**Week 6. Write a C++ program that uses function templates to perform the following:**

**a) Search for a key element in a list of elements using linear search.**

**Program:-**

#include<iostream>

using namespace std;

template<class T>

class lsearch

{

T data[10],n,key;

public:

void getdata();

void display();

};

template<class T>void lsearch<T> :: getdata()

{

cout<<"\nEnter the length of the array:";

cin>>n;

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

{

cout<<"\nEnter the element in the "<<(i+1)<<" position of the array:";

cin>>data[i];

}

cout<<"\nEnter the key to find the element in the array:";

cin>>key;

}

template<class T>void lsearch<T> :: display()

{

int flag=0;

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

{

if(key == data[i])

{

cout<<"\n\nThe element "<<key<<" is present in the position "<<(i+1)<<" of the array";

flag++;

}

}

if(flag==0)

cout<<"\nGiven key "<<key<<" is not present in the array";

}

int main()

{

lsearch<int>ob;

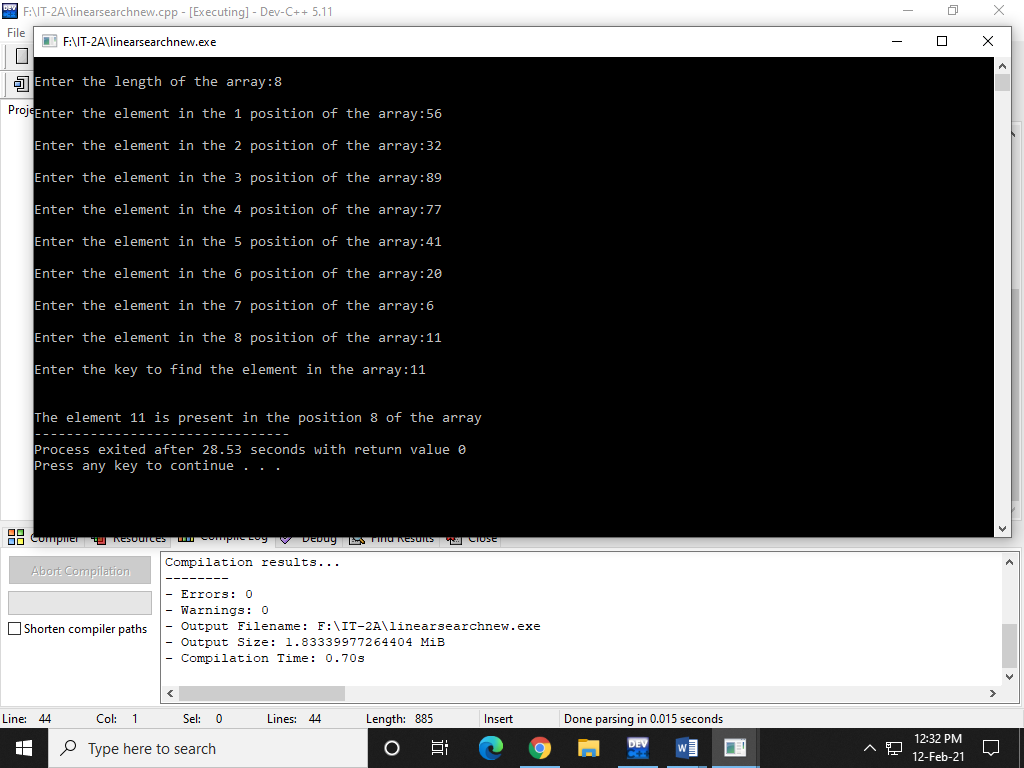
ob.getdata();

ob.display();

return 0;

}

**Output:-**



**b) Search for a key element in a list of sorted elements using binary search.**

**Program:-**

#include<iostream>

#include<stdlib.h>

using namespace std;

template<class T>

T bin(T a[20],T n,T t)

{

int l,r,mid;

l=0;

r=n-1;

while(l<=r)

{

mid=(l+r)/2;

if(t==a[mid])

return mid;

else

if(t<a[mid])

r=mid-1;

else

l=mid+1;

}

return -1;

}

int main()

{

int a[20],n,t,p,i;

cout<<"\nEnter number of elements :";

cin>>n;

cout<<"\nAs binary search algorithm works in sorted array!";

cout<<"\nEnter "<< n <<" values in ascending order : ";

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

cin>>a[i];

cout<<"\nEnter the element to find : ";

cin>>t;

p=bin(a,n,t);

if(p!=-1)

cout<<"\nThe element is found at "<<p+1<<" position";

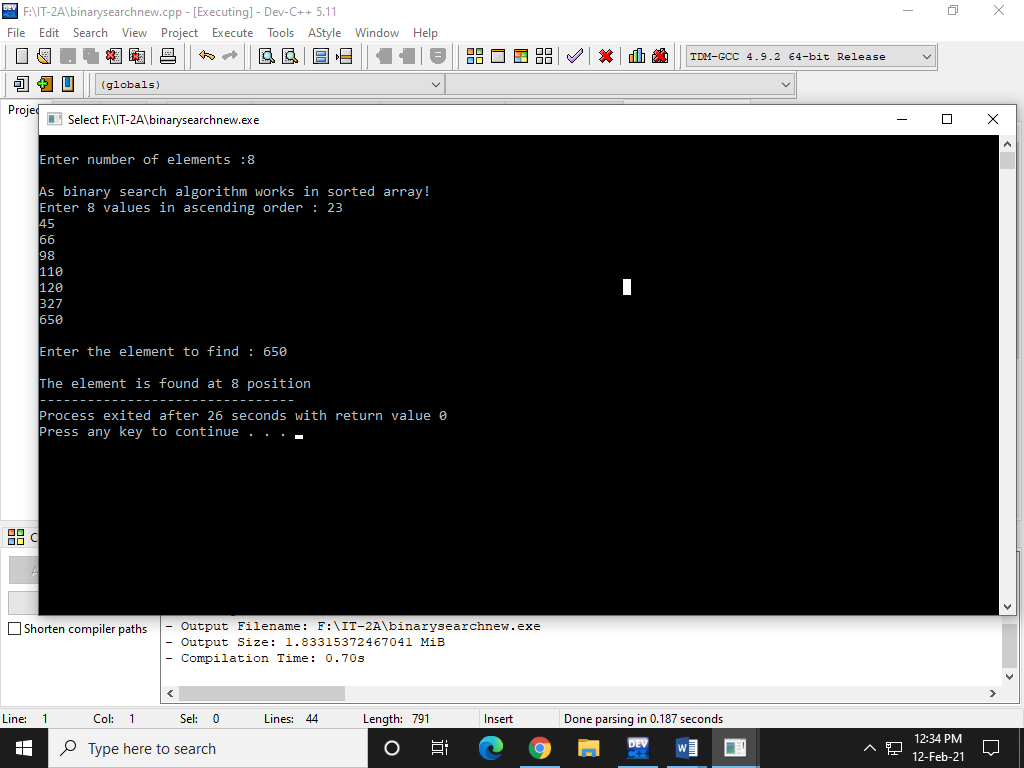
else

cout<<"\nElement Not Found!!";

return 0;

}

**Output:-**



**Week-7. Write a C++ program that implements Insertion sort algorithm to arrange a list of integers in ascending order.**

**Program:-**

#include<iostream>

using namespace std;

int main ()

{

int i,j, k,temp;

int a[10],n,l;

cout<<"Enter size of array\n";

cin>>n;

cout<<"Enter array elements : "<<endl;

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

cin>>a[l];

cout<<"\nprinting sorted elements...\n";

for(k=1; k<l; 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<l;i++)

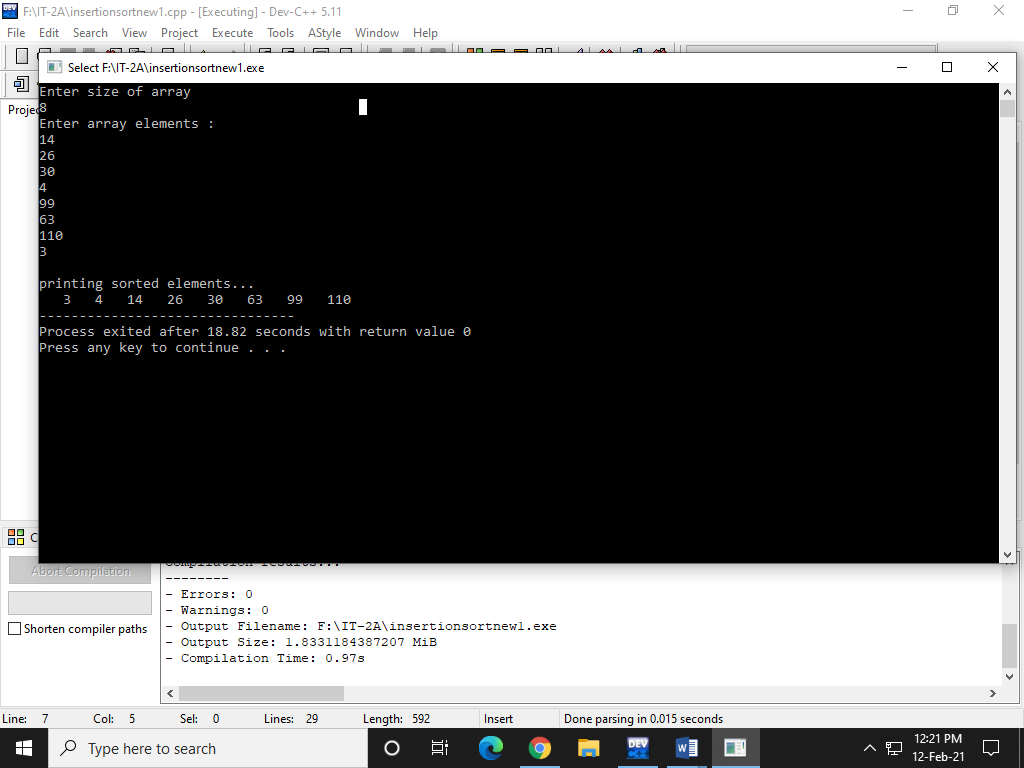
{

cout<<" "<<a[i];

}

}

**Output:-**



**8.Write a template based C++ program that implements selection sort algorithm to arrange a list of elements in descending order.**

**Program:-**

#include<iostream>

using namespace std;

int selectionsort(int a[], int n)

{

int i,j,temp,min;

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

{

min=i;

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

{

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

min=j;

}

temp=a[i];

a[i]=a[min];

a[min]=temp;

}

}

int main()

{

int a[20],n,i;

cout<<"Enter the size of an array :";

cin>>n;

cout<<"Enter array elements :";

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

cin>>a[i];

selectionsort(a,n);

cout<<"Sorted array : ";

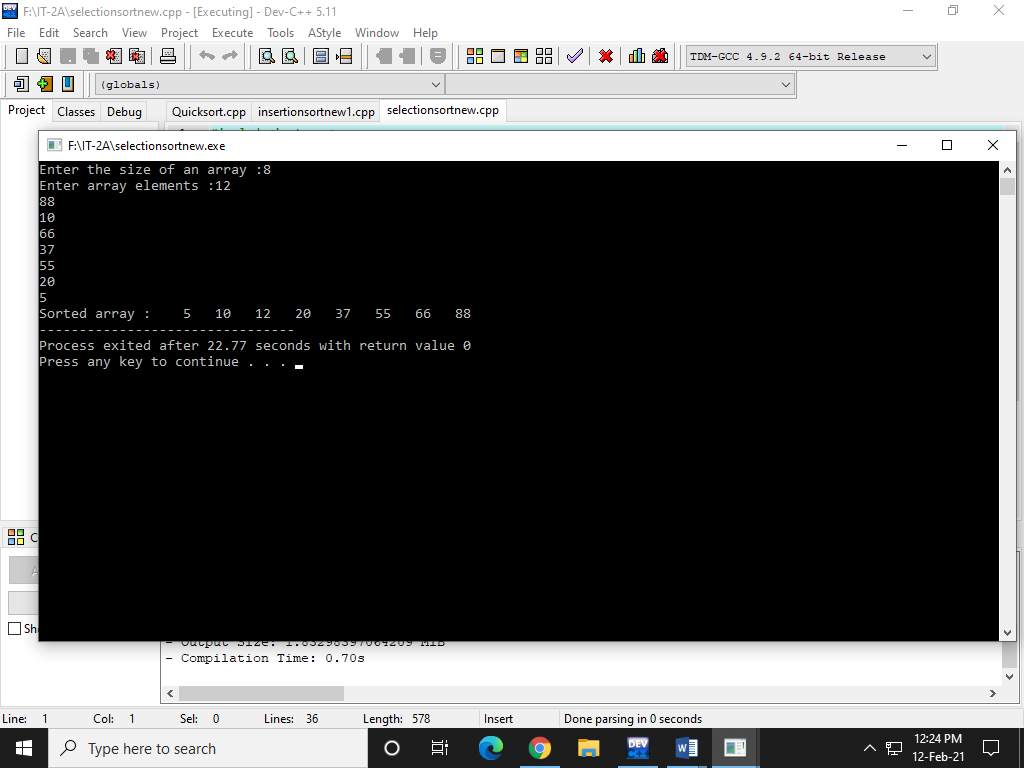
for(i=0;i<n;i++)

cout<<" "<<a[i];

return 0;

}

**Output:-**



**9.Write a template based C++ program that implements Quick sort algorithm to arrange a list of elements in ascending order.**

**Program:-**

#include<iostream>

using namespace std;

template <class T>

void qsort(T a[],int lb,int ub)

{

int i,j,key,b;

int flag=1;

if(lb<ub)

{

i=lb;

j=ub+1;

key=a[lb];

while(flag==1)

{

i++;

while((a[i]<key)&&(i<=ub))

i++;

j--;

while((a[j]>key)&&(j>=lb))

j--;

if(i<j)

{

b=a[i];

a[i]=a[j];

a[j]=b;

}

else

flag=-1; }

b=a[lb];

a[lb]=a[j];

a[j]=b;

}

else

return;

qsort(a,lb,j-1);

qsort(a,j+1,ub);

}

int main()

{

int a[20],i,n,size;

cout<<"Enter size of array\n";

cin>>n;

cout<<"Enter array elements : "<<endl;

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

cin>>a[i];

size=n-1;

qsort(a,0,size);

cout<<"Sorted array elements :";

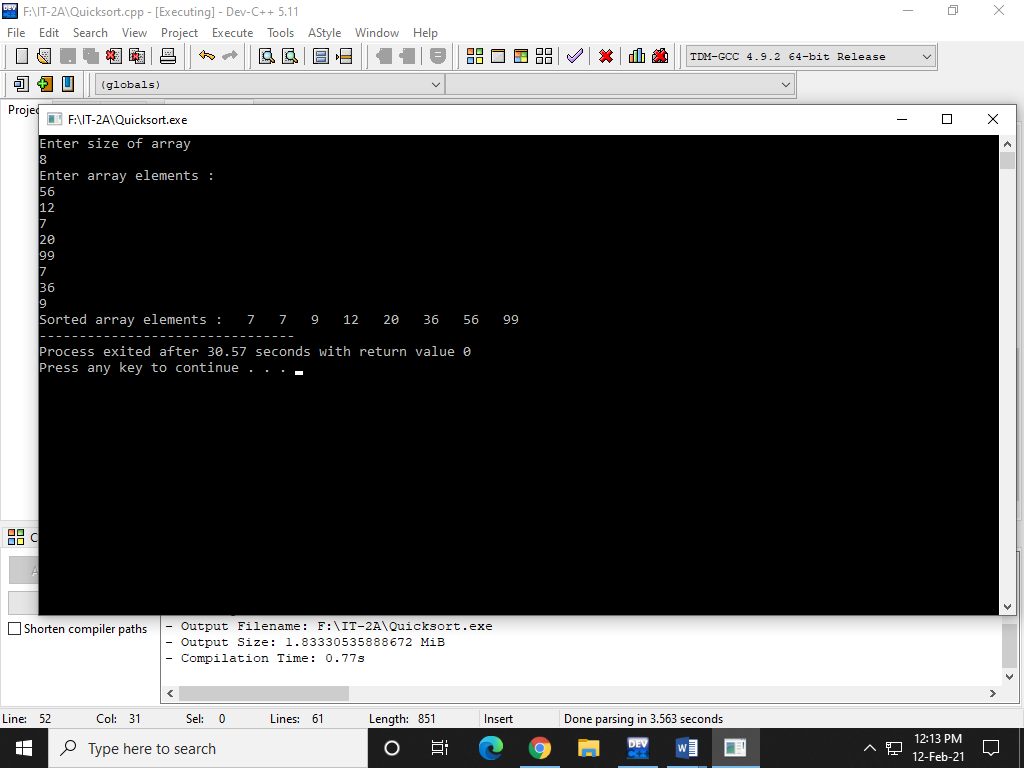
for(i=0;i<n;i++)

cout<<" "<<a[i];

return 0;

}

**Output:-**



**10. Write a C++ program that implements Heap sort algorithm for sorting a list of integers in ascending order.**

**Program:-**

#include<iostream>

using namespace std;

void MaxHeapify(int a[], int i, int n)

{

int j, temp;

temp = a[i];

j = 2\*i;

while (j <= n)

{

if (j < n && a[j+1] > a[j])

j = j+1;

// Break if parent value is already greater than child value.

if (temp > a[j])

break;

// Switching value with the parent node if temp < a[j].

else if (temp <= a[j])

{

a[j/2] = a[j];

j = 2\*j;

}

}

a[j/2] = temp;

return;

}

void HeapSort(int a[], int n)

{

int i, temp;

for (i = n; i >= 2; i--)

{

// Storing maximum value at the end.

temp = a[i];

a[i] = a[1];

a[1] = temp;

// Building max heap of remaining element.

MaxHeapify(a, 1, i - 1);

}

}

void Build\_MaxHeap(int a[], int n)

{

int i;

for(i = n/2; i >= 1; i--)

MaxHeapify(a, i, n);

}

int main()

{

int n, i;

cout<<"\nEnter the number of data element to be sorted: ";

cin>>n;

n++;

int arr[n];

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

{

cout<<"Enter element "<<i<<": ";

cin>>arr[i];

}

// Building max heap.

Build\_MaxHeap(arr, n-1);

HeapSort(arr, n-1);

// Printing the sorted data.

cout<<"\nSorted Elements are: ";

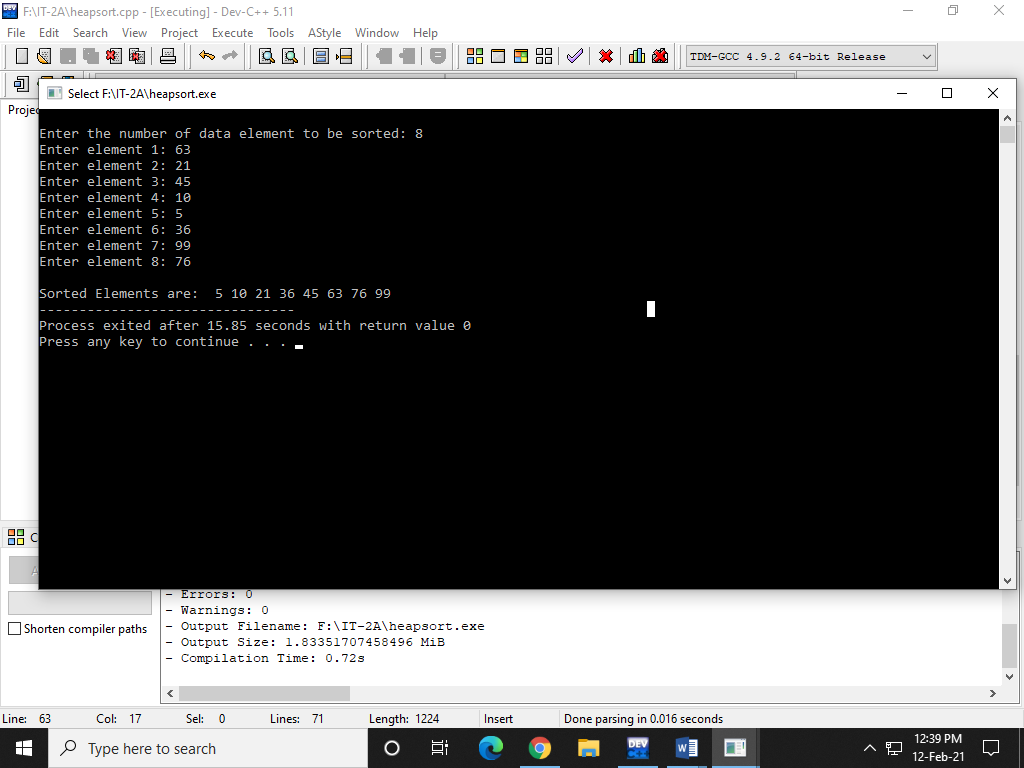
for (i = 1; i < n; i++)

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

return 0;

}

**Output:-**



**11.Write a C++ program that implements Merge sort algorithm for sorting a list of integers in ascending order ?**

**Program:-**

#include<iostream>

using namespace std;

int a[20];

void merge(int low, int mid, int high)

{

int i,j,k,h,b[20];

h=low;

i=low;

j=mid+1;

while(h<=mid && j<=high)

{

if(a[h]<=a[j])

{

b[i]=a[h];

h++;

}

else

{

b[i]=a[j];

j++;

}

i++;

}

if(h>mid)

for(k=j; k<=high; k++)

{

b[i]=a[k];

i++;

}

else

for(k=h; k<=mid; k++)

{

b[i]=a[k];

i++;

}

for(k=low; k<=high; k++)

a[k]=b[k];

}

void mergesort(int low, int high)

{

int mid;

if(low<high)

{

mid=(low+high)/2;

mergesort(low,mid);

mergesort(mid+1,high);

merge(low,mid,high);

}

}

int main()

{

int n,i,size;

cout<<"Enter the size of an array :";

cin>>n;

cout<<"Enter array elements :";

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

cin>>a[i];

size=n-1;

mergesort(0,size);

cout<<"Sorted array : ";

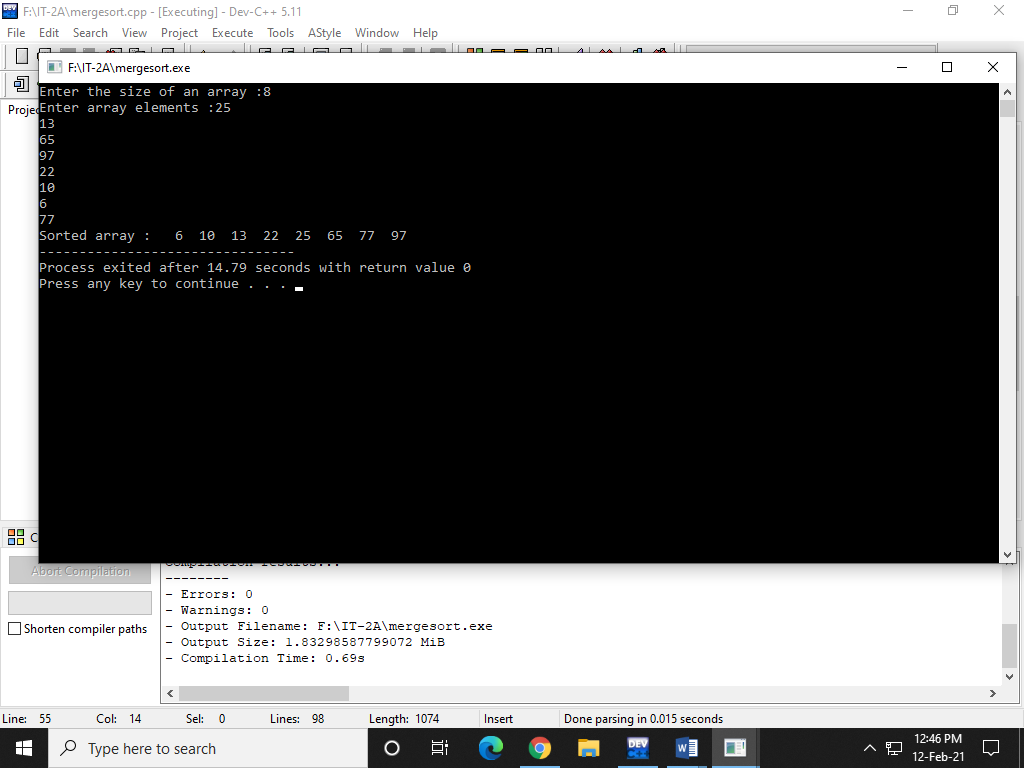
for(i=0;i<n;i++)

cout<<" "<<a[i];

return 0;

}

**Output:-**



**12. Write a C++ program to implement all the functions of a dictionary (ADT) using hashing.**

**Program:-**

#include<iostream>

#include<stdlib.h>

using namespace std;

# define max 10

typedef struct list

{

int data;

struct list \*next;

}node\_type;

node\_type \*ptr[max],\*root[max],\*temp[max];

class Dictionary

{

public:

int index;

Dictionary();

void insert(int);

void search(int);

void delete\_ele(int);

};

Dictionary::Dictionary()

{

index=-1;

for(int i=0;i<max;i++)

{

root[i]=NULL;

ptr[i]=NULL;

temp[i]=NULL;

}

}

void Dictionary::insert(int key)

{

index=int(key%max);

ptr[index]=(node\_type\*)malloc(sizeof(node\_type));

ptr[index]->data=key;

if(root[index]==NULL)

{

root[index]=ptr[index];

root[index]->next=NULL;

temp[index]=ptr[index];

}

else

{

temp[index]=root[index];

while(temp[index]->next!=NULL)

temp[index]=temp[index]->next;

temp[index]->next=ptr[index];

}

}

void Dictionary::search(int key)

{

int flag=0;

index=int(key%max);

temp[index]=root[index];

while(temp[index]!=NULL)

{

if(temp[index]->data==key)

{

cout<<"\nSearch key is found!!";

flag=1;

break;

}

else temp[index]=temp[index]->next;

}

if (flag==0)

cout<<"\nsearch key not found.......";

}

void Dictionary::delete\_ele(int key)

{

index=int(key%max);

temp[index]=root[index];

while(temp[index]->data!=key && temp[index]!=NULL)

{

ptr[index]=temp[index];

temp[index]=temp[index]->next;

}

ptr[index]->next=temp[index]->next;

cout<<"\n"<<temp[index]->data<<" has been deleted.";

temp[index]->data=-1;

temp[index]=NULL;

free(temp[index]);

}

int main()

{

int val,ch,n,num;

char c;

Dictionary d;

do

{

cout<<"\nMENU:\n1.Create";

cout<<"\n2.Search for a value\n3.Delete an value";

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:cout<<"\nEnter the number of elements to be inserted:";

cin>>n;

cout<<"\nEnter the elements to be inserted:";

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

{

cin>>num;

d.insert(num);

}

break;

case 2:cout<<"\nEnter the element to be searched:";

cin>>n;

d.search(n);

break;

case 3:cout<<"\nEnter the element to be deleted:";

cin>>n;

d.delete\_ele(n);

break;

default:cout<<"\nInvalid choice....";

}

cout<<"\nEnter y to continue......";

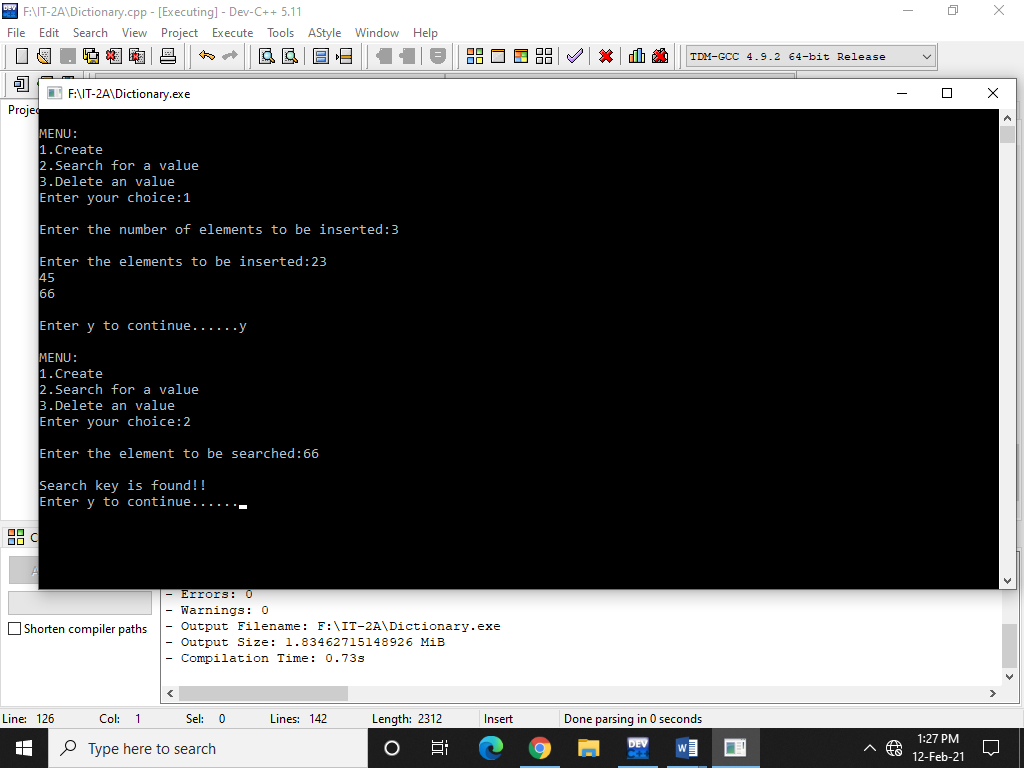
cin>>c;

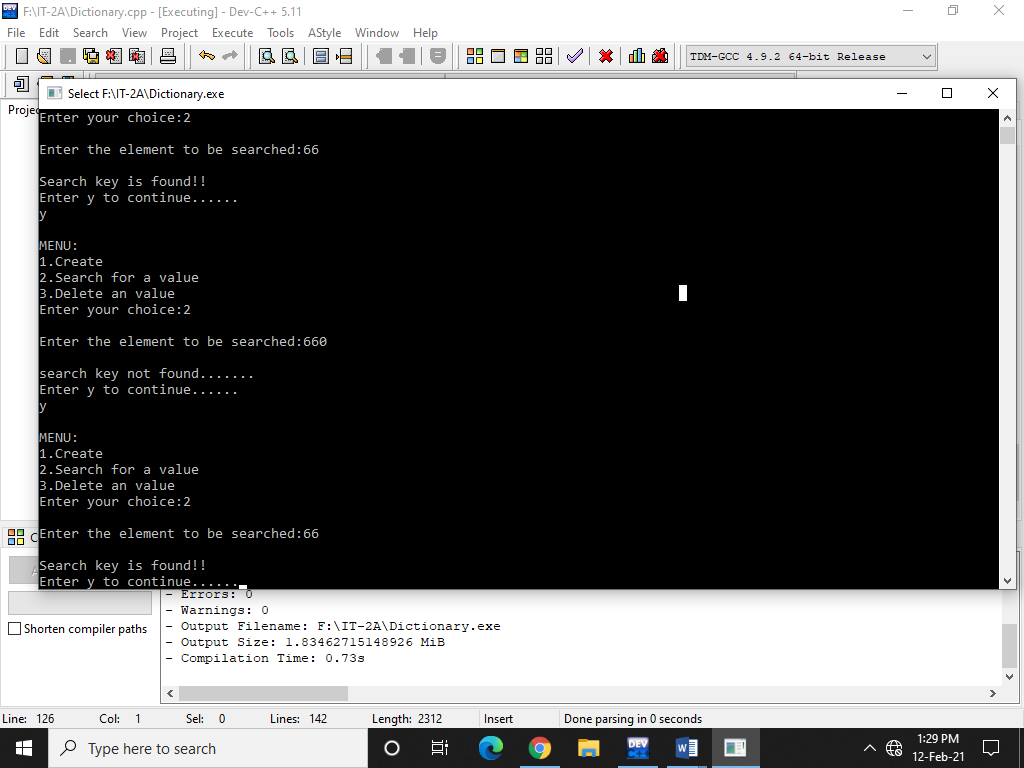
}while(c=='y');

return 0;

}

**Output:-**





**13. Write a C++ program that implements Radix sort algorithm for sorting a list of integers in ascending order**

**Program:-**

#include<iostream>

using namespace std;

int largest(int a[]);

void radix\_sort(int a[]);

int main()

{

int i;

int a[10]={90,23,101,45,65,53,67,89,34,93};

radix\_sort(a);

cout<<"\n The sorted array is: \n";

for(i=0;i<10;i++)

cout<<" "<< a[i];

return 0;

}

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; } }

**Output:-**

