Programming with Indirect Pointers

Tibor Ásványi
Faculty of Informatics, Eötvös Loránd Univ.
Budapest, Pázmány Péter sétány 1/c, H-1117
asvanyi@inf.elte.hu

Highly dynamic data collections are often represented by linked data structures, that is, linked lists, trees etc. Inserting into, and deleting from lists and trees are frequent actions while performing search-and-update operations on these data structures. In this paper we consider some non-recursive variants of these operations, and develop effective and simple implementations of them.

In order to understand the problem, let us suppose, that we have a simple linked list, and we want to insert a new node into the list, or we want to remove one of the nodes of the list. In both cases we need one program code to perform the update at the beginning of the list, and another for the general case. Similarly, updates at the root of a tree are special, compared to the updates at deeper levels.

In order to symplify the list operations, and to make them a bit more effective, traditionally sentinel nodes are proposed.[1] However, sentinel nodes need extra memory. This memory overhead may be significant, if we have many short lists in our application. (For example, consider hash tables, linked representation of sparse matrices and graphs.)

This paper shows that the sentinel nodes of the one-way lists are usually unnecessary. We get rid of the special cases using indirect pointers, that is pointers referring to other pointers. And our method is often useful even for linked trees. (We have to mention that indirect pointers are supported by many widely used, high level programming languages. Consider C, C++, Pascal, Ada, Modula, and so on.)

[1] `http://en.wikipedia.org/wiki/Linked_list`