Nama : Muhammad Iqbal Maulana
NIM : 1203230066
Kelas : IF-03-03

TUGAS ASD PRAKTIKUM - Circular Double Linked List

Source Code Circular Double Linked List

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

struct Node
{
    int number;
    struct Node *prev;
    struct Node *next;
};
typedef struct Node Node;

typedef struct
{
    Node *head;
    Node *tail;
    int size;
} DoubleLinkedList;

DoubleLinkedList newDoubleLinkedList()
{
    DoubleLinkedList doubleLinkedList;
    doubleLinkedList.head = NULL;
    doubleLinkedList.tail = NULL;
    doubleLinkedList.size = 0;

    return doubleLinkedList;
}

Node *newNode(int number)
```

```
{
    Node *node = (Node *)malloc(sizeof(Node));

    node->number = number;
    node->prev = NULL;
    node->next = NULL;

    return node;
}

void dispose(DoubleLinkedList doubleLinkedList)
{
    if (doubleLinkedList.size == 0)
    {
        return;
    }

    Node *currNode = doubleLinkedList.head;
    for (size_t i = 0; i < doubleLinkedList.size; i++)
    {
        Node *temp = currNode;
        currNode = currNode->next;

        temp->next = NULL;
        free(temp);
    }
}

void append(DoubleLinkedList *doubleLinkedList, char alphabet)
{
    Node *node = newNode(alphabet);

    if (doubleLinkedList->size > 0)
    {
        node->next = doubleLinkedList->head;
        doubleLinkedList->head->prev = node;
    }
    else
    {
```

```
        doubleLinkedList->tail = node;
    }

    doubleLinkedList->head = node;
    doubleLinkedList->tail->next = doubleLinkedList->head;
    doubleLinkedList->head->prev = doubleLinkedList->tail;

    doubleLinkedList->size++;
}

void display(DoubleLinkedList doubleLinkedList)
{
    Node *walker = doubleLinkedList.head;
    for (int i = 0; i < doubleLinkedList.size; i++)
    {
        printf("Address %p, Data %d\n", walker, walker->number);

        walker = walker->next;
    }

    printf("\n");
}

void swap(Node *a, Node *b)
{
    b->prev = a->prev;
    a->prev->next = b;

    a->next = b->next;
    b->next->prev = a;

    b->next = a;
    a->prev = b;
};

void sort(DoubleLinkedList *doubleLinkedList)
{
    int swapped;
    for (size_t i = 0; i < doubleLinkedList->size - 1; i++)
```

```
    {
        swapped = 0;
        Node *walker = doubleLinkedList->head;
        for (size_t j = 0; j < doubleLinkedList->size - i - 1; j++)
        {

            Node *next = walker->next;
            if (walker->number > next->number)
            {
                swap(walker, walker->next);
                swapped = 1;

                if (walker == doubleLinkedList->head)
                {
                    doubleLinkedList->head = next;
                }

                if (next == doubleLinkedList->tail)
                {
                    doubleLinkedList->tail = walker;
                }

                walker = next->next;
            }
            else
            {
                walker = next;
            }
        }

        // If no two elements were swapped by inner loop,
        // then break
        if (!swapped)
            break;
    }
};

int main(int argc, char const *argv[])
{
```

```
    DoubleLinkedList doubleLinkedList = newDoubleLinkedList();

    // generate random int
    // ref: https://stackoverflow.com/a/822368/8070785
    srand(time(NULL));
    for (size_t i = 0; i < 30; i++)
    {
        int r = rand() % 19; // random 0 - 19
        append(&doubleLinkedList, r);
    }

    printf("Double Linked List Items: \n\n");
    display(doubleLinkedList);

    printf("====================\n\n");

    printf("Sorted Double Linked List Items (Ascending): \n\n");
    sort(&doubleLinkedList);
    display(doubleLinkedList);

    dispose(doubleLinkedList);

    return 0;
}
```

Output Circular Double Linked List

```
→  double_linked_list git:(main) ✗ cmp circular_double_linkedlist
Double Linked List Items:

Address 0x120e06080, Data 10
Address 0x120e06060, Data 8
Address 0x120e06040, Data 0
Address 0x120e06020, Data 3
Address 0x120e06000, Data 9
Address 0x120e05fe0, Data 4
Address 0x120e05fc0, Data 3
Address 0x120e05fa0, Data 13
Address 0x120e05f80, Data 15
Address 0x120e05f60, Data 10
Address 0x120e05f40, Data 7
Address 0x120e05f20, Data 12
Address 0x120e05c40, Data 13
Address 0x120e05c20, Data 12
Address 0x120e05c00, Data 10
Address 0x120e05be0, Data 7
Address 0x120e05bc0, Data 8
Address 0x120e05ba0, Data 9
Address 0x120e05b80, Data 11
Address 0x120e05b60, Data 13
Address 0x120e05b40, Data 14
Address 0x120e05b20, Data 15
Address 0x120e05b00, Data 18
Address 0x120e05ae0, Data 1
Address 0x120e05ac0, Data 11
Address 0x120e05e20, Data 15
Address 0x120e05e00, Data 1
Address 0x120e05ea0, Data 1
Address 0x120e05e80, Data 3
Address 0x120e05da0, Data 7

======================

Sorted Double Linked List Items (Ascending):

Address 0x120e06040, Data 0
Address 0x120e05ae0, Data 1
Address 0x120e05e00, Data 1
Address 0x120e05ea0, Data 1
Address 0x120e06020, Data 3
Address 0x120e05fc0, Data 3
Address 0x120e05e80, Data 3
Address 0x120e05fe0, Data 4
Address 0x120e05f40, Data 7
Address 0x120e05be0, Data 7
Address 0x120e05da0, Data 7
Address 0x120e06060, Data 8
Address 0x120e05bc0, Data 8
Address 0x120e06000, Data 9
Address 0x120e05ba0, Data 9
Address 0x120e06080, Data 10
Address 0x120e05f60, Data 10
Address 0x120e05c00, Data 10
Address 0x120e05b80, Data 11
Address 0x120e05ac0, Data 11
Address 0x120e05f20, Data 12
Address 0x120e05c20, Data 12
Address 0x120e05fa0, Data 13
Address 0x120e05c40, Data 13
Address 0x120e05b60, Data 13
Address 0x120e05b40, Data 14
Address 0x120e05f80, Data 15
Address 0x120e05b20, Data 15
Address 0x120e05e20, Data 15
Address 0x120e05b00, Data 18
```