E1 Array ADT

#include<stdio.h>

#include<stdlib.h>

struct array

{

int \*A;

int array\_length;

int size;

}\*a1,\*a2;

void create()

{

a1=(struct array\*)malloc(sizeof(struct array));

a2=(struct array\*)malloc(sizeof(struct array));

a1->A=(int\*)malloc(a1->size\*sizeof(int));

a2->A=(int\*)malloc(a2->size\*sizeof(int));

a1->size;

a1->array\_length;

a2->size;

a2->array\_length;

}

void traverse(struct array \*B)

{

int i=0;

while (i < B->array\_length)

{

printf("%d ",B->A[i]);

i++;

}

printf("\n");

return;

}

void insert\_element(struct array \*B,int num1,int num2)

{

int i;

B->A=realloc(B->A,(B->size+1)\*sizeof(int));

for(i=B->array\_length-1;i>=num2;i--)

{

B->A[i+1]=B->A[i];

}

B->A[num2]=num1;

B->array\_length++;

}

void delete\_element(struct array \*B,int num)

{

int i;

for(i=num;i<B->array\_length;i++)

{

B->A[i]=B->A[i+1];

}

B->A=realloc(B->A,(B->size-1)\*sizeof(int));

B->array\_length--;

}

void sort(struct array \*B)

{

int i,j,n=B->array\_length,small\_pos,temp;

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

{

small\_pos=i;

for(j=i+1;j<n;j++)

{

if(B->A[j]<B->A[small\_pos])

{

small\_pos=j;

}

}

temp=B->A[small\_pos];

B->A[small\_pos]=B->A[i];

B->A[i]=temp;

}

}

int search(struct array \*B,int num)

{

int i;

for(i=0;i<B->array\_length;i++)

{

if(B->A[i]==num)

{

return 1;

break;

}

}

if(i==B->array\_length)

{

return 0;

}

}

void merge(struct array \*B1,struct array \*B2)

{

int i,j=0;

B1->A=realloc(B1->A,(B1->size+B2->size)\*sizeof(int));

for(i=B1->array\_length;i < (B1->array\_length + B2->array\_length);i++)

{

B1->A[i]=B2->A[j];

j++;

}

B1->array\_length=B1->array\_length+B2->array\_length;

}

void copy(struct array \*B,struct array \*B2)

{

int i;

for(i=0;i<B->array\_length;i++)

{

B2->A[i]=B->A[i];

}

}

int isFull(struct array \*B)

{

if(B->array\_length==B->size)

return 1;

else

return 0;

}

int isEmpty(struct array \*B)

{

if(B->array\_length==0)

return 1;

else

return 0;

}

int length(struct array \*B)

{

return B->array\_length;

}

void reverse(struct array \*b)

{

int temp,i=0,j=b->array\_length-1;

while(i<j)

{

temp=b->A[i];

b->A[i]=b->A[j];

b->A[j]=temp;

i++;

j--;

}

}

void copy\_alternate\_elements(struct array \*Q1,struct array \*Q2,int num)

{

int i,j=0;

Q2->size=Q1->size;

if(Q1->array\_length%2==0)

{

if(num==1)

{

Q2->array\_length=(Q1->array\_length)/2;

for(i=0;i<Q1->array\_length;i++)

{

if(i%2==0)

{

Q2->A[j]=Q1->A[i];

j++;

}

}

}

if(num==2)

{

Q2->array\_length=(a1->array\_length)/2;

for(i=0;i<Q1->array\_length;i++)

{

if(i%2!=0)

{

Q2->A[j]=Q1->A[i];

j++;

}

}

}

}

if(Q1->array\_length%2!=0)

{

if(num==1)

{

Q2->array\_length=((Q1->array\_length)/2)+1;

for(i=0;i<Q1->array\_length;i++)

{

if(i%2==0)

{

Q2->A[j]=Q1->A[i];

j++;

}

}

}

if(num==2)

{

Q2->array\_length=(a1->array\_length)/2;

for(i=0;i<Q1->array\_length;i++)

{

if(i%2!=0)

{

Q2->A[j]=Q1->A[i];

j++;

}

}

}

}

}

int main()

{

void create();

void traverse(struct array \*a1);

void insert\_element(struct array \*a1,int,int);

void delete\_element(struct array \*a1,int);

void sort(struct array \*a1);

int isFull(struct array \*a1);

int isEmpty(struct array \*a1);

int length(struct array \*a1);

int search(struct array \*a1,int);

void merge(struct array \*a1,struct array \*a2);

void copy(struct array \*a1,struct array \*a2);

void reverse(struct array \*a1);

void copy\_alternate\_elements(struct array \*a1,struct array \*a2,int);

int taker2,helper,choice\_made,newr,taker,i,k,val,insert,pos,del,searcher,full,empty,len\_gth,search\_result;

create();

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

scanf("%d",&a1->size);

printf("[ NOTE: LENGTH will be <= MAX-SIZE given.]\n");

printf("Enter length of array to allocate : ");

scanf("%d",&a1->array\_length);

for(k=0;k <a1->array\_length;k++)

{

printf("enter element %d: \n",k+1);

scanf("%d",&a1->A[k]);

}

printf("THIS IS A MENU DRIVEN PROGRAM\npress the number to perform the corresponding operation\n\n");

printf("1)traverse\n2)insert\_element\n3)delete\_element\n4)sort\n5)search\n6)merge\n7)copy\n8)create\n9)isFull\n10)isEmpty\n11)length\n12)reverse\n13)Copy\_alternate\_elements\n");

do{

printf("ENTER YOUR CHOICE 1-13 || 0 TO EXIT\n");

scanf("%d",&val);

switch(val)

{

case 1:traverse(a1);

break;

case 2:printf("Enter the element to be inserted and index position\n");

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

insert\_element(a1,insert,pos);

break;

case 3:printf("Enter index of element to be deleted\n");

scanf("%d",&del);

delete\_element(a1,del);

break;

case 4:sort(a1);

break;

case 5:printf("Enter element to be searched : ");

scanf("%d",&searcher);

search\_result=search(a1,searcher);

if(search\_result==1)

printf("FOUND\n");

else

printf("NOT FOUND\n");

break;

case 6:

printf("Enter max size of new array:\n");

scanf("%d",&a2->size);

printf("Enter length of new array:\n");

scanf("%d",&a2->array\_length);

printf("Enter Elements Of new array :\n");

for(k=0;k < a2->array\_length;k++)

{

printf("enter element %d: \n",k+1);

scanf("%d",&a2->A[k]);

}

merge(a1,a2);

break;

case 7:

a2->size=a1->size;

a2->array\_length=a1->array\_length;

copy(a1,a2);

printf("To cheak if array was coppied successfully enter 1 || if not enter 0\n");

scanf("%d",&newr);

if(newr==1)

{

for(i=0;i<a1->array\_length;i++)

{

if(a1->A[i]==a2->A[i])

{continue;

}

else

{printf("ERROR OCCURED WHILE COPYING || some elements do not match\n");

break;

}

}

if(i>=a1->array\_length-1)

{

printf("COPPIED SUCCESSFULLY\n");

}

}

if(newr==0)

{

break;

}

printf("Do you want to Traverse the coppied array for mannual confirmation ?\n");

printf("[yes-press '1' || No-press '0']\n");

scanf("%d",&taker);

if(taker==1)

{traverse(a2);}

if(taker==0)

{

continue;

}

break;

case 8:create();

printf("Enter max size of new array:\n");

scanf("%d",&a1->size);

printf("[ NOTE: LENGTH will be <= MAX-SIZE given.]\n");

printf("Enter length of new array:\n");

scanf("%d",&a1->array\_length);

printf("Enter Elements Of new array :\n");

for(k=0;k < a1->array\_length;k++)

{

printf("enter element %d: \n",k+1);

scanf("%d",&a1->A[k]);

}

break;

case 9:

full=isFull(a1);

if(full==1)

printf("array is full\n");

else

printf("array is not full\n");

break;

case 10:

empty=isEmpty(a1);

if(empty==1)

printf("array is empty\n");

else

printf("array is not empty\n");

break;

case 11:

printf("Enter (1) If you want length of original array\nEnter (2) If you want length of modified array(result of operations performed)\n");

scanf("%d",&helper);

if(helper==1)

len\_gth=length(a1);

if(helper==2)

len\_gth=length(a2);

printf("Array length is %d.\n",len\_gth);

break;

case 12:

reverse(a1);

printf("Array Reversed ,[Traverse to Check]\n");

break;

case 13:

printf("Enter the alternate coppying scheme.\n\tpress.1) to keep all even index elements\n\t NOTE:(to start coppying from first element and accordingly)\n\n\tpress.2) to keep all odd index elements\n\t NOTE:(to start coppying from second element and accordingly)\n");

scanf("%d",&choice\_made);

copy\_alternate\_elements(a1,a2,choice\_made);

printf("Do you want to Traverse the coppied array for mannual confirmation ?\n");

printf("[yes-press '1' || No-press '0']\n");

scanf("%d",&taker2);

if(taker2==1)

{

traverse(a2);

}

if(taker2==0)

{

continue;

}

break;

default:printf("\n\nPLEASURE GIVING SERVICE :-)\n\n");

}

}while(val!=0);

return 0;

}