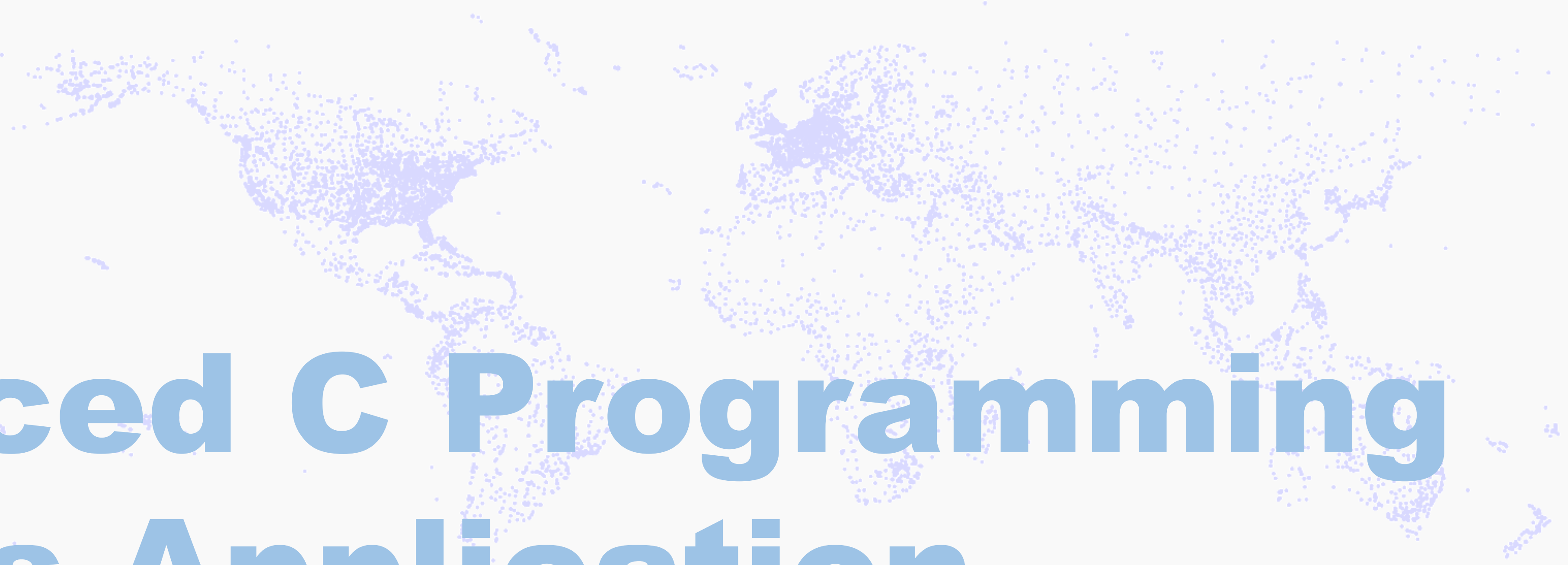


Advanced C Programming And It’s Application

Linked List Part. II

Assistant Prof. Chan, Chun-Hsiang

Department of Artificial Intelligence, Tamkang University

May. 24, 2022

<Outline/>

大綱

Part I.

[1] Concept

[2] Define a linked list

[3] Search

[4] Insert

[5] Delete

[6] Add with DMA

Part II.

[7] Stack

[8] Push

[9] Pop

[10] Release

[11] Insertion in Order

[12] Delete

[13] Assignments

[14] References

</Outline>

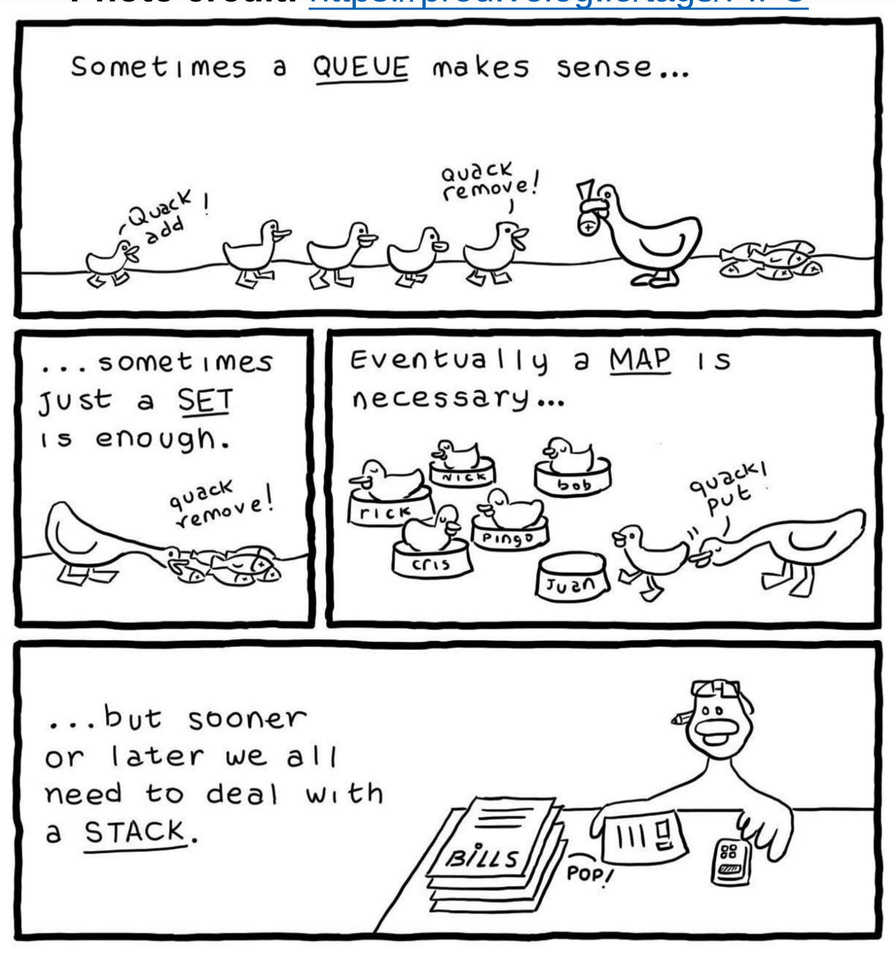
Linked List Applications

Linked list 應用最多莫過於你們之後的課程 – 資料結構與演算法。在資料結構的部分，你們應該會學到 stack, queue, set, map等。不過在這堂課之中，我們只教到堆疊(Stack)與佇列(Queue)。

https://bit.ly/33FzVRu

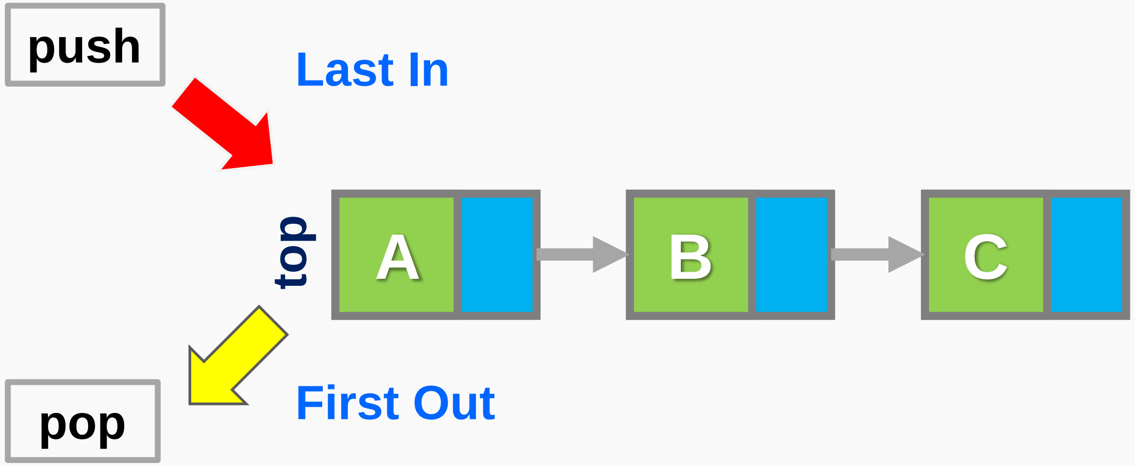
https://nyti.ms/3qwemf3

Photo credit: https://prod.velog.io/tags/FIFO

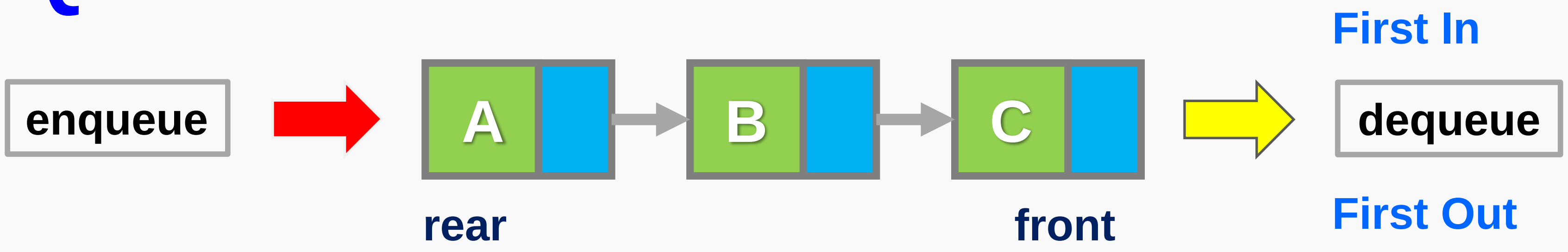

Sometimes a QUEUE makes sense...
Quack! add
quack remove!
...sometimes just a SET is enough.
quack remove!
Eventually a MAP is necessary...
nick
bob
rick
pingo
cris
Juan
quack! put
...but sooner or later we all need to deal with a STACK.
BILLS
POP!

Stack and Queue

Stack



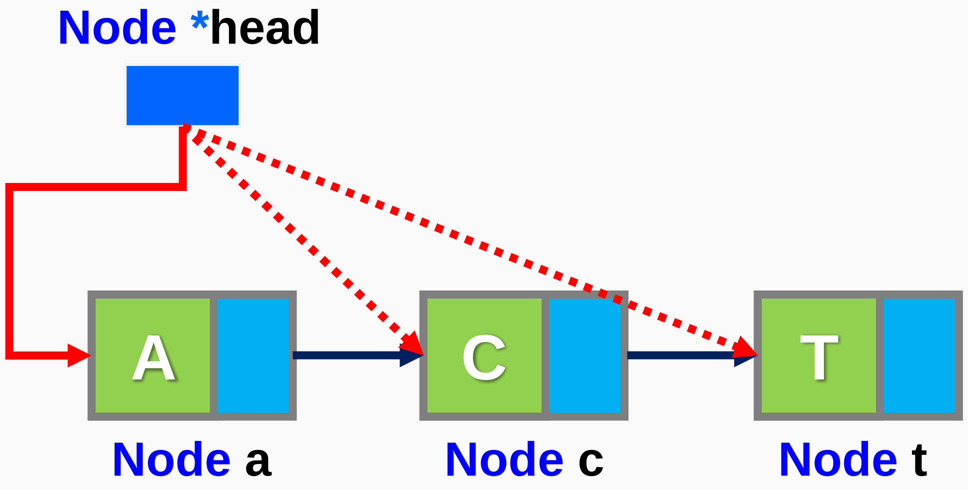
Queue



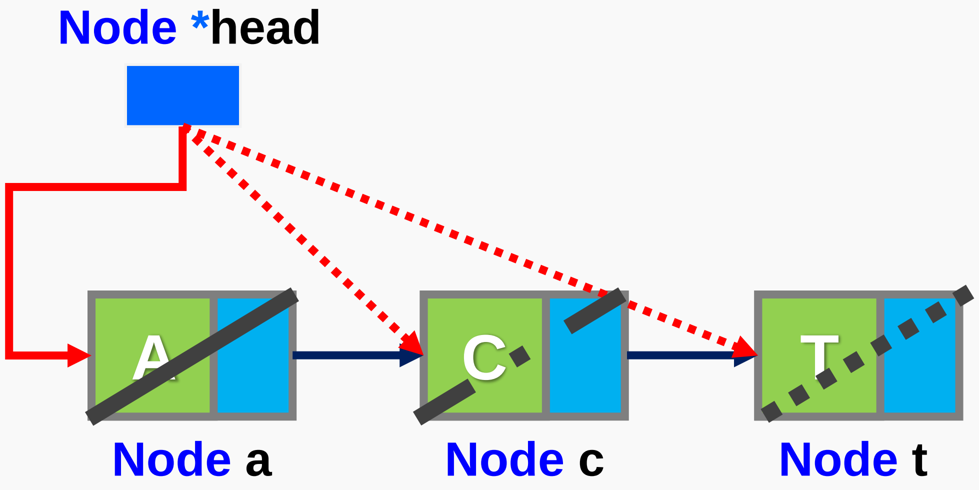
2022/05/24

<push/>

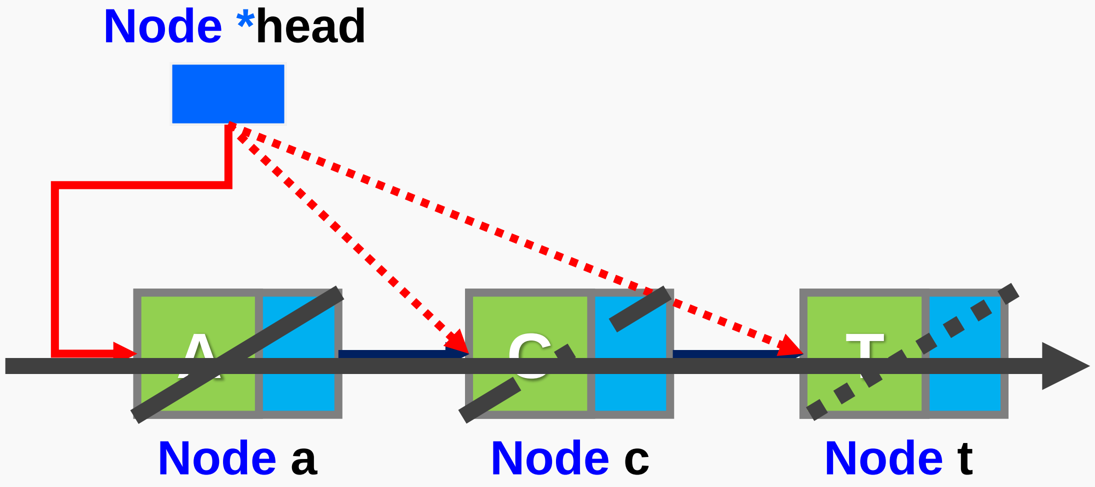
Push



Pop



Release

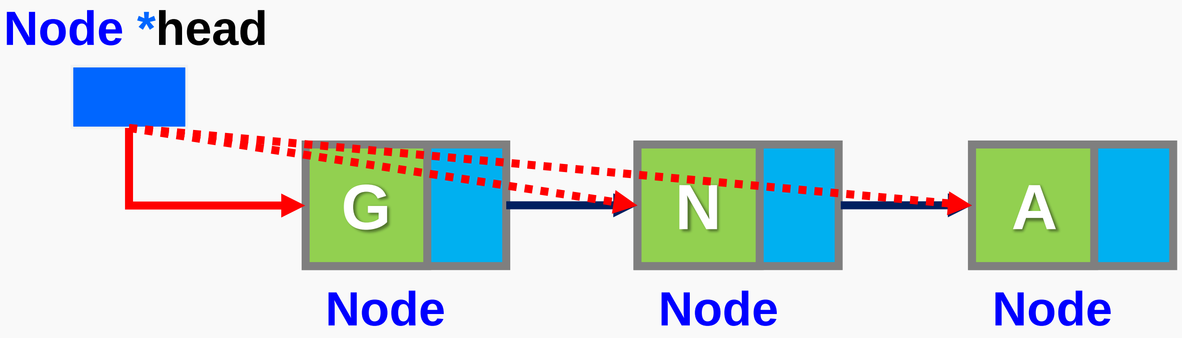


2022/05/24

</push>

<push/>

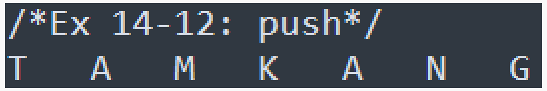
Push



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
typedef struct node{…SKIP…} Node;
void printNode(const Node *head){…SKIP…}

void push(Node **stack, char letter){
        Node *temp = (Node*) malloc (sizeof(Node));
        temp->alpha = letter;
        temp->next = *stack;
        *stack = temp;
}
```

```
int main(){
        /*Ex 14-12: push*/
        printf("/*Ex 14-12: push*/\n");
        Node *head = 0;
        push(&head, 'G');
        push(&head, 'N');
        push(&head, 'A');
        push(&head, 'K');
        push(&head, 'M');
        push(&head, 'A');
        push(&head, 'T');
        printNode(head);
}
```

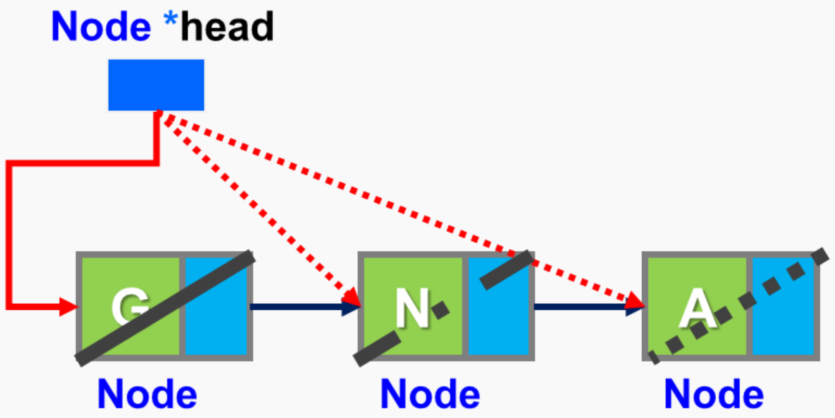


```
/*Ex 14-12: push*/
T   A   M   K   A   N   G
```

</push>

<pop/>

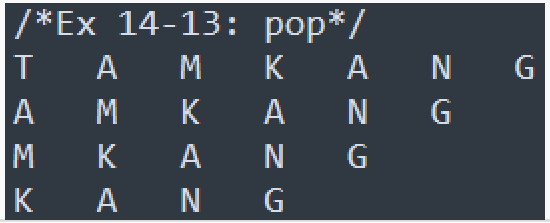
Pop



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
typedef struct node{…SKIP…} Node;
void printNode(const Node *head){…SKIP…}
void push(Node **stack, char letter){…SKIP…}

void pop(Node **stack){
        Node *temp = *stack;
        *stack = temp->next;
        free(temp);
}
```

```
int main(){
        /*Ex 14-13: pop*/
        printf("/*Ex 14-13: pop*/\n");
        Node *head = 0;
        push(&head, 'G');
        push(&head, 'N');
        push(&head, 'A');
        push(&head, 'K');
        push(&head, 'M');
        push(&head, 'A');
        push(&head, 'T');
        printNode(head);
        pop(&head);
        printNode(head);
        pop(&head);
        printNode(head);
        pop(&head);
        printNode(head);
}
```

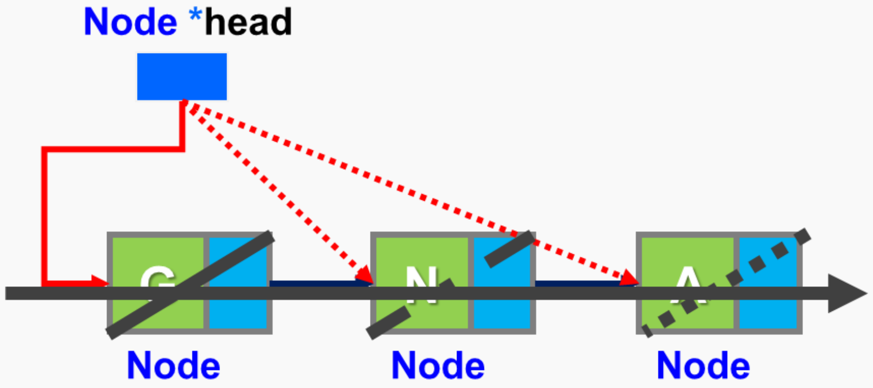


```
/*Ex 14-13: pop*/
T   A   M   K   A   N   G
A   M   K   A   N   G
M   K   A   N   G
K   A   N   G
```

</pop>

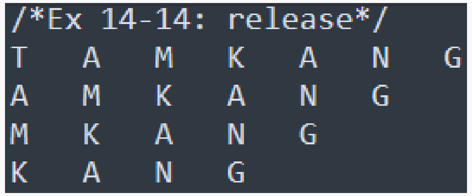
<release/>

Release



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
typedef struct node{…SKIP…} Node;
void printNode(const Node *head){…SKIP…}
void push(Node **stack, char letter){…SKIP…}
void pop(Node **stack){…SKIP…}

void release(Node **stack){
        while(*stack){
                Node *temp = *stack;
                *stack = temp->next;
                free(temp);
        }
}
```



```
/*Ex 14-14: release*/
T  A  M  K  A  N  G
A  M  K  A  N  G
M  K  A  N  G
K  A  N  G
```

```
int main(){
        /*Ex 14-14: release*/
        printf("/*Ex 14-14: release*/\n");
        Node *head = 0;
        push(&head, 'G');
        push(&head, 'N');
        push(&head, 'A');
        push(&head, 'K');
        push(&head, 'M');
        push(&head, 'A');
        push(&head, 'T');
        printNode(head);
        pop(&head);
        printNode(head);
        pop(&head);
        printNode(head);
        pop(&head);
        printNode(head);
        release(&head);
        printNode(head);
}
```

</release>

<header file/>

Packing as a C header file (*.h)

W16_header.h

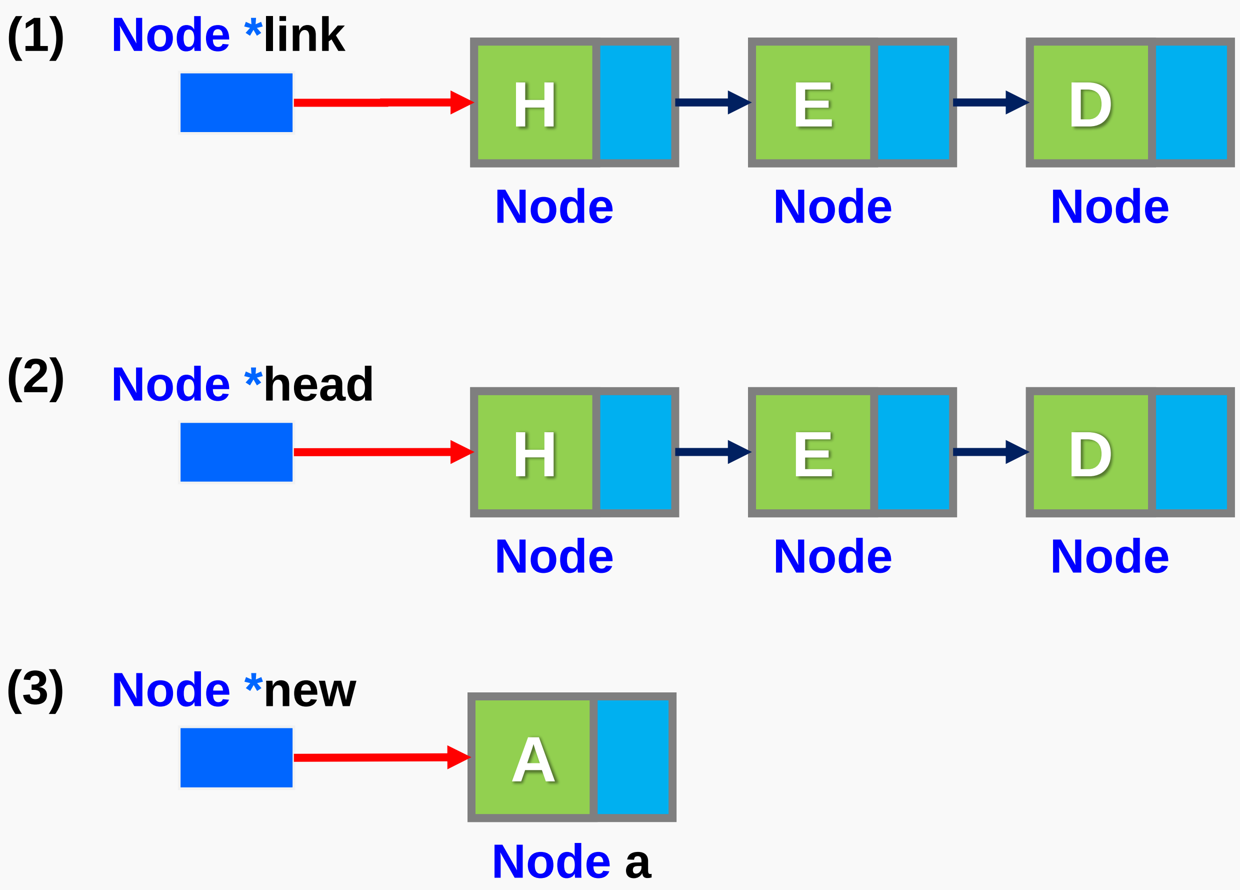
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
typedef struct node{…SKIP…} Node;
void bulitLLByLoop(const char letter[], Node act[]){…SKIP…}
void printNode(const Node *head){…SKIP…}
void push(Node **stack, char letter){…SKIP…}
void pop(Node **stack){…SKIP…}
void release(Node **stack){…SKIP…}
```

</header file>

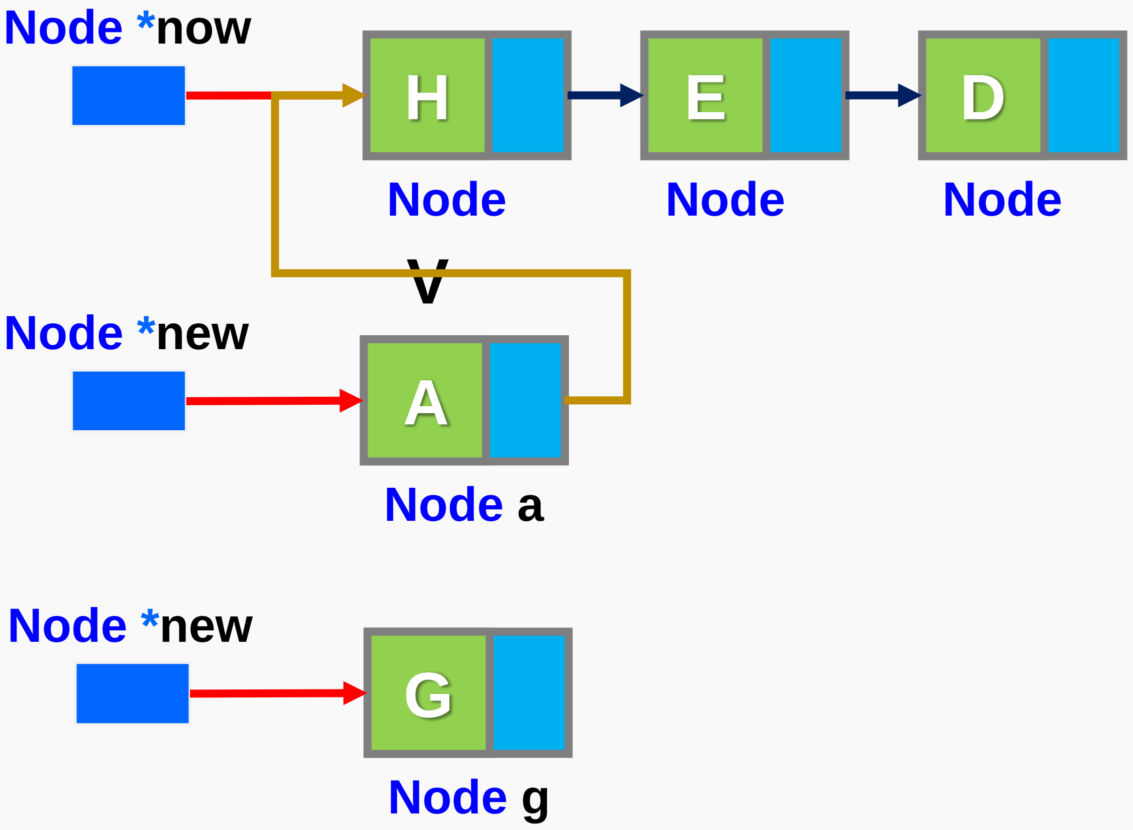
<insert/>

Insert :: situ 1 :: first

Preparation



Search → Compare → Insert



2022/05/24

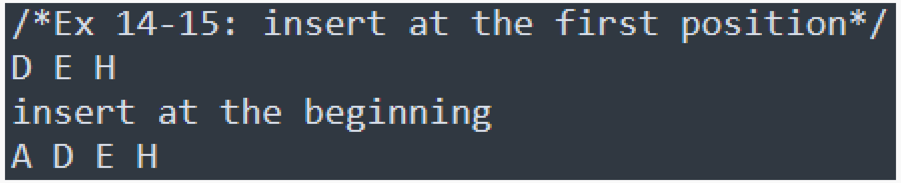
</insert>

<insert/>

Insert :: situ 1 :: first

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "W16_header.h"

int main(){
        /*Ex 14-15: insert at the first position*/
        printf("/*Ex 14-15: insert at the first position*/\n");
        // original linked list
        Node *link = 0;
        push(&link, 'H');
        push(&link, 'E');
        push(&link, 'D');
        printNode(link);
        // new Node a
        Node a;
        a.alpha = 'A';
        a.next = 0;
```

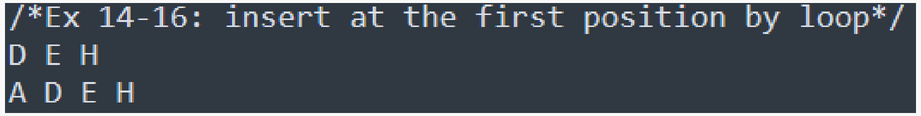


```
/*Ex 14-15: insert at the first position*/
D E H
insert at the beginning
A D E H
```

```
        // set Node ptrs for search and insertion
        Node *head = 0, *now = 0, *new = 0;
        // store the memory space of the new Node a
        new = &a;
        // store the starting point of original linked list
        head = link;
        // store the first Node and second Node location
        now = head;
        // compare the alphabet ranking
        if (new->alpha < now->alpha){
                printf("insert at the beginning\n");
        }else{
                printf("insert at other positions\n");
        }
        // insert at the beginning
        head = new;
        new->next = now;
        // print all nodes
        printNode(head);
        now = 0;}
```

</insert>

<insert/>



```
/*Ex 14-16: insert at the first position by loop*/
D E H
A D E H
```

Insert :: situ 1 :: first by loop

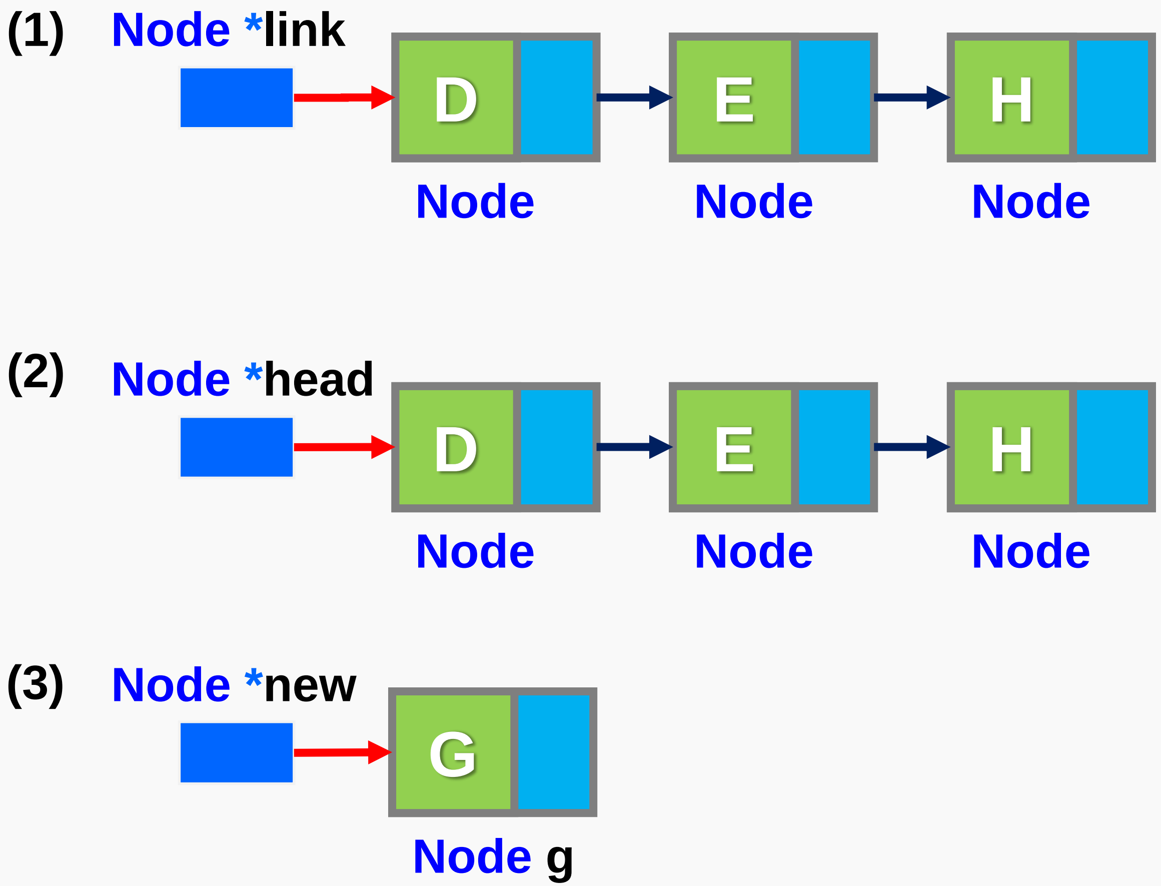
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "W16_header.h"

int main(){
	/*Ex 14-16: insert …by loop*/
	printf("/*Ex 14-16: insert …by loop*/\n");
	// original linked list
	Node *link = 0;
	push(&link, 'H');
	push(&link, 'E');
	push(&link, 'D');
	printNode(link);
	// new Node a
	Node a;
	a.alpha = 'A';
	a.next = 0;
	// set Node ptrs for search and insertion
	Node *head = link, *pre = 0;
	Node *now = head, *new = &a;
	while (now && now->alpha < new->alpha){
		// store the Node location
		pre = now;
		now = now->next;
	}
	if (pre==0){
		// if the node at the beginning
		new->next = head;
		head = new;
	}
	// print all nodes
	printNode(head);
}
```

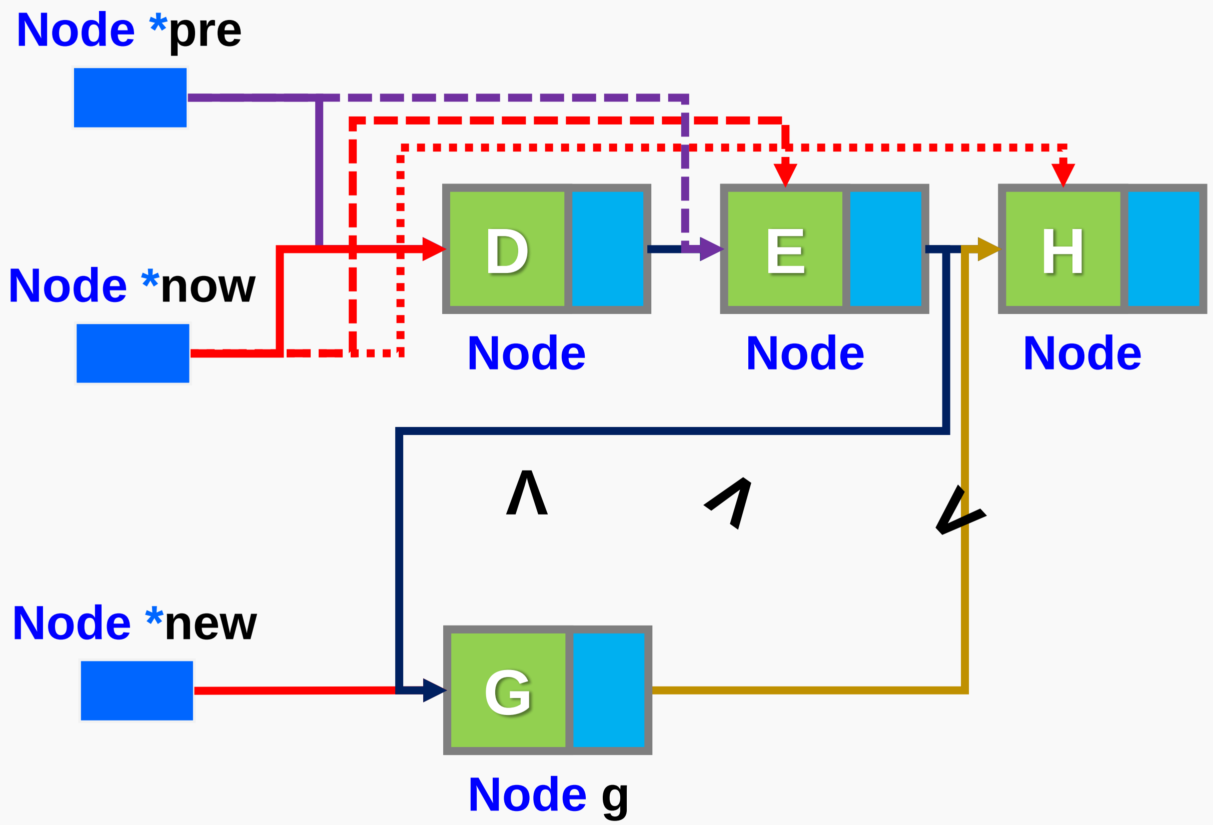
</insert>

Insert :: situ 2 :: non first

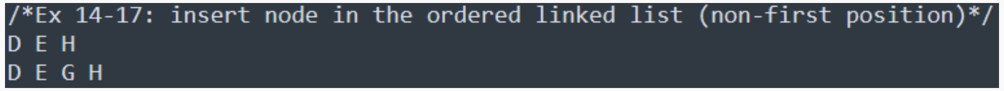
Preparation



Search → Compare → Insert



<insert/>



```
/*Ex 14-17: insert node in the ordered linked list (non-first position)*/
D E H
D E G H
```

Insert :: situ 2 :: non first

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "W16_header.h"

int main(){
        /*Ex 14-17: insert …position)*/
        printf("/*Ex 14-17: insert …position)*/\n");
        // original linked list
        Node *link = 0;
        push(&link, 'H');
        push(&link, 'E');
        push(&link, 'D');
        printNode(link);
        // new Node g
        Node g;
        g.alpha = 'G';
        g.next = 0;
```

```
// set Node ptrs for search and insertion
Node *head = link, *pre = 0;
Node *now = head, *new = &g;
while (now && now->alpha < new->alpha){
        // store the Node location
        pre = now;
        now = now->next;
}
if (pre==0){
        // if the node at the beginning
        new->next = head;
        head = new;
}else{
        // if the node at the other positions
        pre->next = new;
        new->next = now;
}
printNode(head);}
```

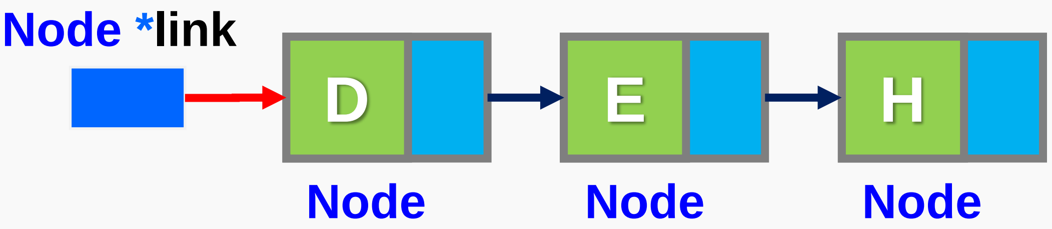
</insert>

<insert/>

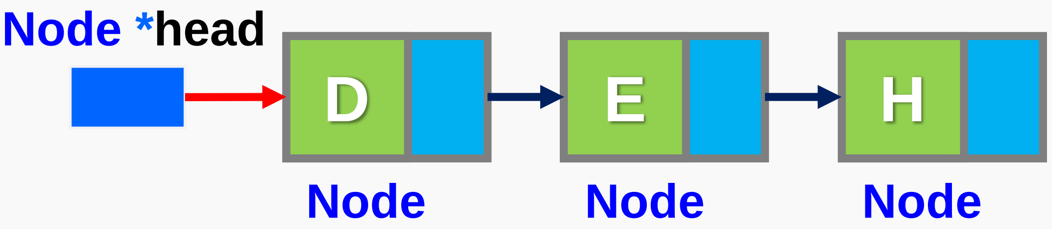
Insert :: situ 3 :: last

Preparation

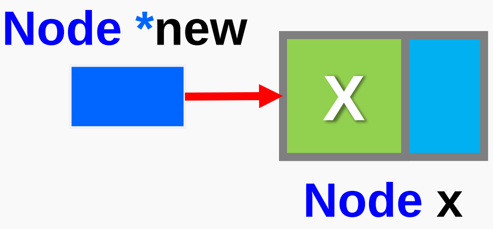
(1)



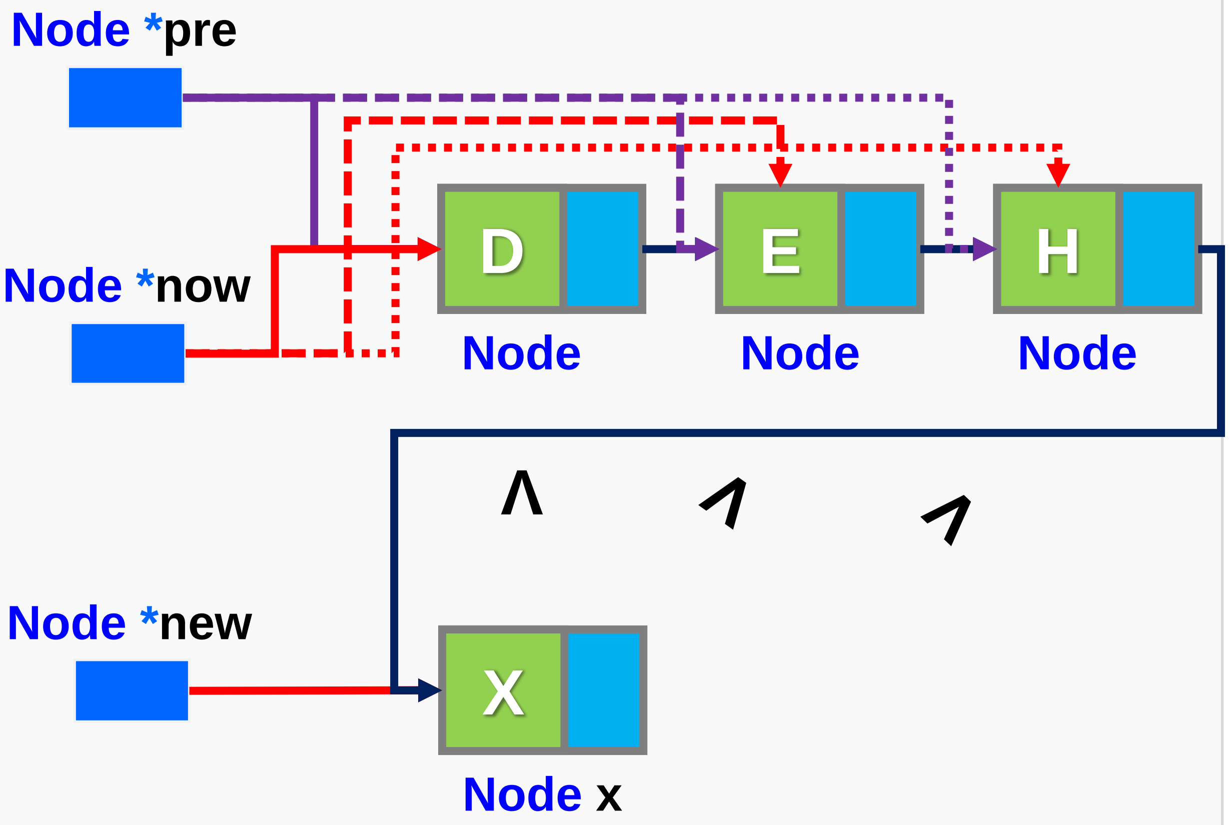
(2)



(3)



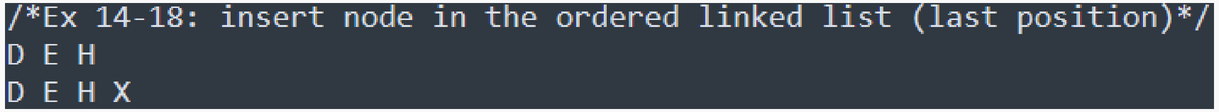
Search → Compare → Insert



2022/05/24

</insert>

<insert/>



```
/*Ex 14-18: insert node in the ordered linked list (last position)*/
D E H
D E H X
```

Insert :: situ 3 :: last

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "W16_header.h"

int main(){
        /*Ex 14-18: insert …position)*/
        printf("/*Ex 14-18: insert …position)*/\n");
        // original linked list
        Node *link = 0;
        push(&link, 'H');
        push(&link, 'E');
        push(&link, 'D');
        printNode(link);
        // new Node x
        Node x;
        x.alpha = 'X';
        x.next = 0;
```

```
// set Node ptrs for search and insertion
Node *head = link, *pre = 0;
Node *now = head, *new = &g;
while (now && now->alpha < new->alpha){
        // store the Node location
        pre = now;
        now = now->next;
}
if (pre==0){
        // if the node at the beginning
        new->next = head;
        head = new;
}else{
        // if the node at the other positions
        pre->next = new;
        new->next = now;
}
printNode(head);}
```

</insert>

<insert/>

Insert :: scanf – 1/3

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "W16_header.h"

int main(){
        /*Ex 14-19: insert node in the ordered linked list with scanf*/
        printf("/*Ex 14-19: insert node in the ordered linked list with scanf*/\n");
        // empty linked list
        Node *head = 0;
        char endSym = '*';

        printf("plz enter a alphabet character (end loop: enter *) >>> ");
        scanf("%c", &endSym);
```

</insert>

Insert :: scanf – 2/3

```
while (endSym!='*'){
    // store a new Node
    Node *new = (Node*)malloc(sizeof(Node));
    new->alpha = endSym;
    new->next = 0;
    // search the position for insertion
    Node *pre = 0, *now = head;
    while (now && now->alpha < new->alpha){
        // store the first Node and second Node location
        pre = now;
        now = now->next;
    }
```

<insert/>

Insert :: scanf – 3/3

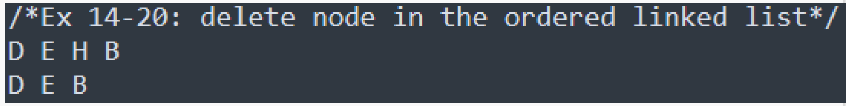
```
        if (pre==0){
            // if the node (to be inserted) at the beginning
            new->next = head;
            head = new;
        } else{
            // if the node (to be inserted) at the other positions
            pre->next = new;
            new->next = now;
        }
        // print all nodes
        printf("current status: ");
        printNode(head);
        printf("plz enter a alphabet character (end loop: enter *) >>> ");
        scanf("%c", &endSym);
    }
    // free memory space
    release(&head);}
```

</insert>

<delete/>

Delete

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "W16_header.h"
int main(){
        /*Ex 14-20: delete … list*/
        printf("/*Ex 14-20: delete …list*/\n");
        // bulid linked list
        Node *link = 0;
        push(&link, 'B');
        push(&link, 'H');
        push(&link, 'E');
        push(&link, 'D');
        printNode(link);
        // new Node a
        Node h;
        h.alpha = 'H';
        h.next = 0;
// set Node ptrs for search and deletion
Node *head = link, *pre = 0, *now = head, *new = &h;
while (now && now->alpha != new->alpha){
        // store the first Node and second Node location
        pre = now;
        now = now->next;
}
if (pre==0){
        // if the node (to be deleted) at the beginning
        head = head->next;
        free(now);
} else{
        // if the node (to be deleted) at the other positions
        pre->next = now->next;
        free(now);
}
// print all nodes
printNode(head);
// free memory space
release(&head);}
```



```
/*Ex 14-20: delete node in the ordered linked list*/
D E H B
D E B
```

</delete>

<Assignment/>

作業一

撰寫一組函數可以自動依照字母大小排列:

(1)決定新的Node要插入的位置，插入現存的鏈結中。

(2)尋找指定的Node是否存在於現存的鏈結中，如有印出其位置；若無回傳0。

(3)刪除現存的鏈結中指定的Node，若不存在回傳0。

</Assignment>

<References/>

參考資料

1. 堆疊(stack) 資料結構
2. Data Structure - Doubly Linked List
3. [資料結構] 雙向鏈結串列教學[1]: 新增與印出
4. Queue: Intro(簡介)，並以Linked list實作
5. 以連結串列 (Linked List) 為基礎的佇列 (Queue)
6. Stack Data Structure (Introduction and Program)
7. C 語言：鏈結串列(Linked List)的建立與刪除
8. [資料結構]Stack — 堆疊和Queue — 佇列
9. Linked List: 新增資料、刪除資料、反轉
10. 蔣宗哲教授講義

</References>