Ανάπτυξη και Σχεδίαση Λογισμικού

Η γλώσσα προγραμματισμού C

Γεώργιος Δημητρίου

Δυναμική Κατανομή Μνήμης

- Δυναμική εκχώρηση μνήμης
  - Σωρός
  - Συναρτήσεις malloc(), calloc(), realloc(), free()
- Δυναμικές δομές δεδομένων
  - Λίστες, δέντρα
  - Προσοχή στη διαρροή μνήμης!

Στατική Κατανομή Μνήμης

- Στατικά δεδομένα
  - Καθολικές μεταβλητές
  - Στατικές μεταβλητές συναρτήσεων
  - Στατικές σταθερές
- Δεδομένα στοίβας
  - Παράμετροι και τοπικές μεταβλητές
  - Βοηθητικά στοιχεία για κλήσεις συναρτήσεων

Δυναμική Κατανομή Μνήμης

- Η έννοια του σωρού
  - Περιοχή μνήμης που παρέχει σκόρπια στοιχεία μνήμης όταν ζητούνται
  - Εξοικονόμηση μνήμης, αφού δεν δεσμεύεται στατικός χώρος για κάποιο μέγιστο μέγεθος δεδομένων
- Διαχείριση του σωρού έναντι της στοίβας
  - Σαφώς πιο πολύπλοκη
  - Συνεργασία προγραμματιστή και συστήματος

Παράδειγμα Πίνακα

- Αντί της ακολουθίας:
  - Ορίζουμε κάποιο μέγιστο μέγεθος με οδηγία #define
  - Δηλώνουμε πίνακα αυτού του μεγέθους
  - Ζητάμε το πραγματικό μέγεθος
  - Χρησιμοποιούμε αυτό μόνο το μέγεθος
  - Το υπόλοιπο δεν χρησιμοποιείται, είναι όμως δεσμευμένο!

Παράδειγμα Πίνακα

- Εφαρμόζουμε την ακολουθία:
  - Δηλώνουμε ένα δείκτη στον τύπο στοιχείου του πίνακα
  - Ζητάμε το πραγματικό μέγεθος
  - Ζητάμε από το σύστημα να μας εκχωρήσει από το σωρό μνήμη πραγματικού μεγέθους
  - Αν αυτή είναι διαθέσιμη εκχωρείται στο δείκτη
  - Χρησιμοποιούμε το δείκτη ως πίνακα
  - Δεν υπάρχει μη χρησιμοποιούμενο υπόλοιπο!

Διαχείριση Σωρού

- Η βιβλιοθήκη stdlib παρέχει συναρτήσεις για τη διαχείριση του σωρού
  - Άρα το σύστημα κάνει τη διαχείριση στη δυναμική κατανομή μνήμης
- Εμείς όμως τις καλούμε!
  - Άρα εμείς έχουμε την ευθύνη για τη *σωστή* διαχείριση στη δυναμική κατανομή μνήμης

Συνάρτηση malloc()

- Με τη συνάρτηση αυτή ζητάμε την εκχώρηση μνήμης από το σωρό:

  void * malloc(size_t);

  - όπου size_t προκαθορισμένος τύπος, συνώνυμος του unsigned int, που χρησιμοποιείται για να εκφράσει πλήθος από bytes
- Η συνάρτηση επιστρέφει NULL αν δεν υπάρχει αρκετή μνήμη στο σωρό
  - Έλεγχος τιμής επιστροφής!

Συνάρτηση malloc()

- Σε επιτυχή επιστροφή, λαμβάνουμε μια διεύθυνση στο σωρό με το πλήθος των bytes που ζητήσαμε
  - Το σύστημα δε γνωρίζει τον τύπο δεδομένων για τον οποίο θα χρησιμοποιηθεί αυτός ο χώρος μνήμης, οπότε θα πρέπει εμείς να τον μετατρέψουμε!

```
p = (char*) malloc(100*sizeof(char));
```

για πίνακα 100 χαρακτήρων

Εδώ συνηθίζεται η χρήση του τελεστή sizeof για τον ακριβή υπολογισμό του πλήθους των bytes που θα χρειαστούμε!

Συνάρτηση free()

- Με τη συνάρτηση αυτή απελευθερώνουμε χώρο στο σωρό που δε χρειαζόμαστε άλλο:

```
void free(void *);
```

Εδώ δε χρειάζεται μετατροπή σε τύπο void* μια που κάθε δείκτης είναι συμβατός με τέτοιον τύπο

- Ίσως η πιο σημαντική συνάρτηση!
  - Με τη σωστή χρήση της free() αποφεύγουμε τη *διαρροή μνήμης*, δηλαδή τη συνεχή δέσμευση μέχρι εξαντλήσεως του σωρού!

Συνάρτηση calloc()

- Η συνάρτηση αυτή χρησιμοποιείται συχνά για δέσμευση μνήμης για πίνακες:

  void * calloc(size_t, size_t);

  - όπου η πρώτη παράμετρος είναι το πλήθος στοιχείων μεγέθους όση είναι η δεύτερη παράμετρος
- Επιπλέον της εκχώρησης, η μνήμη που επιστρέφεται αρχικοποιείται σε 0
  - Χρήσιμη αν επιθυμούμε το μηδενισμό

Συνάρτηση realloc()

- Με τη συνάρτηση αυτή τροποποιούμε τη δέσμευση που έχουμε κάνει νωρίτερα, κυρίως για αύξηση του δεσμευμένου μεγέθους:

  void * realloc(void *, size_t);

  - όπου η πρώτη παράμετρος είναι η προηγούμενη δεσμευμένη μνήμη
- Η διεύθυνση που επιστρέφεται – αν δεν είναι NULL – δίνει τη νέα δεσμευμένη μνήμη, όπου έχουν μεταφερθεί τα δεδομένα μας
  - Αν είναι NULL, δε συμβαίνει καμία αλλαγή

Δυναμικός Πίνακας Ακεραίων

```
int main() {
        int n, *pinakas;
        printf("Please type the number of integers: ");
        scanf("%d",&n);
        pinakas = (int *) malloc (n * sizeof(int));
        if (pinakas == NULL) {
                printf("No memory available!\n");
                exit(1);
        }
        // κώδικας επεξεργασίας του πίνακα ως int[n]
        …
        free(pinakas);
}
```

Δυναμικές Δομές Δεδομένων

- Λίστες
  - Απλά ή διπλά συνδεδεμένες
  - Τυπικά τύπου struct όπου τουλάχιστον ένα πεδίο είναι δείκτης σε ίδιο τύπο struct

```
struct lista {
        int a;
        struct lista * next;
};
```

Ταξινομημένη Λίστα Ακεραίων

```
struct lista * insert (struct lista * list, int n) {
        struct lista *oldlist;
        struct lista *newlink = (struct lista *) malloc (sizeof(struct lista));
        if (newlink == NULL) { printf("Sorry, no memory!\n"); exit(1); }
        newlink->a = n;
        if (list == NULL) { newlink->next = NULL; return newlink; }
        if (list->a >= n) { newlink->next = list; return newlink; }
        oldlist = list;
        while ((list->next != NULL) && (list->next->a < n))
                list = list->next;
        newlink->next = list->next;
        list->next = newlink;
        return oldlist;
}
```

Ταξινομημένη Λίστα Ακεραίων

```
void print_and_delete_list (struct lista * list) {
        struct lista * oldlist = list;
        while (list != NULL) {
                printf("%d\n", list->a);
                oldlist = list->next;
                free(list);
                list = oldlist;
        }
}
```

Ταξινομημένη Λίστα Ακεραίων

```
int main(int argc, char *argv[]) {
        FILE * intfile = fopen(argv[1], "r");
        struct lista * intlist = NULL;
        int x;
        if (!intfile) { printf("File does not exist!\n"); exit(1); }
        while (!feof(intfile)) {
                fscanf(intfile, "%d", &x);
                intlist = insert(intlist, x);
        }
        print_and_delete_list (intlist);
        fclose(intfile);
}
```

Άλλες Δυναμικές Δομές

- Παραπλήσιες με τη λίστα:
  - Δομή ουράς (FIFO)
  - Δομή στοίβας (LIFO)

- Δομές γραφημάτων:
  - Δέντρα
  - Γενικά γραφήματα

Τι Μάθαμε Σήμερα

- Δυναμική κατανομή μνήμης
- Σωρός και συναρτήσεις διαχείρισης σωρού
  - Δέσμευση και αποδέσμευση μνήμης
- Δυναμικοί πίνακες
- Δυναμικές δομές δεδομένων
  - Λίστες
  - Άλλες δυναμικές δομές