Προγραμματισμός Ι

Εργαστήριο 3

Διδάσκοντας: Κωνσταντίνος Τσερπές

Ακ. Έτος 2013-2014

1 Δυναμική Μνήμη

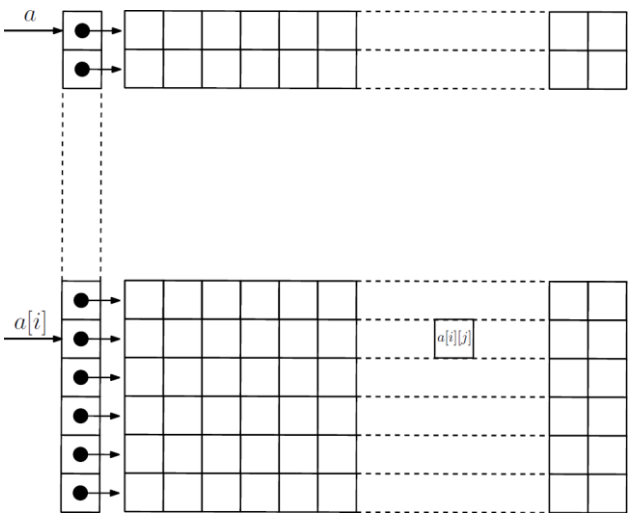
1.1 Εισαγωγή Πίνακα από τον Χρήστη

Γράψτε ένα πρόγραμμα που να ζητάει από τον χρήστη έναν ακέραιο `n>0` και στην συνέχεια `n` ακεραίους $x_0, x_1, \ldots, x_{n-1}$. Το πρόγραμμα σας πρέπει να δημιουργήσει ένα πίνακα μεγέθους `n` με την χρήση της συνάρτησης **`malloc`**, και να αποθηκεύσει τους αριθμούς αυτούς.

Στην συνέχεια τυπώστε τους αριθμούς σε ανάποδη σειρά από την σειρά εισαγωγής. Μην ξεχάσετε να ελέγξετε την επιτυχή κλήση της συνάρτησης **`malloc`** και να ελευθερώσετε την μνήμη με την συνάρτηση **`free`** κατά το τέλος του προγράμματος σας.

1.2 Δισδιάστατος Πίνακας

Γράψτε ένα πρόγραμμα που να ζητάει από τον χρήστη δύο ακεραίους `n`, `m > 0` και να φτιάχνει έναν δυναμικό πίνακα ακεραίων μεγέθους `n x m`. Στη συνέχεια ζητήστε από τον χρήστη `n*m` αριθμούς ώστε να γεμίσετε τον πίνακα. Τυπώστε τον πίνακα σε μορφή δισδιάστατου πίνακα στην οθόνη. Μην ξεχάσετε να ελευθερώσετε την δεσμευμένη μνήμη.



Η μορφή του δισδιάστατου πίνακα πρέπει να φτιαχτεί σε δύο φάσεις. Πρώτα πρέπει να φτιάξετε ένα μονοδιάστατο δυναμικό πίνακα `a` μεγέθους `n` ο οποίος να περιέχει `n` δείκτες ακεραίων.

Στην συνέχεια πρέπει για κάθε θέση `a[i]` όπου $0 \leq i \leq n-1$ να φτιάξετε ένα μονοδιάστατο πίνακα ακεραίων μεγέθους `m`. Δηλώστε λοιπόν ένα δείκτη σε δείκτη ακεραίων (διπλός δείκτης) ως

```
int **a;
```

και μέσω μιας κλήσης της **`malloc`** φτιάξτε ένα πίνακα δεικτών σε ακεραίους μεγέθους `n`. Στην συνέχεια κάντε μια επανάληψη και για κάθε θέση του πίνακα φτιάξτε ένα πίνακα ακεραίων μεγέθους `m`.

Μην ξεχάσετε στο τέλος του προγράμματος σας να ελευθερώσετε την μνήμη με την ανάποδη διαδικασία, καλώντας `free(a[i])` για κάθε $0 \leq i \leq n-1$ και μετά `free(a)`.

2 Πίνακας Απεριόριστου Μεγέθους

Στην άσκηση αυτή θα φτιάξουμε μια πρώτη δομή δεδομένων η οποία θα υλοποιεί ένα πίνακα ακεραίων απεριόριστου μεγέθους. Θα χρειαστούμε να δηλώσουμε μια δομή που θα περιέχει τις απαραίτητες μεταβλητές και μερικές συναρτήσεις οι οποίες θα διαχειρίζονται τον πίνακα.

```
typedef struct
{
      int size;
      int *a;
} array;
```

Η παραπάνω δομή περιέχει ένα δείκτη που θα δείχνει στις πραγματικές θέσεις μνήμης όπου θα αποθηκεύουμε τα στοιχεία και μια μεταβλητή `size` όπου θα κρατάμε το μέγεθος του πίνακα. Κρατώντας το μέγεθος του πίνακα μπορούμε να αυξομειώνουμε το μέγεθος ανάλογα με τις θέσεις που χρειάζεται ο χρήστης.

Οποιαδήποτε αλλαγή στον πίνακα θα γίνεται **μόνο** μέσω των παρακάτω συναρτήσεων.

- `array* unbounded_create();`

Η συνάρτηση αυτή πρέπει να φτιάχνει δυναμικά μια δομή τύπου `array`, και να βάζει τον δείκτη `a` να δείχνει σε ένα δυναμικό πίνακα μεγέθους `64`.

- `void unbounded_destroy( array* a );`

Η συνάρτηση αυτή πρέπει να ελευθερώνει την μνήμη του πίνακα `a->a` και της δομής.

- `void unbounded_double( array* a );`

Η συνάρτηση πρέπει να διπλασιάζει το μέγεθος του πίνακα `a->a`.

- `void unbounded_set( array* a, int i, int value );`

Η συνάρτηση πρέπει να βάζει στη θέση `i` του πίνακα `a->a` την τιμή `value`. Σε περίπτωση που ο πίνακας δεν είναι αρκετά μεγάλος πρέπει να τον διπλασιάσει (όσες φορές είναι απαραίτητο) ώστε να γίνει αρκετά μεγάλος.

- `int unbounded_get( array* a, int i );`

Η συνάρτηση επιστρέφει τον ακέραιο που βρίσκεται στην θέση `i` του πίνακα `a->a`. Σε περίπτωση που ο πίνακας δεν είναι αρκετά μεγάλος πρέπει να τον διπλασιάσει (όσες φορές είναι απαραίτητο) ώστε να γίνει αρκετά μεγάλος.

2.1 Κυρίως Πρόγραμμα

Ακολουθεί ένα μικρό κυρίως πρόγραμμα ώστε να ελέγξετε την υλοποίηση σας.

```
int main()
{
      int i;
      array *A = unbounded_create();
      for( i = 0; i < 1000; i++ )
      unbounded_set( A, i, i );
      for( i = 0; i < 1000; i++ )
      printf("%d ", unbounded_get( A, i ) );
      printf("\n");
      unbounded_destroy( A );
}
```