Array Sheet

4. insert in unsorted arr:

int insertfun(int arr[],int n,int size,int key){

if(n>=size){

return n;

}

else

arr[n]=key;

return (n+1);

}

int main()

{

int arr[20] = {1, 2, 3, 1, 3, 7};

int size = sizeof(arr) / sizeof(arr[0]);

int n=6;

int key;

cin>>key;

cout<<"\n before insertion";

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

cout<<arr[i]<<" ";

}

n = insertfun(arr,n,size,key);

cout<<"\n after insertion";

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

cout<<arr[i]<<" ";

}

}

5. delete item in unsorted arr:

int Binary\_Search(int arr[],int i,int j,int target){

if(i>j){

return -1;

}

int mid;

mid=(i+j)/2;

if(target<mid)

return Binary\_Search(arr,i,mid-1,target);

else if(target>mid)

return Binary\_Search(arr,mid+1,j,target);

else

return mid;

}

int delete\_item(int arr[],int n,int target){

int position=Binary\_Search(arr,0,n-1,target);

if(position==-1){

cout<<"not founded";

return n;

}

for(int i=position;i<n-1;i++){

arr[i]=arr[i+1];

}

return n-1;

}

6.

7.

8. Quick sort:

int Partition(int arr[],int left,int right){

int pivot,i,j;

pivot=arr[left];

i=left;

j=right+1;

while(1){

do{

++i;

}while(arr[i]<=pivot&&i<=right);

do{

--j;

}while(arr[j]>pivot);

if(i>=j)break;

int temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

int temp =arr[left];

arr[left]=arr[j];

arr[j]=temp;

return j;

}

void Quick\_sort(int arr[],int left,int right){

int j;

if(left<right){

j=Partition(arr,left,right);

Quick\_sort(arr,left,j-1);

Quick\_sort(arr,j+1,right);

}

}

9. binary search:

int Binary\_Search(int arr[],int i,int j,int target){

if(i>j){

return -1;

}

int mid;

mid=(i+j)/2;

if(target<mid)

return Binary\_Search(arr,i,mid-1,target);

else if(target>mid)

return Binary\_Search(arr,mid+1,j,target);

else

return mid;

}

10. selection sort:

void Selection\_sort(int arr[],int n){

for(int i=0;i<n-1;i++){

int min=i;

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

if(arr[j]<arr[min])

min=j;

}

if(arr[min]!=arr[i])

{

int temp=arr[min];

arr[min]=arr[i];

arr[i]=temp;

}

}

}

11. insertion sort:

void Insertion\_sort(int arr[], int n){

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

int j;

j=i-1;

int key;

key=arr[i];

while(j>=0&&arr[j]>key){

arr[j+1]=arr[j];

j--;

}

arr[j+1]=key;

}

}

12. merge sort:

void Merge(int arr[], int i1,int i2, int j1, int j2){

int temp[50];

int i=i1,j=j1, k=0;

while(i<=i2 && j<=j2){

if(arr[i]<arr[j])

temp[k++]=arr[i++];

else

temp[k++]=arr[j++];

}

while(i<=i2){

temp[k++]=arr[i++];

}

while(j<=j2){

temp[k++]=arr[j++];

}

for(i=i1,j=0;i<=j2;i++,j++){

arr[i]=temp[j];

}

}

void Merge\_sort(int arr[],int i,int j){

int mid;

if(i<j){

mid=(i+j)/2;

Merge\_sort(arr,i,mid);

Merge\_sort(arr,mid+1,j);

Merge(arr,i,mid,mid+1,j);

}

}

13. bubble sort:

void Bubble\_sort(int arr[],int n){

for(int i=0;i<n-1;i++){

for(int j =0;j<n-1-i;j++){

if(arr[j]>arr[j+1]){

int temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

}