

// Program reads integers and stores them in a linked list and prints them.

```
#include <iostream>
#include <cstdint> // to access NULL
#include <fstream>
#include <cassert>
using namespace std;
typedef int ItemType;
struct NodeType; // forward declaration
typedef NodeType* NodePtr;
struct NodeType
{
    ItemType data;
    NodePtr next;
};
void GetList(ifstream &, NodePtr &);
void PrintList(NodePtr);
void Delete(NodePtr &, ItemType);
int main ()
{
    ItemType delItem;
    NodePtr head = NULL; // head of the list
    ifstream myIn;

    myIn.open("int.dat");
    assert(myIn);

    GetList(myIn, head); // Build the list from values in data file
    PrintList(head); // Display the values in the list

    cout << "Enter a value to delete: ";
    cin >> delItem;
    Delete(head, delItem);
    PrintList(head); // print the list again after the delete operation
    return 0;
}

//*****
// This function reads integer values from the data file and build a list to store these values
void GetList(ifstream& myIn, NodePtr & head) {
    ItemType tempValue;
    NodePtr currentNodePtr; // extra pointer
    NodePtr newNodePtr;

    head = NULL; // list is empty to start with
    // keep building the list til the end of the data file is reached
    // With this code, the new values are always added at the end of the list
    while (myIn>>tempValue) // File is not empty. First value is read successfully
    {
        // Generate and set up a new node for the value just read.
        newNodePtr = new NodeType;
```

```

        newNodePtr -> data = tempValue;
        newNodePtr -> next = NULL;
        if (head == NULL) { // empty list case
            head = newNodePtr;
            currentNodePtr = head; // get ready to build the rest of the list
        }
        else { // the list is not empty
            currentNodePtr->next = newNodePtr;
            currentNodePtr = currentNodePtr -> next;
        }
    }
}
// *****
// *****
// This function displays the values in a linked list
void PrintList(NodePtr head) {
    NodePtr currentNodePtr; // extra pointer
    // Start from the first node and go down the list node by node
    // For each node, print out its value

    currentNodePtr = head;
    while (currentNodePtr != NULL)
    {
        cout << currentNodePtr -> data << endl;
        currentNodePtr = currentNodePtr -> next;
    }
}
// *****
// This function deletes an data from a list
void Delete(NodePtr & head, ItemType data)
{
    NodePtr currPtr, prevPtr; // need two extra pointers
    prevPtr=NULL;
    currPtr=head;
    // search for the data in the list
    while (currPtr != NULL && currPtr->data != data) {
        prevPtr = currPtr;
        currPtr = currPtr->next;
    }
    if (head!=NULL && head->data == data) { // case 1: delete from the beginning of the list
        head = head->next;
    }
    else if (currPtr != NULL) { // found the data in the list (middle or end)
        prevPtr->next = currPtr->next;
    }
    delete currPtr;
    currPtr = NULL;
}
}

```