Intermediate C Programming

Lesson 10

List

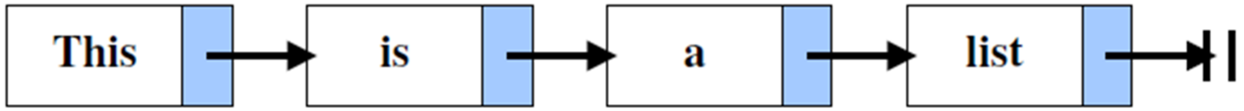
Today’s outline

- List
- Exercise

List

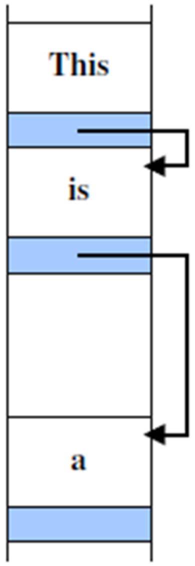
- What is the list?
- Fundamental data structure
- Linked list: a dynamic data structure which is able to increase or decrease its length at run time
- Be able to control any of the data
  - Stack: only can add/remove the top
  - Queue: read from the head and add to the tail

List



Each element has

1) its value
2) its next pointer is assigned the address of the next.



List

Structure is used to control

1) its value
2) its next pointer is assigned the address of the next item.

```
struct wordlist {
     char word [MAXSTRING];
     struct wordlist * next;
 };
```

List

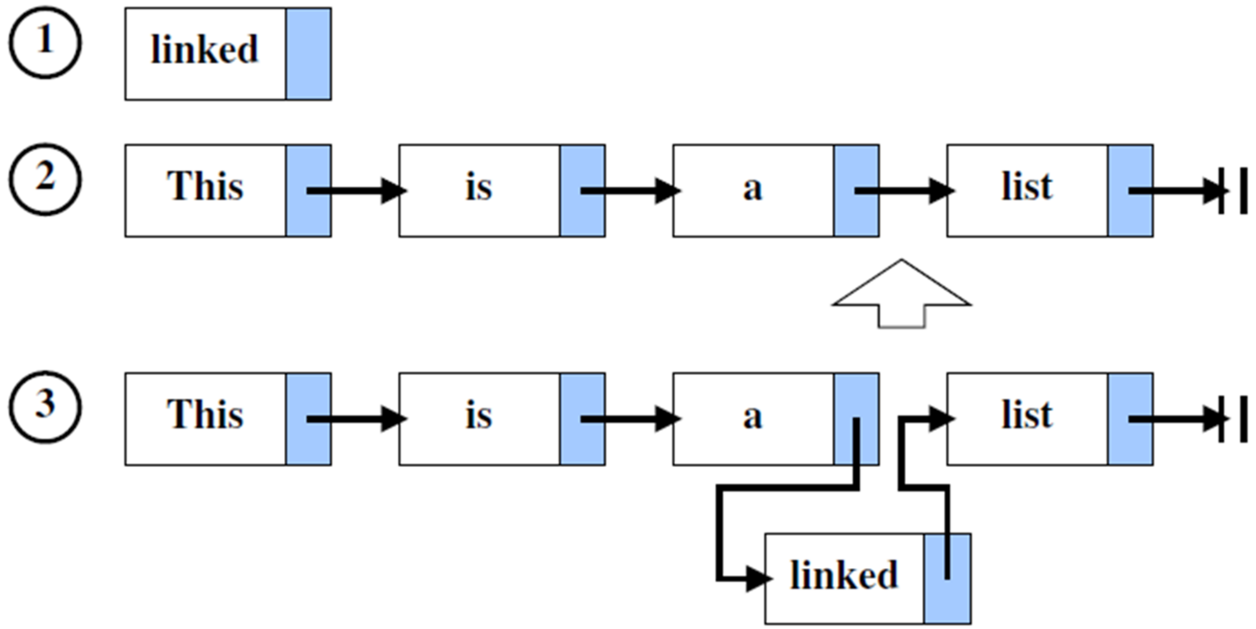
To append an element,

```
elem = (struct wordlist*)malloc( sizeof(struct wordlist) );
              strncpy( elem->word, input, MAXSTRING );
              elem->next = NULL;
/* check the list is empty or not*/
             if( head == NULL ){
                  head = elem;
              } else {
                  for( tmp=head; tmp->next!=NULL; tmp=tmp->next );
                  tmp->next = elem;
              }
```

List

To insert a new element (linked),

1) Make the new element
2) Decide the position to insert
3) Link



List

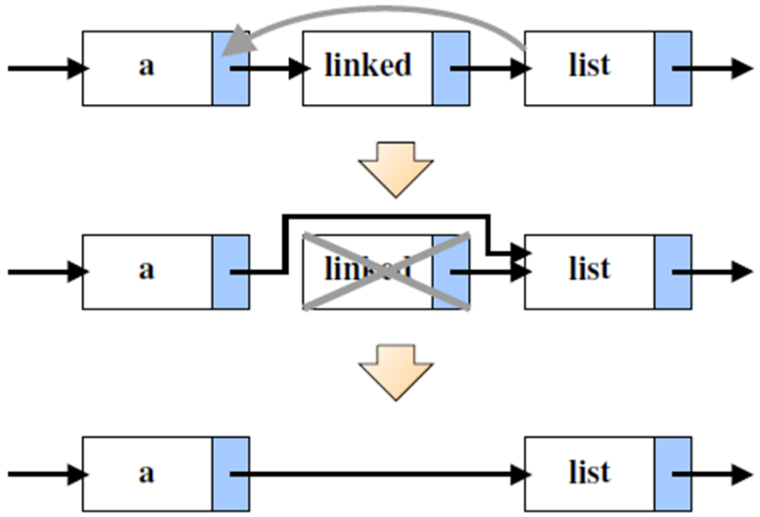
To insert a new element (linked),

```
elem = (struct wordlist*)malloc( sizeof(struct wordlist) );
        strncpy( elem->word, input, MAXSTRING );
        elem->next = NULL;
        /* check the list is empty or not*/
        if( pos == 0 ){
            elem->next = head;
            head = elem;
        } else {
            tmp = head;
            for( i=1; i<pos; i++ ) tmp=tmp->next;
            elem->next = tmp->next;
            tmp->next = elem;
        }
```

■List

To remove an element,

1) Move to the element to be removed
2) Link
3) Remove



List

To remove an element,

```
if( pos == 0 ){
            tmp = head->next;
            free( head );
            head = tmp;
        } else {
            tmp = head;
            for( i=1; i<pos; i++ ) tmp=tmp->next;
            remove = tmp->next;
            tmp->next = remove->next;
            free( remove );
        }
```

List

```
#include <stdio.h>
  #include <string.h>
  #define MAXSTRING 20
  struct wordlist {
      char word [MAXSTRING];
      struct wordlist * next;
  };

int main(){
       char input[MAXSTRING];
      int i, op, pos;
      struct wordlist *elem, *tmp, *head=NULL, *remove;

    return 0;
}
```

Exercise

Finish the program of page11

```
$ ./a.out
Select 1:append 2:insert 3:remove 4:quit>> 1
word>> This
This
Select 1:append 2:insert 3:remove 4:quit>> 1
word>> is
This is
Select 1:append 2:insert 3:remove 4:quit>> 1
word>> a
This is a
Select 1:append 2:insert 3:remove 4:quit>> 1
word>> list
This is a list
Select 1:append 2:insert 3:remove 4:quit>> 2
where[head=0]?>> 3
word>> linked
This is a linked list
Select 1:append 2:insert 3:remove 4:quit>> 3
where[head=0]?>> 0
is a linked list
Select 1:append 2:insert 3:remove 4:quit>> 2
where[head=0]?>> 0
word>> It
It is a linked list
Select 1:append 2:insert 3:remove 4:quit>> 4
$
```