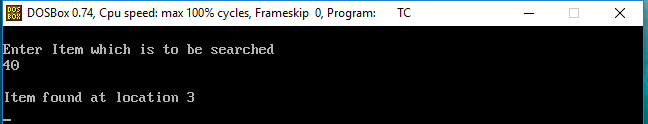
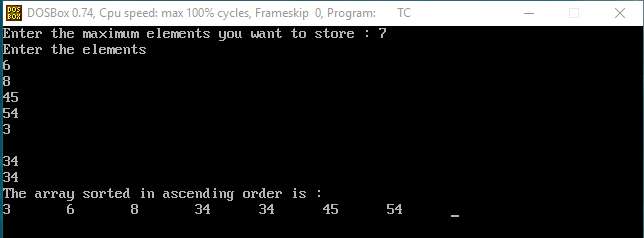
Linear Search

|  |  |
| --- | --- |
| #include<stdio.h>  #include<conio.h>  void main ()  {  int a[10] = {10, 23, 40, 1, 2, 0, 14, 13, 50, 9};  int item, i, flag;  printf("\nEnter Item which is to be searched\n");  scanf("%d",&item);  for (i = 0; i< 10; i++)  {  if(a[i] == item)  {  flag = i+1;  break;  }  else | {  flag = 0;  }  }  if(flag != 0)  {  printf("\nItem found at location %d\n",flag);  }  else  {  printf("\nItem not found\n");  }  getch();  } |



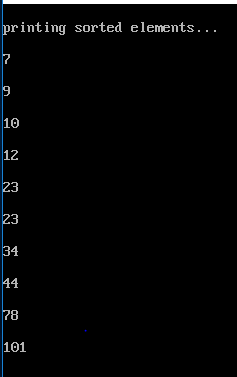
Bubble Sort

|  |  |
| --- | --- |
| #include <stdio.h>  #include <stdlib.h>  #include<conio.h>  int main()  {  int i, n, temp, j, arr[10];  printf("Enter the maximum elements you want to store : ");  scanf("%d", &n);  printf("Enter the elements \n");  for(i=0;i<n;i++)  {  scanf("%d", & arr[i]);  } | for(i=0;i<n;i++)  {  for(j=0;j<n-1;j++)  {  if(arr[j]>arr[j+1])  {  temp = arr[j];  arr[j] = arr[j+1];  arr[j+1] = temp;  }  }  }  printf("The array sorted in ascending order is :\n");  for(i=0;i<n;i++)  printf("%d\t", arr[i]);  getch();  return 0;  } |



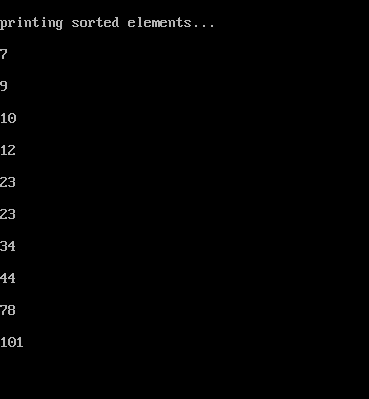
Insertion sort

|  |  |
| --- | --- |
| #include<stdio.h>  # include<conio.h>  void main()  {  int i,j,k,temp;  int a[10]= {10,9,7,101,23,44,12,78,34,23};  printf ("\nprinting sorted elements...\n");  for (k=1;k<10;k++)  {temp = a[k];  j=k-1;  while (j>=0&&temp<=a[j])  {a[j+1]= a[j];  j=j-1; | }  a[j+1]=temp;  }  for (i=0;i<10;i++)  {printf("\n%d\n",a[i]);  }  getch();  } |



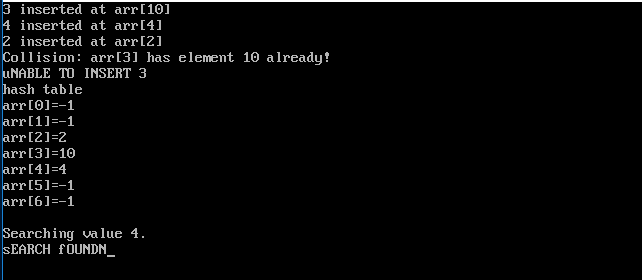
Selection sort

|  |  |
| --- | --- |
| #include<stdio.h>  # include<conio.h>  int smallest(int[],int,int);  void main()  {  int a[10]={10,9,7,101,23,44,12,78,34,23};  int i,j,k,pos,temp;  for(i=0;i<10;i++){  pos=smallest(a,10,i);  temp = a[i];  a[i] = a[pos];  a[pos] = temp;  }  printf("\nprinting sorted element\n");  for(i=0;i<10;i++){  printf("%d\n",a[i]);  }  } | int smallest(int a[],int n,int i){  int small, pos, j;  small =a[i];  pos=i;  for(j=i+1;j<10;j++){  if(a[j]<small){  small=a[j];  pos=j;  }  }  getch();  return pos;  } |
|  |  |



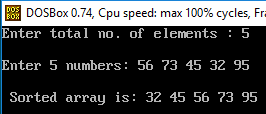
Modulo Division

|  |  |
| --- | --- |
| #include<stdio.h>  #include<conio.h>  #define size 7  int arr[size];  void init()  {  int i;  for(i=0;i<size;i++){  arr[i]=-1;  }  }  void insert(int value)  {  int key = value%size;  if(arr[key]==-1)  {  arr[key]=value;  printf("%d inserted at arr[%d]\n",key,arr[key]);  }  else{  printf("Collision: arr[%d] has element %d already!\n",key,arr[key]);  printf("uNABLE TO INSERT %d\n",value);  }  }  void search(int value)  { | int key = value%size;  if(arr[key]==value)  {  printf("sEARCH fOUND\N");  }  else{  printf("Search not found\n");  }  }  void display(){  int i;  for(i=0;i<size;i++)  {  printf("arr[%d]=%d\n",i,arr[i]);  }  }  int main(){  init();  insert(10);  insert(4);  insert(2);  insert(3);  printf("hash table\n");  display();  printf("\n");  printf("Searching value 4.\n");  search(4);  getch();  return 0;  } |



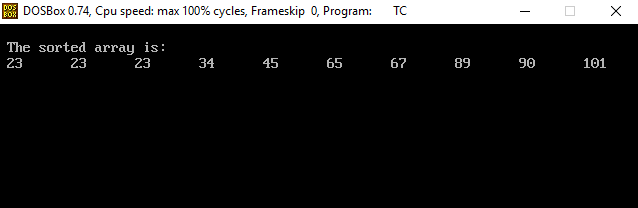
Shell sort

|  |  |
| --- | --- |
| #include <stdio.h>  #include <conio.h>  void shellsort(int arr[], int num)  {  int i, j, k, tmp;  for (i = num / 2; i > 0; i = i / 2)  {  for (j = i; j < num; j++)  {  for(k = j - i; k >= 0; k = k - i)  {  if (arr[k+i] >= arr[k])  break;  else  {  tmp = arr[k];  arr[k] = arr[k+i];  arr[k+i] = tmp;  }  } | int main()  {  int arr[30];  int k, num;  printf("Enter total no. of elements : ");  scanf("%d", &num);  printf("\nEnter %d numbers: ", num);  for (k = 0 ; k < num; k++)  {  scanf("%d", &arr[k]);  }  shellsort(arr, num);  printf("\n Sorted array is: ");  for (k = 0; k < num; k++)  printf("%d ", arr[k]);  return 0;  }  }  }  } |



Radiz sort

|  |  |
| --- | --- |
| #include <stdio.h>  #include <conio.h>  int largest(int a[]);  void radix\_sort(int a[]);  void main()  {  int i;  int a[10]={90,23,101,45,65,23,67,89,34,23};  radix\_sort(a);  printf("\n The sorted array is: \n");  for(i=0;i<10;i++)  printf(" %d\t", a[i]);  }  int largest(int a[])  {  int larger=a[0], i;  for(i=1;i<10;i++)  {  if(a[i]>larger)  larger = a[i];  }  return larger;  }  void radix\_sort(int a[])  {  int bucket[10][10], bucket\_count[10];  int i, j, k, remainder, NOP=0, divisor=1, larger, pass; | larger = largest(a);  while(larger>0)  {  NOP++;  larger/=10;  }  for(pass=0;pass<NOP;pass++) // Initialize the buckets  {  for(i=0;i<10;i++)  bucket\_count[i]=0;  for(i=0;i<10;i++)  {  // sort the numbers according to the digit at passth place  remainder = (a[i]/divisor)%10;  bucket[remainder][bucket\_count[remainder]] = a[i];  bucket\_count[remainder] += 1;  }  // collect the numbers after PASS pass  i=0;  for(k=0;k<10;k++)  {  for(j=0;j<bucket\_count[k];j++)  {  a[i] = bucket[k][j];  i++;  }  }  divisor \*= 10;  }  getch();  } |



**Aim: Implement program for Fold Boundary**

|  |  |
| --- | --- |
| #include<stdio.h>  #include<string.h>  #include <math.h>  int count\_digits(int key)  {  int count=0;  while(key != 0)  {  key /= 10;  ++count;  }  return count;  }  int fold\_boundary(int key, int size)  {  int key\_roll=key;  int key\_sum=0;  int key\_frac=0;  int middle=0;  int left=0;  int right=0;  int digits=0;  int key\_length=0;  int fraction = size;  key\_length = count\_digits(key\_roll);  key\_frac = key\_roll / (int)pow(10, (key\_length - fraction));// start digit  left=reversDigits(key\_frac);  key\_roll = key\_roll % (int)pow(10,3); | right=reversDigits(key\_roll);  digits = (int)log10(key) + 1;  middle= (int)(key / pow(10, digits/ 2)) % 10;  key\_sum = left +middle+ right;  return key\_sum % (int)pow(10, (fraction)); //ignore carry  }  int reversDigits(int num)  {  int rev\_num = 0;  while (num > 0)  {  rev\_num = rev\_num \* 10 + num % 10;  num = num / 10;  }  return rev\_num;  }  int main()  {  printf("\n\n%d",fold\_boundary(3347878, 3)); //318  printf("\n\n%d",fold\_boundary(1234678, 3)); //201  return 0;  } |

