Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών

Εισαγωγή στον Προγραμματισμό

Η γλώσσα προγραμματισμού C

Δυναμική Διαχείριση Μνήμης

- ***Calloc, Malloc,***
  για δέσμευση μνήμης
- ***Realloc***
  για αλλαγή μεγέθους δεσμευμένης μνήμης
- ***Free***
  για απελευθέρωση μνήμης

Malloc()

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

```
void *malloc(size_t size);
```

Η συνάρτηση malloc δεσμεύει χώρο μεγέθους τουλάχιστον size bytes.

Επιστρέφει ένα δείκτη στη δεσμευθείσα μνήμη ή NULL, αν δεν υπάρχει διαθέσιμη μνήμη. Τα bytes του χώρου μνήμης δεν παίρνουν αρχική τιμή

x=(int *)malloc(n*sizeof(int)); // int *x;

x=(int **)mallloc(n*sizeof(int *));

for (i=0; i<n; i++) *(x+i)=(int *)malloc(m*sizeof(int));

&x[i][j] ---- *(x+i)+j

x[i][j] ---- *(*(x+i)+j)

Calloc()

```
#include <stdlib.h>

void *calloc(size_t n,size_t size);
```

Η συνάρτηση calloc δεσμεύει χώρο για ένα πίνακα n στοιχείων, μεγέθους size το καθένα. Διασφαλίζει την αρχικοποίησή της με 0.

Επιστρέφει ένα δείκτη στη δεσμευθείσα μνήμη ή NULL, αν δεν υπάρχει διαθέσιμη μνήμη.

p=(int *)calloc(n, sizeof(int));

Realloc()

```
#include <stdlib.h>

void *realloc(void *buffer, size_t size);
```

Η συνάρτηση realloc μεταβάλλει το μέγεθος ενός τμήματος μνήμης που είχε προηγούμενα δεσμευτεί. Το νέο μέγεθος ορίζεται από τη size. Διασφαλίζει τα υπάρχοντα περιεχόμενα στη μνήμη. Επιστρέφει ένα δείκτη στο νέο τμήμα μνήμης που μπορεί να είναι ίδιος με τον buffer ή διαφορετικός αν το τμήμα μετακινηθεί. Αν ο buffer είναι NULL, η realloc λειτουργεί σαν τη malloc.

p=(int *) realloc(p, n*sizeof(int));

Free ()

```
#include <stdlib.h>

void free(void * buffer);
```

Η συνάρτηση free αποδεσμεύει τη μνήμη η οποία σηματοδοτείται από το όρισμα buffer και η οποία είχε προηγούμενα δεσμευτεί με κλήση μίας εκ των *calloc*, *malloc*, *realloc*.

Δείκτες – Πίνακες (1)

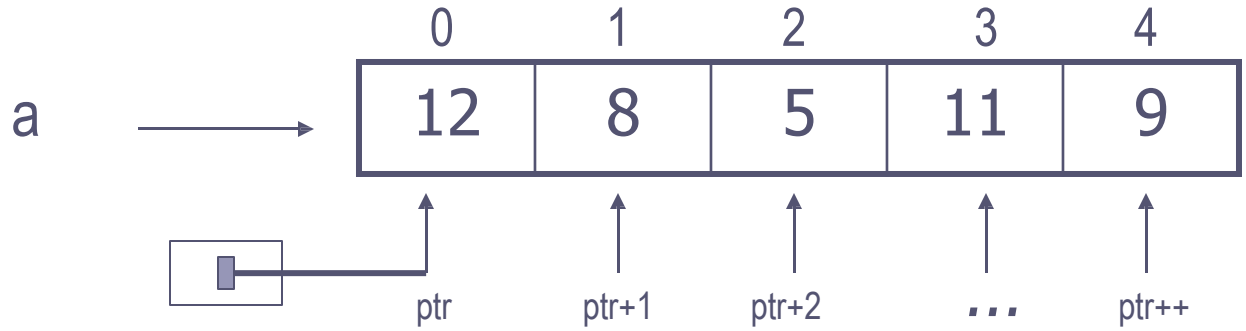
- Το όνομα ενός πίνακα είναι μία «σταθερά τύπου δείκτη» με τιμή τη διεύθυνση του πρώτου στοιχείου του πίνακα.
- Λόγω αυτού του γεγονότος μπορούμε να ορίσουμε ένα δείκτη στον πίνακα και να χειριστούμε τον πίνακα μέσω αυτού του δείκτη.
- Δεν είναι εφικτή η ανάθεση τιμής και η αριθμητική με δείκτες

Δείκτες – Πίνακες (2)

Έστω

```
int a[5] = { 12, 8, 5, 11, 9};
int *ptr;

ptr = a; ή  ptr = &a[0];
```



Προσοχή: a++; a-- (το όνομα του πίνακα δεν είναι δείκτης (δηλ. μεταβλητή δείκτη))

Δείκτες – Πίνακες (3)

Ισοδύναμες εκφράσεις

*(a+i) ⟷ a[i]

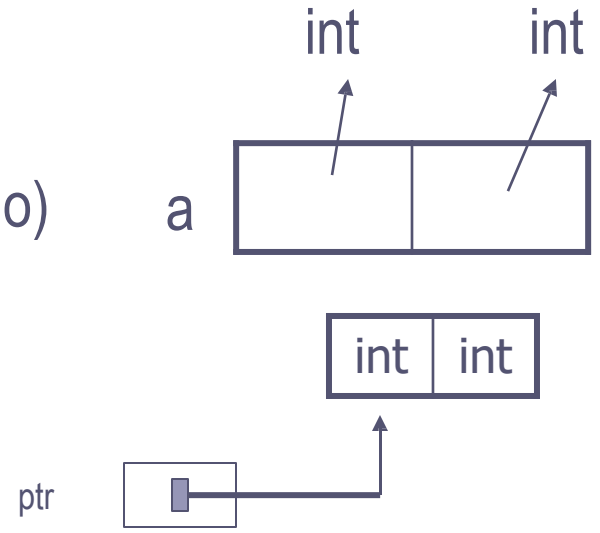
a+i ⟷ &a[i]

Δείκτης σε δείκτη

Προτεραιότητα [ ] μεγαλύτερη από *

π.χ. int *a[2] πίνακας (δύο) δεικτών

int (*ptr)[2] δείκτης σε πίνακα (δύο) ακεραίων



Ασκήσεις

1.Δημιουργήστε κώδικα ο οποίος χειρίζεται δύο μονοδιάστατους πίνακες ακεραίων αριθμών ίδιου μεγέθους, και στη συνέχεια προσθέτει τα περιεχόμενα τους ανά δύο και τα τοποθετεί στον πρώτο πίνακα. Τό μέγεθος των πινάκων θα πρέπει να δίνεται από τον χρήστη ενώ η ανάγνωση δεδομένων και η πρόσθεση των στοιχείων των πινάκων θα πρέπει να γίνεται με κλήση δύο κατάλληλα ορισμένων συναρτήσεων.

2.Δημιουργήστε κώδικα που χειρίζεται ένα πίνακα δύο διαστάσεων οριζόμενο με τη μορφή δείκτη σε δείκτη, διαβάζει τα περιεχόμενα του πίνακα και στη συνέχεια τα εκτυπώνει. Οι διαστάσεις του πίνακα θα πρέπει να δίνονται από τον χρήστη, η ανάγνωση των περιεχομένων και η εκτύπωση θα πρέπει να γίνεται με κλήση συναρτήσεων.

3.Δημιουργήστε κώδικα ο οποίος χειρίζεται δύο δισδιάστατους πίνακες ακεραίων αριθμών A και B μεγέθους n x k και k x m αντίστοιχα (τα n, m, k δίνονται από τον χρήστη). Το πρόγραμμα θα πρέπει να διαβάζει τα περιεχόμενα των δύο πινάκων και στη συνέχεια να αποθηκεύει το γινόμενό τους σε ένα πίνακα C διαστάσεων nxm. Η ανάγνωση των περιεχομένων και ο πολλαπλασιασμός θα πρέπει να γίνεται με την κλήση κατάλληλων συναρτήσεων.

Δείκτες - Αλφαριθμητικά (1)

Δήλωση δείκτη αλφαριθμητικού

char * <όνομα – δείκτη>;

π.χ. char * pmsg;

Ανάθεση σε δείκτη αλφαριθμητικού

<όνομα-δείκτη> = < αλφαριθμητικό >;

π.χ. pmsg = “Today is Thursday”;

Δείκτες - Αλφαριθμητικά (2)

Προσοχή στις διαφορές

char msg[ ] = “Today is Thursday”; (πίνακας χαρακτήρων)

char * pmsg = “Today is Thursday”; (δείκτης σε πίνακα, μπορεί να πάρει άλλη τιμή)

char * msg[18]; (πίνακας δεικτών χαρακτήρα)

msg[1] -> 2ος δείκτης

* (msg[1]) -> ο 1ος χαρακτήρας του δεύτερου δείκτη

Παράδειγμα (2)

```
void strcpy(char *s, char *t)
{
int  i=0;
while ((s[i]=t[i])!='\0')
i++;
}
```

```
void strcpy(char *s, char *t)
{
while ((*s=*t) != '\0') { s++; t++;}
}
```

```
void strcpy(char *s, char *t)
{
while ((* s++=*t++) != '\0') ;
}
```

Παράδειγμα (3)

```
int strcmp(char *s, char *t)
{
int  i=0;
for (ı=0; (s[i]==t[i])&&s[i]!='\0'; i++);
return s[i]-t[i];
}
```

```
int strcmp(char *s, char *t)
{
for (; (*s==*t) && *s!='\0'; s++, t++);
return  *s-*t;
}
```

Ορίστε μία συνάρτηση που παίρνει σαν όρισμα ένα αλφαριθμητικό και μετατρέπει το αλφαριθμητικό από μικρά γράμματα σε κεφαλαία..
Θεωρήστε σαν δεδομένο την ύπαρξη των συναρτήσεων:
int islower(int c)
Int toupper(int c)

Δομές

- Στην C μπορούμε να προσθέσουμε τους δικούς μας τύπους (π.χ. FILE)
- Βασικά εργαλεία: *struct*, *typedef*.

  struct: ορίζει ένα νέο τύπο δεδομένων.

  typedef: συσχετίζει ένα όνομα με αυτόν.
- Οι μεταβλητές δομών μπορούν να αποδίδονται σαν τιμές, να μεταβιβάζονται σε συναρτήσεις και να επιστρέφονται από συναρτήσεις. Επίσης, μπορεί να εφαρμόζεται σε αυτές ο τελεστής *sizeof*().
- Αρχικοποιούνται με λίστα τιμών όπως οι πίνακες.
- ΔΕΝ συγκρίνονται με χρήση τελεστή == ή !=
- Η αναφορά σε ένα μέλος γίνεται: *όνομα_μεταβλητής_δομής.μέλος*
- Ειδικός συμβολισμός για προσπέλαση μέλους μεταβλητής δείκτη σε δομή.

Η εντολή *struct (1)*

```
#include <stdio.h>
struct line {
    float x1, y1;/* co-ords of 1 end of line*/
    float x2, y2;/* co-ords of other end */
    };
main()
{
struct line line1={1,2,3,4};
.
.
}
```

Typedef

- Η *typedef* επιτρέπει τη συσχέτιση ενός ονόματος με μία δομή (ή άλλο τύπο δεδομένων)
- Τοποθέτησε την *typedef* στην αρχή του προγράμματος.

```
typedef struct line {
        float x1, y1;
        float x2, y2;
        } LINE;

int main()
{
LINE line1;
}
```

Αναφορά στα Μέλη Δομής

Χρησιμοποιείται ο τελεστής . και έχουμε:

<όνομα μεταβλητής>.<όνομα μέλους>

```
LINE line1;
line1.x1= 3;
line1.y1= 5;
line1.x2= 7;

if (line1.y2 == 3) {
    printf ("Y co-ord of end is 3\n");
    }
```

Παράδειγμα

```
typedef struct point{
    float x, y;
    } POINT;
typedef struct line {
    POINT left;
    POINT right;
    } LINE;

int main()
{
LINE line1;
line1.left.x=2;
}
```

Παρατηρήσεις

Μπορούμε να περνάμε και να επιστρέφουμε δομές σε συναρτήσεις (το πρωτότυπο όμως θα πρέπει να έχει οριστεί μετά τη δήλωση δομής)

```
typedef struct line {
    float x1,y1;
    float x2,y2;
} LINE;

LINE find_perp_line (LINE x);
/* Function takes a line and returns a
perpendicular line */
```

Παράδειγμα

```
typedef struct complex {
    float imag;
    float real;
} CPLX;

CPLX mult_complex (CPLX x, CPLX y);

int main()
{
/* Main goes here */
}

CPLX mult_complex (CPLX no1, CPLX no2)
{
     CPLX answer;
     answer.real= no1.real*no2.real - no1.imag*no2.imag;
     answer.imag= no1.imag*no2.real + no1.real*no2.imag;
     return answer;
}
```

Χρησιμότητα

- Διευκολύνουν τον προγραμματισμό.
- FILE * είναι μία *typedef* δομή αλλά μπορούμε να τη χρησιμοποιήσουμε χωρίς να ξέρουμε τι εμπεριέχει.
- Μπορούμε να επεκτείνουμε τη γλώσσα, για παράδειγμα αν χρησιμοποιούμε μιγαδικούς αριθμούς μπορούμε να *επεκτείνουμε* τη γλώσσα με τον τύπο αυτό.

Παράδειγμα

Να υλοποιήσετε ένα πρόγραμμα που να διαχειρίζεται τις καρτέλες με τα στοιχεία μέχρι 20 μαθητών (επώνυμο, όνομα, βαθμός).

Το πρόγραμμα θα πρέπει να έχει τις δυνατότητες:

1. Εισαγωγής στοιχείων μαθητή
2. Παρουσίαση στην οθόνη όλων των καρτελών
3. Εντοπισμού και παρουσίασης της καρτέλας ενός συγκεκριμένου μαθητή, βασιζόμενο στο επώνυμο
4. Παρουσίαση στην οθόνη του μέσου όρου του βαθμού του συνόλου των μαθητών καθώς και τις καρτέλες με τη μικρότερη και μεγαλύτερη βαθμολογία
5. Εξόδου από το πρόγραμμα μετά από επιλογή του χρήστη.