C for Beginners

Friday, 8 November 2013

Generate n of random integers in a given range and sort them using quick sort. Apply both binary search and Interpolation search to locate a given integer and compare the search algorithms based on the number of comparisons/probes required for a successful as well as unsuccessful search..

Generate n of random integers in a given range and sort them using quick sort. Apply both binary search and Interpolation search to locate a given integer and compare the search algorithms based on the number of comparisons/probes required for a successful as well as unsuccessful search..

```
#include<stdlib.h>
#include<conio.h>
#include<stdio.h>

void quicksort(int [10],int,int);
void interpolation(int [10],int,int,int);
main()
{
    int x[10],i,n,e;
    clrscr();
    printf("Enter the Size : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
 x[i]=rand()%100;
    }

    printf("Before Sorting        : ");
    for(i=0;i<n;i++)
 printf("%d\t",x[i]);

    quicksort(x,0,n-1);
    printf("\nAfter Quick Sorting : ");
    for(i=0;i<n;i++)
      printf("%d\t",x[i]);

    printf("\nEnter the Search Key  : ");
    scanf("%d",&e);
    interpolation(x,0,n-1,e);
    b_search(x,0,n-1,e);
    getch();

}

void quicksort(int x[10],int first, int last)
{
   int pivot,j,temp,i;
   if(first<last)
```

```
    {
 pivot=first;
 i=first;
 j=last;
 while(i<j)
 {
     while(x[i]<=x[pivot]&&i<last)
  i++;
     while(x[j]>x[pivot])
  j--;
     if(i<j)
     {
  temp=x[i];
  x[i]=x[j];
  x[j]=temp;
     }
 }
 temp=x[pivot];
 x[pivot]=x[j];
 x[j]=temp;
 quicksort(x,first,j-1);
 quicksort(x,j+1,last);
    }
}

void interpolation(int a[10],int low,int high,int item)
{
     int mid,f=0,i,c=0;
     printf("\nInterpolation Search :\n");
     while(low<=high)
     {
     c++;
     mid=low+(high-low)*((item-a[low])/(a[high]-a[low]));
     if(a[mid]==item)
     {
   printf("\nElements found at POS %d ",mid+1);
   f=1;
   break;
     }
     else if(item<a[mid])
   high=mid-1;
     else
   low=mid+1;
     }
     if(f==0)
   printf("\nThe Element is not found");

     printf("\nSuccess Comparsions : %d",c);

}

b_search(int a[10],int top,int bot,int item)
{
     int mid,flag=0,loc,count=0,i;

     printf("\n\nBianary Search Result :\n");
     while(top<=bot)
     {
   count++;
   mid=(top+bot)/2;
   if(item==a[mid])
   {
       loc=mid;
       flag=1;
       break;
```

```
  }
  else if(item<a[mid])
      bot=mid-1;
  else
      top=mid+1;

   }
  if(flag==1)
      printf("\nElement found at POS : %d",loc+1);
  else
      printf("\nThe Element is not found ");
  printf("\nSuccess Comparisions : %d",count);

}
```

Posted by Prathuish Kumar at 07:19 No comments:

Saturday, 19 October 2013

Evaluation of Post-fix Expression

ADT Stack implementation and use it for evaluation of Post-fix Expression

```
/*Progam for Postfix Evalution */
#define SIZE 50            /* Size of Stack */
#include <ctype.h>
#include<stdio.h>
int s[SIZE];
int top=-1;       /* Global declarations */

push(int elem)
{                          /* Function for PUSH operation */
 s[++top]=elem;
}

int pop()
{                         /* Function for POP operation */
 return(s[top--]);
}

main()
{                          /* Main Program */
 char pofx[50],ch;
 int i=0,op1,op2;
 printf("\n\nRead the Postfix Expression ? ");
 scanf("%s",pofx);
 while( (ch=pofx[i++]) != '\0')
 {
  if(isdigit(ch))
  push(ch-'0'); /* Push the operand */
  else
  {        /* Operator,pop two  operands */
 op2=pop();
 op1=pop();
 switch(ch)
   {
 case '+':push(op1+op2);break;
 case '-':push(op1-op2);break;
   case '*':push(op1*op2);break;
   case '/':push(op1/op2);break;
   }
  }
 }
 printf("\n Given Postfix Expn: %s\n",pofx);
 printf("\n Result after Evaluation: %d\n",s[top]);
}
```

Posted by Prathuish Kumar at 07:57 No comments:

G+1 +1 Recommend this on Google

Labels: Abstract Data Type, ADT, Evaluation of Post-fix Expression, Post fix Evaluation

Thursday, 17 October 2013

Merge Sort Program

Merge Sort Program

```
#include<stdio.h>
#define MAX 50

void mergeSort(int arr[],int low,int mid,int high);
void partition(int arr[],int low,int high);

int main(){

  int merge[MAX],i,n;

  printf("Enter the total number of elements: ");
  scanf("%d",&n);

  printf("Enter the elements which to be sort: ");
  for(i=0;i<n;i++){
   scanf("%d",&merge[i]);
  }

  partition(merge,0,n-1);

  printf("After merge sorting elements are: ");
  for(i=0;i<n;i++){
   printf("%d ",merge[i]);
  }

 return 0;
}

void partition(int arr[],int low,int high){

  int mid;

  if(low<high){
   mid=(low+high)/2;
   partition(arr,low,mid);
   partition(arr,mid+1,high);
   mergeSort(arr,low,mid,high);
  }
}

void mergeSort(int arr[],int low,int mid,int high){

  int i,m,k,l,temp[MAX];

  l=low;
  i=low;
  m=mid+1;

  while((l<=mid)&&(m<=high)){

   if(arr[l]<=arr[m]){
     temp[i]=arr[l];
     l++;
   }
   else{
     temp[i]=arr[m];
     m++;
   }
         i++;
  }

  if(l>mid){
         for(k=m;k<=high;k++){
     temp[i]=arr[k];
            i++;
```

```
  }
   }
 else{
        for(k=l;k<=mid;k++){
    temp[i]=arr[k];
           i++;
  }
   }

   for(k=low;k<=high;k++){
  arr[k]=temp[k];
   }
}
```

Posted by Prathuish Kumar at 10:17 No comments:

G+1 +1 Recommend this on Google

Labels: Merge Sort, Merge Sort Program

Quick Sort Program

Quick Sort Program

```
#include<stdio.h>

void quicksort(int [10],int,int);

int main(){
  int x[20],size,i;

  printf("Enter size of the array: ");
  scanf("%d",&size);

  printf("Enter %d elements: ",size);
  for(i=0;i<size;i++)
  scanf("%d",&x[i]);

  quicksort(x,0,size-1);

  printf("Sorted elements: ");
  for(i=0;i<size;i++)
  printf(" %d",x[i]);

  return 0;
}

void quicksort(int x[10],int first,int last){
  int pivot,j,temp,i;

   if(first<last){
   pivot=first;
   i=first;
   j=last;

   while(i<j){
     while(x[i]<=x[pivot]&&i<last)
       i++;
     while(x[j]>x[pivot])
       j–;
     if(i<j){
       temp=x[i];
      x[i]=x[j];
      x[j]=temp;
     }
   }

   temp=x[pivot];
   x[pivot]=x[j];
   x[j]=temp;
   quicksort(x,first,j-1);
   quicksort(x,j+1,last);

  }
}
```

Posted by Prathuish Kumar at 10:05 No comments:

G+1 +1 Recommend this on Google

Labels: Quick Sort, Quick Sort Program, Sorting, Sorting Methods

Circular Queue Using Double linked list

Circular Queue Using Double linked list

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

struct node
{
int data;
struct node *next;
struct node *prev;
};

struct node *head=NULL,*temp;

void insert(int ele)
{
 struct node *tempe;
 temp=(struct node *)malloc(sizeof(struct node));
 temp->data=ele;
 temp->next=NULL;
 if(head==NULL)
 {
  head=temp;
  temp->next=head;
 }
 else
 {
  tempe=head;
  while(tempe->next!=head)
  {
   tempe=tempe->next;
  }
  tempe->next=temp;
  temp->next=head;
      temp->prev=tempe;
 }
}

void display()
{
 struct node *temp1;
 temp1=head;
 while(temp1->next!=head)
 {
  printf("%d\t",temp1->data);
  temp1=temp1->next;
 }
 printf("%d",temp1->data);
}


void del(int pos)
{
 struct node *temp1,*temp2;

 if(head==NULL)
 {
  printf("Circular Queue is Emplty");
 }
 else if(pos==1)
 {
  temp1=head;
  temp2=head;
  while(temp1->next!=head)
  {
   temp1=temp1->next;
  }
```

```
 head=head->next;
 temp1->next=head;
 free(temp2);
}
else
{
 temp2=head;
 while(--pos!=0)
 {
  temp1=temp2;
  temp2=temp2->next;

 }
 temp1->next=temp2->next;
 free(temp2);
}
}
 void main()
{
 int n,e,ch;
 head=NULL;
 while(1)
 {
  printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit");
  printf("\nEnter your Choice : ");
  scanf("%d",&ch);
  switch(ch)
  {
   case 1: printf("\nEnter the Number : ");
      scanf("%d",&n);
      insert(n);break;
   case 2:  printf("\nEnter the Position for Deleting : ");
      scanf("%d",&e);
      del(e);  break;
   case 3:
      display();  break;
   case 4:  exit(0);   break;
   default: printf("\nYour Choice is Wrong");
  }
 }

}
```

Posted by Prathuish Kumar at 10:02 No comments:

G+1 +1 Recommend this on Google

Labels: Circular Queue Using Double linked list

Queue Operations using Single Linked List

Queue Operations using Single Linked List

```
#include<stdio.h>
#include<conio.h>
#include <stdlib.h>

struct node
{
 int data;
 struct node *next;
};
struct node *front=NULL,*rear=NULL;

void insert()
{
 int item;
 struct node *temp,*tempe;

 printf("\nEnter the Number : ");
 scanf("%d",&item);
 temp=(struct node *) malloc(sizeof(struct node));

 temp->data=item;
 temp->next=NULL;
```

```
 if(rear==NULL)
 {
  front=temp;
  rear=temp;
 }
 else
 {
  rear->next=temp;
  rear=temp;
 }
}



void del()
{


 if(front==NULL)
  printf("\Queue is Empty ");
 else if(rear==front)
  front=rear=NULL;
 else
 {
  printf("%d is Deleted ",front->data);
  front=front->next;
 }
}

void display()
{
 struct node *temp;

 if(front==NULL && rear==NULL)
  printf("\nQueue is Empty");
 else
 {
  temp=front;
  printf("\n Elements are : ");
  while(temp->next!=NULL)
  {
   printf("%d\t",temp->data);
   temp=temp->next;
  }
  printf("%d",temp->data);
 }
}

void main()
{
 int ch;
 while(1)
 {
  printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit");
  printf("\nEnter your Choice : ");
  scanf("%d",&ch);
  switch(ch)
  {
   case 1:  insert();break;
   case 2:  del();break;
   case 3:  display();break;
   case 4:  exit(0);break;
          default: printf("\nYour Choice is Woring");
  }
 }

}
```

Posted by Prathuish Kumar at 09:51 No comments:

G+1 +1 Recommend this on Google

Labels: C Program for Linked list, Queue Operations using Single Linked List, Queue using Linked List

Wednesday, 4 September 2013

Infix to Postfix Conversion using stack in C

Infix to Postfix Conversion using stack in C

```
#define SIZE 50 /* Size of Stack */
#include<string.h>
#include <ctype.h>
#include<stdio.h>
char s[SIZE]; int top=-1; /* Global declarations */
push(char elem)
{ /* Function for PUSH operation */
s[++top]=elem;
}
char pop()
{ /* Function for POP operation */
return(s[top--]);
}
int pr(char elem)
{ /* Function for precedence */
switch(elem)
{
case '#': return 0;
case ')': return 1;
case '+':
case '-': return 2;
case '*':
case '/':return 3;
}
}
main()
{ /* Main Program */
char infx[50],prfx[50],ch,elem;
int i=0,k=0;
printf("\n\nRead the Infix Expression ? ");
scanf("%s",infx);
push('#');

while( (ch=infx[i++]) != '\0')
{
if( ch == ')')
push(ch);
else if(isalnum(ch))
prfx[k++]=ch;
else if( ch == '(')
{
while( s[top] != ')')
prfx[k++]=pop();
elem=pop(); /* Remove ) */
}
else
{ /* Operator */
while( pr(s[top]) >= pr(ch) )
prfx[k++]=pop(); push(ch);
}
}
while( s[top] != '#') /* Pop from stack till empty */
prfx[k++]=pop();
prfx[k]='\0'; /* Make prfx as valid string */

printf("\n\nGiven Infix Expn: %s \nPrefix Expn: %s\n",infx,prfx);
}
```

Posted by Prathuish Kumar at 19:42 No comments:

G+1 +1 Recommend this on Google

Labels: Infix to Postfix Conversion, Infix to Postfix Conversion using stack, Infix to Postfix Conversion using stack in C

Home Older Posts

Subscribe to: Posts (Atom)