Stack

#include <stdio.h>

void print\_current\_queue();

int stack\_full();

int stack\_empty();

int pop();

int push(int num);

int stack[10];

int top = -1;

int main(){

    print\_current\_queue();

    int check1;

    for (int i = 0; i<11; i++){

        check1 = push(i);

        if (check1 == 0){

            printf("failed to push %d\n", i);

        }

        else{

            print\_current\_queue();

        }

    }

    int check2;

    for (int i = 0; i<11; i++){

        check2 = pop();

        if (check2 != 0){

            print\_current\_queue();

        }

    }

    return 0;

}

void print\_current\_queue(){

    if (top==-1){

        printf("The stack is empty\n");

    }

    else{

        for (int i = 0; i<top+1; i++)

            printf("%d ", stack[i]);

        printf("\n");

    }

}

int stack\_full(){

    if (top == 9){

        return 1;   //full

    }

    return 0;       //not full

}

int stack\_empty(){

    if (top==-1){

        return 1;   //empty

    }

    return 0;       //not empty

}

int push(int num){

    if (stack\_full()!=1){

        stack[++top] = num;

        return 1;   //success to push

    }

    return 0;       //failed to push

}

int pop(){

    if (stack\_empty()!=1){

        printf("popped number : %d\n", stack[top]);

        stack[top--] = NULL;

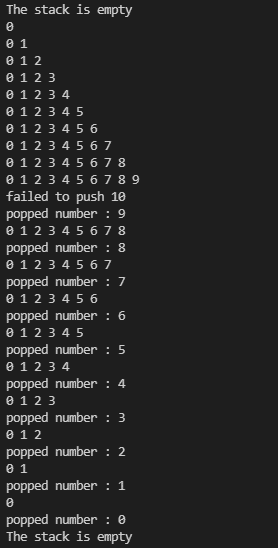
        return 1;   //success to pop

    }

    return 0;       //failed to pop

}

Result :



Queue

#include <stdio.h>

int queue\_full();

int queue\_empty();

int enqueue(int num);

int dequeue();

void print\_the\_queue();

int queue[10];

int front = -1;

int rear = -1;

int main(){

    int check1;

    dequeue();

    for (int i = 0; i<11; i++){

        check1 = enqueue(i);

        if (check1!=0){

            print\_the\_queue();

        }

    }

    int check2;

    for (int i = 0; i<11; i++){

        check2 = dequeue();

        if (check2!=0){

            print\_the\_queue();

        }

    }

}

int queue\_full(){

    if (rear==9)

        return 1;       //full

    return 0;           //not full

}

int queue\_empty(){

    if (rear==-1)

        return 1;       //empty

    return 0;           //not empty

}

int enqueue(int num){

    if (rear==-1){

        queue[++front] = num;

        rear++;

        return 1;       //success to enqueue num

    }

    if (queue\_full()!=1){

        queue[++rear] = num;

        return 1;       //success to enqueue num

    }

    printf("failed to enqueue %d\n", num);

    return 0;           //failed to enqueue num

}

int dequeue(){

    if (queue\_empty()==1){

        printf("failed to dequeue\n");

        return 0;   //failed to dequeue

    }

    for (int i = 0; i<rear; i++){

        queue[i] = queue[i+1];

    }

    queue[rear--] = NULL;

    return 1;       //success to dequeue

}

void print\_the\_queue(){

    if (rear==-1){

        printf("the queue is empty\n");

    }

    else{

        for (int i = 0; i<rear+1; i++){

            printf("%d ", queue[i]);

        }

        printf("\n");

    }

}

Result :

화살이(가) 표시된 사진

자동 생성된 설명