο κυλικείο από 1 (απαίσιο) έως και 10 (αστέρι michelin).
- Να συνοψίσουμε τα αποτελέσματα.
- Από Deitel & Deitel.

- Λεκτική περιγραφή – Προστακτικός προγραμματισμός: (οργάνωση προγράμματος – συναρτήσεις)

  Διάβασε τις απαντήσεις

  Υπολόγισε το ιστόγραμμα => generatehist()

  Παρουσίασε το αποτέλεσμα => printhist()

- Αναπαράσταση δεδομένων
  - Πίνακας για απαντήσεις ⇩ responses[ ]
  - Πίνακας για ιστόγραμμα ⇩ frequency[ ]

Πρότυπα συναρτήσεων

Υλοποίηση συναρτήσεων

```c
#include <stdio.h>
#include <stdlib.h>

#define RESPONSE_SIZE 40
#define FREQUENCY_SIZE 11

void generatehist(int [], int []);
void printhist(int []);
void printstars(int );

int main()
{
   int frequency[ FREQUENCY_SIZE ] = { 0 };
   int responses[ RESPONSE_SIZE ] =
      { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
        1, 6, 3, 8, 6, 10, 3, 8, 2, 7,
        6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
        5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };

   generatehist(frequency, responses);

   printhist(frequen

   return 0;
}
```

```c
void generatehist(int frequency[], int responses[]){
 int answer;

 for ( answer = 0; answer <= RESPONSE_SIZE - 1; answer++ )
            ++frequency[ responses [ answer ] ];
}

void printhist(int frequency[]) {
 int rating;

 printf( "%s%17s Bar\n", "Rating", "Frequency" );
 for ( rating = 1; rating <= FREQUENCY_SIZE - 1; rating++ ) {
    printf( "%6d%17d ", rating, frequency[ rating ] );
    printstars(frequency[rating]);
    printf("\n");
 }
}

void printstars(int len){
 int i;

 for (i=0;i<len;i++)
   putchar('*');
}
```

Η λεκτική περιγραφή αντιστοιχίζεται σε κλήση συναρτήσεων.

```c
#include <stdio.h>
#define RESPONSE_SIZE 40
#define FREQUENCY_SIZE 11

int main()
{
   int answer, rating, frequency[ FREQUENCY_SIZE ] = { 0 };
   int responses[ RESPONSE_SIZE ] =
      { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
        1, 6, 3, 8, 6, 10, 3, 8, 2, 7,
        6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
        5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };

   for ( answer = 0; answer <= RESPONSE_SIZE - 1; answer++ )
      ++frequency[ responses [ answer ] ];

   printf( "%s%17s\n", "Rating", "Frequency" );

   for ( rating = 1; rating <= FREQUENCY_SIZE - 1; rating++ )
      printf( "%6d%17d\n", rating, frequency[ rating ] );

   return 0;
}
```

Αλφαριθμητικά (strings)

πρόκειται για **πίνακες χαρακτήρων**:

- **char** name[30];

- αρχικοποίηση με
  char name[30] = "abcd";
  το οποίο ισοδυναμεί με
  name[0] = 'a';
  name[1] = 'b';
  name[2] = 'c';
  name[3] = 'd';
  name[4] = **0** ; /* δηλώνει το τέλος ενός αλφαριθμητικού */

Ανάγνωση και εκτύπωση αλφαριθμητικού

- **char** str[N_MAX];
- scanf ("%s", str);
- printf ("%s\n", str);

- %s → αντιστοιχεί σε αλφαριθμητικό
- str[0] είναι ο πρώτος χαρακτήρας
- str είναι η **διεύθυνση του πρώτου στοιχείου**
  - str είναι το **ίδιο** με &str[0]
  - ισχύει για κάθε τύπο πίνακα

Παράδειγμα

- Το σύστημα ζητά από το χρήστη το όνομά του και τυπ ώνει "hello" ακολουθούμενο από το όνομα του χρήστη.

Υλοποίηση σε C

```
#include <stdio.h>
#include <string.h>
#define N 10

int main (  ) {

 char username[N];

 printf("Please enter user name: ");
 scanf("%s", username);

 printf("Hello, %s\n", username);
 printf("Your name is %d letters long", strlen(username));


return  0;
}
```

το όνομα του πίνακα είναι η διεύθυνση του πρώτου στοιχείου (⇒ δεν βάζουμε &)

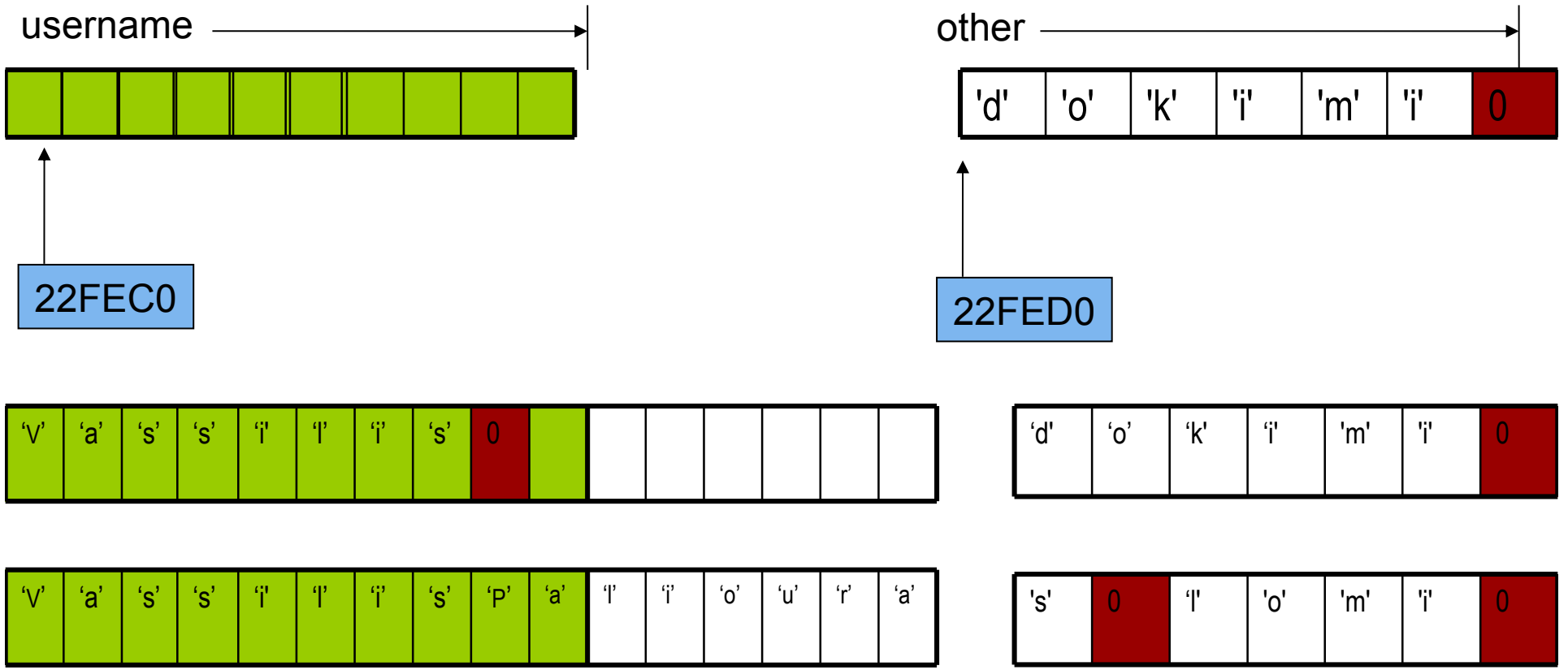
συνάρτηση βιβλιοθήκης strlen()

Υπάρχουν όρια;

```
#include <stdio.h>
#include <string.h>
#define N 10

int main (  ) {

 char other[ ] = "dokimi";
 char username[N];

 printf("Please enter user name: ");
 scanf("%s", username);

 printf("Hello, %s\n", username);
 printf("Your name is %d letters long stored at %X\n", strlen(username), username);

printf("Value of other: %s at %X", other, other);

return 0;
}
```

Ναι, αλλά δεν γίνεται έλεγχος...

Σχηματικά η μνήμη – buffer overflow



Καταστρέφονται τα περιεχόμενα της other!

Επεξεργασία ανά χαρακτήρα

```
#include <stdio.h>
int main()
{
   char string1[ 20 ], string2[] = "string literal";
   int i;

   printf(" Enter a string: ");
   scanf( "%s", string1 );
   printf( "string1 is: %s\nstring2: is %s\n"
           "string1 with spaces between characters is:\n",
           string1, string2 );

   for ( i = 0; string1[ i ] != '\0'; i++ )
      printf( "%c ", string1[ i ] );

   printf( "\n" );
   return 0;
}
```

Μια τεχνική

- **Εξασφαλίζουμε** ότι μια συνάρτηση μπορεί να αλλάξει τι μές πίνακα μόνο αν αναλυτικά το επιτρέψουμε.
- Εφαρμογή της αρχής *ελαχίστου δικαιώματος.*

Παράδειγμα

```c
#include <stdio.h>
void DisplayName(char []);

int main ( ) {
char astring[10] = "Hello";

DisplayName(astring);
DisplayName(astring);

return 0;
}

void DisplayName(char x[]) {
int i;
for (i=0; x[i]!=0 ; i++)
    printf("%c", x[i]);
x[0]='h';
printf("\n");
}
```

```c
#include <stdio.h>
void DisplayName(const char []);

int main ( ) {
char astring[10] = "Hello";

DisplayName(astring);
DisplayName(astring);

return 0;
}

void DisplayName(const char x[]) {
int i;
for (i=0; x[i]!=0 ; i++)
    printf("%c", x[i]);
x[0]='h';
printf("\n");
}
```

Error: assignment of read-only location

Παράδειγμα

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
   char alphabet[27]; // 26 letters plus trailing zero
   char c;

   for (c='A'; c<='Z'; c++)
        alphabet[c-'A'] = c;

   alphabet[c-'A'] = 0;

   printf("%s", alphabet);

   return 0;
}
```

Δείκτες (Pointers)

- **Δείκτης**: μεταβλητή στην οποία αποθηκεύουμε **διεύθυνση** θέσης μνήμης.
  - δείχνει ***που*** είναι αποθηκευμένα δεδομένα
- **Δήλωση Δείκτη**

  `<τύπος>  *<όνομα δείκτη>;`
- ΠΡΟΣΟΧΗ: Το όνομα πίνακα **είναι** διεύθυνση, αλλά **δεν είναι** μεταβλητή!!!

Παράδειγμα δήλωσης δείκτη

- `char *ch_ptr;`
- Η μεταβλητή ch_ptr περιέχει ***διεύθυνση μνήμης*** στην οποία ε ίναι αποθηκευμένο δεδομένο τύπου χαρακτήρα.
- `char ch;`
- Η μεταβλητή ch έχει ως αξία χαρακτήρα.

- Μπορούμε να δηλώσουμε δείκτες σε δεδομένα διαφόρων τύπ ων
  - Βασικών
  - Κατασκευασμένων

Παράδειγμα χρήσης δείκτη

```
void main ( ) {
char ch = 'a', ch2;
char *ch_ptr ;

ch_ptr = &ch ;
ch2 = *ch_ptr ;

printf ("%c", ch2);

}
```

Δήλωση δείκτη σε χαρακτήρα

* ⓿ **Περιεχόμενα της θέσης** στην οποία δείχνει ο δείκτης

Πίνακες και δείκτες

- `int arr[10], n ;`
- `*(arr + n)` ⟺ `arr[n]`
- `arr + n` ⟺ `&arr[n]`
- χρησιμοποιούμε δείκτες για να περάσουμε ως όρισμα σε συνάρτηση πίνακες
  - ακριβέστερα: σε ποια διεύθυνση μνήμης βρίσκεται το πρώτο στοιχείο του πίνακα.

Συναρτήσεις βασικής βιβλιοθήκης για αλ φαριθμητικά

- πρότυπα στο <string.h>
- **char** *strcpy (**char** *, **const char** *) ;
- **int** strcmp (**const char** *, **const char** *) ;
- **char** *strcat (**char** *, **const char** *) ;
- **char** *strchr (**const char** *, **char**) ;
- size_t strlen (**const char** *) ;
- …

Παράδειγμα

- Διάβασε ένα αλφαριθμητικό
- Μέτρησε πόσες φορές περιλαμβάνει τον χαρακτήρα ‘a’
- Τύπωσε το αποτέλεσμα.

```
void readstring(char *) ;
int countA(char *);
void printresult( int ) ;
```

Η main ( ) του παραδείγματος

```
main ( ) {
 char astring[N];
 int acount ;

 readstring (astring) ;

 acount = countA(astring);

 printresult(acount) ;
}
```

```
#define N 50
#include <stdio.h>
void readstring(char *) ;
int countA(char *);
void printresult( int ) ;
```

```
void readstring(char *s ) {
        printf ("alpharithmitiko? ");
        scanf("%s", s);
}
```

```
void printresult ( int a) {
        printf("The result is %d\n", a);
}
```

Υλοποίηση της **int** countA(**char** *);

```
int countA(char *s) {
int count = 0 ;
int i = 0;
  while ( s[i] != 0 ) {
    if (s[i] == 'a')
       count ++;
    i ++ ;
    }
  return count ;
}
```

γιατί το μηδέν δηλώνει τέλος του αλφαριθμητικού

αν ο τρέχων χαρακτήρας είναι ίσος με ‘a’, αύξησε το μετρητή count κατά ένα

προχώρησε στον επόμενο χαρακτήρα του αλφαριθμητικού

Πρόθεμα και Επίθεμα (prefix και postfix)

- i++; /*postfix */
- ++ i; /* prefix */
- i = 0;
- myprint(i++);
- i = 0;
- myprint(++i);

Η myprint ( ) καλείται με διαφορετικό όρισμα (διαφορετική τιμή) στις δύο περιπτώσεις!

Πρόβλημα

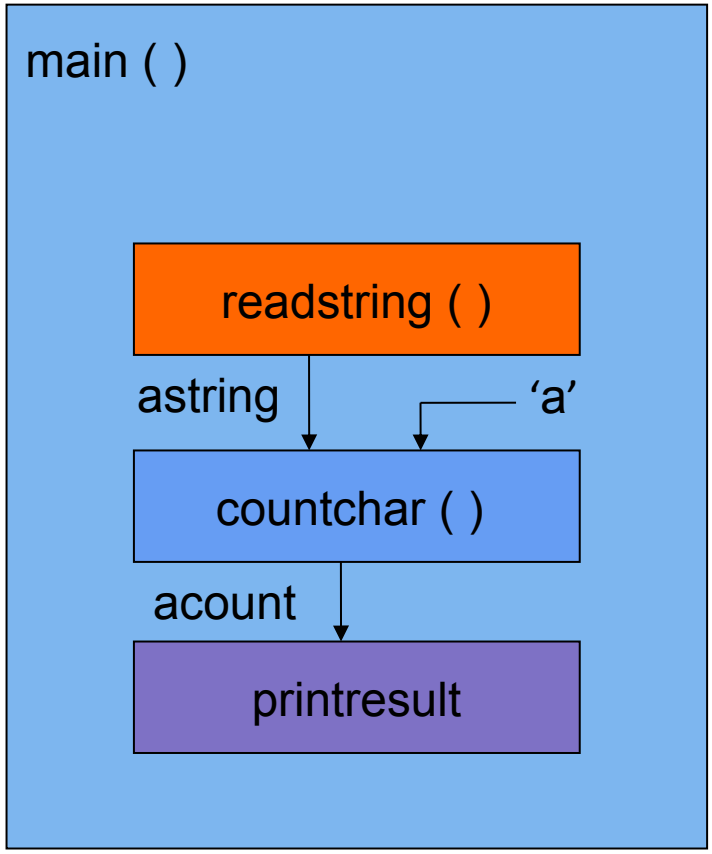
- Πώς θα γράφαμε πρότυπο και τον ορισμό συνάρτησης η οποία θα δέχεται ως ορίσματα:
  α) το αλφαριθμητικό και
  β) τον χαρακτήρα για τον οποίο γίνεται ο έλεγχος.
- Πρότυπο αυτής:

  int countchar(**char** *, **char**);
- Παράδειγμα κλήσης

  acount = countchar(astring, 'a');
- Τι πλεονεκτήματα έχει να γράφουμε γενικότερο κώδικα;

Χρήση της countchar(**char** *, **char**)

```
main ( ) {
 char astring[N];
 int acount ;

 readstring (astring) ;

 acount = countchar(astring, ‘a’);

 printresult(acount) ;
}
```



Πιθανές υλοποιήσεις

```
int countA(char *s, char ch)
    {
int count = 0 ;
int i = 0;
  while ( s[i] != 0 ) {
     if (s[i] == ch)
        count ++;
     i ++ ;
     }
  return count ;
}
```

```
int countA(char *s, char ch)
    {
int count = 0 ;
int i ;
  for (i=0; s[i] != 0; i++)
     if (s[i] == ch)
                count ++;
   return count ;
}
```

int countchar(**char** *, **char**);

```
int countchar(char *s, char c) {
int count = 0 ;
int i = 0;
  while ( s[i] ) {
                count += (s[i] == c);
              i ++ ;
              }
  return count ;
}
```

```
int countchar(char *s, char c) {
int count = 0 ;
int i = 0;                          postfix notation
  while ( s[i] )                          ↓
                count += (s[i++] == c);
return count ;
}
```

int countchar(**char** *, **char**);

```
int countchar(char *s, char c) {
int count = 0 ;
int i = 0;
  while ( s[i] ) {
                count += (s[i] == c);
              i ++ ;
              }
  return count ;
}
```

```
int countchar(char *s, char c) {
int count = 0 ;
int i = 0;
  while ( s[i] )                    postfix notation
                count += (s[i++] == c);
return count ;
}
```

```
int countchar(char *s, char c) {
int count = 0 ;
int i ;

  for( i = 0; s[i]; count += (s[i++] == c));

return count ;
}
```

```
int countchar(char *s, char c) {
int count = 0 ;
int i = 0;

  for( ; s[i]; count += (s[i++] == c));

return count ;
}
```

Μηχανισμοί κλήσης συναρτήσεων

Τι θα τυπωθεί;

```
#include <stdio.h>

int myfun(int, int);

main ( ) {
int a = 3, b = 3;

myfun(a, b);
printf("main: a:%d b:%d\n", a, b);

}
```

```
int myfun(int a, int b) {
a++;
b++;
printf("myfun: a:%d b:%d\n",
    a, b);
return 0;
}
```

Κλήση συνάρτησης κατ' αναφορά (call by reference)

- πρότυπο: **void** swap(**int** *a, **int** *b);

- κλήση: swap(&value1, &value2);

1. Η συνάρτηση επενεργεί **απευθείας στις θέσεις μνήμης** των μεταβλ ητών που χρησιμοποιούνται ως πραγματικά ορίσματα.
2. **Δεν** δημιουργούνται τοπικά αντίγραφα των δεδομένων.
   - Ισχύει για τις διευθύνσεις;
3. Στη C, ουσιαστικά, υλοποιείται ως κατ' αξία πέρασμα διευθύνσεων.

Ορισμός συνάρτησης swap( )

```
void swap(int *a, int *b) {
        int temp ;
        temp = *b ;
        *b = *a ;
        *a = temp ;
}
```

Παράδειγμα χρήσης κλήσης κατ΄αναφορά

```
#include <stdio.h>
void swap (int *, int *);

main ( ) {
 int value1 = 5;
 int value2 = 3;

 printf("value1: %d value2: %d\n", value1, value2);

 swap (&value1, &value2) ;

 printf("value1: %d value2: %d\n", value1, value2);
}
```

```
void swap(int *a , int *b) {
int temp;
 temp = *a ;
 *a = *b ;
 *b = temp ;
}
```

```
Paliuras@MOB ~/nlect9
$ ./a
value1: 5 value2: 3
value1: 3 value2: 5
```

Μερικές διαφορές

- Μηχανισμός κλήσης κατ' αξία
  - δημιουργούνται τοπικά αντίγραφα των ορισμάτων, τα οποία χρη σιμοποιεί η συνάρτηση
  - Αν τα ορίσματα έχουν μήκος πολλών bytes, επιβαρύνεται η εκτέ λεση.
  - Η συνάρτηση δεν μπορεί να αλλάξει τις τιμές ορισμάτων στο σημ είο κλήσης.
- Μηχανισμός κλήσης με αναφορά
  - δεν δημιουργούνται τοπικά αντίγραφα, η συνάρτηση ενημερώνετ αι για τη θέση μνήμης στην οποία είναι αποθηκευμένο ένα όρισμ α.
    - μπορεί να είναι ταχύτερο
  - είναι δυνατόν να αλλάξουν οι τιμές ορισμάτων στο σημείο κλήση ς.
    - χρειάζεται προσοχή, μπορεί να γίνει εκ παραδρομής...

Διάρκεια μεταβλητής

```
void function (void);
void function1 (void);
main () {
        function1();
        function ( ) ;
        function ( ) ;
        function ( ) ;
}
void function1() {
int j = 0;
}
void function ( ) {
int i;
        i++ ;
        printf("%d\n", i);
```

Η τοπική μεταβλητή *i* δεν αρχικοποιείται!

Δουλεύει «σωστά»(;;;)

Γιατί;

Διάρκεια μεταβλητής (2)

```
void function1 (void);
void function2 (void);
void function(void);
main ( ) {
                function();
                function1( ) ;
                function2( ) ;
                function1( ) ;
                function2( ) ;
                function1( ) ;
                function2( ) ;
}
void function() {
int k = 0;
}
void function1( ) {
int i;
                i++ ;
                printf("f1:%d\n", i);
}
void function2( ) {
int j;
                j++ ;
                printf("f2:%d\n", j);
```

Αν η κλήση της function1( ) γίνεται εναλλάξ με την function2( ), η συμπεριφορά αλλάζει...

Εμφανίζονται εξαρτήσεις (είναι δυνατόν να ...)

Λύση;;;

```
void function1 (void);
void function2 (void);

main ( ) {
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
        function1( ) ;
        function2( ) ;
}
void function1 () {
int i = 0;
        i++ ;
        printf("f1:%d\n", i);
}
void function2 () {
int j = 0;
        j++ ;
        printf("f2:%d\n", j);
}
```

Αρχικοποιώντας τις μεταβλητές

κάθε φορά που καλείται η συνάρτηση,

οι τοπικές μεταβλητές μηδενίζονται κάθε

φορά που καλείται η συνάρτηση.

Η λέξη κλειδί **static** σε δηλώσεις τοπικών μεταβλητών

```
void function1 (void);
void function2 (void);

main ( ) {
                function1( )  ;
                function2( ) ;
                function1( ) ;
                function2( ) ;
                function1( ) ;
                function2( ) ;
}
void function1 ( ) {
static int i = 0;
                i++ ;
                printf("f1:%d\n", i);
}
void function2 ( ) {
static int j = 0;
                j++ ;
                printf("f2:%d\n", j);
}
```

static: ο χώρος μνήμης της μεταβλητής δεν αποδεσμεύεται όταν ολοκληρωθεί μια εκτέλεση της συνάρτησης.

Άλλη χρήση της **static**: Ορισμός εμβέλειας αρχείου

- αρχείο πηγαίου κώδικα: code1.c

```
int func (int);
void report(void);

main () {
int i ;

for ( i=0; i< 5; i++)
    printf("%d\n", func(i));

report();
}
```

- αρχείο πηγαίου κώδικα: code2.c

```
static int sum = 0;

int func(int k) {
int i;
for (i=0;i<5; i++)
    sum += k ;  /* sum = sum + k */
return 2 * k;
}
void report() {
    printf("%d\n", sum);
}
```

Σημείωμα αναφοράς

- Copyright Πανεπιστήμιο Πατρών, Παλιουράς Βασίλειος , Δερματάς Ευάγγελος «Αρχές Προγραμματισμού ». Έκδοση: 1.0. Πάτρα 2015
- Διαθέσιμο από τη δικτυακιακή διέυθυνση https://eclass.upatras.gr/modules/