CSE101-Lec#35

Self Referential Structures

Created By:
Amanpreet Kaur &
Sanjeev Kumar
SME (CSE) LPU

Introduction

- Dynamic data structures
  - Data structures that grow and shrink during execution
- Linked lists
  - Allow insertions and removals anywhere

Self-Referential Structure

- A structure which contains a pointer to itself is called a self-referential structure.
- A structure cannot contain instance of itself.
- A self referential structure is used to create data structures like linked lists, stacks, etc. Following is an example of this kind of structure:

```
struct struct_name
    {
        datatype datatypename;
        struct_name * pointer_name;
    };
```

Self-Referential Structures

- Self-referential structures

```
struct node {
   int data;
   struct node *nextPtr;
}
```

- `nextPtr`
  - Points to an object of type node
  - Referred to as a link
    - Ties one node to another **`node`**

- A variable of type struct car cannot be declared in the definition of struct car.
- However, pointer to car may be included.

```
Struct car{
   char name[20];
   struct car newCar;   // wrong
   struct car *newCarPtr;   // right
};
```

Linked Lists

- Linked list
  - Linear collection of self-referential class objects, called nodes
  - Connected by pointer links
  - Accessed via a pointer to the first node of the list
  - Subsequent nodes are accessed via the link-pointer member of the current node
  - Link pointer in the last node is set to NULL to mark the list's end
- Use a linked list instead of an array when
  - You have an unpredictable number of data elements
  - Your list needs to be sorted quickly

Linked Lists

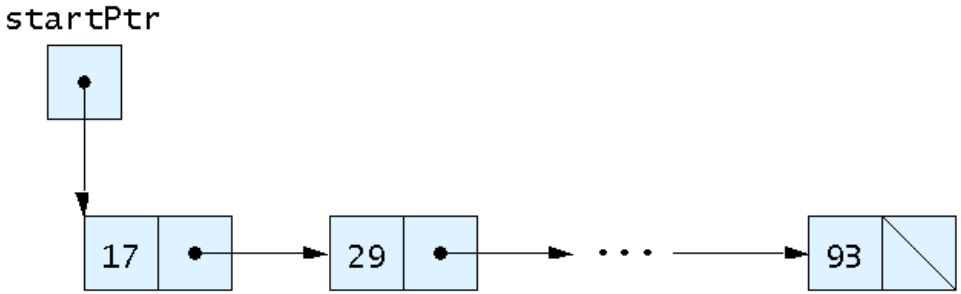

Fig. 12.2 A graphical representation of a linked list.

Linked Lists

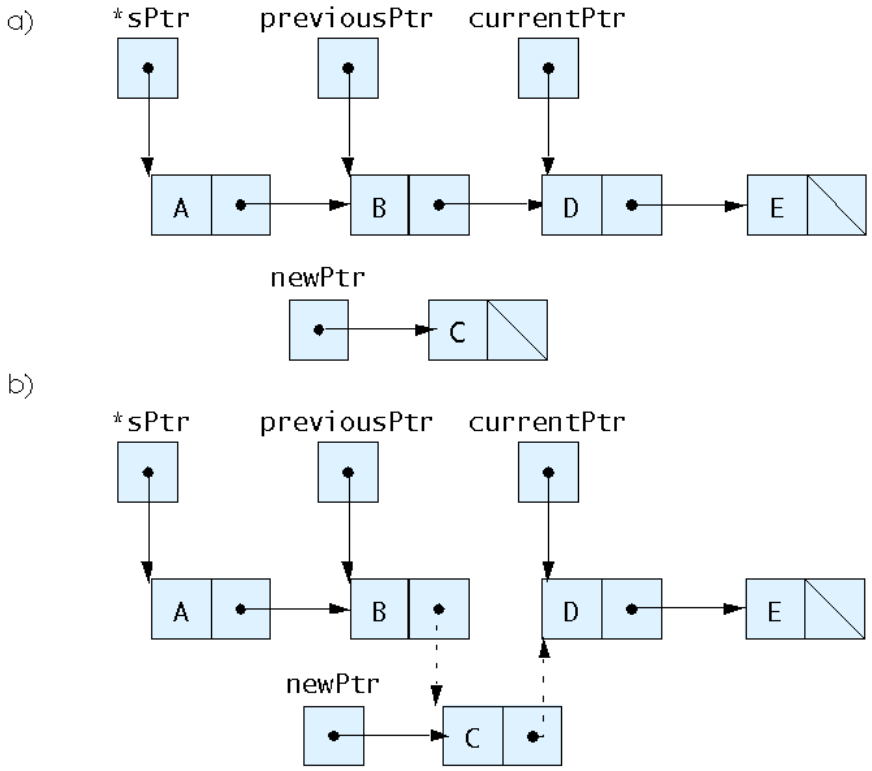

Fig. 12.5 Inserting a node in order in a list.

Link List

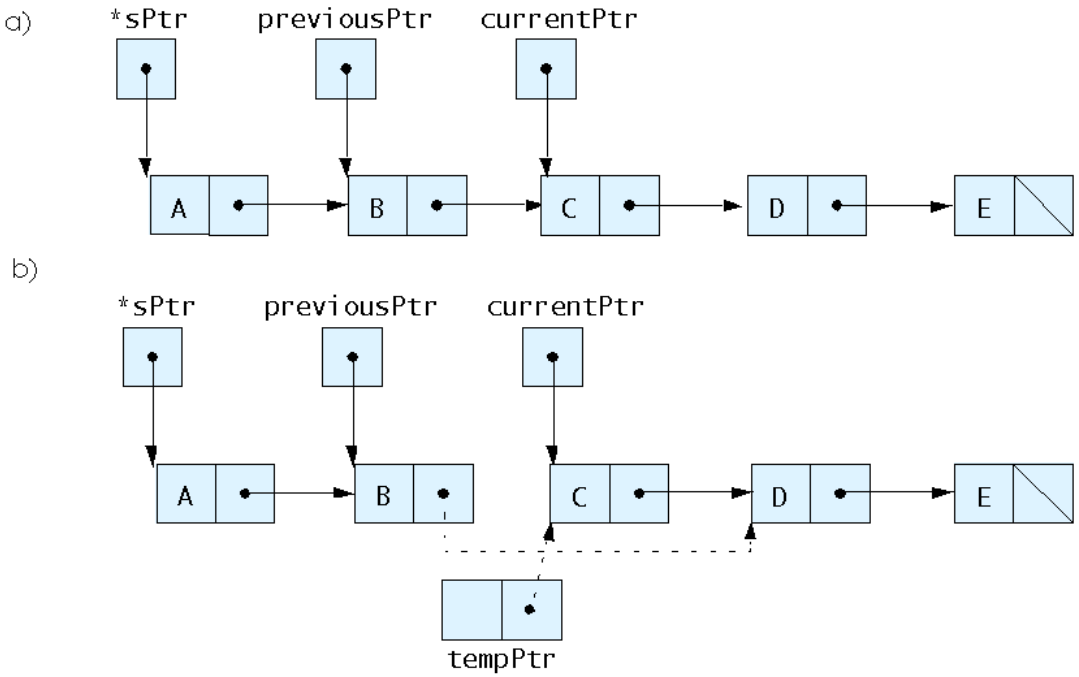

Fig. 12.6 Deleting a node from a list.

Program of insertion at beginning in the link list

```
#include <stdio.h>
struct link                 //one element of list
   {
   int data;               //data item
   link* next;             //pointer to next link
   };
void additem(int d);  //add one link
void display();  //display all links

void additem(int d)    //add data item
   {
   link *newlink;
   newlink->data = d;
   newlink->next = first;
   first = newlink;
   }
```

Program of insertion at beginning in the link list

```
Void display()
    {
    link* current = first; //set ptr to first
link
    while( current != NULL ) //quit on last link
        {
        printf("%d\n", current->data);
        current = current->next;//move to next
link
        }
    }
main()
    {
    link li;          //make linked list

    additem(25);      //add four items to list
    additem(36);
    additem(49);
    additem(64);

    display();        //display entire list
    return 0;
    }
```

Next Class: Nested Structure Union

cse101@lpu.co.in