#include<stdio.h>

#define MAX 5

int top = -1;

int main()

{

int a[MAX] = {0};

int choice = 5;

int e;

printf("Enter 1 to Push an element\n");

printf("Enter 2 to Pop\n");

printf("Enter 3 to Print Stack\n");

printf("Enter 4 to Exit\n");

while(choice != 4)

{

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter Element to be pushed\n");

scanf("%d",&e);

push(a,e);

break;

case 2:

pop(a);

break;

case 3:

printstack(a);

break;

}

}

return 0;

}

int push(int a[],int element)

{

if(top == MAX-1)

{

printf("Stack overflow");

}

else {

top++;

a[top] = element;

}

return 0;

}

int pop(int a[])

{

if(top == -1)

{

printf("Stack underflow");

}

else {

a[top] = 0;

top--;

}

return 0;

}

int printstack(int a[])

{

int i;

printf("\n");

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

{

if(a[i] != 0) {

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

}

}

return 0;

}

**OUTPUT:**

Enter 1 to Push an element

Enter 2 to Pop

Enter 3 to Print Stack

Enter 4 to Exit

Enter your choice

1

Enter Element to be pushed

10

Enter your choice

1

Enter Element to be pushed

20

Enter your choice

1

Enter Element to be pushed

30

Enter your choice

1

Enter Element to be pushed

40

Enter your choice

1

Enter Element to be pushed

50

Enter your choice

3

50

40

30

20

10

Enter your choice

2

Enter your choice

3

40

30

20

10

**Queue ADT Array Implementation**

#include<stdio.h>

#define MAX 5

int rear=0,front=0;

int main()

{

int a[MAX] = {0};

int choice = 5;

int e;

printf("Enter 1 to Enqueue an element\n");

printf("Enter 2 to Dequeue\n");

printf("Enter 3 to Print Queue\n");

printf("Enter 4 to Exit\n");

while(choice != 4)

{

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter Element to be enqueued\n");

scanf("%d",&e);

queue(a,e);

break;

case 2:

dequeue(a);

break;

case 3:

printqueue(a);

break;

}

}

return 0;

}

int queue(int a[],int element)

{

if(rear == MAX)

{

printf("Queue full");

}

else {

a[rear] = element;

rear++;

}

return 0;

}

int dequeue(int a[])

{

if(rear==front)

{

printf("Queue empty");

}

else {

a[front] = 0;

front++;

}

return 0;

}

int printqueue(int a[])

{

int i;

printf("\n");

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

{

if(a[i] != 0){

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

}

}

return 0;

}

**OUTPUT:**

Enter 1 to Enqueue an element

Enter 2 to Dequeue

Enter 3 to Print Queue

Enter 4 to Exit

Enter your choice

1

Enter Element to be enqueued

10

Enter your choice

1

Enter Element to be enqueued

20

Enter your choice

1

Enter Element to be enqueued

30

Enter your choice

3

10 20 30

Enter your choice

2

Enter your choice

3

20 30

**Circular Queue**

#include<stdio.h>

#include<conio.h>

#define MAX 5

int q[MAX];

int front=0,rear=0;

void enqueue(int);

void dequeue();

void display();

int main()

{

int choice;

int ele;

clrscr();

while(choice != 4)

{

printf("Enter 1 to enqueue\n");

printf("Enter 2 to dequeue\n");

printf("Enter 3 to display\n");

printf("Enter 4 to exit\n");

scanf("%d",&choice);

switch (choice)

{

case 1:

printf("Enter element to be enqueued\n");

scanf("%d",&ele);

enqueue(ele);

break;

case 2:

dequeue();

break;

case 3:

printf("The queue elements are\n");

display();

break;

}

}

return 0;

}

void enqueue(int element)

{

if(rear == MAX)

{

rear = 0;

return;

}

q[rear] = element;

rear++;

return;

}

void dequeue()

{

if (front == rear)

{

printf("Queue is empty\n");

}

else

{

q[front] = 0;

front++;

}

}

void display()

{

int i;

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

{

if(q[i] != 0)

{

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

}

}

printf("\n");

}

**OUTPUT:**

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

1

Enter element to be enqueued

10

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

1

Enter element to be enqueued

20

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

1

Enter element to be enqueued

30

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

1

Enter element to be enqueued

40

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

1

Enter element to be enqueued

50

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

3

The queue elements are

10 20 30 40 50

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

1

Enter element to be enqueued

60

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

3

The queue elements are

10 20 30 40 50

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

1

Enter element to be enqueued

70

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

1

Enter element to be enqueued

80

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

3

The queue elements are

70 80 30 40 50

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

2

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

3

The queue elements are

80 30 40 50

Enter 1 to enqueue

Enter 2 to dequeue

Enter 3 to display

Enter 4 to exit

**Singly Linked List**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void ins(int element,int pos);

void del(int element);

void insatend(int element);

void insatbeg(int element);

void search(int element);

void display();

struct node

{

int data;

struct node \*next;

};

struct node \*head;

int main()

{

int choice=4;

int element;

int pos;

while (choice != 0)

{

printf("Enter 1 to insert anywhere\n");

printf("Enter 2 to delete anywhere\n");

printf("Enter 3 to insert at end\n");

printf("Enter 4 to insert at beginning\n");

printf("Enter 5 to search for an element\n");

printf("Enter 6 to display\n");

printf("Enter 0 to exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter element and position\n");

scanf("%d %d",&element,&pos);

ins(element,pos);

break;

case 2:

printf("Enter position\n");

scanf("%d",&pos);

del(pos);

break;

case 3:

printf("Enter element\n");

scanf("%d",&element);

insatend(element);

break;

case 4:

printf("Enter element\n");

scanf("%d",&element);

insatbeg(element);

break;

case 5:

printf("Enter element\n");

scanf("%d",&element);

search(element);

break;

case 6:

display();

break;

}

}

return 0;

}

void insatend(int element)

{

struct node \*temp;

struct node \*newnode;

if(head == 0)

{

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

head->data = element;

head->next = 0;

}

else {

temp = head;

while(temp->next != 0)

{

temp = temp->next;

}

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

newnode->data = element;

newnode->next = 0;

temp->next = newnode;

}

}

void insatbeg(int element)

{

struct node \*temp;

struct node \*newnode;

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

newnode->data = element;

newnode->next = head;

head = newnode;

}

void ins(int element,int pos)

{

struct node \*temp;

struct node \*newnode;

int cnt=0;

temp = head;

while(temp->next != 0)

{

if(cnt == pos-1)

{

break;

}

temp = temp->next;

cnt++;

}

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

newnode->data = element;

newnode->next = temp->next;

temp->next = newnode;

}

void del(int pos)

{

struct node \*temp;

struct node \*old;

int cnt=0;

temp = head;

while(temp->next != 0)

{

if(cnt == pos)

{

break;

}

old = temp;

temp = temp->next;

cnt++;

}

old->next = temp->next;

temp->next = 0;

free(temp);

}

void display()

{

struct node \*temp;

temp = head;

while(temp->next != 0)

{

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

temp = temp->next;

}

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

}

void search(int element)

{

struct node \*temp;

int cnt = 1;

temp = head;

while(temp->next != 0)

{

if(temp->data == element)

{

printf("%d found in %d\n",element,cnt);

}

temp = temp->next;

cnt++;

}

}

**OUTPUT:**

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

3

Enter element

10

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

3

Enter element

20

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

3

Enter element

30

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

1

Enter element and position

25 2

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

4

Enter element

50

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

6

50 --> 10 --> 20 --> 25 --> 30

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

5

Enter element

10

10 found in 2

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

2

Enter position

1

6

50 --> 20 --> 25 --> 30

**Doubly Linked List**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void ins(int element,int pos);

void del(int element);

void insatend(int element);

void insatbeg(int element);

void search(int element);

void display();

struct node

{

int data;

struct node \*next;

struct node \*prev;

};

struct node \*head;

int main()

{

int choice=4;

int element;

int pos;

while (choice != 0)

{

printf("Enter 1 to insert anywhere\n");

printf("Enter 2 to delete anywhere\n");

printf("Enter 3 to insert at end\n");

printf("Enter 4 to insert at beginning\n");

printf("Enter 5 to search for an element\n");

printf("Enter 6 to display\n");

printf("Enter 0 to exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter element and position\n");

scanf("%d %d",&element,&pos);

ins(element,pos);

break;

case 2:

printf("Enter position\n");

scanf("%d",&pos);

del(pos);

break;

case 3:

printf("Enter element\n");

scanf("%d",&element);

insatend(element);

break;

case 4:

printf("Enter element\n");

scanf("%d",&element);

insatbeg(element);

break;

case 5:

printf("Enter element\n");

scanf("%d",&element);

search(element);

break;

case 6:

display();

break;

}

}

return 0;

}

void insatend(int element)

{

struct node \*temp;

struct node \*newnode;

if(head == 0)

{

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

head->data = element;

head->next = 0;

head->prev = 0;

}

else {

temp = head;

while(temp->next != 0)

{

temp = temp->next;

}

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

newnode->data = element;

newnode->prev = temp;

newnode->next = 0;

temp->next = newnode;

}

}

void insatbeg(int element)

{

struct node \*temp;

struct node \*newnode;

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

newnode->data = element;

newnode->next = head;

head->prev = newnode;

newnode->prev = 0;

head = newnode;

}

void ins(int element,int pos)

{

struct node \*temp;

struct node \*newnode;

int cnt=0;

temp = head;

while(temp->next != 0)

{

if(cnt == pos-1)

{

break;

}

temp = temp->next;

cnt++;

}

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

newnode->data = element;

newnode->next = temp->next;

newnode->prev = temp;

temp->next->prev = newnode;

temp->next = newnode;

}

void del(int pos)

{

struct node \*temp;

int cnt=0;

temp = head;

while(temp->next != 0)

{

if(cnt == pos)

{

break;

}

temp = temp->next;

cnt++;

}

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

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

free(temp);

}

void display()

{

struct node \*temp;

temp = head;

while(temp->next != 0)

{

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

temp = temp->next;

}

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

}

void search(int element)

{

struct node \*temp;

int cnt = 1;

temp = head;

while(temp->next != 0)

{

if(temp->data == element)

{

printf("%d found in %d\n",element,cnt);

}

temp = temp->next;

cnt++;

}

}

**OUTPUT:**

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

3

Enter element

10

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

3

Enter element

20

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

3

Enter element

30

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

1

Enter element and position

2

2

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

4

Enter element

60

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

6

60 <--> 10 <--> 20 <--> 2 <--> 30

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

2

Enter position

1

Enter 1 to insert anywhere

Enter 2 to delete anywhere

Enter 3 to insert at end

Enter 4 to insert at beginning

Enter 5 to search for an element

Enter 6 to display

Enter 0 to exit

5

Enter element

60

60 found in 1

6

60 <--> 20 <--> 2 <--> 30

**Polynomial Addition**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void addatend(int,int);

void display();

struct poly

{

int coeff;

int exp;

struct poly \*next;

}\*p1,\*p2,\*p3;

int main()

{

int x,cnt,exp,coeff;

struct poly \*spare,\*temp,\*newnode,\*temp1,\*temp2;

int c,e;

cnt = 0;

exp = 0;

p1 = (struct poly\*)malloc(sizeof(struct poly));

p2 = (struct poly\*)malloc(sizeof(struct poly));

p3 = (struct poly\*)malloc(sizeof(struct poly));

printf("Enter the data for P1\n");

while(exp >= 0)

{

printf("Enter the coefficient\n");

scanf("%d",&coeff);

printf("Enter the exponent\n");

scanf("%d",&exp);

if(coeff == -1 && exp == -1)

{

break;

}

if(cnt==0)

{

p1->coeff = coeff;

p1->exp = exp;

p1->next = 0;

}

else

{

temp = p1;

while(temp->next != 0)

{

temp = temp->next;

}

newnode = (struct poly\*)malloc(sizeof(struct poly));

newnode->exp = exp;

newnode->coeff = coeff;

newnode->next = 0;

temp->next = newnode;

}

cnt++;

}

exp = 0;

cnt =0;

printf("Enter the data for P2\n");

while(exp >= 0)

{

printf("Enter the coefficient\n");

scanf("%d",&coeff);

printf("Enter the exponent\n");

scanf("%d",&exp);

if(coeff == -1 && exp == -1)

{

break;

}

if(cnt==0)

{

p2->coeff = coeff;

p2->exp = exp;

p2->next = 0;

}

else

{

temp = p2;

while(temp->next != 0)

{

temp = temp->next;

}

newnode = (struct poly\*)malloc(sizeof(struct poly));

newnode->exp = exp;

newnode->coeff = coeff;

newnode->next = 0;

temp->next = newnode;

}

cnt++;

}

cnt = 0;

temp1 = p1;

temp2 = p2;

while(temp1 != 0 && temp2 != 0)

{

if(temp1->exp == temp2->exp)

{

c = temp1->coeff + temp2->coeff;

e = temp1->exp;

temp1 = temp1->next;

temp2 = temp2->next;

}

else if(temp1->exp > temp2->exp) {

c = temp1->coeff;

e = temp1->exp;

temp1 = temp1->next;

}

else if(temp1->exp < temp2->exp)

{

c = temp2->coeff;

e = temp2->exp;

temp2 = temp2->next;

}

if(cnt == 0)

{

p3->coeff = c;

p3->exp = e;

p3->next = 0;

}

else

{

temp = p3;

while(temp->next != 0)

{

temp = temp->next;

}

newnode = (struct poly\*)malloc(sizeof(struct poly));

newnode->coeff = c;

newnode->exp = e;

newnode->next = 0;

temp->next = newnode;

}

cnt++;

}

if(temp1 != 0)

{

temp = temp1;

while(temp != 0)

{

addatend(temp->coeff,temp->exp);

temp = temp->next;

}

}

if(temp2 != 0)

{

temp = temp2;

while(temp != 0)

{

addatend(temp->coeff,temp->exp);

temp = temp->next;

}

}

display();

getch();

return;

}

void addatend(int coeff,int exp)

{

struct poly \*newnode,\*temp;

temp = p3;

while(temp->next != 0)

{

temp = temp->next;

}

newnode = (struct poly\*)malloc(sizeof(struct poly));

newnode->coeff = coeff;

newnode->exp = exp;

newnode->next = 0;

temp->next = newnode;

}

void display()

{

struct poly \*temp;

temp=p3;

printf("The sum is : ");

while(temp->next != 0)

{

printf("%dx^%d + ",temp->coeff,temp->exp);

temp=temp->next;

}

printf("%dx^%d \n",temp->coeff,temp->exp);

}

**OUTPUT:**

Enter the data for P1

Enter the coefficient

5

Enter the exponent

3

Enter the coefficient

4

Enter the exponent

2

Enter the coefficient

3

Enter the exponent

1

Enter the coefficient

8

Enter the exponent

0

Enter the coefficient

-1

Enter the exponent

-1

Enter the data for P2

Enter the coefficient

8

Enter the exponent

4

Enter the coefficient

9

Enter the exponent

2

Enter the coefficient

10

Enter the exponent

1

Enter the coefficient

11

Enter the exponent

0

Enter the coefficient

-1

Enter the exponent

-1

The sum is : 8x^4 + 5x^3 + 13x^2 + 13x^1 + 19x^0

**OUTPUT:**

Enter 1 to insert Enter 2 to delete Enter 3 to search Enter 4 to display Enter 5 to exit

Enter your choice

1

Enter element to be inserted

20

Enter your choice

1

Enter element to be inserted

10

Enter your choice

1

Enter element to be inserted

36

Enter your choice

1

Enter element to be inserted

46

Enter your choice

1

Enter element to be inserted

32

Enter your choice

1

Enter element to be inserted

4

Enter your choice

1

Enter element to be inserted

11

Enter your choice

3

Enter element to be search

20

20 found. 10 is its left and 36 is its right

Enter your choice

2

Enter element to be deleted

11

Enter your choice

4

The in-order is

4 10 20 32 36 46