**STACK & QUEUE**

**STACK:**

Last In First Out

Stack will grow from bottom to top.

If top reaches we can stop adding data.

If the bottom is equal to top then the container is empty.

If the top is equal to limit then the container is full.

In stack we need to declare the size/limit and bottom also.

**How to avoid multiple definition of variables?**

**EX:**

#include <stdio.h>

#include<stdlib.h>

#define MAX 5

int bottom=-1;

int top=-1;

int push(int);

int pop();

void dispStack();

int stackCount[MAX];

int push(int v)

{

if(top == MAX-1)

{

printf("\nStack is Full\n");

return 0;

}

top++;

stackCount[top]=v;

return 0;

}

int pop()

{

int v;

if(top == bottom)

{

printf("\nStack is Empty\n");

return 0;

}

v=stackCount[top];

top--;

return v;

}

void dispStack()

{

int iter;

if(top == bottom)

{

printf("\nStack is Empty\n");

return;

}

printf("\nStack elements are:\n");

for(iter=top; iter>bottom;iter--)

{

printf("\n%d",stackCount[iter]);

}

printf("\n\n");

}

int main() {

//STACK USING ARRAYS

int v;

/\* v=pop();

printf("\n%d Element is deletde from the stack",v);

\*/

push(10);

push(20);

push(30);

push(40);

push(50);

dispStack();

for(int i=0;i<MAX;i++)

{

v=pop();

printf("\n%d Element is deletde from the stack",v);

dispStack();

}

printf("\n\n");

return 0;

}

**EX:**

#include <stdio.h>

#include <stdlib.h>

#define CAP 5

int top, bottom;

int push(int [], int);

void dispStack(int []);

void pop();

int main()

{

int arr[CAP];

int ch=1, ele;

top = 0;

bottom = 0;

while(ch)

{

printf("\nEnter the Value to be insert in stack: ");

scanf("%d",&ele);

push(arr, ele);

printf("\nDo you want to add/insert new element to the stack (0/1): ");

scanf("%d",&ch);

}

dispStack(arr);

pop(arr);

dispStack(arr);

pop(arr);

dispStack(arr);

pop(arr);

dispStack(arr);

pop(arr);

dispStack(arr);

pop(arr);

dispStack(arr);

printf("\n\n");

return 0;

}

int push(int stackArr[], int ele)

{

if(top > (CAP-1))

{

printf("\nStack is full\n");

return 0;

}

stackArr[top] = ele;

top++;

return 1;

}

void pop(int a[])

{

if(top == bottom)

{

printf("\nStack is Empty\n");

return ;

}

else

{

printf("\n%d element is popped out successfully",a[top-1]);

top--;

}

}

void dispStack(int stackArr[])

{

int i;

if(top == bottom)

{

printf("\nStack is Empty\n");

return ;

}

printf("\nELements in the stack are\n");

printf("\t\t");

for(i=top-1;i>=0;i--)

printf("%d ",stackArr[i]);

printf("\n\n");

}

**QUEUE:**

If front is equal to rear then the queue is empty.

Rear is increasing in queue.

If rear is equal to MAX then the queue is full.

While deleting the front is growing.

If front is equal to MAX then the queue is empty.

The rare is present at the MAX so we can say it as, if front is equal to rare then the queue is empty.

After doing this we have to reset the rare and front---🡪 f=0,r=0.

For displaying we start from the front to the rare.

**EX:**

#include <stdio.h>

#define MAX 5

int rear=0;

int front=0;

int q[MAX];

void dequeue();

void enqueue(int );

void dispQueue();

void dispQueue()

{

int i;

if(front == rear)

{

printf("\nEmpty queue\n");

return;

}

printf("\nQueue is:\n");

for(i=front;i<rear;i++)

{

printf("%d ",q[i]);

}

}

void enqueue(int val)

{

if(rear == MAX)

{

printf("\nQueue is full\n");

return;

}

q[rear]=val;

rear++;

}

void dequeue()

{

if(front == rear)

{

front=0;//reset the queue

rear=0;

printf("\nQueue is Empty\n");

return;

}

printf("\n%d is dequeued from the queue\n",q[front]);

front++;

}

int main() {

enqueue(10);

enqueue(20);

enqueue(30);

enqueue(40);

enqueue(50);

dispQueue();

dequeue();

dequeue();

dequeue();

dequeue();

dequeue();

dequeue();

dispQueue();

printf("\n\n");

return 0;

}

Create the Employee records dynamically display it using stack operation.