자료구조론 CC343_2207

Reading assignment 6

경기대학교 컴퓨터공학부

201511837 이상민

Review Questions

1. Make a comparison between a linked list and a linear array. Which one will you prefer to use and when?
연계 목록과 선형 배열을 비교한다. 어떤 것을 사용하고 언제 쓰시겠습니까?

Key Differences Between Array and Linked List

1. An array is the data structure contains a collection of similar type data elements whereas the Linked list is considered as non-primitive data structure contains a collection of unordered linked elements known as nodes.
2. In the array the elements belong to indexes, i.e., if you want to get into the fourth element you have to write the variable name with its index or location within the square bracket.
3. In a linked list though, you have to start from the head and work your way through until you get to the fourth element.
4. While accessing an element array is fast while Linked list takes linear time so, it is quite bit slower.
5. Operations like insertion and deletion in arrays consume a lot of time. On the other hand, the performance of these operations in Linked lists is fast.
6. Arrays are of fixed size. In contrast, Linked lists are dynamic and flexible and can expand and contract its size.
7. In an array, memory is assigned during compile time while in a Linked list it is allocated during execution or runtime.
8. Elements are stored consecutively in arrays whereas it is stored randomly in Linked lists.

9. The requirement of memory is less due to actual data being stored within the index in the array. As against, there is a need for more memory in Linked Lists due to storage of additional next and previous referencing elements.

10. In addition memory utilization is inefficient in the array. Conversely, memory utilization is efficient in the array.

2. Why is a doubly linked list more useful than a singly linked list?
단독 연계 리스트보다 이중 연계 리스트가 더 유용한 이유는?

In general, a singly linked list is more useful than a doubly linked list. All the cons stated so far can be resolved by a singly linked list implementations. You can try the following functions as examples:

- ✓ Print the elements of a singly linked list in reverse.
- ✓ Delete a random element from a linked list (whose pointer you know) in constant time (it should take the same amount of time to delete an element near the start of the list, as it does to delete an element near the end of the list)

3. Give the advantages and uses of a circular linked list.
원형 연계 리스트의 장점과 활용도를 부여한다.

Advantages of a circular linked list

- ✓ Some problems are circular and a circular data structure would be more natural when used to represent it
- ✓ The entire list can be traversed starting from any node (traverse means visit every node just once)
- ✓ fewer special cases when coding(all nodes have a node before and after it)

Applications of Circular Linked List

- ✓ Circular lists are used in applications where the entire list is accessed one-by-one in a loop. Example: Operating systems may use it to switch between various running applications in a circular loop.
- ✓ It is also used by Operating system to share time for different users, generally uses Round-Robin time sharing mechanism.
- ✓ Multiplayer games uses circular list to swap between players in a loop.
- ✓ Implementation of Advanced data- structures like Fibonacci Heap.

4. Specify the use of a header node in a header linked list.
헤더 링크 목록에서 헤더 노드의 사용을 명시한다.

A Header linked list is one more variant of linked list. In Header linked list, we have a special node present at the beginning of the linked list. This special node is used to store number of nodes present in the linked list. In other linked list variant, if we want to know the size of the linked list we use traversal method. But in Header linked list, the size of the linked list is stored in its header itself.

5. Give the linked representation of the following polynomial : $7x^3y^2 - 8x^2y + 3\text{xy} + 11 - 4$
다음과 같은 다항식의 연계표현을 한다.

	0	**1**	**3**	**4**	**5**
계수	*7*	*8*	*3*	*11*	*4*
X 차수	*3*	*2*	*1*	*0*	*0*
Y 차수	*2*	*1*	*1*	*0*	*0*

6. Explain the difference between a circular linked list and a singly linked list.
원형 링크 리스트와 단일 링크 리스트의 차이점을 설명하라.

The singly linked list is a linked list which is a set of multiple nodes (contains information and address of another node) connected with each other and contains null in the address part of the last node. It is used to traverse the nodes until the last node is not reached.

The circular linked list is a linked list which is also the set of multiple nodes (contains information and address of another node) connected with each other and does not contain null in the address part of the last node. It is because the last node contains the address of the first node which makes the linked list circular and there is no endpoint found on this list.

7. Form a linked list to store students' details.

학생들의 세부사항을 저장하기 위해 링크된 목록을 작성하라.

Project1 (전역 범위)

```c
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

struct student {
    char name[100];
    char roll[10];
    struct student *next;
};
 struct student *first = NULL, *last = NULL, *k;
void create(int n)
{
    int i;
    first = (struct student*)malloc(sizeof(struct student));
    printf("\nEnter the first name of the student:");
    scanf("%s", first->name);
    printf("\nEnter the roll number of the student:");
    scanf("%s", first->roll);
    first->next = NULL;
    last = first;
    for (i = 1; i < first; i++) {
    k = (struct student*)malloc(sizeof(struct student));
    printf("\nEnter the first name of the student:");
    scanf("%s",k->name);
    printf("\nEnter the roll number of the student:");
    scanf("%s",k->roll);
    k->next = NULL;
    last->next = k;
    last = k;
    }
}
```

```c
void display()
{
    struct student *t;
    t = first;
    while (t != NULL)
    {
        printf("\nThe roll number of the student:%s", t->roll);
        printf("\nFirst name of the student:%s", t->name);
        t = t->next;
    }
}
```

```c
void search()
{
    char r[10];
    int flag = 0;
    printf("\nEnter the roll number u wanna search:");
    scanf("%s", r);
    struct student *t;
    t = first;
    while (t != NULL)
    {
        if (strcmpi(r, t->roll) == 0)
        {
            printf("\nThe roll number found in the list!!!\nHis name is %s", t->name);
            flag = 1;
            break;
        }t = t->next;
    }
    if (flag == 0)
        printf("\nThe roll number not in database!!");
}
```

```c
int main()
{
    int n, o;
    while (o != 0)
    {
        printf("\nMENU\n");
        printf("\nEnter 1 for creating database");
        printf("\nEnter 2 for displaying database");
        printf("\nEnter 3 for inserting an record after another");
        printf("\nEnter 4 for deleting a record");
        printf("\nEnter 5 for searching a record");
        printf("\nEnter 0 for exit!");
        printf("\nEnter the choice:");
        scanf("%d", &o);
        switch (o)
        {
        case 1:printf("\nEnter the maximum size of the database:");
            scanf("%d", &n);
            create(n);
            break;
        case 2:display(); break;
        case 3:insertafter(); break;
        case 4:del(); break;
        case 5:search(); break;
        case 0:exit(0); break;
        default:printf("\nYou have entered a wrong choice!!!");
        }
    }
    getche();
}
```

8. Use the linked list of the above question to insert the record of a new student in the list.
위 질문의 링크된 목록을 사용하여 목록에 신입생의 기록을 삽입한다.

```
void insertafter()
{
    char r[10];
    int flag = 0;
    printf("\nEnter the roll number u wanna insert after that:");
    scanf("%s", r);
    struct student *t;
    t = first;
    while (t != NULL)
    {
        if (strcmpi(r, t->roll) == 0)
        {
            k = (struct student*)malloc(sizeof(struct student));
            printf("\nEnter the first name of the student:");
            scanf("%s", k->name);
            printf("\nEnter the roll number of the student:");
            scanf("%s", k->roll);
            k->next = t->next;
            t->next = k;
            flag = 1;
            break;
        }
        t = t->next;
    }
    if (flag == 0)
        printf("\nThe element not found!!!");
}
```

9. Delete the record of a student with a specified roll number from the list maintained in Question 7.
문제 7에서 유지되는 목록에서 지정된 롤 번호를 가진 학생의 기록을 삭제한다.

```
void del()
{
    struct student *back, *t, *k;
    char r[10];
    int flag = 0;
    printf("\nEnter the roll number u wanna delete:");
    scanf("%s", r);
    if (strcmpi(r, first->roll) == 0)
    {
        first = first->next;
        flag = 1;
    }
    else
    {
        back = first;
        k = first->next;
        while (k != NULL)
        {
            if (strcmpi(r, k->roll) == 0)
            {
                back->next = k->next;
                flag = 1;
                break;
            }
        }
    }
    if (flag == 0)
        printf("\nThe element not found!!!");
}
```

7,8,9번의 전체 소스코드 :

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

struct student {
        char name[100];
        char roll[10];
        struct student *next;
};
struct student *first = NULL, *last = NULL, *k;
void create(int n)
{
        int i;
        first = (struct student*)malloc(sizeof(struct student));
        printf("₩nEnter the first name of the student:");
        scanf("%s", first->name);
        printf("₩nEnter the roll number of the student:");
        scanf("%s", first->roll);
        first->next = NULL;
        last = first;
        for (i = 1; i < first; i++) {
        k = (struct student*)malloc(sizeof(struct student));
        printf("₩nEnter the first name of the student:");
        scanf("%s",k->name);
        printf("₩nEnter the roll number of the student:");
```

```
            scanf("%s",k->roll);
            k->next = NULL;
            last->next = k;
            last = k;
            }
}

void display()
{
            struct student *t;
            t = first;
            while (t != NULL)
            {
                        printf("₩nThe roll number of the student:%s", t->roll);
                        printf("₩nFirst name of the student:%s", t->name);
                        t = t->next;
            }
}

void insertafter()
{
            char r[10];
            int flag = 0;
            printf("₩nEnter the roll number u wanna insert after that:");
            scanf("%s", r);
            struct student *t;
```

```
        t = first;
        while (t != NULL)
        {
                if (strcmpi(r, t->roll) == 0)
                {
                        k = (struct student*)malloc(sizeof(struct student));
                        printf("₩nEnter the first name of the student:");
                        scanf("%s", k->name);
                        printf("₩nEnter the roll number of the student:");
                        scanf("%s", k->roll);
                        k->next = t->next;
                        t->next = k;
                        flag = 1;
                        break;
                }
                t = t->next;
        }
        if (flag == 0)
                printf("₩nThe element not found!!!");
}

void del()
{
        struct student *back, *t, *k;
        char r[10];
        int flag = 0;
```

```
        printf("₩nEnter the roll number u wanna delete:");
        scanf("%s", r);
        if (strcmpi(r, first->roll) == 0)
        {
                first = first->next;
                flag = 1;
        }
        else
        {
                back = first;
                k = first->next;
                while (k != NULL)
                {
                        if (strcmpi(r, k->roll) == 0)
                        {
                                back->next = k->next;
                                flag = 1;
                                break;
                        }
                }
        }
        if (flag == 0)
                printf("₩nThe element not found!!!");
}
```

```
void search()
{
	char r[10];
	int flag = 0;
	printf("₩nEnter the roll number u wanna search:");
	scanf("%s", r);
	struct student *t;
	t = first;
	while (t != NULL)
	{
		if (strcmpi(r, t->roll) == 0)
		{
			printf("₩nThe roll number found in the list!!!₩nHis name is %s", t->name);
			flag = 1;
			break;
		}t = t->next;
	}
	if (flag == 0)
		printf("₩nThe roll number not in database!!");
}

int main()
{
	int n, o;
	while (o != 0)
	{
```

```
                printf("₩nMENU₩n");
                printf("₩nEnter 1 for creating database");
                printf("₩nEnter 2 for displaying database");
                printf("₩nEnter 3 for inserting an record after another");
                printf("₩nEnter 4 for deleting a record");
                printf("₩nEnter 5 for searching a record");
                printf("₩nEnter 0 for exit!");
                printf("₩nEnter the choice:");
                scanf("%d", &o);
                switch (o)
                {
                case 1:printf("₩nEnter the maximum size of the database:");
                        scanf("%d", &n);
                        create(n);
                        break;
                case 2:display(); break;
                case 3:insertafter(); break;
                case 4:del(); break;
                case 5:search(); break;
                case 0:exit(0); break;
                default:printf("₩nYou have entered a wrong choice!!!");
                }
        }
        getche();
}
```

10. Given a linked list that contains English alphabet. The characters may be in upper case or in lower case. Create two linked lists—one which stores upper case characters and the other that stores lower case characters.
어 알파벳이 포함된 링크된 목록이 주어진다. 문자는 대문자 또는 소문자일 수 있다. 대문자를 저장하는 목록과 소문자를 저장하는 목록 등 두 개의 연결된 목록을 만드십시오.

```
#include<stdio.h>

#include<stdlib.h>

void main() {

	char str[] = "#GeeKs01fOr@gEEks07";

	int upper = 0, lower = 0;

	int number = 0, special = 0;

	for (int i = 0; i < len(str); i++)

	{

		char ch = str[i];

		if (ch >= 'A' && ch <= 'Z')

			upper++;

		else if (ch >= 'a' && ch <= 'z')

			lower++;

		else if (ch >= '0' && ch <= '9')

			number++;

		else

			special++;

	}

	printf("Upper case letters : " + upper);

	printf("Lower case letters : " + lower);

	printf("Number : " + number);

	printf("Special characters : " + special);

	return 0;

}
```

11. Create a linked list which stores names of the employees. Then sort these names and re-display the contents of the linked list.
직원의 이름을 저장하는 연결 목록을 작성한다. 그런 다음 이 이름을 정렬하고 링크된 목록의 내용을 다시 표시하십시오.

```c
/* Program to insert in a sorted list */
#include<stdio.h>
#include<stdlib.h>

/* Link list node */
struct Node
{
    char data;
    struct Node* next;
};
```

```c
/* function to insert a new_node in a list. Note that this
 function expects a pointer to head_ref as this can modify the
 head of the input linked list (similar to push())*/
void sortedInsert(struct Node** head_ref, struct Node* new_node)
{
    struct Node* current;
    /* Special case for the head end */
    if (*head_ref == NULL || (*head_ref)->data >= new_node->data)
    {
        new_node->next = *head_ref;
        *head_ref = new_node;
    }
    else
    {
        /* Locate the node before the point of insertion */
        current = *head_ref;
        while (current->next != NULL &&
               current->next->data < new_node->data)
        {
            current = current->next;
        }
        new_node->next = current->next;
        current->next = new_node;
    }
}
```

```c
/* A utility function to create a new node */
struct Node *newNode(char new_data)
{
    /* allocate node */
    struct Node* new_node =
        (struct Node*) malloc(sizeof(struct Node));

    /* put in the data  */
    new_node->data = new_data;
    new_node->next = NULL;

    return new_node;
}

/* Function to print linked list */
void printList(struct Node *head)
{
    struct Node *temp = head;
    while (temp != NULL)
    {
        printf("%d  ", temp->data);
        temp = temp->next;
    }
}
```

```c
/* Driver program to test count function*/
void main()
{
    /* Start with the empty list */
    struct Node* head = NULL;
    struct Node *new_node = newNode(5);
    sortedInsert(&head, new_node);
    new_node = newNode("a");
    sortedInsert(&head, new_node);
    new_node = newNode("c");
    sortedInsert(&head, new_node);
    new_node = newNode("z");
    sortedInsert(&head, new_node);
    new_node = newNode("f");
    sortedInsert(&head, new_node);
    new_node = newNode("e");
    sortedInsert(&head, new_node);
    printf("₩n Created Linked List₩n");
    printList(head);

    return 0;
}
```

```
 Created Linked List
5  56  60  64  68  72
C:₩Users₩이상민₩source₩repo
디버깅이 중지될 때 콘솔을
록 설정합니다.
```

전체 소스코드 :

```
#include<stdio.h>
#include<stdlib.h>

/* Link list node */
struct Node
{
        char data;
        struct Node* next;
};

/* function to insert a new_node in a list. Note that this
   function expects a pointer to head_ref as this can modify the
   head of the input linked list (similar to push())*/
void sortedInsert(struct Node** head_ref, struct Node* new_node)
{
        struct Node* current;
        /* Special case for the head end */
        if (*head_ref == NULL || (*head_ref)->data >= new_node->data)
        {
                new_node->next = *head_ref;
                *head_ref = new_node;
        }
        else
        {
                /* Locate the node before the point of insertion */
```

```
                current = *head_ref;
                while (current->next != NULL &&
                        current->next->data < new_node->data)
                {
                        current = current->next;
                }
                new_node->next = current->next;
                current->next = new_node;
        }
}

/* BELOW FUNCTIONS ARE JUST UTILITY TO TEST sortedInsert */

/* A utility function to create a new node */
struct Node *newNode(char new_data)
{
        /* allocate node */
        struct Node* new_node =
                (struct Node*) malloc(sizeof(struct Node));

        /* put in the data  */
        new_node->data = new_data;
        new_node->next = NULL;

        return new_node;
}
```

```
/* Function to print linked list */
void printList(struct Node *head)
{
        struct Node *temp = head;
        while (temp != NULL)
        {
                printf("%d  ", temp->data);
                temp = temp->next;
        }
}

/* Driver program to test count function*/
void main()
{
        /* Start with the empty list */
        struct Node* head = NULL;
        struct Node *new_node = newNode(5);
        sortedInsert(&head, new_node);
        new_node = newNode("a");
        sortedInsert(&head, new_node);
        new_node = newNode("c");
        sortedInsert(&head, new_node);
        new_node = newNode("z");
        sortedInsert(&head, new_node);
        new_node = newNode("f");
```

```
        sortedInsert(&head, new_node);
        new_node = newNode("e");
        sortedInsert(&head, new_node);
        printf("₩n Created Linked List₩n");
        printList(head);

        return 0;
}
```

Multiple-choice Questions

1. A linked list is a

(a) Random access structure

(b) Sequential access structure

(c) Both

(d) None of these

2. An array is a

(a) Random access structure

(b) Sequential access structure

(c) Both

(d) None of these

3. Linked list is used to implement data structures like

(a) Stacks (b) Queues (c) Trees **(d) All of these**

4. Which type of linked list contains a pointer to the next as well as the previous node in the sequence?

(a) Singly linked list (b) Circular linked list **(c) Doubly linked list** (d) All of these

5. Which type of linked list does not store NULL in next field?

(a) Singly linked list **(b) Circular linked list** (c) Doubly linked list (d) All of these

6. Which type of linked list stores the address of the header node in the next field of the last node?
(a) Singly linked list (b) Circular linked list

(c) Doubly linked list **(d) Circular header linked list**

7. Which type of linked list can have four pointers per node?

(a) Circular doubly linked list

(b) Multi-linked list

(c) Header linked list

(d) Doubly linked list

true or False

1. A linked list is a linear collection of data elements. **: True**

2. A linked list can grow and shrink during run time. **: True**

3. A node in a linked list can point to only one node at a time. **: False**

4. A node in a singly linked list can reference the previous node. **: False**

5. A linked list can store only integer values. **: False**

6. Linked list is a random access structure. **: False**

7. Deleting a node from a doubly linked list is easier than deleting it from a singly linked list. **: True**

8. Every node in a linked list contains an integer part and a pointer. **: False**

9. START stores the address of the first node in the list. **: True**

10. Underflow is a condition that occurs when we try to delete a node from a linked list that is empty. **: True**

Fill in the Blanks

1. _____ is used to store the address of the first free memory location.

: AVAIL

2. The complexity to insert a node at the beginning of the linked list is _____.

: O(1)

3. The complexity to delete a node from the end of the linked list is _____.

: O(n)

4. Inserting a node at the beginning of the doubly linked list needs to modify _____ pointers.

: Two

5. Inserting a node in the middle of the singly linked list needs to modify _____ pointers.

: One

6. Inserting a node at the end of the circular linked list needs to modify _____ pointers.

: Two

7. Inserting a node at the beginning of the circular doubly linked list needs to modify _____ pointers.

: Two

8. Deleting a node from the beginning of the singly linked list needs to modify _____ pointers

: One

9. Deleting a node from the middle of the doubly linked list needs to modify _____ pointers.

: Two

10. Deleting a node from the end of a circular linked list needs to modify _____ pointers.

: One

11. Each element in a linked list is known as a _____.

: Node

12. First node in the linked list is called the _____.

: START

13. Data elements in a linked list are known as _____.

: Node

14. Overflow occurs when _____.

: There is no memory that can be allocated for the new node to be inserted

15. In a circular linked list, the last node contains a pointer to the _____ node of the list. **: First**