Δομές Δεδομένων
Σημειώσεις εργαστηρίου

Μιχαήλ Ανάργυρος Ζαμάγιας
ΤΠ5000

18 Ιουνίου 2020

Περιεχόμενα

Εργαστήριο 1

1.1 Μονοδιάστατος πίνακας

Ισχύουν τα εξής:

```
1 pin == &pin[0]
2 pin+k == &pin[k]
3 *pin == pin[0]
4 *(pin+k) == pin[k]
```

Η τιμή ενός δείκτη ισούται με τη διεύθυνση μνήμης του byte στο οποίο είναι τοποθετημένος ο δείκτης και εμφανίζεται στην οθόνη με την χρήση του προσδιοριστή %p.

1.2 Πίνακας ως ορίσματα συνάρτησης

Για να «περάσω» σε μια συνάρτηση ως παράμετρο ένα πίνακα, περνάω ένα δείκτη στην αρχή του πίνακα και (αν χρειάζεται) το μέγεθος του πίνακα.

Στον ορισμό μιας συνάρτησης (για παράδειγμα, της `parad`) οι παρακάτω συμβολισμοί είναι ισοδύναμοι:

```
1 void parad(int *pin)
2 void parad(int pin[])
```

Σε κάθε περίπτωση, το `pin` είναι δείκτης σε ακέραιο.

1.3 Ταξινόμηση πίνακα

Υπάρχουν διάφοροι αλγόριθμοι ταξινόμησης ενός πίνακα. Αυτός που περιγράφεται εδώ είναι γνωστός ως «Ταξινόμηση με επιλογή».

Η συνάρτηση θα ξεκινά από το στοιχείο της πρώτης θέσης τοτ πίνακα, το `list[0]`, με στόχο να τοποθετηθεί στην θέση αυτή η μικρότερη τιμή του πίνακα.Η συνάρτηση να διατρέχει όλα τα υπόλοιπα στοιχεία, από το `list[1]` μέχρι το `list[N-1]` και να συγκρίνει καθένα με

το πρώτο. Αν βρει κάποιο μικρότερο από το πρώτο, τα στοιχεία εναλλάσσονται μεταξύ τους. Τελικά το `list[0]` θα έχει την μικρότερη τιμή του πίνακα.

Αφού τελειώσουμε με το πρώτο στοιχείο επαναλαμβάνεται η διαδικασία, προσπαθώντας να βάλουμε στη θέση 1 του πίνακα τη δεύτερη σε μέγεθος τιμή. Συγκρίνονται δηλαδή όλα τα στοιχεία από το `list[2]` και μετά με το `list[1]`. Αν βρεθεί κάποιο στροιχείο μικρότερο από το `list[1]`, τα στοιχεία εναλλάσσονται μεταξύ τους, κ.ο.κ.

Χρειάζεστε δύο εμφωλευμένες επαναλήψεις.

Εργαστήριο 2

2.1 Δισδιάστατος πίνακας

Αναφερόμαστε σε κάθε στοιχείο ενός πίνακα δύο διαστάσεων χρησιμοποιώντας δύο αριθμητικές ετικέτες, για παράδειγμα το στοιχείο της γραμμής `3`, της στήλης `5` ενός πίνακα `pin`, λέγεται `pin[3][5]`.

Χρειαζόμαστε για να διαβάσουμε ή να γράψουμε τα στοιχεία ενός πίνακα δύο διαστάσεων διπλή επαναληπτική εντολή (εμφωλευμένες επαναλήψεις).

Κατά το πέρασμα ενός πίνακα δύο διαστάσεων σε μια συνάρτηση πρέπει να «περνάμε» στην συνάρτηση τον αριθμό των στηλών του πίνακα.

2.2 Πίνακας συμβολοσειρών

Με δεδομένη την δήλωση:

```
char students[R][C];
```

- Η `k` γραμμή του πίνακα είναι μονοδιάστατος πίνακας χαρακτήρων και λέγεται `students[k]`.
- Το `students[k]` είναι πίνακας χαρακτήρων, άρα και δείκτης σε χαρακτήρα.
- Η ταξινόμηση (κατάταξη αλφαβητικά) συμβολοσειρών μοιρεί να γίνει με τον ίδιο αλγόριθμο που γίνεται η ταξινόμηση ακεραίων. Η σύγκριση των συμβολοσειρών θα γίνεται με την συνάρτηση `strcmp()`.

Εργαστήριο 3

3.1 Δομή (struct): περιγραφή, περδία δομής, δηλώσεις και δεδομένα

Η περιγραφή της δομής προϋπάρχει της δήλωσης για να «διδάξει» στον compiler πως είναι το νέο είδος μεταβλητών που δημιουργούμε.

Η περιγραφή της δομής βρίσκεται ανάμεσα σε άγκιστρα, μετά τα οποία υπάρχει το `;`.

Αναφερόμαστε στο κάθε πεδίο μιας δομής ως εξής: `struct_name.member_name`.

Η εμφάνιση μόνο του είδους της δομής (της λέξης δηλαδή που ακολυθεί την λέξη struct) αποτελεί συντακτικό λάθος.

Η εμφάνιση μόνο του ονόματος κάποιου πεδίου της δομής (των λέξεων δηλαδή που εμφανίζονται στην περιγραφή μιας δομής) αποτελεί επίσης συντακτικό λάθος.

Μια δομή `a` κάποιου τύπου μπορεί να γίνει ίση με μια δομή `b` του ίδιου τύπου με απλή απόδοση τιμής, δηλαδή:

```
a = b;
```

3.2 Φωλιασμένη δομή

Φωλιασμένη δομή λέγεται η δομή των οποίων κάποιο ή κάποια πεδία είναι δομή, η οποία έχει ήδη περιγραφεί προηγουμένως.

Η προσπέλαση των πεδίων σε Φωλιασμένη δομή γίνεται με την πολλαπλή χρήση της τελείας (`.`).

3.3 Πίνακας δομών

Σε ένα πίνακα δομών, κάθε στοιχείο του πίνακα είναι μια δομή. Στην άσκηση 3, το `pin` είναι ένας πίνακας `N` δομών του είδους `student`. Άρα, το `pin[0]`, `pin[0]` κ.λ.π. είναι δομές του

είδους `student`.

Για παράδειγμα ο χαρακτήρας της τρίτης θέσης του πίνακα `name` του τέταρτου στοιχείου του πίνακα `pin` λέγεται `pin[3].name[2]`.

Το `fflush(stdin)` αδειάζει τον buffer εισόδου κι έτσι η `gets` δεν επηρεάζεται από το `Enter` της `scanf` που έχει προηγηθεί.

Η άσκηση 3:

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

#define number_of_students 2
#define name_length 30

struct student
{
    int student_id;
    char student_name[name_length];
};

int main()
{
    //  struct array
    struct student pin[number_of_students];
    int k;
    char buf[1024];

    //  enter data to array
    for (k = 0; k < number_of_students; k++)
    {
        //  print message
        printf("Enter Student's %i ID:\t", k + 1);
        //  read number as string
        fgets(buf, 1024, stdin);
        //  convert string to int
        pin[k].student_id = atoi(buf);
        //  print message
        printf("Enter Student's %i name:\t", k + 1);
        //  read string
        fgets(pin[k].student_name, name_length, stdin);
        //  trim \n at the end of the string
        pin[k].student_name[strcspn(pin[k].student_name, "\n")] = 0;
    }

    //  print data from array
    for (k = 0; k < number_of_students; k++)
    {
        printf("Student %2i\n", k + 1);
        printf("Student ID:\t%4i\n", pin[k].student_id);
        printf("Student name:\t%s\n", pin[k].student_name);
    }

    return 0;
}
```

3.4 Δομή ως παράμετρος και ως τιμή επιστροφής συναρτήσεων

Μια συνάρτηση μπορεί να δέχεται ως ορίσματα δομές, όπως οποιοδήποτε άλλο τύπο δεδομένων. Για παράδειγμα η επικεφαλίδα μιας συνάρτησης που είναι `void` και δέχεται ως παραμέτρους δύο δομές, τις `s1` και `s2` του τύπου `stype`, θα είναι:

```
void example(struct stype s1, struct stype s2)
```

Η κλήση της συνάρτησης θα είναι:

```
example(s1, s2);
```

Μια συνάρτηση μπορεί να δέχεται ως όρισμα πίνακα δομών, όπως και οποιοδήποτε ένα ακέραιο, τον αριθμό των byte που θέλουμε να δεσμεύσουμε.

Εάν η δέσμευση χώρου είναι επιτυχής, επιστρέφει ένα δείκτη στην αρχή αυτού του χώρου.

Η τιμή επιστροφής της είναι δείκτης σε `void` και γι' αυτό χρειάζεται προσαρμογή τύπου.

Σε αδυναμία δέσμεσυης μνήμης επιστρέφει δείκτη ίσο με `NU:L`άλλο πίνακα. Για παράδειγμα η επικεφαλίδα μια συνάρτησης που είναι `void` και δέχεται ως παράμετρο ένα πίνακα δομών του τύπου `funds`, τον `pin`, θα είναι:

```
void example(struct funds *pin)
```

ή

```
void example(struct funds pin[])
```

Η κλήση της συνάρτησης θα είναι:

```
example(pin);
```

Μια συνάρτηση μπορεί να έχει τιμή επιστροφής δομή, όπως και οποιοδήποτε άλλο είδος δεδομένων. Για παράδειγμα η επικεφαλίδα μιας συνάρτησης που είναι `void` και δέχεται ως μια δομή του τύπου `funds`, την `str`, θα είναι:

```
struct funds reading (struct funds str)
```

Η κλήση της συνάρτησης θα είναι:

```
x = reading(str);
```

Εργαστήριο 4

Ο τελεστής `sizeof` δίνει το μέγεθος σε byte του τελεστέου που βρίσκεται στα δεξιά του. Ο τελεστής μπορεί να είναι ένας προσδιοριστής τύπου σε παρενθέσεις, όπως για παράδειγμα:

```
k = sizeof(float);
```

όπου ζητούμε από τον τελεστή `sizeof` το μέγεθος σε byte των δεδομένων του είδους float. Όταν αναφερόμαστε σε συγκεκριμένη μεταβλητή (και όχι γενικά σε είδος), της οποίας ζητούμε το μέγεθος σε byte, η μεταβλητή αυτή δεν τίθεται σε παρενθέσεις. Για παράδειγμα:

```
1 float fp;
2 int ak;
3 ak = sizeof fp;
```

4.1 Συναρτήσεις δυναμικής δέσμευσης μνήμης

4.1.1 calloc

Η συνάρτηση `calloc()` δεσμεύει μνήμη την ώρα της εκτέλεσης ενός προγράμματος.

Δέχεται ως ορίσματα δύο ακεραίους, τον αριθμό των στοιχείων μνήμης που θα δεσμευτούν, κάθως και το πλήθος των byte ανά στοιχείο μνήμης.

Εάν η δέσμευση χώρου είναι επιτυχής, μηδενίζει τα περιεχόμενα της μνήμης που δεσμεύει και επιστρέφει ένα δείκτη στην αρχή αυτού του χώρου.

Η τιμή επιστροφής της είναι δείκτης σε `void` και γι' αυτό χρει προσαρμογή του τύπου.

Σε αδυναμία δέσμεσυης μνήμης επιστρέφει δείκτη ίσο με `NULL\`.

4.1.2 malloc

Η συνάρτηση `malloc()` δεσμεύει μνήμη την ώρα της εκτέλεσης ενός προγράμματος.

Δέχεται ως όρισμα ένα ακέραιο, τον αριθμό των byte που θέλουμε να δεσμεύσουμε.

Εάν η δέσμευση χώρου είναι επιτυχής, επιστρέφει ένα δείκτη στην αρχή αυτού του χώρου.

Η τιμή επιστροφής της είναι δείκτης σε `void` και γι' αυτό χρειάζεται προσαρμογή τύπου. Σε αδυναμία δέσμευσης μνήμης επιστρέφει δείκτη ίσο με `NULL`.

4.1.3 realloc

Η συνάρτηση `realloc()` τροποποιεί την ποσότητα μνήμης που είχε προηγουμένως δεσμεύτεί από κλήση της `malloc()` ή της `calloc()`.

Δέχεται δύο ορίσματα:

1. Ένα δείκτη στη θέση μνήμης, της οποίας το μέγεθος θέλουμε να τροποιήσουμε (δείκτης τον οποίο έχει επιστρέψει μια κλήση της `malloc()` ή της `calloc()`).
2. Το πλήθος των byte που θα δεσμευτούν.

Αν τα byte που θα δεσμευτούν είναι λιγότερα από τα ήδη δεσμευμένα και υπάρχει ελεύθερος συνεχόμενος χώρος μετά τον ήδη δεσμευμένο, η `realloc()` δεσμεύει τον χώρο που χρειάζεται επιπλέον.

Αν τα byte που θα δεσμευτούν είναι περισσότερα από τα ήδη δεσμευμένα και δεν υπάρχει ελεύθερος συνεχόμενος χώρος μετά τον ήδη δεσμευμένο, η `realloc()` δεσμεύει χώρο σε νέα θέση, τα υπάρχονται δεδομένα αναγράφονται στη νέα θέση και η συνάρτηση επιστρέφει ένα δείκτη στην αρχ του νέου μπλοκ.

Σε αδυναμία δέσμευσης μνήμης επιστρέφει δείκτη ίσο με `NULL`.

Η τιμή επιστροφής της είναι δείκτης σε `void` και γι' αυτό δεν χρειάεται προσαρμογή τύπου.

4.2 Πίνακας δεικτών

Με την δήλωση:

```
char *pin[N];
```

δημιουργούμε ένα πίνακα δεικτών σε χαρακτήρες, δηλαδή ένα πίνακα σε κάθε θέση του οποίου υπάρχει ένας δείκτης σε χαρακτήρα. Άρα για παράδειγμα: το `pin[3]` είναι δείκτης σε χαρακτήρα, ενώ το `*pin[3]` είναι χαρακτήρας.

Η επικόλληση μιας συμβολοσειράς σε άλλη γίνεται με την χρήση της συνάρτησης `strcat()`.

Η `strcat()` δέχεται ως ορίσματα δύο δείκτες σε χαρακτήρα. Επικολλά την συμβολοσειρά που ξεκινάει από εκεί που δείχνει ο δεύτερος δείκτης, στο τέλος της συμβολοσειράς που δείχνει ο πρώτος δείκτης.

Έχει τιμή επιστροφής δείκτη σε χαρακτήρα, όσο το πρώτο της όρισμα.

Η αντιγραφή μιας συμβολοσειράς σε άλλη γίνεται με την χρήση της συνάρτησης `strcpy()`.

Η `strcpy()` δέχεται ως ορίσματα δύο δείκτες σε χαρακτήρα. Αντιγράφει την συμβολοσειρά που ξεκινάει από εκεί που δείχνει ο δεύτερος δείκτης, εκεί που δείχνει ο πρώτος δείκτης.

Έχει τιμή επιστροφής δείκτη σε χαρακτήρα, όσο το πρώτο της όρισμα.

Εργαστήριο 5

5.1 Στοίβα, υλοποίηση με πίνακα

Στην υλοποίηση στοίβας με πίνακα εξυπηρετεί η δήλωση του πίνακα ως εξωτερικής μεταβλητής, δεδομένου ότι μόνο δύο συναρτήσεις θα χειριστούν την στοίβα (ώθηση και ανάκληση).

Το σημείο εισόδου και εξόδου (head) στοιχείων στην στοίβα εξυπηρετεί επίσης να δηλωθεί ως εξωτερική μεταβλητή, για παράδειγμα ακέραια μεταβλητή που θα ισούται με θέση του πίνακα.

Πριν την τοποθέτηση στοιχείου στην στοίβα πρέπει να γίνεται έλεγχος πλήρους στοίβας, ενώ πριν την ανάκληση στοιχείου πρέπει αν γίνεται έλεγχος κενής στοίβας.

Η υλοπο της στοίβας μπορεί να γίνει με δύο λογικές:

1. Εάν το head σημαίνει την θέση της στοίβας όπου θα τοποθετηθεί στοιχείο, τότε κατά την ώθηση τοποθετούμε στοιχείο στην στοίβα και αυξάνουμε το head, ενώ κατά την ανάκληση ελαττώνουμε πρώτα το head και μετά παίρνουμε το στοιχείο.

2. Εάν το head σημαίνει την θέση της στοίβας όπου έχει τοποθετηθεί το τελευταίο στοιχείο, τότε κατά την ώθηση αυξάνουμε το head και τοποθετούμετο στοιχείο στην στοίβα, ενώ κατά την ανάκληση παίρνουμε πρώτα το στοιχείο και μετά ελαττώνουμε το head.

Μια υλοποίηση στοίβας:

```
#include<stdio.h>

#define MAX 3

typedef struct
{
    int TOP;
    int ele[MAX];
}Stack;

void init(Stack *s)
{
    s->TOP = -1;
}

int isFull(Stack *s)
{
    if(s->TOP == MAX-1)
    {
        return 0;
    }
    return -1;
}

int isEmpty(Stack *s)
{
    if(s->TOP == -1)
    {
        return 0;
    }
    return -1;
}

void push(Stack *s, int item)
{
    if( !isFull(s) )
    {
        printf("\nStack is full");
        return;
    }
    s->TOP = s->TOP + 1;
    s->ele[s->TOP] = item;
}

int pop(Stack *s, int *item)
{
    if(!isEmpty(s))
    {
        printf("\nStack is empty");
        return -1;
    }
    *item = s->ele[s->TOP];
    s->TOP = s->TOP - 1;
    return 0;
}

int main()
{
    Stack s;
    int item;

    clrscr();

    init(&s);

    push(&s, 10);
    push(&s, 20);
    push(&s, 30);

    pop(&s, &item);
    printf("\nPoped Item : %d",item);

    pop(&s, &item);
    printf("\nPoped Item : %d",item);

    pop(&s, &item);
    printf("\nPoped Item : %d",item);
    getch();

    return 0;
}
```

Εργαστήριο 6

6.1 Απλά συνδεδεμένες λίστες (δημιουργία)

Η δομή που περιγραφεί κάθε κόμβο μιας απλά συνδεδεμένης λίστας περιέχει «χρήσιμα» δεδομένα και ένα δείκτη προς ίδιου τύπου δομές.

Τον χώρο των κόμβων που δεν Χρειαζόμαστε πρέπει να τον ελευθερώνουμε (συνάρτηση `free()`).

Στον πρώτο κόμβο (κεφαλή) μιας απλά συνδεδεμένης λίστας πρέπει να υπάρχει ένας δείκτης, ο οποίος να μη μετακινηθεί ποτέ από εκ. Μέσω αυτού μπορούμε να διατρέχουμε την λίστα.

Ο δείκτης του τελευταίου κόμβου της λίστας πρέπει να είναι `NULL`. Ελέγχοντας αν έχουμε φθάσει σε τιμή `NULL`, ξέτουμε αν φθάσαμε ή όχι στο τέλος της λίστας.

6.2 Λειτουργίες στις απλά συνδεδεμένες λίστες: αναζήτηση, εισαγωγή, διαγραφή, μετακίνηση, συνένωση λιστών

Χρήσιμες πηγές

C Cheatsheet Data structures runtime Stack, in Python Linked list, in Python