11η Φροντιστηριακή Άσκηση στη C++: Διαχείριση Μνήμης και Δομές Δεδομένων

Άσκηση 1: Δυναμική Δέσμευση Μνήμης

Εκφώνηση:
Γράψτε ένα πρόγραμμα που δεσμεύει δυναμικά μνήμη για έναν ακέραιο αριθμό, ζητά τιμή από τον χρήστη, την αποθηκεύει και την εμφανίζει. Στη συνέχεια, αποδεσμεύστε τη μνήμη.

Λύση:

```
#include <iostream>
using namespace std;

int main() {
    int* ptr = new int;
    cout << "Εισάγετε έναν αριθμό: ";
    cin >> *ptr;
    cout << "Αποθηκευμένος αριθμός: " << *ptr << endl;
    delete ptr;
    return 0;
}
```

Άσκηση 2: Δυναμικός Πίνακας

Εκφώνηση:
Γράψτε ένα πρόγραμμα που δεσμεύει δυναμικά μνήμη για έναν πίνακα ακεραίων, δέχεται τιμές από τον χρήστη, τις εμφανίζει και αποδεσμεύει τη μνήμη.

Λύση:

```
#include <iostream>
using namespace std;

int main() {
    int size;
    cout << "Εισάγετε το μέγεθος του πίνακα: ";
    cin >> size;
    int* arr = new int[size];

    cout << "Εισάγετε " << size << " αριθμούς: ";
    for (int i = 0; i < size; i++) {
        cin >> arr[i];
    }

    cout << "Τα στοιχεία του πίνακα είναι: ";
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
        delete[] arr;
    return 0;
}
```

Άσκηση 3: Μονόσυνδετη Λίστα

Εκφώνηση:
Υλοποιήστε μια απλή μονόσυνδετη λίστα που επιτρέπει την εισαγωγή και εμφάνιση κόμβων.

Λύση:

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
    Node(int val) : data(val), next(nullptr) {}
};

void printList(Node* head) {
    while (head) {
        cout << head->data << " -> ";
        head = head->next;
    }
    cout << "NULL" << endl;
}

int main() {
    Node* head = new Node(10);
    head->next = new Node(20);
    head->next->next = new Node(30);

    printList(head);

    delete head->next->next;
    delete head->next;
    delete head;
    return 0;
}
```

Άσκηση 4: Στοίβα με Δυναμική Μνήμη

Εκφώνηση:
Υλοποιήστε μια στοίβα (Stack) χρησιμοποιώντας δυναμική μνήμη και λειτουργίες `push` και `pop`.

Λύση:

```
#include <iostream>
using namespace std;

class Stack {
private:
    int* arr;
    int top;
    int capacity;
public:
    Stack(int size) : capacity(size), top(-1) { arr = new int[size];
}
    ~Stack() { delete[] arr; }
    void push(int x) {
        if (top == capacity - 1) {
            cout << "Η στοίβα είναι γεμάτη!" << endl;
            return;
        }
        arr[++top] = x;
    }
    void pop() {
        if (top == -1) {
            cout << "Η στοίβα είναι άδεια!" << endl;
            return;
        }
        top--;
    }
    void display() {
        for (int i = top; i >= 0; i--) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
};

int main() {
    Stack s(5);
    s.push(10);
    s.push(20);
    s.push(30);
    s.display();
    s.pop();
    s.display();
    return 0;
}
```