Δυναμική δέσμευση μνήμης

#10

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων (Άρτα)
Γκόγκος Χρήστος

Δέσμευση μνήμης

- Υπάρχουν δύο τύποι δέσμευσης μνήμης
  - **Στατική δέσμευση μνήμης** - πραγματοποιείται αυτόματα από το μεταγλωττιστή (υπονοούμενη=implicitly).
    - **καθολικές μεταβλητές**: η μνήμη δεσμεύεται στην αρχή του προγράμματος και απελευθερώνεται όταν το πρόγραμμα τερματίζει, συνεπώς οι καθολικές μεταβλητές είναι διαθέσιμες σε όλη τη διάρκεια εκτέλεσης του προγράμματος.
      - Μπορούν να προσπελαστούν από οποιοδήποτε σημείο του προγράμματος.
    - **τοπικές μεταβλητές** (δηλώνονται σε μια συνάρτηση): η μνήμη δεσμεύεται όταν η συνάρτηση ξεκινά και ελευθερώνεται όταν η συνάρτηση επιστρέφει.
      - Μια τοπική μεταβλητή δεν μπορεί να προσπελαστεί από μια άλλη συνάρτηση.
    - Η δέσμευση και η αποδέσμευση μνήμης γίνεται αυτόματα (υπονοούμενα).
    - Η μη αναγκαιότητα διαχείρισης της μνήμη είναι βολική αλλά έχει περιορισμούς:
      - Για παράδειγμα στη στατική δέσμευση μνήμης το μέγεθος των πινάκων πρέπει να είναι καθορισμένο κατά τη δήλωση τους.

Δέσμευση μνήμης

- Ο δεύτερος τύπος δέσμευσης μνήμης είναι η δυναμική δέσμευση.
  - Δυναμική δέσμευση μνήμης - πραγματοποιείται από τον προγραμματιστή (σαφής=explicitly).
    - Ο προγραμματιστής με σαφείς εντολές ζητά από το σύστημα να δεσμεύσει μνήμη και να επιστρέψει την αρχική διεύθυνση από τη μνήμη που έχει δεσμευθεί. Η διεύθυνση αυτή μπορεί να χρησιμοποιηθεί από τον προγραμματιστή για να προσπελάσει τη μνήμη που έχει δεσμευθεί.
    - Όταν πλέον δεν χρειαζόμαστε τη μνήμη που έχει δεσμευθεί, η μνήμη θα πρέπει να ελευθερωθεί με σαφείς εντολές που θα δώσει ο προγραμματιστής.

Δέσμευση μνήμης: ο τελεστής **new**

- Ο τελεστής new χρησιμοποιείται για να δεσμεύσει δυναμικά μνήμη.
- Μπορεί να χρησιμοποιηθεί για να δεσμεύσει μια απλή μεταβλητή ή έναν πίνακα μεταβλητών.
- Ο τελεστής new επιστρέφει έναν δείκτη προς το τύπο δεδομένων που έχει δεσμευθεί. Παραδείγματα:

```
char *my_char_ptr = new char;
int *my_int_array = new int[20];
Fraction *f = new Fraction(1,2);
```

- Πριν την ανάθεση τιμής ο δείκτης μπορεί να δείχνει ή να μην δείχνει σε έγκυρη θέση μνήμης.
- Μετά την ανάθεση τιμής ο δείκτης δείχνει σε έγκυρη θέση μνήμης.

https://github.com/chgogos/oop/blob/master/various/COP3330/lect10/sample1.cpp

https://github.com/chgogos/oop/blob/master/various/COP3330/lect10/sample2.cpp

Απελευθέρωση μνήμης: ο τελεστής `delete`

- Ο τελεστής `delete` χρησιμοποιείται για να ελευθερώσει μνήμη που έχει δεσμευθεί με τον τελεστή `new`.
- Ο τελεστής `delete` θα πρέπει να καλείται σε έναν δείκτη προς μια δεσμευμένη μνήμη όταν η μνήμη αυτή πλέον δεν χρειάζεται.
- Μπορεί να διαγράψει μια απλή μεταβλητή ή έναν πίνακα:

```
delete pointerName;
delete [] arrayName;
```

- Από τη στιγμή που έχει κληθεί η delete για μια περιοχή μνήμης, δεν θα πρέπει να γίνονται πλέον προσπελάσεις σε αυτή τη περιοχή μνήμης.
- Η σύμβαση είναι να θέτουμε το δείκτη προς μια μνήμη που έχει διαγραφεί σε `NULL`.
  - Κάθε `new` θα πρέπει να έχει το αντίστοιχο `delete` (αλλιώς το πρόγραμμα έχει διαρροή μνήμης).
  - Το `new` και το `delete` μπορεί να μην βρίσκονται στην ίδια συνάρτηση.

https://github.com/chgogos/oop/blob/master/various/COP3330/lect10/sample3.cpp
https://github.com/chgogos/oop/blob/master/various/COP3330/lect10/sample4.cpp

Ο σωρός (heap)

- Ο σωρός είναι μια μεγάλη περιοχή μνήμης ελεγχόμενη από το σύστημα χρόνου εκτέλεσης που είναι υπεύθυνο να εξυπηρετεί αιτήματα δέσμευσης και αποδέσμευσης δυναμικής μνήμης.
- Είναι δυνατόν να δεσμευθεί δυναμικά μνήμη και στη συνέχει να «χαθεί» ο δείκτης προς τη μνήμη αυτή. Σε αυτή την περίπτωση έχουμε διαρροή μνήμης.
- Οι διαρροές μνήμες μπορεί να προκαλέσουν την εξάντληση του χώρου του σωρού.
- Αν γίνει απόπειρα να δεσμευθεί μνήμη από το σωρό και δεν υπάρχει αρκετή μνήμη, τότε παράγεται εξαίρεση (exception) που υποδηλώνει το λάθος.

https://github.com/chgogos/oop/blob/master/various/COP3330/lect10/sample5.cpp

Γιατί να χρησιμοποιηθεί δυναμική δέσμευση μνήμης;

- Επιτρέπει στα δεδομένα (ειδικά στους πίνακες) να λάβουν μεταβλητά μεγέθη (π.χ. να γίνεται ερώτηση προς το χρήστη για τον αριθμό των ακεραίων που επιθυμεί να αποθηκεύσει, και στη συνέχεια να δημιουργεί έναν πίνακα ακεραίων με αυτό ακριβώς το μέγεθος).
- Επιτρέπει σε μεταβλητές που έχουν δημιουργηθεί τοπικά να εξακολουθούν να υπάρχουν και μετά την ολοκλήρωση της συνάρτησης.
- Επιτρέπει τη δημιουργία πολλών δομών δεδομένων που χρησιμοποιούνται στις Δομές Δεδομένων και στους Αλγορίθμους.

https://github.com/chgogos/oop/blob/master/various/COP3330/lect10/sample6.cpp

https://github.com/chgogos/oop/blob/master/various/COP3330/lect10/sample7.cpp

Οι τελεστές . και ->

- Ο τελεστής . χρησιμοποιείται για να προσπελάσουμε τα μέλη ενός αντικειμένου.

```
f1.Evaluate();
f1.num=5;
```

- Πως γίνεται η πρόσβαση στα μέλη ενός αντικειμένου αν διαθέτουμε μόνο έναν δείκτη προς αυτό;
  - Αν έχουμε έναν δείκτη `f_ptr = &f;`
    - θα μπορούσαμε να χρησιμοποιήσουμε: `(*f_ptr).Evaluate();`
    - Ωστόσο, υπάρχει ένας συντομότερος τρόπος: `f_ptr->Evaluate();`
    - Οι δύο παραπάνω τρόποι πρόσβασης στα μέλη ενός αντικειμένου είναι ισοδύναμοι.

https://github.com/chgogos/oop/blob/master/various/COP3330/lect10/sample8.cpp

Δέσμευση και αποδέσμευση μνήμης στη C (λειτουργεί επίσης και στη C++)

- Οι συναρτήσεις `malloc` και `free`
  - Οι συναρτήσεις `malloc` και `free` ορίζονται στο `<stdlib.h>`
  - Η `malloc` είναι παρόμοια με την new, αλλά θα πρέπει να καθοριστεί το ακριβές μέγεθος μνήμης.
    - Επιστρέφει `void*` (πρέπει να μετατραπεί στον κατάλληλο τύπο δείκτη)
  - Η `free` είναι ισοδύναμη με την `delete` (ωστόσο, δεν υπάρχει διαφοροποίηση ανάμεσα στην αποδέσμευση μνήμης για απλή μεταβλητή και για πίνακα)

Ερωτήσεις σύνοψης

- Σε ποια περιοχή της μνήμης επενεργεί η στατική δέσμευση και σε ποια η δυναμική δέσμευση μνήμης;
- Πότε θα πρέπει να χρησιμοποιούμε δυναμική δέσμευση μνήμης;
- Πώς γίνεται η δέσμευση και η αποδέσμευση μνήμης για ένα απλό στοιχείο δεδομένων και πως για έναν πίνακα;
- Πότε παρατηρείται διαρροή μνήμης;
- Ποια είναι η αντίστοιχη εντολή της new με την οποία στη C γίνεται δέσμευση μνήμης;
- Ποια είναι η αντίστοιχη εντολή της delete με την οποία στη C γίνεται αποδέσμευση μνήμης;
- Με τι μπορεί να αντικατασταθεί ο τελεστής -> ;

Αναφορές

- http://www.cs.fsu.edu/~xyuan/cop3330/