21BDS0340

Abhinav Dinesh Srivatsa

Data Structures and Algorithms

Assignment – I

1. Stack Data Structure

Aim

To make a stack and discuss its application.

Algorithm

Push:

Data is added to the stack

Pop:

The latest piece of data is removed from the stack

Peek:

The element to be popped is displayed

Code

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

int top;
int *stack;
int size;

void init(int n)
{
    size = n;
    stack = malloc(size * sizeof(int));
    top = -1;
```

```
}

void del()
{
    free(stack);
}

bool isEmpty()
{
    return top == -1 ? true : false;
}

bool isFull()
{
    return top == size - 1 ? true : false;
}

bool push(int n)
{
    if (isFull())
        return false;
    *(stack + ++top) = n;
    return true;
}

int pop()
{
    if (isEmpty())
        return -1;
    return *(stack + top--);
}

int peek()
{
    if (isEmpty())
        return -1;
    return *(stack + top);
}

int display()
{
    if (isEmpty())
        printf("Stack is empty.");
    else
        for (int x = 0; x < top + 1; x++)
            printf("%d ", *(stack + x));
}
```

```c
int main()
{
    int inp;
    printf("Enter size: ");
    scanf("%d", &inp);
    init(inp);
    bool flag = true;
    while (flag)
    {
        printf("Enter 1 to push\n      2 to pop\n      3 to peek\n      4 to
display\n      5 to exit\n");
        scanf("%d", &inp);
        switch (inp)
        {
        case 1:
            scanf("%d", &inp);
            if (push(inp))
                printf("Pushed %d successfully.", inp);
            else
                printf("Stack Overflow.");
            break;

        case 2:
            inp = pop();
            if (inp == -1)
                printf("Stack Underflow.");
            else
                printf("Popped %d successfully.", inp);
            break;

        case 3:
            inp = peek();
            if (inp == -1)
                printf("Stack Underflow.");
            else
                printf("Peek is %d.", inp);
            break;

        case 4:
            display();
            break;

        case 5:
            del();
            flag = false;
        }
        printf("\n");
    }
```

}

Output

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Django\sleepy-rabbit\p2pteach> cd "d:\College Work\Year 2 Semester 1 (Sem 3)\Data Structures and Algorithms Lab\Assignment 1\" ; if ($?) { gcc stack.c -o stack } ; if ($?)
{ .\stack }
Enter size: 5
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
1
1
Pushed 1 successfully.
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
1
2
Pushed 2 successfully.
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
1
3
Pushed 3 successfully.
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
1
4
Pushed 4 successfully.
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
```

```
1
5
Pushed 5 successfully.
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
1
6
Stack Overflow.
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
2
Popped 5 successfully.
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
4
1 2 3 4
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
2
Popped 4 successfully.
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
4
1 2 3
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
2
```

```
Popped 3 successfully.
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
4
1 2
Enter 1 to push
      2 to pop
      3 to peek
      4 to display
      5 to exit
5

PS D:\College Work\Year 2 Semester 1 (Sem 3)\Data Structures and Algorithms Lab\Assignment 1>
```

Result

Stacks are used in various places like solving Towers of Hanoi, finding and solving Postfix and prefix problems.

2. Queue Data Structure

Aim

To make a queue and discuss its applications.

Algorithm

Enqueue:

Add element to queue

Dequeue:

Remove oldest element from queue

Peek:

Display the element closest to removal

Code

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

int top;
int *queue;
int size;

void init(int n){
    size = n;
    queue = malloc(size * sizeof(int));
```

```
    top = -1;
}

void del(){
    free(queue);
}

bool isEmpty(){
    return top == -1 ? true : false;
}

bool isFull(){
    return top == size - 1 ? true : false;
}

bool enqueue(int n){
    if(isFull())
        return false;
    *(queue + ++top) = n;
    return true;
}

int dequeue(){
    if(isEmpty())
        return -1;
    int temp = *queue;
    for(int x = 0; x < top; x++)
        *(queue + x) = *(queue + x + 1);
    top--;
    return temp;
}

int peek(){
    if(isEmpty())
        return -1;
    return *queue;
}

void display(){
    if(top == -1)
        printf("Queue is empty.");
    for(int x = 0; x < top + 1; x++)
        printf("%d ", *(queue + x));
}

int main()
{
    int inp;
```

```
    printf("Enter size: ");
    scanf("%d", &inp);
    init(inp);
    bool flag = true;
    while (flag)
    {
        printf("Enter 1 to enqueue\n      2 to dequeue\n      3 to
peek\n      4 to display\n      5 to exit\n");
        scanf("%d", &inp);
        switch (inp)
        {
        case 1:
            scanf("%d", &inp);
            if (enqueue(inp))
                printf("Queued %d successfully.", inp);
            else
                printf("Queue Overflow.");
            break;

        case 2:
            inp = dequeue();
            if (inp == -1)
                printf("Queue Underflow.");
            else
                printf("Dequeued %d successfully.", inp);
            break;

        case 3:
            inp = peek();
            if (inp == -1)
                printf("Queue Underflow.");
            else
                printf("Dequeued is %d.", inp);
            break;

        case 4:
            display();
            break;

        case 5:
            del();
            flag = false;
        }
        printf("\n");
    }
}
```

Output

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Django\sleepy-rabbit\p2pteach> cd "d:\College Work\Year 2 Semester 1 (Sem 3)\Data Structures and Algorithms Lab\Assignment 1\" ; if ($?) { gcc queue.c -o queue } ; if ($?)
{ .\queue }
Enter size: 5
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
1
1
Queued 1 successfully.
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
1
2
Queued 2 successfully.
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
1
3
Queued 3 successfully.
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
1
4
Queued 4 successfully.
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
```

```
1
5
Queued 5 successfully.
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
1
6
Queue Overflow.
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
2
Dequeued 1 successfully.
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
3
Dequeued is 2.
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
4
2 3 4 5
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
2
Dequeued 2 successfully.
Enter 1 to enqueue
      2 to dequeue
      3 to peek
      4 to display
      5 to exit
5
```

Result

Queues are used in various applications, like waiting lists, CPU scheduling and music players.

3. List Data Structure

Aim

To make a list and discuss its applications

Algorithm

Append:

Add an element to the end of the list

Insert:

Add an element at any position in the list

Delete:

Remove an element from a certain index in the list

Display:

Show all the elements in the list

Code

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

struct Node
{
    int val;
    struct Node *next;
};

struct Node *root;
struct Node *curr;
struct Node *last;
int length;

void init()
{
```

```
    length = 0;
    root = malloc(sizeof(struct Node));
    last = root;
}

void del()
{
    struct Node *temp;
    curr = root;
    while (curr != NULL)
    {
        temp = curr;
        curr = curr->next;
        free(temp);
    }
}

void append(int n)
{
    last->val = n;
    last->next = malloc(sizeof(struct Node));
    last = last->next;
    last->next = NULL;
    length++;
}

bool insert(int n, int i)
{
    if (i > length + 1 || i < 0)
        return false;
    else if (i == 0)
    {
        curr = malloc(sizeof(struct Node));
        curr->val = n;
        curr->next = root;
        root = curr;
        length++;
    }
    else if (i == length + 1)
        append(n);
    else
    {
        curr = root;
        for (int x = 0; x < i - 1; x++)
            curr = curr->next;
        struct Node *temp = curr->next;
        curr->next = malloc(sizeof(struct Node));
        curr->next->val = n;
```

```
        curr->next->next = temp;
        length++;
    }
}

bool delete (int i)
{
    if (i > length - 1 || i < 0)
        return false;
    if (i == 0)
    {
        curr = root;
        root = root->next;
        free(curr);
        length--;
        return true;
    }
    curr = root;
    for (int x = 0; x < i - 1; x++)
        curr = curr->next;
    struct Node *temp = curr->next;
    curr->next = curr->next->next;
    free(temp);
    length--;
    return true;
}

void display()
{
    curr = root;
    while (curr->next != NULL)
    {
        printf("%d ", curr->val);
        curr = curr->next;
    }
}

int main()
{
    init();
    int choice, temp;
    bool flag = true;
    while (flag)
    {
        printf("Enter 1 to append\n      2 to insert\n      3 to
delete\n      4 to Display\n      5 to exit\n");
        scanf("%d", &choice);
        switch (choice)
```

```
        {
        case 1:
            scanf("%d", &choice);
            append(choice);
            printf("Appended %d successfully.", choice);
            break;

        case 2:
            scanf("%d%d", &choice, &temp);
            if (insert(choice, temp))
                printf("Inserted %d at position %d successfully.", choice,
temp);
            else
                printf("Index not valid.");
            break;

        case 3:
            scanf("%d", &choice);
            if (delete (choice))
                printf("Item at index %d deleted successfully.", choice);
            else
                printf("Index not valid.");
            break;

        case 4:
            display();
            break;

        case 5:
            flag = false;
        }
        printf("\n\n");
    }
    del();
}
```

Output

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Django\sleepy-rabbit\p2pteach> cd "d:\College Work\Year 2 Semester 1 (Sem 3)\Data Structures and Algorithms Lab\Assignment 1\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempC
odeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Enter 1 to append
      2 to insert
      3 to delete
      4 to Display
      5 to exit
1
1
Appended 1 successfully.

Enter 1 to append
      2 to insert
      3 to delete
      4 to Display
      5 to exit
2
0
0
Inserted 0 at position 0 successfully.

Enter 1 to append
      2 to insert
      3 to delete
      4 to Display
      5 to exit
4
0 1

Enter 1 to append
      2 to insert
      3 to delete
      4 to Display
      5 to exit
1
2
Appended 2 successfully.

Enter 1 to append
      2 to insert
```

```
5
1
Inserted 5 at position 1 successfully.

Enter 1 to append
      2 to insert
      3 to delete
      4 to Display
      5 to exit
4
0 5 1 2

Enter 1 to append
      2 to insert
      3 to delete
      4 to Display
      5 to exit
2
2
4
Inserted 2 at position 4 successfully.

Enter 1 to append
      2 to insert
      3 to delete
      4 to Display
      5 to exit
3
2
Item at index 2 deleted successfully.

Enter 1 to append
      2 to insert
      3 to delete
      4 to Display
      5 to exit
4
0 5 2 2

Enter 1 to append
      2 to insert
      3 to delete
      4 to Display
      5 to exit
5
```

Result

Lists can be used to implement other data structures, dynamic memory allocation and for maintaining multiple entries of anything.