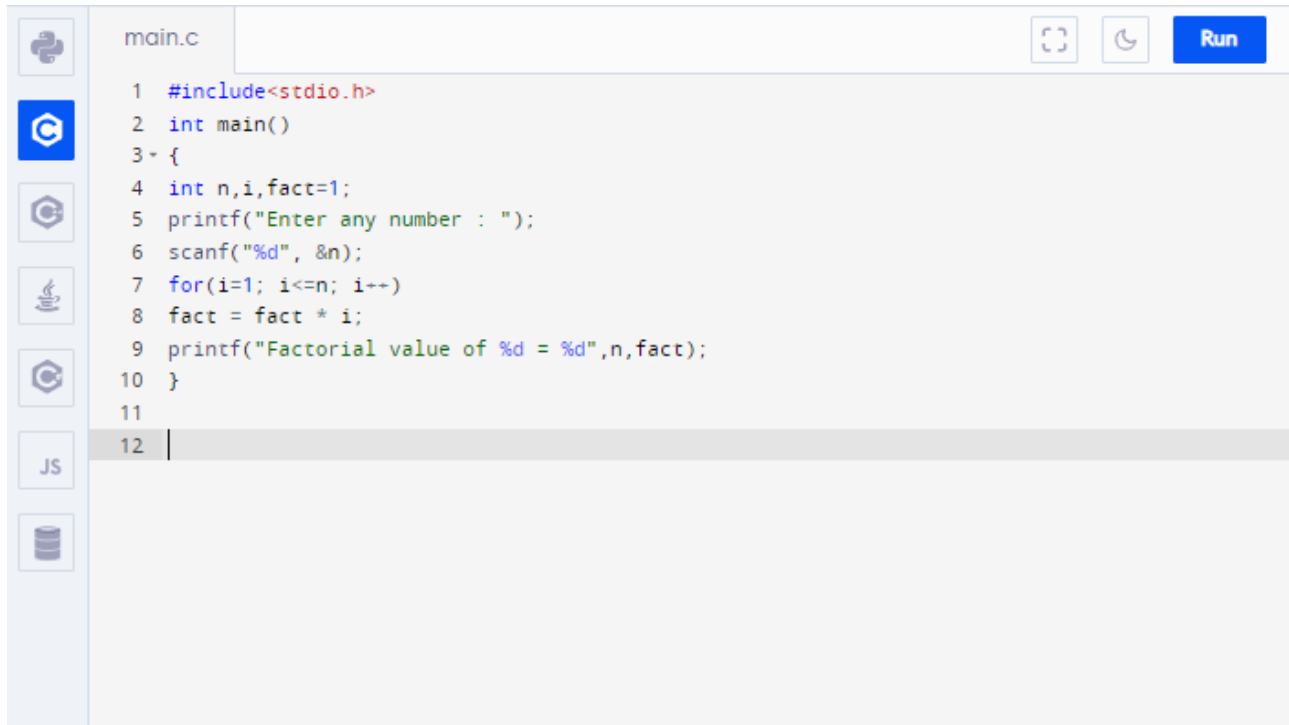


## Program:



The image shows a screenshot of a C programming IDE. On the left is a vertical sidebar with icons for Python, C, C++, Java, JavaScript, and a database. The main editor area is titled 'main.c' and contains the following C code for calculating a factorial. The code is syntax-highlighted: keywords are in blue, integers in green, and strings in red. Line numbers 1 through 12 are visible on the left side of the code. At the top right of the editor, there are icons for a code viewer, a theme selector (showing a moon icon), and a blue 'Run' button.

```
1  #include<stdio.h>
2  int main()
3  {
4  int n,i,fact=1;
5  printf("Enter any number : ");
6  scanf("%d", &n);
7  for(i=1; i<=n; i++)
8  fact = fact * i;
9  printf("Factorial value of %d = %d",n,fact);
10 }
11
12 |
```

### **Output:**

```
Output Clear  
/tmp/MhkJeqhMuJ.o  
Enter any number : 19  
Factorial value of 19 = 109641728|
```

## Program:

The image shows a code editor window with a file named 'main.c'. On the left side, there is a vertical toolbar with icons for Python, C, C++, Java, JavaScript, and a database. The code in the editor is a C program that implements a sorting algorithm. It starts with including the standard input/output library. The main function declares variables for indices, count, a temporary variable, and an array of 25 integers. It prompts the user for the number of elements to enter, reads the count, and then prompts for the elements. A for loop reads the elements into the array. Another for loop then sorts the array using a bubble sort-like logic: it compares adjacent elements and swaps them if they are in the wrong order, repeating this process until the array is sorted. Finally, it prints the sorted elements. The code is as follows:

```
1 #include<stdio.h>
2 int main(){
3     int i, j, count, temp, number[25];
4     printf("How many numbers u are going to enter?: ");
5     scanf("%d",&count);
6     printf("Enter %d elements: ", count);
7     for(i=0;i<count;i++)
8         scanf("%d",&number[i]);
9     for(i=1;i<count;i++){
10         temp=number[i];
11         j=i-1;
12         while((temp<number[j])&&(j>=0)){
13             number[j+1]=number[j];
14             j=j-1;
15         }
16         number[j+1]=temp;
17     }
18     printf("Order of Sorted elements: ");
19     for(i=0;i<count;i++)
20         printf(" %d",number[i]);
21 }
22
23
24
```

### **Output:**

Output

Clear

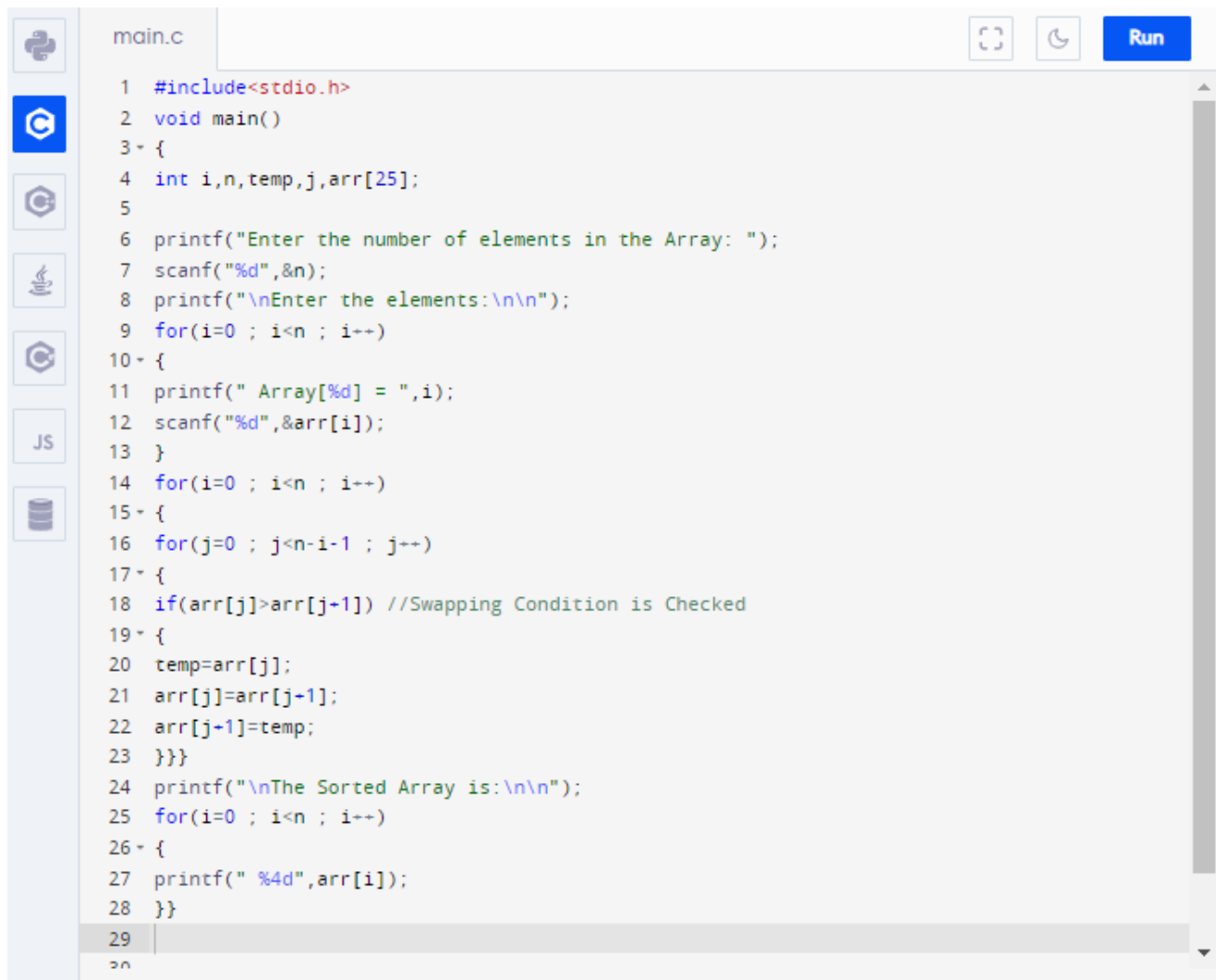
```
/tmp/MhkJeqhMuJ.o
```

```
How many numbers u are going to enter?: 6
```

```
Enter 6 elements: 21 92 12 7 0 4
```

```
Order of Sorted elements: 0 4 7 12 21 92
```

## Program:



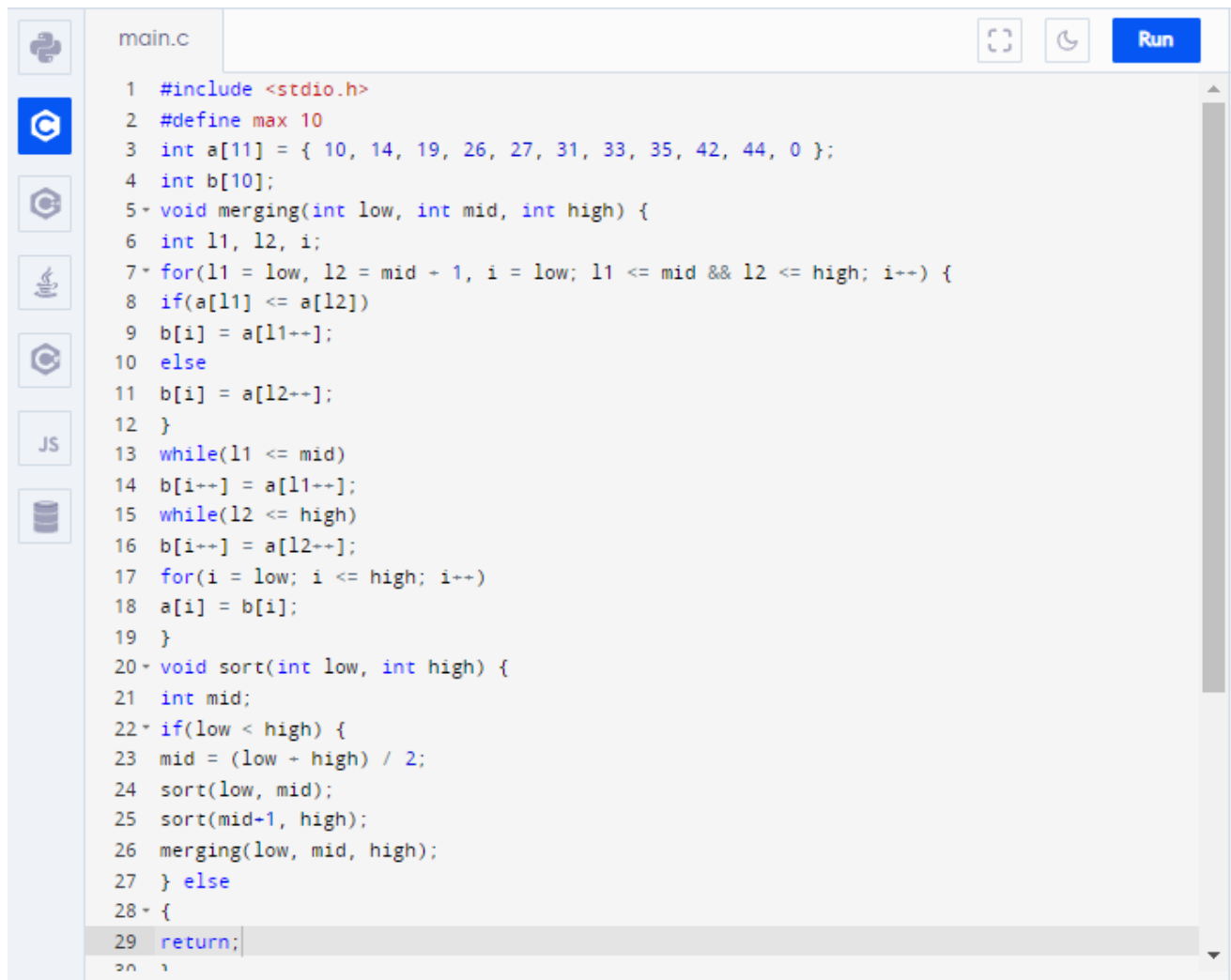
The image shows a code editor window with a file named 'main.c'. The editor has a sidebar on the left with icons for file explorer, search, and other tools. The main area displays C code for a bubble sort algorithm. The code includes a header, a main function, and several loops for input, output, and sorting. A 'Run' button is visible in the top right corner of the editor.

```
1  #include<stdio.h>
2  void main()
3  {
4  int i,n,temp,j,arr[25];
5
6  printf("Enter the number of elements in the Array: ");
7  scanf("%d",&n);
8  printf("\nEnter the elements:\n\n");
9  for(i=0 ; i<n ; i++)
10 {
11 printf(" Array[%d] = ",i);
12 scanf("%d",&arr[i]);
13 }
14 for(i=0 ; i<n ; i++)
15 {
16 for(j=0 ; j<n-i-1 ; j++)
17 {
18 if(arr[j]>arr[j+1]) //Swapping Condition is Checked
19 {
20 temp=arr[j];
21 arr[j]=arr[j+1];
22 arr[j+1]=temp;
23 }}
24 printf("\nThe Sorted Array is:\n\n");
25 for(i=0 ; i<n ; i++)
26 {
27 printf(" %4d",arr[i]);
28 }}
29
30
```








## **Output:**

```
Output Clear  
/tmp/MhkJeqhMuJ.o  
Enter the number of elements in the Array: 10  
Enter the elements:  
  
Array[0] = 1  
Array[1] = 34  
Array[2] = 21  
Array[3] = 0  
Array[4] = 19  
Array[5] = 72  
Array[6] = 82  
Array[7] = 3  
Array[8] = 6  
Array[9] = 2  
The Sorted Array is:  
  
0    1    2    3    6    19   21   34   72   82
```



## Program:



```
main.c
1  #include <stdio.h>
2  #define max 10
3  int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
4  int b[10];
5  void merging(int low, int mid, int high) {
6  int l1, l2, i;
7  for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
8  if(a[l1] <= a[l2])
9  b[i] = a[l1++];
10 else
11 b[i] = a[l2++];
12 }
13 while(l1 <= mid)
14 b[i++] = a[l1++];
15 while(l2 <= high)
16 b[i++] = a[l2++];
17 for(i = low; i <= high; i++)
18 a[i] = b[i];
19 }
20 void sort(int low, int high) {
21 int mid;
22 if(low < high) {
23 mid = (low + high) / 2;
24 sort(low, mid);
25 sort(mid+1, high);
26 merging(low, mid, high);
27 } else
28 {
29 return;
30 }
```



main.c



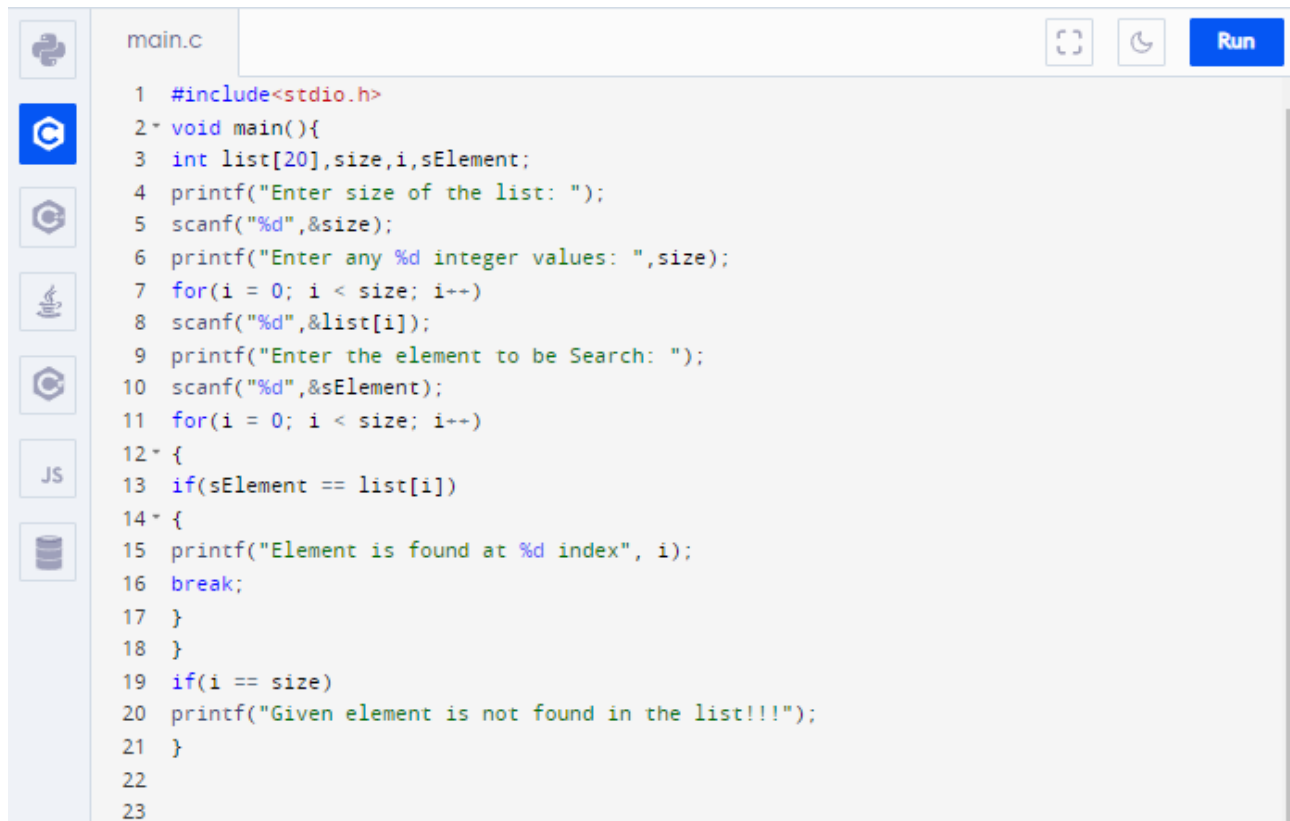
```
28+ {
29  return;
30 }
31 }
32 int main()
33+ {
34  int i;
35  printf("List before sorting\n");
36  for(i = 0; i <= max; i++)
37  printf("%d ", a[i]);
38  sort(0, max);
39  printf("\nList after sorting\n");
40  for(i = 0; i <= max; i++)
41  printf("%d ", a[i]);
42 }
43
44
45
46
47
48
```



## **Output:**

```
Output Clear  
/tmp/MhkJeqhMuJ.o  
List before sorting  
10 14 19 26 27 31 33 35 42 44 0  
List after sorting  
0 10 14 19 26 27 31 33 35 42 44 |
```

## Program:



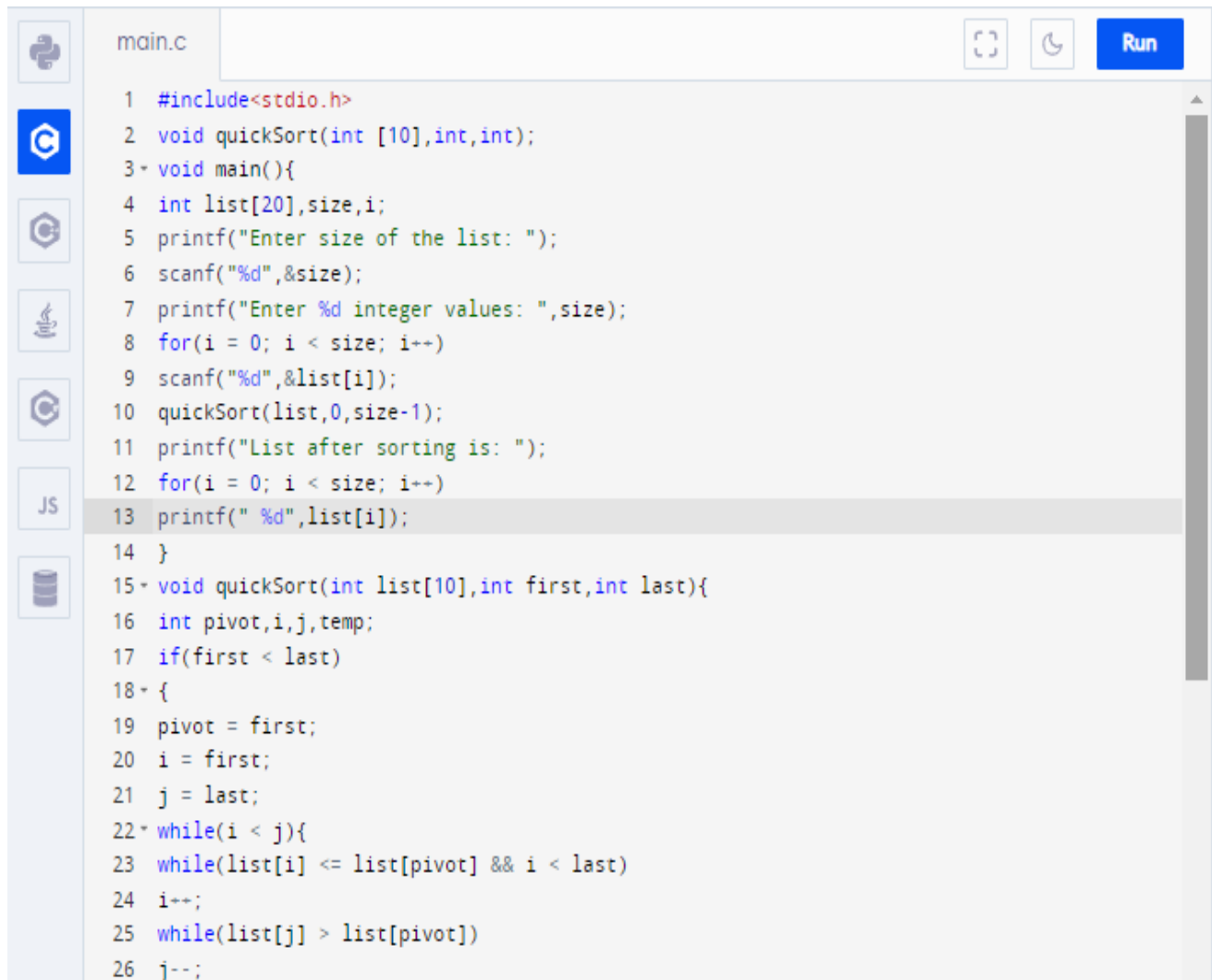
The image shows a screenshot of a C programming IDE. On the left, there is a vertical sidebar with icons for Python, C, C++, Java, JavaScript, and a database. The main area displays a C program named 'main.c'. The program prompts the user to enter the size of a list, then enters integer values for that size. It then prompts for an element to search and checks if it exists in the list. If found, it prints the index; otherwise, it prints a message that the element was not found. The code is as follows:

```
1  #include<stdio.h>
2  void main(){
3  int list[20],size,i,sElement;
4  printf("Enter size of the list: ");
5  scanf("%d",&size);
6  printf("Enter any %d integer values: ",size);
7  for(i = 0; i < size; i++)
8  scanf("%d",&list[i]);
9  printf("Enter the element to be Search: ");
10 scanf("%d",&sElement);
11 for(i = 0; i < size; i++)
12 {
13 if(sElement == list[i])
14 {
15 printf("Element is found at %d index", i);
16 break;
17 }
18 }
19 if(i == size)
20 printf("Given element is not found in the list!!!");
21 }
22
23
```

### **Output:**

```
Output Clear  
/tmp/MhkJeqhMuJ.o  
Enter size of the list: 5  
Enter any 5 integer values: 9  
2  
56  
72  
49  
Enter the element to be Search: 72  
Element is found at 3 index|
```

## Program:



The image shows a screenshot of a C++ IDE. On the left is a sidebar with icons for file explorer, search, and other IDE features. The main area displays a C++ program named 'main.c'. The code implements a Quick Sort algorithm. It starts with including `<stdio.h>`, then defines `quickSort` and `main` functions. The `main` function prompts the user for the size of the list and the integer values, then calls `quickSort`. The `quickSort` function uses a pivot-based partitioning scheme to sort the array. The code is as follows:

```
1  #include<stdio.h>
2  void quickSort(int [10],int,int);
3  void main(){
4  int list[20],size,i;
5  printf("Enter size of the list: ");
6  scanf("%d",&size);
7  printf("Enter %d integer values: ",size);
8  for(i = 0; i < size; i++)
9  scanf("%d",&list[i]);
10 quickSort(list,0,size-1);
11 printf("List after sorting is: ");
12 for(i = 0; i < size; i++)
13 printf(" %d",list[i]);
14 }
15 void quickSort(int list[10],int first,int last){
16 int pivot,i,j,temp;
17 if(first < last)
18 {
19 pivot = first;
20 i = first;
21 j = last;
22 while(i < j){
23 while(list[i] <= list[pivot] && i < last)
24 i++;
25 while(list[j] > list[pivot])
26 j--;
```



main.c









JS





Run

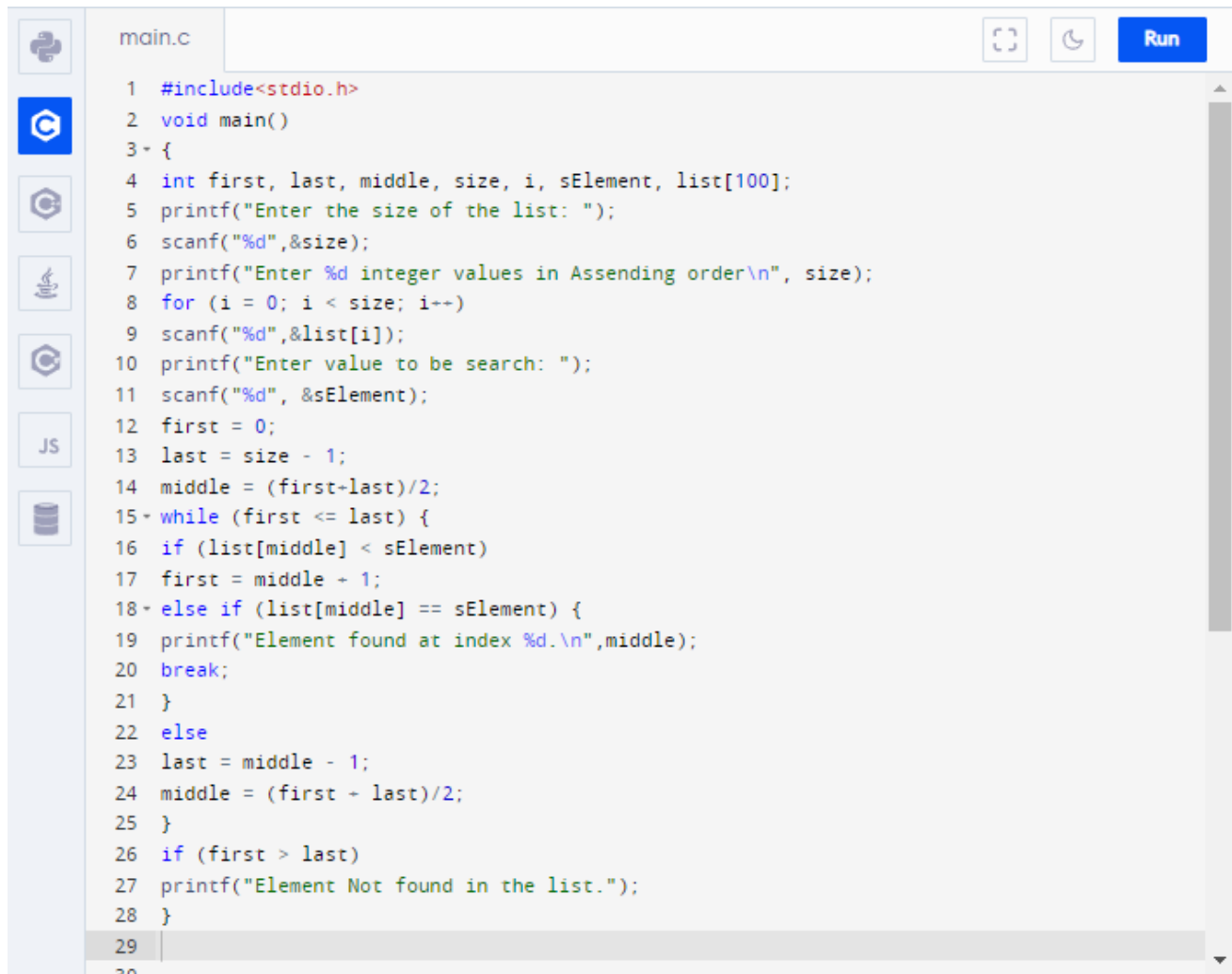
```
25 while(list[j] > list[pivot])
26 j--;
27 if(i < j)
28 {
29     temp = list[i];
30     list[i] = list[j];
31     list[j] = temp;
32 }
33 }
34 temp = list[pivot];
35 list[pivot] = list[j];
36 list[j] = temp;
37 quickSort(list,first,j-1);
38 quickSort(list,j+1,last);
39 }
40 }
41
42
43
44
45
46
47
```

## **Output:**

Output Clear

```
/tmp/MhkJeqhMuJ.o
Enter size of the list: 4
Enter 4 integer values: 21 37 100 29
List after sorting is: 21 29 37 100
```

## Program:



The image shows a screenshot of a C programming IDE. On the left, there is a sidebar with icons for Python, C, C++, Java, JavaScript, and a database. The main editor window is titled 'main.c' and contains the following C code:

```
1  #include<stdio.h>
2  void main()
3  {
4  int first, last, middle, size, i, sElement, list[100];
5  printf("Enter the size of the list: ");
6  scanf("%d",&size);
7  printf("Enter %d integer values in Assending order\n", size);
8  for (i = 0; i < size; i++)
9  scanf("%d",&list[i]);
10 printf("Enter value to be search: ");
11 scanf("%d", &sElement);
12 first = 0;
13 last = size - 1;
14 middle = (first+last)/2;
15 while (first <= last) {
16 if (list[middle] < sElement)
17 first = middle + 1;
18 else if (list[middle] == sElement) {
19 printf("Element found at index %d.\n",middle);
20 break;
21 }
22 else
23 last = middle - 1;
24 middle = (first + last)/2;
25 }
26 if (first > last)
27 printf("Element Not found in the list.");
28 }
29
30
```

The code implements a binary search algorithm. It starts by including `stdio.h` and defining the `main` function. It declares variables for `first`, `last`, `middle`, `size`, `i`, `sElement`, and an array `list` of size 100. It prompts the user to enter the size of the list and then enters `size` integer values in ascending order. It then prompts the user to enter a value to be searched and stores it in `sElement`. The search process begins with `first` at 0 and `last` at `size - 1`. A `while` loop continues as long as `first` is less than or equal to `last`. Inside the loop, it checks if the element at the `middle` index is less than `sElement`. If so, it updates `first` to `middle + 1`. If it is equal, it prints the index and breaks the loop. Otherwise, it updates `last` to `middle - 1` and recalculates `middle`. If the loop ends without finding the element, it prints "Element Not found in the list."

### **Output:**

```
Output Clear  
/tmp/MhkJeqhMuJ.o  
Enter the size of the list: 5  
Enter 5 integer values in Assending order  
1 2 3 4 5  
Enter value to be search: 6  
Element Not found in the list.
```