**A picture containing text, clipart

Description automatically generated**

**Data Structure practical FILE**



**Submitted to: Submitted by:**

**MR. SHIVASHISH ABHAY SINGH RAWAT**

**University Roll. No.:** **2018062 Class Roll. No./Section: 02/L**



**Q1.** Write a the C program to create an array by inserting N elements in it then find second non repeating element from the array.

# ANS:-

#include <stdio.h>

int main()

{

printf("ABHAY SINGH RAWAT \n");

int n,c=0;

printf("enter the size of array\n");

scanf("%d",&n);

int a[n];

printf("enter the element\n");

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

{

scanf("%d",&a[i]);

}

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

{

int f=0;

for(int j=0;j<n;j++)

{

if(a[i]==a[j] && i!=j)

{

f=1;

break;

}

}

if(f==0)

{

c++;

}

if(c==2)

{

printf("second non repeating element %d",a[i]);

break;

}

}

return 0;

}

# OUTPUT:-

# Text Description automatically generated

**Q2.** Write a the C program to create an array by inserting N elements in it then find third repeating element from the array.

# ANS:-

#include<stdio.h>

int main()

{

printf("ABHAY SINGH RAWAT \n");

int n,i,j,c,d=0;

printf("Enter the limit of array: ");

scanf("%d",&n);

int a[n];

printf("\nEnter the elements:\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

for(i=0;i<n;i++)

{

c=0;

for(j=0;j<n;j++)

{

if(a[j]!=-1&&i!=j)

{

if(a[i]==a[j]&&i!=j)

{

c++;

a[j]=-1;

}

}

}

if(c>0)

d++;

if(d==3)

{

break;

}

}

if(d==3)

printf("\nThird repeating element is %d",a[i]);

else

printf("\nThird repeating element not found");

return 0;

}

# OUTPUT:-

# Text Description automatically generated

**Q3.** Write a C program Create a Dynamic array and then Reverse the array using recursion and then finally print the array.

# ANS:-

#include<stdio.h>

#include<stdlib.h>

void Input();

void Reverse(int \*,int,int);

void print(int \*,int);

int main()

{

printf("ABHAY SINGH RAWAT\n");

Input();

return 0;

}

void Input()

{

int num,i=0;

printf("Enter the size of the array : ");

scanf("%d",&num);

int \*arr=(int\*)malloc(sizeof(int)\*num);

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

{

printf("Enter the number %d :",(i+1));

scanf("%d",&arr[i]);

}

Reverse(arr,0,num-1);

print(arr,num);

}

void Reverse(int \*arr,int start, int end)

{

int store;

if(start>=end)

return;

store=arr[start];

arr[start]=arr[end];

arr[end]=store;

Reverse(arr,start+1,end-1);

}

void print(int \*arr,int num)

{

printf("Array elements after reversing are : ");

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

{

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

}

}

# OUTPUT:-

**Q4.** Write a C Program implement STACK using array in menu driven form.

# ANS:-

#include<stdio.h>

#include<stdlib.h>

#define size 5

int Insert(int \*arr,int top)

{

if(top==size-1)

{

printf("STACK IS FULL \n");

}

else

{

int num;

printf("Enter the value: \n");

scanf("%d",&num);

top++;

arr=(int \*)realloc(arr,sizeof(int)\*(top+1));

arr[top]=num;

printf("ADDED %d SUCCESSFULLY\n",arr[top]);

return top;

}

}

int Delete(int \*arr,int top)

{

if(top==-1)

{

printf("STACK IS EMPTY\n");

}

else{

printf("DELETED %d SUCCESSFULLY\n",arr[top]);

top--;

arr=(int \*)realloc(arr,sizeof(int)\*(top+1));

return top;

}

}

void Display(int \*arr,int top)

{

if(top==-1)

{

printf("STACK IS EMPTY\n");

}

else

{

printf("Stack is \n");

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

{

printf("----------\n");

printf("| %d |\n",arr[i]);

printf("----------\n");

}

}

}

int main()

{

printf("ABHAY SINGH RAWAT \n");

int \*stack,top=-1,counter=0,choice;

stack=(int\*)malloc(sizeof(int )\*(top+1));

while(counter==0)

{

printf("1.INSERT \n2.DELETE \n3.DISPLAY \n4.EXIT \n");

scanf("%d",&choice);

switch(choice)

{

case 1:

top=Insert(stack,top);

break;

case 2:

top=Delete(stack,top);

break;

case 3:

Display(stack,top);

break;

case 4:

printf("EXITED \n");

counter++;

break;

default:

printf("WRONG CHOICE \n");

}

}

return 0;

}

# OUTPUT:-

**Q5.** Write a C Program to Convert Infix to Postfix Expression using Stack.

# ANS: -

#include<stdio.h>

#include<ctype.h>

void push(char stack[],int \*top,char x)

{

\*top=\*top+1;

stack[\*top] = x;

}

char pop(char stack[],int \*top)

{

if(\*top == -1)

return -1;

else

{

char temp= stack[\*top];

\*top=\*top-1;

return temp;

}

}

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

if(x == '^')

return 3;

return 0;

}

int main()

{

printf("ABHAY SINGH RAWAT \n");

char stack[100];

int top = -1;

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

while(\*e !='\0')

{

if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(stack,&top,\*e);

else if(\*e == ')')

{

while((x = pop(stack,&top)) != '(')

printf("%c ", x);

}

else

{

while(priority(stack[top]) >= priority(\*e))

printf("%c ",pop(stack,&top));

push(stack,&top,\*e);

}

e++;

}

while(top != -1)

{

printf("%c ",pop(stack,&top));

}return 0;

}

# OUTPUT:-

**Q6.** Write a C Program to create singly linked list by adding nodes in the right hand side and delete alternate node from the list and then print the final list.

# ANS:-

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

typedef struct Node node;

void insertAtRightSide(node \*\*,int);

void deleteAtAlternate(node\*);

void display(node \*head);

int main()

{

printf("ABHAY SINGH RAWAT \n");

node \*head;

int counter=0,choice,value,pos;

head=NULL;

while(counter==0)

{

printf("1.INSERT AT RIGHT \n2.DELETE ALTERNATE NODES \n3.DISPLAY \n4.EXIT\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter a value : ");

scanf("%d",&value);

insertAtRightSide(&head,value);

break;

case 2:

deleteAtAlternate(head);

display(head);

break;

case 3:

display(head);

break;

case 4:

printf("EXITED\n");

counter++;

break;

default:

printf("WRONG CHOICE\n");

break;

}

}

return 0;

}

void insertAtRightSide(node \*\*head,int value)

{

node \*temp=(node\*)malloc(sizeof(node));

temp->data=value;

if(\*head==NULL)

{

temp->next=\*head;

\*head=temp;

}

else

{

node \*temp2=\*head;

while(temp2->next!=NULL)

{

temp2=temp2->next;

}

temp2->next=temp;

temp->next=NULL;

}

}

void display(node \*head)

{

node \*temp;

temp=head;

printf("List is : ");

while(temp!=NULL)

{

printf(" %d ",temp->data);

temp=temp->next;

}

printf("\n");

}

void deleteAtAlternate(node\* head)

{

if (head == NULL)

return;

node \*temp=head->next;

if(temp==NULL)

return;

head->next=temp->next;

free(temp);

deleteAtAlternate(head->next);

}

# OUTPUT:-

**Q7.** Write a C Program implement STACK using Link List in menu driven form.

# ANS:-

#include<stdio.h>

#include<stdlib.h>

#define max 5

struct Node

{

int data;

struct Node \*next;

};

typedef struct Node node;

void insert(node \*\*,int,int\*);

void delete(node\*\*,int\*);

void display(node \*top);

int main()

{

printf("ABHAY SINGH RAWAT \n");

node \*top;

int counter=0,choice,value,pos,sizeRef=-1;

top=NULL;

while(counter==0)

{

printf("1.INSERT \n2.DELETE \n3.DISPLAY \n4.EXIT\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

if(sizeRef==max-1)

{

printf("Stack is Full\n");

}

else{

printf("Enter a value : ");

scanf("%d",&value);

insert(&top,value,&sizeRef);

}

break;

case 2:

if(sizeRef==-1)

{

printf("Stack is Empty\n");

}

else{

delete(&top,&sizeRef);

}

break;

case 3:

display(top);

break;

case 4:

printf("EXITED\n");

counter++;

break;

default:

printf("WRONG CHOICE\n");

break;

}

}

return 0;

}

void insert(node \*\*top,int value,int \*sizeRef)

{

\*sizeRef=\*sizeRef+1;

node \*temp=(node\*)malloc(sizeof(node));

temp->data=value;

temp->next=\*top;

\*top=temp;

}

void display(node \*top)

{

node \*temp;

temp=top;

printf("List is : ");

while(temp!=NULL)

{

printf(" %d ",temp->data);

temp=temp->next;

}

printf("\n");

}

void delete(node\*\*top,int \*sizeRef)

{

if (top == NULL)

return;

\*sizeRef=\*sizeRef-1;

node \*temp=\*top;

printf("%d Deleted Successfully\n",temp->data);

\*top=temp->next;

free(temp);

}

# OUTPUT:-

**Q8.** Write a C Program implement QUEUE using Link List in menu driven form.

# ANS:-

#include<stdio.h>

#include<stdlib.h>

#define max 5

struct Node

{

int data;

struct Node \*next;

};

typedef struct Node node;

void insert(node \*\*,int\*,int\*,int);

void delete(node\*\*,int\*,int\*);

void display(node \*top);

int main()

{

printf("ABHAY SINGH RAWAT \n");

node \*top;

int counter=0,choice,value,pos,front=-1,rear=-1;

top=NULL;

while(counter==0)

{

printf("1.INSERT \n2.DELETE \n3.DISPLAY \n4.EXIT\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

if(rear==max-1)

{

printf("Queue is Full\n");

}

else{

printf("Enter a value : ");

scanf("%d",&value);

insert(&top,&front,&rear,value);

}

break;

case 2:

if(front==-1)

{

printf("Queue is Empty\n");

}

else{

delete(&top,&front,&rear);

}

break;

case 3:

display(top);

break;

case 4:

printf("EXITED\n");

counter++;

break;

default:

printf("WRONG CHOICE\n");

break;

}

}

return 0;

}

void insert(node \*\*top,int\* front,int\* rear,int value )

{

if(\*front==-1)

\*front=\*front+1;

node \*temp=(node\*)malloc(sizeof(node));

temp->data=value;

if(\*top==NULL)

{

temp->next=\*top;

\*top=temp;

}

else

{

node \*temp2=\*top;

while(temp2->next!=NULL)

{

temp2=temp2->next;

}

temp2->next=temp;

temp->next=NULL;

}

\*rear=\*rear+1;

}

void display(node \*top)

{

node \*temp;

temp=top;

printf("List is : ");

while(temp!=NULL)

{

printf(" %d ",temp->data);

temp=temp->next;

}

printf("\n");

}

void delete(node\*\*top,int\* front,int\* rear)

{

if (top == NULL)

return;

\*front=\*front+1;

node \*temp=\*top;

printf("%d Deleted Successfully\n",temp->data);

\*top=temp->next;

free(temp);

if(\*front>\*rear)

\*front=\*rear=-1;

}

# OUTPUT:-

**Q9.** Write a C Program implement priority QUEUE using array in menu driven form.

# ANS:-

#include <stdio.h>

#include <stdlib.h>

struct priorityqueue

{

int r;

int \*num;

int \*priority;

int size;

};

int isempty(struct priorityqueue \*pq)

{

if (pq->r == -1)

{

return 1;

}

else

{

return 0;

}

}

int isfull(struct priorityqueue \*pq)

{

if (pq->r == pq->size - 1)

{

return 1;

}

else

{

return 0;

}

}

void enqueue(struct priorityqueue \*pq, int val, int p)

{

if (isfull(pq))

{

printf("The queue is full\n");

}

else

{

pq->r++;

pq->num[pq->r]=val;

pq->priority[pq->r] = p;

}

}

int gethighestpriority(struct priorityqueue \*pq)

{

int i, p;

p = -1;

if (!(isempty(pq)))

{

for (i = 0; i <= pq->r; i++)

{

if (pq->priority[i] > p)

{

p = pq->priority[i];

}

}

}

return p;

}

int deletehighestpriority(struct priorityqueue \*pq)

{

int i, j, x, a;

x = gethighestpriority(pq);

for (i = 0; i <= pq->r; i++)

{

if (pq->priority[i] == x)

{

a = pq->num[i];

break;

}

}

if (i < pq->r)

{

for (j = i; j < pq->r; j++)

{

pq->num[j] = pq->num[j + 1];

pq->priority[j] = pq->priority[j + 1];

}

}

pq->r--;

return a;

}

void display(struct priorityqueue \*pq)

{

int i;

printf("Priority queue: \n");

for (i = 0; i <= pq->r; i++)

{

printf("Element: %d Priority: %d\n", pq->num[i], pq->priority[i]);

}

}

int main()

{

printf("ABHAY SINGH RAWAT \n");

struct priorityqueue pq;

printf("Enter the size of your priority queue: ");

scanf("%d", &pq.size);

pq.r = -1;

pq.num = (int \*)malloc(pq.size \* sizeof(int));

pq.priority = (int \*)malloc(pq.size \* sizeof(int));

int ch, val, p;

do

{

printf("Press 1 to insert\nPress 2 to get highest priority\nPress 3 to delete\nPress 4 to display\nPress 5 to exit\n");

printf("Enter your choice\n");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter element to insert: \n");

scanf("%d", &val);

printf("Enter priority: \n");

scanf("%d", &p);

enqueue(&pq, val, p);

break;

case 2:

if (isempty(&pq))

{

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

}

else

{

p = gethighestpriority(&pq);

printf("The highest priority = %d\n", p);

}

break;

case 3:

if (isempty(&pq))

{

printf("Queue is empty\n");

}

else

{

val = deletehighestpriority(&pq);

printf("%d is deleted\n", val);

}

break;

case 4:

display(&pq);

break;

default:

break;

}

} while (ch != 5);

}

# OUTPUT:-

**Q10.** Write a C Program implement QUEUE using array in menu driven form.

# ANS:-

#include<stdio.h>

#define size 5

void Enqueue(int\*,int\*,int\*);

void Dequeue(int\*,int\*,int\*);

void Display(int\*,int\*,int\*);

int main()

{

printf("ABHAY SINGH RAWAT \n");

int queue[size],front=-1,rear=-1;

int counter=0,choice;

while(counter==0)

{

printf("1.INSERT \n2.DELETE \n3.DISPLAY \n4.EXIT \n");

scanf("%d",&choice);

switch(choice)

{

case 1:

Enqueue(queue,&front,&rear);

break;

case 2:

Dequeue(queue,&front,&rear);

break;

case 3:

Display(queue,&front,&rear);

break;

case 4:

printf("EXITED \n");

counter++;

break;

default:

printf("WRONG CHOICE \n");

break;

}

}

return 0;

}

void Enqueue(int queue[],int \*front ,int \*rear)

{

int value;

if(\*rear==size-1)

{

printf("Queue is Full\n");

}

else

{

printf("Enter a value\n");

scanf("%d",&value);

if(\*front==-1)

{

\*front=\*front+1;

}

\*rear=\*rear+1;

queue[\*rear]=value;

printf("Entered %d SUCCESSFULLY \n",value);

}

}

void Dequeue(int queue[],int \*front,int \*rear)

{

if(\*front==-1)

{

printf("Queue is Empty\n");

}

else

{

printf("Deleted %d Successfully\n",queue[\*front]);

\*front=\*front+1;

if(\*front>\*rear)

{

\*front=\*rear=-1;

}

}

}

void Display(int queue[],int \*front,int \*rear)

{

if(\*front==-1)

{

printf("Queue is Empty\n");

}

else

{

for(int i=\*front;i<=\*rear;i++)

{

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

}

printf("\n");

}

}

# OUTPUT:-

**Q11.** Write a C program to Evaluate Postfix Expression using Stack

# ANS:-

#include<stdio.h>

#include<ctype.h>

#define max 20

void Push(int\*,int\*,int );

int Pop(int \*,int\*);

void Eval\_Postfix(int\*,char[],int\*);

int main()

{

printf("ABHAY SINGH RAWAT n");

int stack[max],top=-1;

char exp[max];

printf("Enter the expression: ");

scanf("%s",exp);

printf("\n");

Eval\_Postfix(stack,exp,&top);

return 0;

}

void Push(int stack[],int \*top,int item)

{

\*top=\*top+1;

stack[\*top]=item;

}

int Pop(int stack[],int \*top)

{

int item=stack[\*top];

\*top=\*top-1;

stack[\*top];

return item;

}

void Eval\_Postfix(int stack[],char exp[],int \*top)

{

int a,b,total;

char \*e;

e=exp;

while(\*e!='\0')

{

total=0;

if(isdigit(\*e))

{

Push(stack,top,\*e-'0');

}

else if(\*e=='+' || \*e=='-' || \*e=='\*' || \*e=='/')

{

b=Pop(stack,top);

a=Pop(stack,top);

switch(\*e)

{

case '+':

total =a+b;

break;

case '-':

total =a-b;

break;

case '\*':

total =a\*b;

break;

case '/':

total =a/b;

break;

}

Push(stack,top,total);

}

e++;

}

printf("The result is : %d",Pop(stack,top));

}

# OUTPUT:-

**Q 12.** Write a C program to create TWO singly linked list L1 and L2 and sort both the list and finally merge both the list such that L2 comes after L1.[ use double pointer]

# ANS:-

#include <stdio.h>

#include <stdlib.h>

struct node

{

int info;

struct node \*next;

};

void push(struct node \*\*head, int val)

{

struct node \*newNode = malloc(sizeof(struct node));

newNode->info = val;

newNode->next = NULL;

if (\*head == NULL)

\*head = newNode;

else

{

struct node \*lastNode = \*head;

while (lastNode->next != NULL)

{

lastNode = lastNode->next;

}

lastNode->next = newNode;

}

}

void sort(struct node \*head)

{

struct node \*temp;

while(head!=NULL)

{

temp=head->next;

while(temp!=NULL)

{

if(head->info>temp->info)

{

int hold=head->info;

head->info=temp->info;

temp->info=hold;

}

temp=temp->next;

}

head=head->next;

}

}

void merge(struct node \*l1,struct node \*l2)

{

while(l1->next!=NULL)

{

l1=l1->next;

}

l1->next=l2;

}

void print(struct node \*ptr)

{

struct node \*temp = ptr;

while (temp != NULL)

{

printf("%d ", temp->info);

temp = temp->next;

}

}

int main()

{

printf("ABHAY SINGH RAWAT \n");

struct node \*l1 = NULL,\*l2 = NULL;

push(&l1,19);

push(&l1,18);

push(&l1,12);

push(&l1,11);

push(&l1,10);

sort(l1);

push(&l2,1);

push(&l2,21);

push(&l2,8);

push(&l2,17);

push(&l2,16);

sort(l2);

merge(l1,l2);

print(l1);

}

# OUTPUT:-

**Q 13.** Write C program to create a doubly link list by adding the node right hand side and then check list is in palindrome form or not.

# ANS:-

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*prev;

struct Node \*next;

};

typedef struct Node node;

void insert(node \*\*,node\*\*,int value);

void PalindromeChecker(node \*,node\*);

int main()

{

printf("ABHAY SINGH RAWAT \n");

int num,value;

node \*head=NULL,\*tail=NULL;

printf("Enter the total number of nodes to enter : ");

scanf("%d",&num);

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

{

printf("Enter node %d data: ",(i+1));

scanf("%d",&value);

insert(&head,&tail,value);

}

PalindromeChecker(head,tail);

return 0;

}

void insert(node \*\*head,node \*\*tail, int value)

{

node \*temp=(node\*)malloc(sizeof(node));

temp->data=value;

temp->prev=temp->next=NULL;

if(\*head==NULL&&\*tail==NULL)

{

\*head=temp;

\*tail=temp;

}

else

{

(\*tail)->next=temp;

temp->prev=\*tail;

\*tail=temp;

}

}

void PalindromeChecker(node \*head,node \*tail)

{

int counter=0;

while(head->next!=NULL||tail->prev!=NULL)

{

if(head->data!=tail->data)

{

counter++;

break;

}

head=head->next;

tail=tail->prev;

}

if(counter)

printf("Not in Palindrome Form\n");

else

printf("In Palindrome Form\n");

}

# OUTPUT:-

**Q14**. Write a C program to create a circular link list by adding the nodes in right hand side and then print the list.

# ANS:-

#include<stdio.h>

#include<stdlib.h>

struct Node{

int data;

struct Node \*next;

};

typedef struct Node node;

void insert(node \*\*,int );

void display(node\*);

int main()

{

node \*last;

printf("ABHAY SINGH RAWAT\n");

int counter=0,choice,value;

last=NULL;

while(counter==0)

{

printf("1.INSERT \n2.DISPLAY \n3.EXIT\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter a value : ");

scanf("%d",&value);

insert(&last,value);

break;

case 2:

display(last);

break;

case 3:

printf("EXITED\n");

counter++;

break;

default:

printf("WRONG CHOICE\n");

break;

}

}

return 0;

}

void insert(node \*\*last,int value)

{

node \*temp=(node\*)malloc(sizeof(node));

temp->data=value;

if(temp==NULL)

temp->next=NULL;

if(\*last==NULL)

{

\*last=temp;

(\*last)->next=temp;

}

else{

temp->next=(\*last)->next;

(\*last)->next=temp;

\*last=(\*last)->next;

}

printf("Entered %d Sucessfully\n",value);

}

void display(node \*last)

{

node \*head=last->next;

while(head!=last)

{

printf("%d ",head->data);

head=head->next;

}

printf("%d \n",head->data);

}

# OUTPUT:-