자료구조 실습과제

덱 회문 검사 프로그램

5671144 강범창

1. 덱을 사용하여 구두점, 스페이스 , 대소문자 등을 무시하는 회문 검사 프로그램을 작성한다.

1.1 덱을 사용하여 회문 검사 프로그램을 작성하기위해 MAX_QUEUE_SIZE , Struct 을 선언합니다.

```
#define MAX_QUEUE_SIZE 20

typedef char element; // char element 즉 이번 덱 에서는 회문 을 검사하기 위해 문자를 사용하게 됩니다.

typedef struct
{
	element data[MAX_QUEUE_SIZE];
	int front;
	int rear;
} DequeType; // 덱 을 구현하기위해 data를 넣어놓을 수 있는 배열과 front , rear 를 가진 구조체를 선언합니다.
```

1.2 error가 일어나면 에러 메시지를 출력하기위해 error 함수를 선언합니다.

```
// error 함수
void error(char *message)
{
	fprintf(stderr, "%s₩n", message);
	return;
}
```

1.3 덱을 구현합니다.

Init 함수입니다. 덱을 초기화하는데 사용합니다.

```
// init
void init(DequeType *queue)
{
	queue->front = 0;
	queue->rear = 0;
}
```

isEmpty 함수 입니다. 덱의 front 와 rear 가 같다면 비었다 보고 1을 return 합니다.

```
int isEmpty(DequeType *queue)
{
	return queue->front == queue->rear ? 1 : 0;
}
```

isFull 함수입니다. 덱의 rear 즉 뒷부분의 값을 MAX QUEUE SIZE 로 나눠 앞부분(0) 과 같다면 1 아니면 0 을 return 을 하여 덱이 가득 찼는지 확인을 합니다.

```
int isFull(DequeType *queue)
{
	return (queue->rear + 1) % MAX_QUEUE_SIZE == queue->front ? 1 : 0;
}
```

printDeque 함수입니다 덱을 출력할 때 사용합니다. 이때 값이 비어 있지 않을 때 모두다 덱의 안에 있는 값들 전부를 출력합니다.

```
void printDeque(DequeType *queue)
{
	printf("DEQUE( front = %d rear = %d ) =", queue->front, queue->rear);
	if (!isEmpty(queue))
	{
```

```
                int index = queue->front;
                do
                {
                        index = (index + 1) % MAX_QUEUE_SIZE;
                        printf(" %d |", queue->data[index]);
                        if (index == queue->rear)
                        {
                                break;
                        }
                } while (index != queue->front);
                printf("₩n");
        }
}
```

addRear 함수입니다. 덱의 뒷부분 에 값을 넣기 위해 사용합니다. 덱이 가득 차 있다면 error메시지를 출력하고 함수를 종료합니다..

```
void addRear(DequeType *queue, element item)
{
        if (isFull(queue))
        {
                error("queue is Full");
                return;
        }
        queue->rear = (queue->rear + 1) % MAX_QUEUE_SIZE;
        queue->data[queue->rear] = item;
}
```

Delete Rear 함수입니다. 뒷부분 에서 값을 빼기 위해 사용합니다. 덱이 비어 있다면 비어 있다는 error 메시지와 -1 을 반환합니다. 아니라면 element 을 반환합니다.

```
element deleteRear(DequeType *queue)
{
```

```
        int prev = queue->rear;
        if (isEmpty(queue))
        {
                error("queue is Empty");
                return -1;
        }
        queue->rear = (queue->rear - 1 + MAX_QUEUE_SIZE) % MAX_QUEUE_SIZE;
        return queue->data[prev];
}
```

Get Rear 함수입니다. 가장 뒷부분에 있는 값이 어떤 값인지 확인하는 함수입니다. 만약 덱이 비어 있다면 비어 있다는 error 메시지와 -1 을 반환합니다. 아니라면 element 값을 반환합니다.

```
element getRear(DequeType *queue)
{
        if (isEmpty(queue))
        {
                error("queue is Empty");
                return -1;
        }
        return queue->data[queue->rear];
}
```

Add Front 함수입니다. 앞쪽으로 값을 넣고 싶을 때 사용하며 Rear 과 마찬가지로 가득 찼을 때 는 error메시지를 출력하고 함수를 종료합니다..

```
void addFront(DequeType *queue, element item)
{
        if (isFull(queue))
        {
                error("queue is Full");
                return;
        }
```

```
        queue->data[queue->front] = item;
        queue->front = (queue->front - 1 + MAX_QUEUE_SIZE) % MAX_QUEUE_SIZE;
}
```

Delete Front 함수 입니다. 앞쪽에서 값을 가져와 삭제하고 그 값을 반환합니다. 만약 비어 있다면 비어 있다는 메시지와 -1 을 반환합니다.

```
element deleteFront(DequeType *queue)
{
        if (isEmpty(queue))
        {
                error("queue is Empty");
                return -1;
        };
        queue->front = (queue->front + 1) % MAX_QUEUE_SIZE;
        return queue->data[queue->front];
}
```

Get front 함수입니다. 가장 앞에 있는 값을 확인하기 위해 사용을 하며 비어 있다면 비어 있다는 메시지와 -1을 반환합니다.

```
element getFront(DequeType *queue)
{
        if (isEmpty(queue))
        {
                error("queue is Empty");
                return -1;
        }
        return queue->data[(queue->front + 1) % MAX_QUEUE_SIZE];
}
```

Is palindrome 함수입니다. 문자열을 파라미터 값으로 받으며 strlen 을 통해 문자열 길이를 알아 낸 후 tolower 함수 isalpha 함수로 공백과, 알파벳이 아닌 값들을 걸러 내고 모두 소문자 로 바꾼후 add Rear 를 통해 뒷부분에 값을 추가합니다 이후 덱의 rear 를 /2

로 나눠 반만큼 돌 수 있게 한 후 rear 의 앞부분, 뒷부분 에서 값을 꺼내 서로 비교를 하여 회문인지를 확인합니다. 만약 아니라면 0 맞다면 1을 반환합니다.

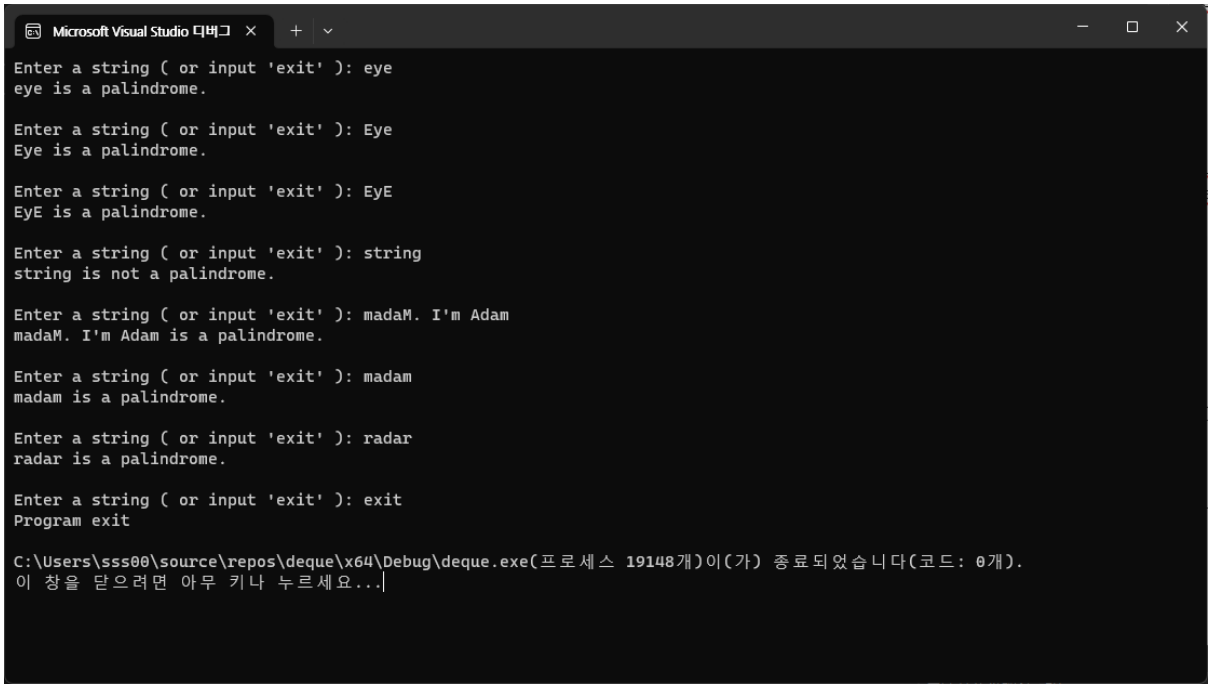
```
int isPalindrome(char inputString[])
{
	DequeType queue;
	int len;
	char c;

	len = strlen(inputString);

	init(&queue);

	for (int i = 0; i < len; i++)
	{
		c = tolower(inputString[i]);
		if (isalpha(c))
		{
			addRear(&queue, c);
		}
	}

	for (int i = 0; i < queue.rear / 2; i++)
	{
		if (deleteFront(&queue) != deleteRear(&queue))
		{
			return 0;
		}
	}
	return 1;
}
```

1.3 메인에서 입력을 받아 처리하는 부분입니다.

입력을 받아 exit 라면 프로그램을 종료하고, 아니라면 회문 을 검사하는 함수를 실행하여 회문인지 아닌지 확인을 할 수 있게 합니다.

```
char input_string[100]; // 문자열을 입력받습니다.
	char yesNo[10];			// 프로그램을 다시시작 할지를 묻는 String 을 저장하기위해 사용합니다.

	while (1)
	{
		printf("Enter a string ( or input 'exit' ): ");
		fgets(input_string, sizeof(input_string), stdin);

		if (strcmp(input_string, "exit₩n") == 0)
		{
			printf("Program exit₩n");
			break;
		}

		input_string[strcspn(input_string, "₩n")] = '₩0'; // 개행 문자 제거
		if (isPalindrome(input_string) == 1)
		{
			printf("%s is a palindrome.₩n", input_string);
		}
		else
		{
			printf("%s is not a palindrome.₩n", input_string);
		}
		printf("₩n");
	}

	return;
```

아래는 실행 결과입니다.


Microsoft Visual Studio 디버그
Enter a string (or input 'exit'): eye
eye is a palindrome.
Enter a string (or input 'exit'): Eye
Eye is a palindrome.
Enter a string (or input 'exit'): EyE
EyE is a palindrome.
Enter a string (or input 'exit'): string
string is not a palindrome.
Enter a string (or input 'exit'): madaM. I'm Adam
madaM. I'm Adam is a palindrome.
Enter a string (or input 'exit'): madam
madam is a palindrome.
Enter a string (or input 'exit'): radar
radar is a palindrome.
Enter a string (or input 'exit'): exit
Program exit
C:\Users\sss00\source\repos\deque\x64\Debug\deque.exe(프로세스 19148개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...