Data Structures in C

Kim Pham

What are Data Structures

- Ways to arrange data
- Arrays, stacks, queues

Arrays

- Linear data structure
- Elements accessible through indexes
- Datatype varname [size] ;
- Datatype varname [] = {ele1, ele2, ele3, ele4};

Arrays

- Replacement of multiple variables
- Easy sorting/iteration

- Fixed size
- Hard to delete/insert
- If array is too big, then array is wasted

Stacks

- Kind of like a stack of plates; only top elements are available to be accessed
- Insertion/deletion takes place from the top
- push(element)
- Pop()
- Show()

```c
#define SIZE 4

int top = -1, inp_array[SIZE];
```

```c
void push()
{
    int x;

    if (top == SIZE - 1)
    {
        printf("\nOverflow!!");
    }
    else
    {
        printf("\nEnter the element to be added onto the stack: ");
        scanf("%d", &x);
        top = top + 1;
        inp_array[top] = x;
    }
}
```

```c
void pop()
{
    if (top == -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nPopped element: %d", inp_array[top]);
        top = top - 1;
    }
}
```

```c
void show()
{
    if (top == -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nElements present in the stack: \n");
        for (int i = top; i >= 0; --i)
            printf("%d\n", inp_array[i]);
    }
}
```

```
printf("\nPerform operations on the stack:");
printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End");
printf("\n\nEnter the choice: ");
scanf("%d", &choice);

switch (choice)
{
case 1:
    push();
    break;
case 2:
    pop();
    break;
case 3:
    show();
    break;
case 4:
    exit(0);

default:
    printf("\nInvalid choice!!");
}
```

Stacks

- Most recent element readily for use
- Fast! O(1) complexity
- Hard to manipulate
- Not really flexible

Queues

- Kind of like a line; bottom/top can get removed
- enqueue(element)
- dequeue()
- Show()

```
# define SIZE 100
void enqueue();
void dequeue();
void show();
int inp_arr[SIZE];
```

```c
void enqueue()
{
    int insert_item;
    if (Rear == SIZE - 1)
       printf("Overflow \n");
    else
    {
        if (Front == - 1)

        Front = 0;
        printf("Element to be inserted in the Queue\n : ");
        scanf("%d", &insert_item);
        Rear = Rear + 1;
        inp_arr[Rear] = insert_item;
    }
}
```

```c
void dequeue()
{
    if (Front == - 1 || Front > Rear)
    {
        printf("Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from the Queue: %d\n", inp_arr[Front]);
        Front = Front + 1;
    }
}
```

```c
void show()
{

    if (Front == - 1)
        printf("Empty Queue \n");
    else
    {
        printf("Queue: \n");
        for (int i = Front; i <= Rear; i++)
            printf("%d ", inp_arr[i]);
        printf("\n");
    }
}
```