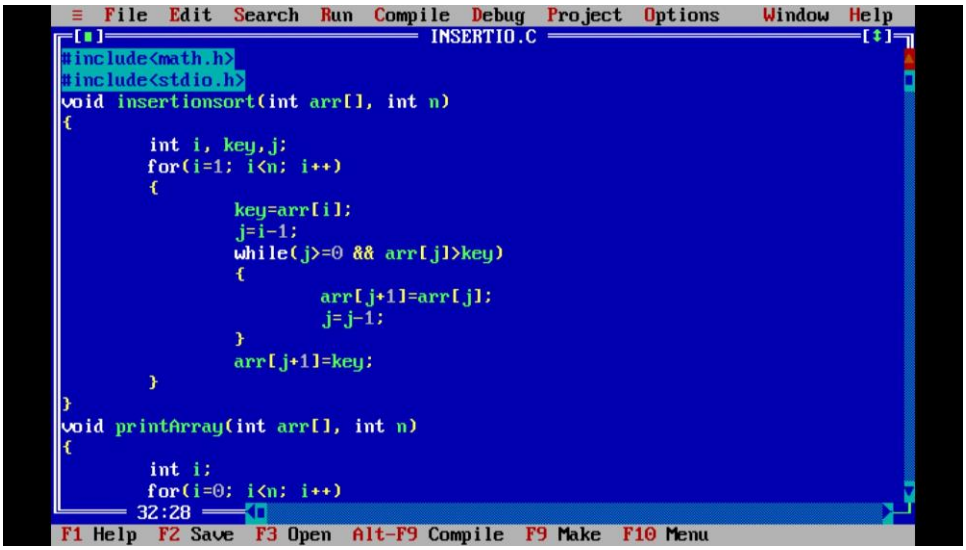
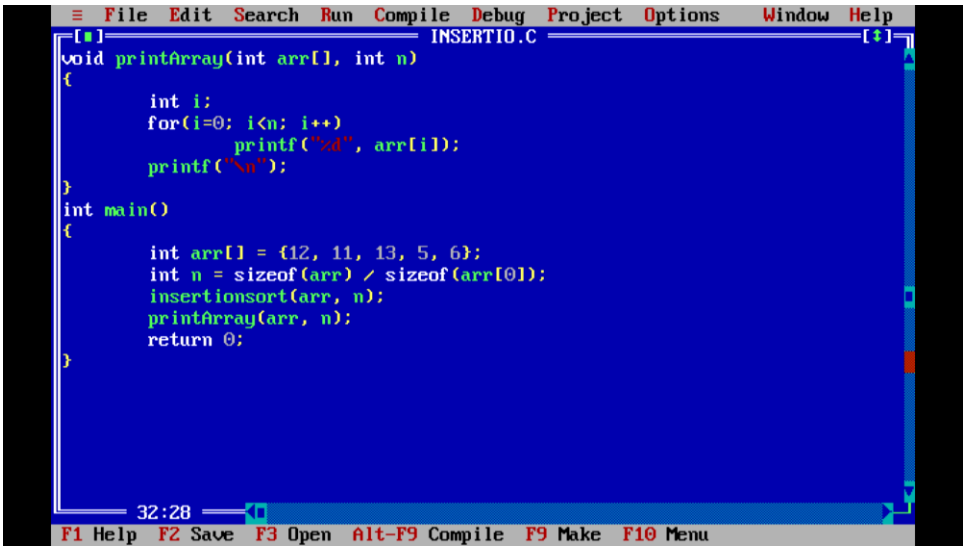
Name : Atharva Karkar
Class : MCA(DS)
Roll no : 8

Assignment 4

Aim1 : Design an algorithm and implement programs for :

a. Insertion Sort

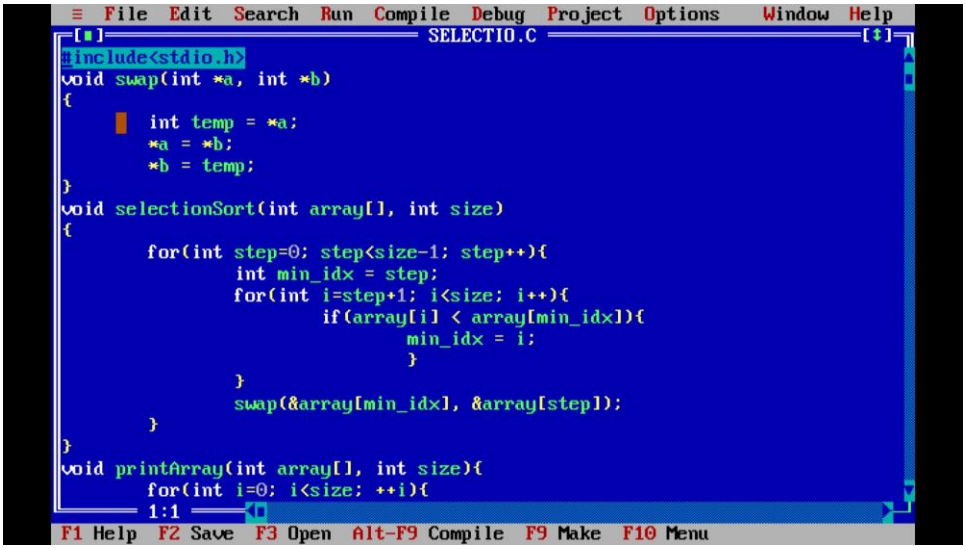




Output :

```
5 6 11 12 13


...Program finished with exit code 0
Press ENTER to exit console.
```

b. Selection Sort

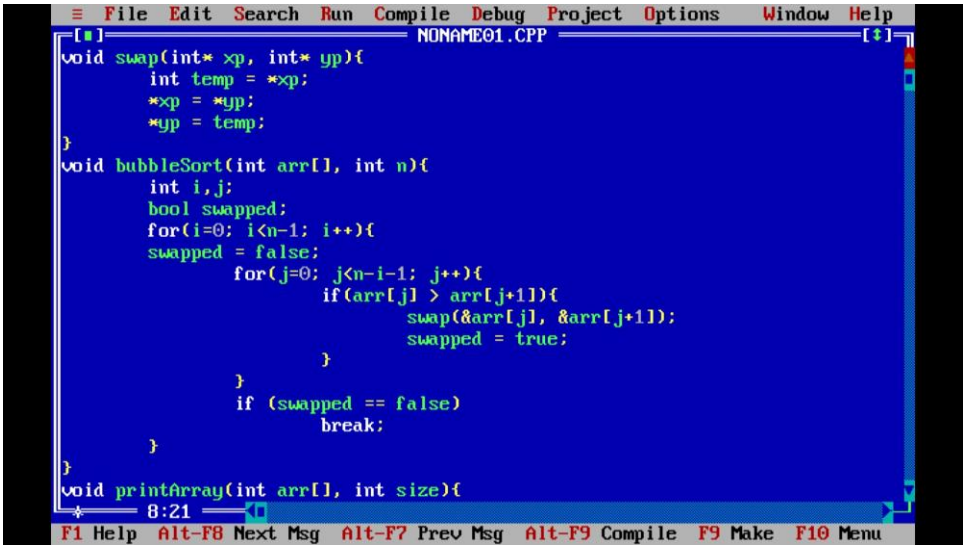


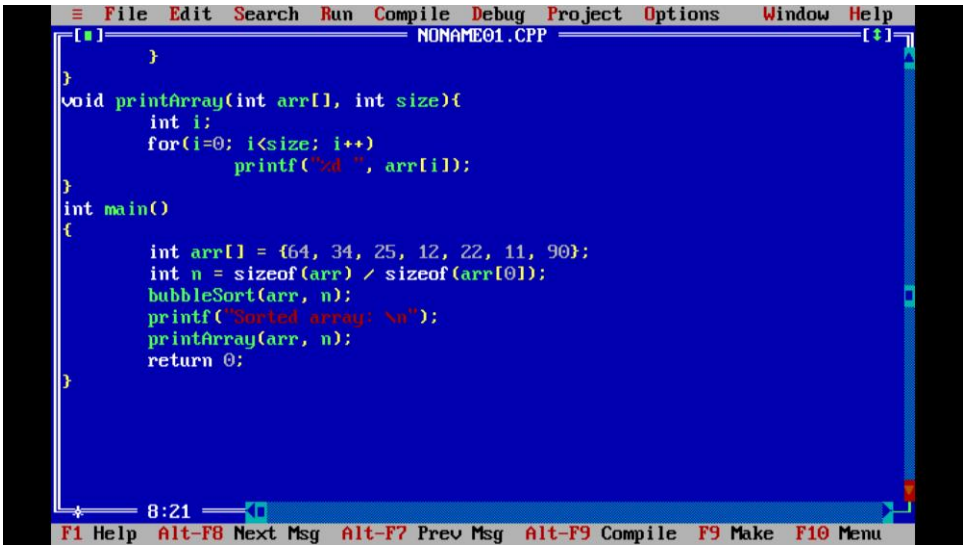
```
 ≡ File  Edit  Search  Run  Compile  Debug  Project  Options    Window  Help
[■]═══════════ SELECTIO.C ═══════════[↕]
}
void printArray(int array[], int size){
        for(int i=0; i<size; ++i){
                printf("%d ", array[i]);
        }
        printf("\n");
}
int main(){
        int data[]={20, 12, 10, 15, 2};
        int size = sizeof(data)/sizeof(data[0]);
        selectionSort(data, size);
        printf("Sorted array in ascending order:\n");
        printArray(data, size);
        return 0;
}
═══ 1:1 ═══
F1 Help  F2 Save  F3 Open  Alt-F9 Compile  F9 Make  F10 Menu
```

Output:

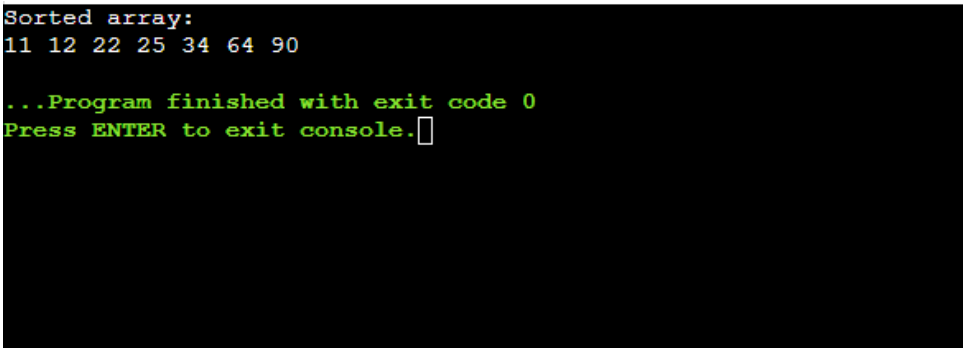
```
Sorted array in Acsending Order:
2  10  12  15  20


...Program finished with exit code 0
Press ENTER to exit console.
```

c. Bubble Sort

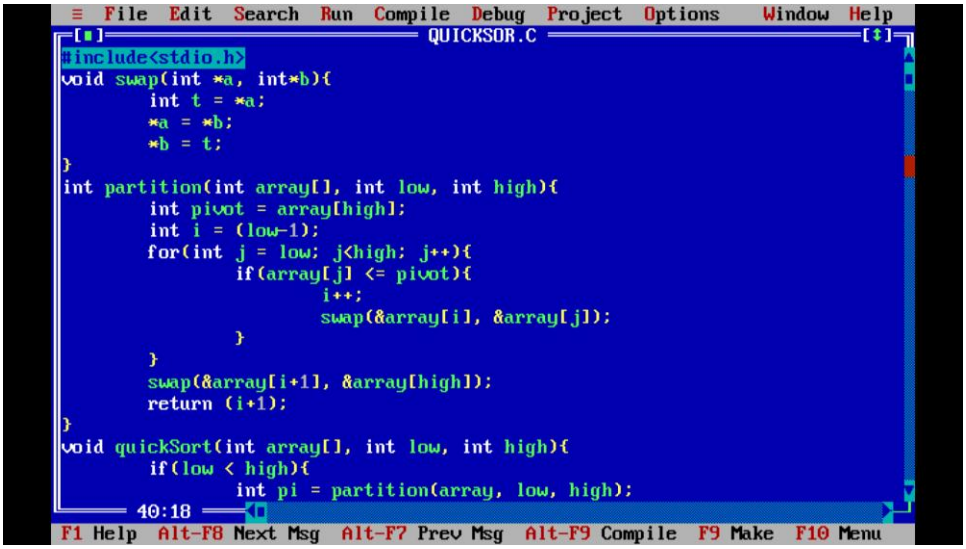


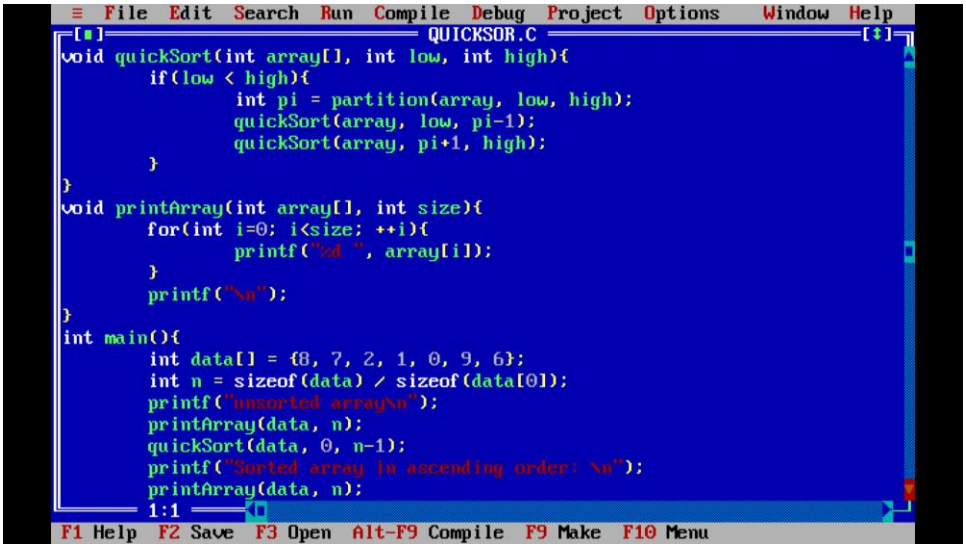


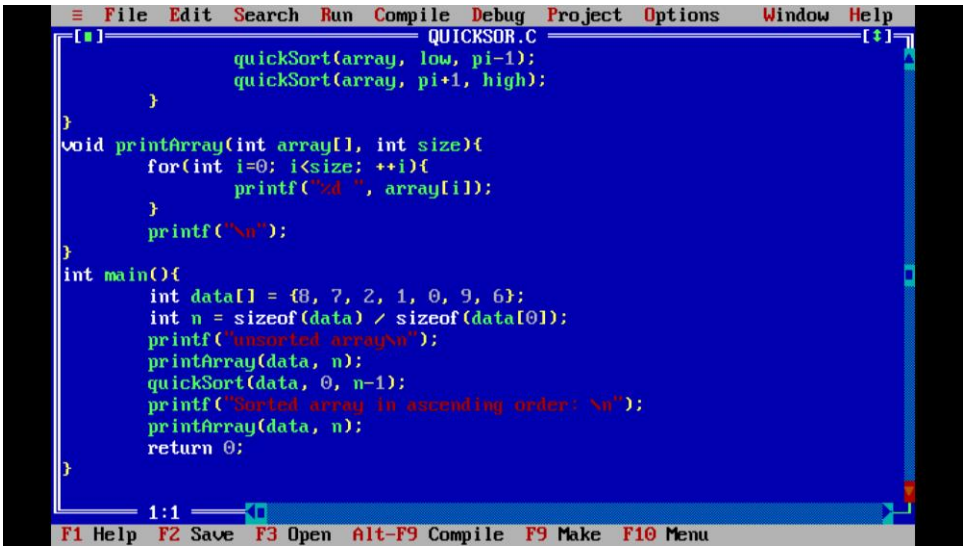
Output:



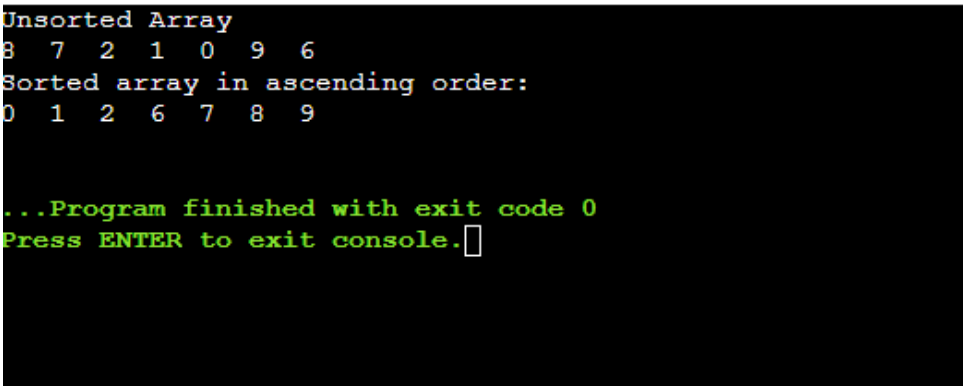
d. Quick Sort


File Edit Search Run Compile Debug Project Options Window Help
QUICKSOR.C
#include<stdio.h>
void swap(int *a, int*b){
int t = *a;
*a = *b;
*b = t;
}
int partition(int array[], int low, int high){
int pivot = array[high];
int i = (low-1);
for(int j = low; j<high; j++){
if(array[j] <= pivot){
i++;
swap(&array[i], &array[j]);
}
}
swap(&array[i+1], &array[high]);
return (i+1);
}
void quickSort(int array[], int low, int high){
if(low < high){
int pi = partition(array, low, high);
40:18
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
QUICKSOR.C
void quickSort(int array[], int low, int high){
if(low < high){
int pi = partition(array, low, high);
quickSort(array, low, pi-1);
quickSort(array, pi+1, high);
}
}
void printArray(int array[], int size){
for(int i=0; i<size; ++i){
printf("%d ", array[i]);
}
printf("\n");
}
int main(){
int data[] = {8, 7, 2, 1, 0, 9, 6};
int n = sizeof(data) / sizeof(data[0]);
printf("unsorted array\n");
printArray(data, n);
quickSort(data, 0, n-1);
printf("Sorted array in ascending order: \n");
printArray(data, n);
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
QUICKSOR.C
quickSort(array, low, pi-1);
quickSort(array, pi+1, high);
}
}
void printArray(int array[], int size){
for(int i=0; i<size; ++i){
printf("%d ", array[i]);
}
printf("\n");
}
int main(){
int data[] = {8, 7, 2, 1, 0, 9, 6};
int n = sizeof(data) / sizeof(data[0]);
printf("unsorted array\n");
printArray(data, n);
quickSort(data, 0, n-1);
printf("Sorted array in ascending order: \n");
printArray(data, n);
return 0;
}
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Output:



Aim2 : Design an algorithm and implement a program for:

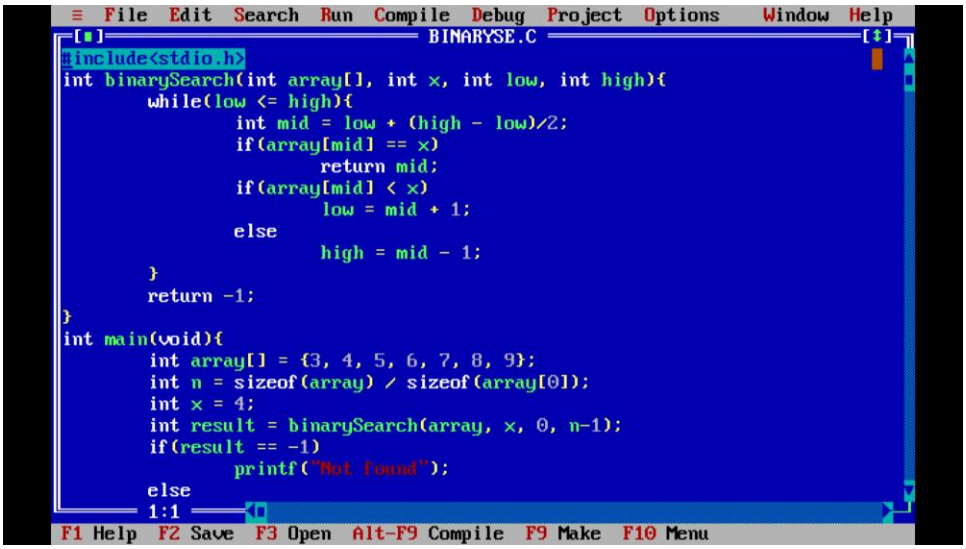
a. Linear Search

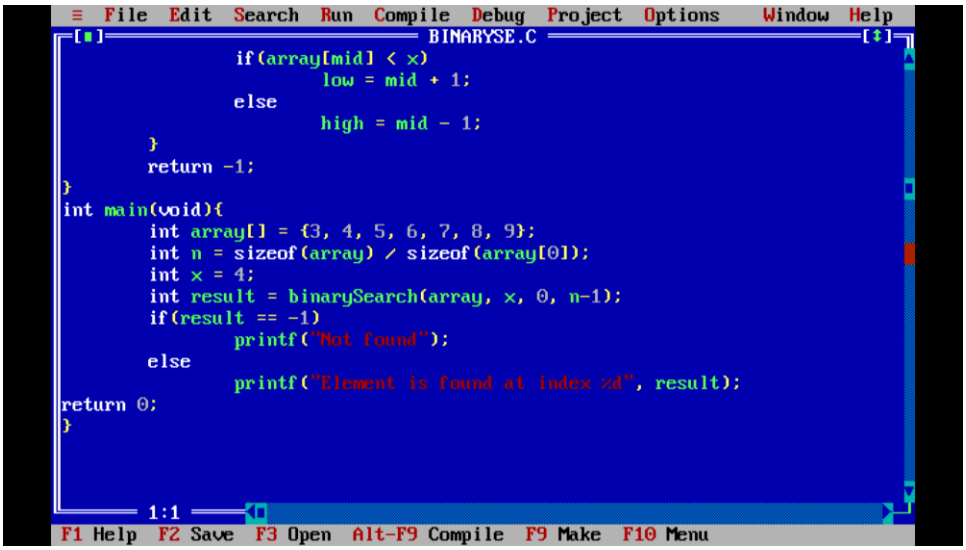
```
#include<stdio.h>
int LINEAR_SEARCH(int inp_arr[], int size, int val)
{
        for(int i=0; i<size; i++)
                if(inp_arr[i] == val)
                        return i;
        return -1;
}
int main(void)
{
        int arr[] = {10, 20, 30, 40, 50, 100, 0};
        int key = 100;
        int size = 10;
        int res = LINEAR_SEARCH(arr, size, key);
        if(res == -1)
        printf("ELEMENT NOT FOUND!!");
        else
        printf("Item is present at index %d", res);
        return 0;
}
```

Output:

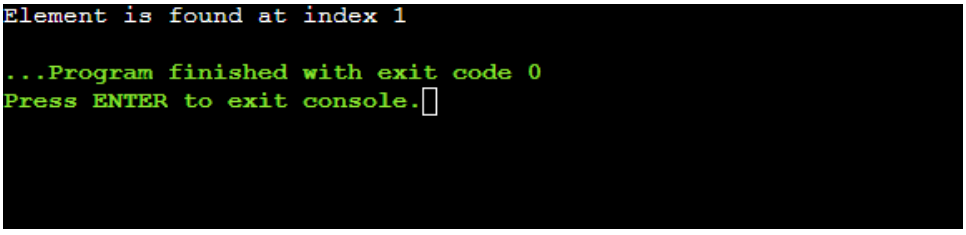
```
Item is present at index 5

...Program finished with exit code 0
Press ENTER to exit console.
```

b. Binary Search


File Edit Search Run Compile Debug Project Options Window Help
BINARYSE.C
#include<stdio.h>
int binarySearch(int array[], int x, int low, int high){
while(low <= high){
int mid = low + (high - low)/2;
if(array[mid] == x)
return mid;
if(array[mid] < x)
low = mid + 1;
else
high = mid - 1;
}
return -1;
}
int main(void){
int array[] = {3, 4, 5, 6, 7, 8, 9};
int n = sizeof(array) / sizeof(array[0]);
int x = 4;
int result = binarySearch(array, x, 0, n-1);
if(result == -1)
printf("Not found");
else
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
BINARYSE.C
if(array[mid] < x)
low = mid + 1;
else
high = mid - 1;
}
return -1;
}
int main(void){
int array[] = {3, 4, 5, 6, 7, 8, 9};
int n = sizeof(array) / sizeof(array[0]);
int x = 4;
int result = binarySearch(array, x, 0, n-1);
if(result == -1)
printf("Not found");
else
printf("Element is found at index %d", result);
return 0;
}
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Output:


Element is found at index 1
...Program finished with exit code 0
Press ENTER to exit console.