Stack & Queue in C

Stack Program in C using Array

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

// N will be the capacity of the Static Stack
#define N 100
typedef struct {
    char name[10];
    int grade;
} st;
// Initializing the top of the stack to be -1
int top = -1;
// Initializing the stack using an array
st stack[N];

// Function prototypes
void push();        // Push element to the top of the stack
st pop();          // Remove and return the top most element of the stack
st peek();         // Return the top most element of the stack
void display();     // Display the elements of the stack
bool isEmpty();     // Check if the stack is in Underflow state or not
bool isFull();      // Check if the stack is in Overflow state or not
```

```
int main()
{
    int choice;
    while (1)
    {
        printf("\nChoose an option:");
        printf("\n1.Push the element\n2.Pop the element\n3.Show
                \n4.Peek the topmost element\n0.Exit");
        printf("\n\nEnter the choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            push();
            break;
        case 2:
            if (isEmpty())
                {
                    printf("\nStack is empty!");
                }
                else
                { st x = pop();
                  printf("%s: %d ", x.name, x.grade);
                }
            break;
        case 3:
            display();
            break;
        case 4:
            if (isEmpty())
                {
                    printf("\nStack is empty!");
                }
                else
                { st x = peek();
                  printf("The topmost element is:\n%s: %d ", x.name, x.grade);
                }
            break;
        case 0:
            exit(0);
        default:
            printf("\nWrong choice!");
        }
    }
return 0;
}
```

```
void push()
{
    if (isFull())
    {
        printf("\nStack is Full!");
    }
    else
    {
        top += 1;
        printf("\n---Enter the element to be added into the stack---");
        printf("\nEnter the name: ");
        scanf("%s",stack[top].name);
        printf("Enter the mark: ");
        scanf("%d", &stack[top].grade);
    }
}
```

```
st pop()
{
    st item = stack[top];
    top -= 1;
    return item;
}

st peek(){
    return stack[top];
}
```

```
void display()
{
    if (isEmpty())
    {
        printf("\nStack is empty!");
    }
    else
    {
        printf("\nStack's elements: \n");
        for (int i = top; i >= 0; --i)
            printf("%s: %d\n", stack[i].name, stack[i].grade);
    }
}
```

```
bool isFull()
{
 if(top == N-1)
    return true;
 else
    return false;
}

bool isEmpty()
{
 if(top == -1)
    return true;
 else
    return false;
}
```

Queue Program in C using Array

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define N 10

typedef struct {
    char nameDoc[15];
    int page;
} print;

print queue[N];
int last = - 1;
int first = - 1;

void enqueue();
void dequeue();
void display();
bool isFull();
bool isEmpty();
```

```
int main()
{
    int choice;
    while (1)
    {
        printf("\nChoose an option:\n");
        printf("1.Insert element to queue \n");
        printf("2.Delete element from queue \n");
        printf("3.Display all elements of queue \n");
        printf("0. Exit \n");
        printf("\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                enqueue(); break;
            case 2:
                dequeue(); break;
            case 3:
                display(); break;
            case 0:
                exit(0);
            default:
                printf("Wrong choice \n");
        } /* End of switch */
    } /* End of while */
    return 0;
} /* End of main() */
```

```
void enqueue()
{
    print item;
    if (isFull())
    {
        printf("Queue is Full!\n");
    }
    else
    {
        if (isEmpty())
        /*If queue is initially empty */
        first = 0;
        printf("---Insert the element in queue---\n");
        printf("Document name: ");
        scanf("%s", item.nameDoc);
        printf("Document pages: ");
        scanf("%d", &item.page);
        last += 1;
        queue[last] = item;
    }
} /* End of enqueue() */
```

```
void dequeue()
{
    if (isEmpty())
    {
        printf("Queue is empty \n");
    }
    else
    {
        printf("Element deleted from queue is : %s -> %d\n",
queue[first].nameDoc, queue[first].page);
        first += 1;
    }
} /* End of dequeue() */
```

```
void display()
{
    if (isEmpty())
        printf("Queue is empty \n");
    else
    {
        printf("Queue elements is : \n");
        for (int i = first; i <= last; ++i)
            printf("%s -> %d\n", queue[i].nameDoc, queue[i].page);
        printf("\n");
    }
} /* End of display() */
```

```
bool isFull()
{
 if(last == N-1)
    return true;
 else
    return false;
}

bool isEmpty()
{
 if(first == - 1 || first > last)
    return true;
 else
    return false;
}
```

Dynamic Stack / Stack Program in C using Linked List

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

// Function prototypes
void push();       // Push element to the top of the stack
void pop();         // Remove and return the top most element of the stack
void peek();        // Return the top most element of the stack
bool isEmpty();    // Check if the stack is in Underflow state or not
void display();

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

struct node *head = NULL;
```

```c
int main (){
    int choice;
    while(1){
        printf("\nChoose an option:");
        printf("\n1: Push \n2: Pop \n3: Peek \n4: Display \n0: Exit\n");
        scanf("%d",&choice);
        switch(choice){
            case 1: push(); break;
            case 2: pop(); break;
            case 3: peek(); break;
            case 4: display(); break;
            case 0: exit(0);
            default: printf("Wrong option!\n");
        }
    }
return 0;
}
```

```c
bool isEmpty(){
    if(head == NULL){
        return true;
    }
    return false;
}
```

```c
void peek(){
    if (isEmpty())
        printf("The stack is empty\n");
    else
        printf("%d is the top most element of the stack\n", head->val);
}
```

```c
void push (){
    int val;
    struct node *ptr = (struct node*) malloc (sizeof(struct node));
    printf("Enter the value to be pushed: ");
    scanf("%d", &val);

    if(isEmpty()){
        ptr->val = val;
        ptr->next = NULL;
        head = ptr;
    }
    else{
        ptr->val = val;
        ptr->next = head;
        head = ptr;
    }
}
```

```c
void pop(){
    struct node *ptr;
    if (isEmpty())
        printf("Stack is empty!\n");
    else{
        ptr = head;
        head = head->next;
        free(ptr);
    }
}
```

```c
void display(){
    struct node *ptr = head;
    if (isEmpty())
        printf("Stack is empty!\n");
    else {
        while(ptr != NULL){
        printf(" %d", ptr->val);
        ptr = ptr->next;
        }
    }
}
```

Dynamic Queue / Queue Program in C using Linked List

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

struct node
{
    int val;
    struct node *next;
} *first = NULL, *last = NULL;

void display();
void enqueue();
void dequeue();

int main()
{
    int ch;
    do
    {
        printf("\nQueue Menu\n1. Add element \n2. Remove element\n
                3. Display\n0. Exit\n");
        printf("\nChoose an option: ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
        }
    }while (ch != 0);
return 0;
}

void enqueue()
{
    struct node *ptr = (struct node*) malloc (sizeof(struct node));
    printf("\nEnter a value: ");
    scanf("%d", &ptr->val);
    ptr->next = NULL;
    if (last == NULL)
    {
        first = ptr;
        last = ptr;
    }
    else
    {
        last->next = ptr;
        last = last->next;
    }
}
```

```
void dequeue()
{
    if (first == NULL)
    {
        printf("\nQueue is empty!\n");
    }
    else
    {
        struct node *temp;
        temp = first;
        first = first->next;
        printf("\n%d deleted", temp->val);
        free(temp);
    }
}
```

```
void display()
{
    if (first == NULL)
    {
        printf("\nQueue is empty!\n");
    }
    else
    {
        struct node *temp;
        temp = first;
        printf("\n");
        while (temp != NULL)
        {
            printf("%d\t", temp->val);
            temp = temp->next;
        }
    }
}
```

Reverse a String

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

void push(char ch);
void pop(char *ptWord);
bool isEmpty();
void reverseWord(char *word);

struct node{
    char ch;
    struct node *next;
};

struct node *head = NULL;
```

```
int main (){
    char word[50];
    printf("Enter the word\n");
    //scanf("%s", word);
    fgets(word, 50, stdin);
    reverseWord(word);
return 0;
}
```

```
void reverseWord(char *word){
    for (int i = 0; i < strlen(word); ++i){
        push(*(word+i));
    }
    for (int i = 0; i < strlen(word); ++i){
        pop(word+i);
    }
    puts(word);
}
void push (char ch){
    struct node *ptr = (struct node*) malloc (sizeof(struct node));

    if(isEmpty()){
        ptr->ch = ch;
        ptr->next = NULL;
        head = ptr;
    }
    else{
        ptr->ch = ch;
        ptr->next = head;
        head = ptr;
    }
}
```

```c
void pop(char *ptWord){
    struct node *ptr;
    if (isEmpty())
        printf("Stack is empty!\n");
    else{
        ptr = head;
        head = head->next;
        *(ptWord) = ptr->ch;
        free(ptr);
    }
}
bool isEmpty(){
    if(head == NULL){
        return true;
    }
    return false;
}
```