Unit 2

Program for selection sort

Aim:

- To implement selection sort

Theory:

- The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning.

- The algorithm maintains two subarrays in a given array.

- The subarray which already sorted.
- The remaining subarray was unsorted.

- In every iteration of the selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray

How selection sort works?

Lets consider the following array as an example: ***arr[] = {64, 25, 12, 22, 11}***

First pass:

- *For the first position in the sorted array, the whole array is traversed from index 0 to 4 sequentially. The first position where* ***64*** *is stored presently, after traversing whole array it is clear that* ***11*** *is the lowest value.*

64	*25*	*12*	*22*	*11*

- *Thus, replace 64 with 11. After one iteration* ***11****, which happens to be the least value in the array, tends to appear in the first position of the sorted list.*

11	*25*	*12*	*22*	*64*

Second Pass:

- *For the second position, where 25 is present, again traverse the rest of the array in a sequential manner.*

11	***25***	*12*	*22*	*64*

- *After traversing, we found that **12** is the second lowest value in the array and it should appear at the second place in the array, thus swap these values.*

11	***12***	*25*	*22*	*64*

Third Pass:

- *Now, for third place, where **25** is present again traverse the rest of the array and find the third least value present in the array.*

11	*12*	***25***	*22*	*64*

- *While traversing, **22** came out to be the third least value and it should appear at the third place in the array, thus swap **22** with element present at third position.*

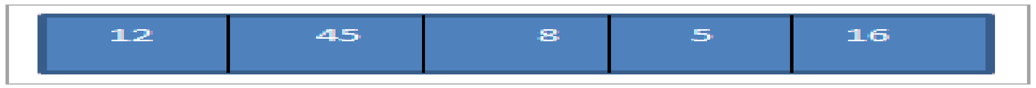
11	*12*	***22***	*25*	*64*

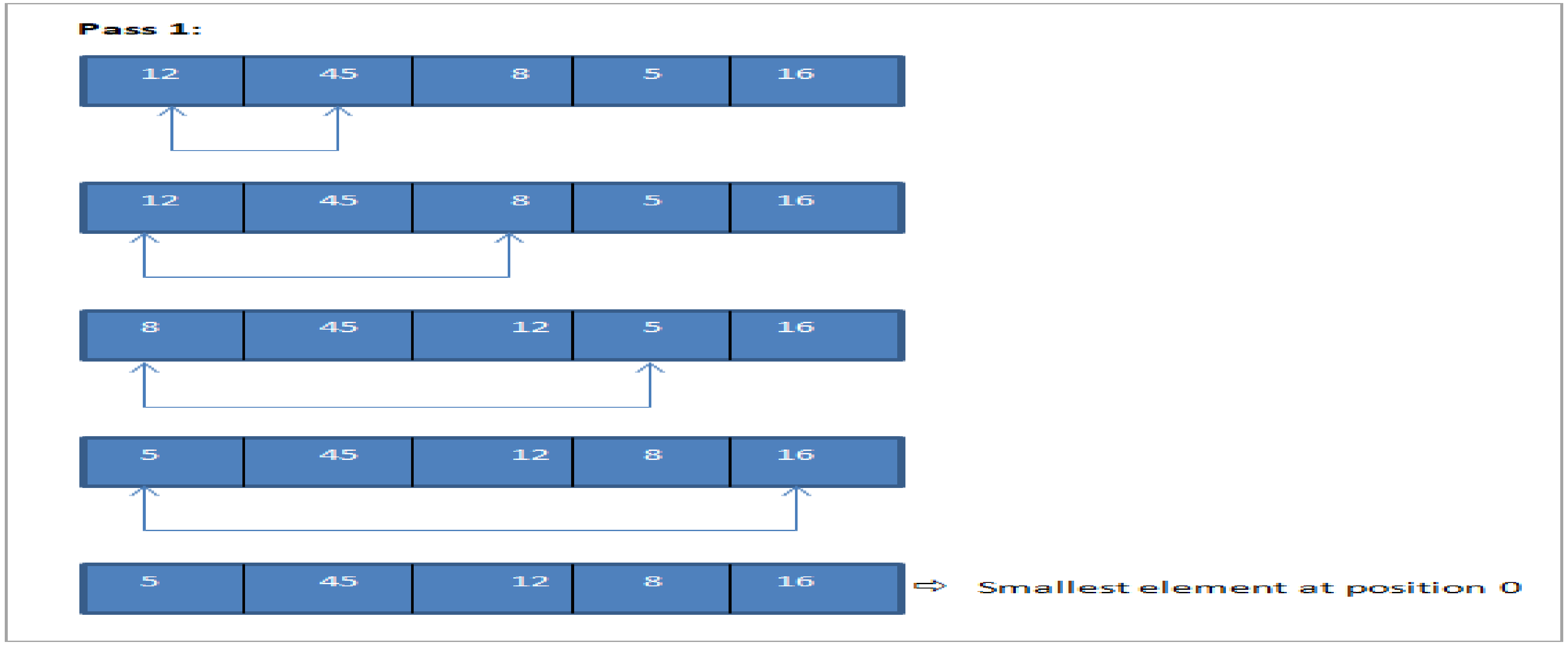
Fourth pass:

- *Similarly, for fourth position traverse the rest of the array and find the fourth least element in the array*
- *As **25** is the 4th lowest value hence, it will place at the fourth position.*

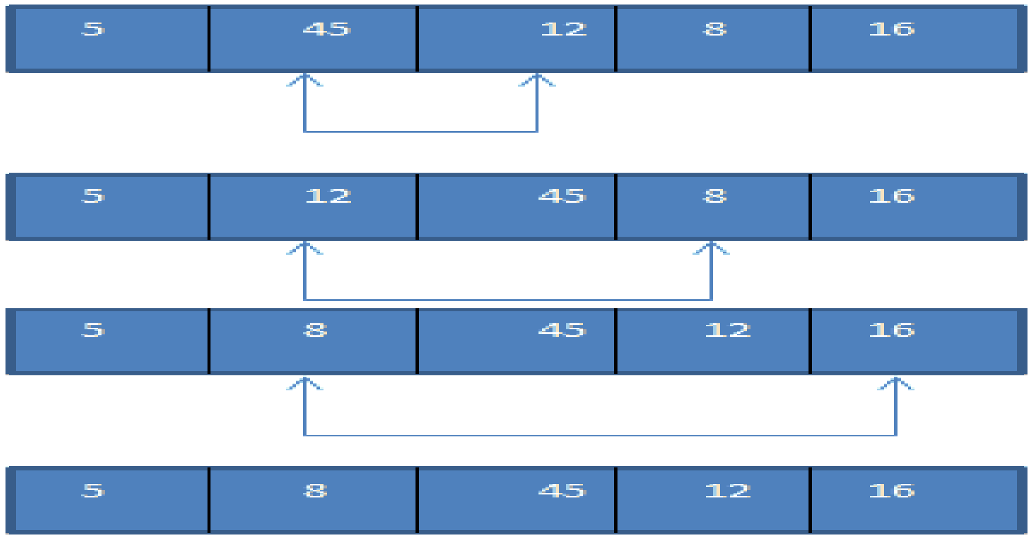
11	*12*	*22*	***25***	*64*

Another example


12
45
8
5
16

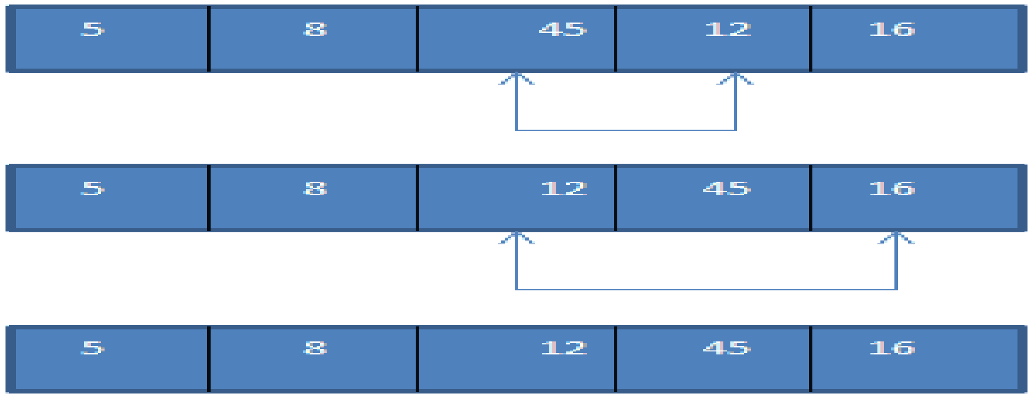

Pass 1:
12
45
8
5
16
12
45
8
5
16
8
45
12
5
16
5
45
12
8
16
5
45
12
8
16
⇨ Smallest element at position 0

Pass 2:



⇨ Next smallest element at position 1

Pass 3:



⇨ Third smallest element at position 2

- As shown in the above illustration,
- for N number of elements we take N-1 passes to completely sort the array.
- At the end of every pass, the smallest element in the array is placed at its proper position in the sorted array.

Procedure:

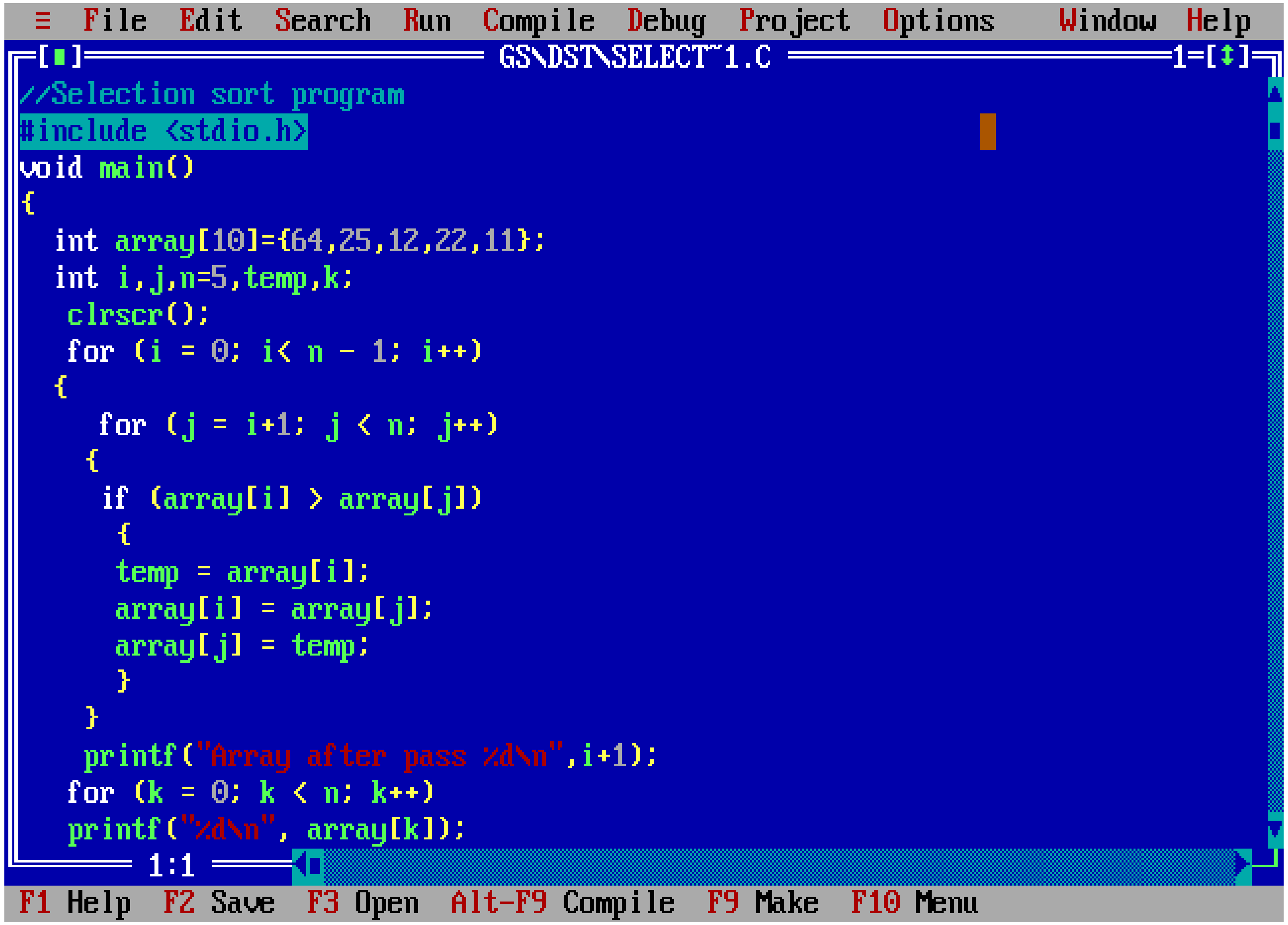
Algorithm

- Initialize minimum value to element at location 0.(min_element)
- Traverse the array to find the minimum element in the array.
- While traversing if any element smaller than min_element is found then swap both the values.
- Then, increment index to point to the next element.
- Repeat until the array is sorted.

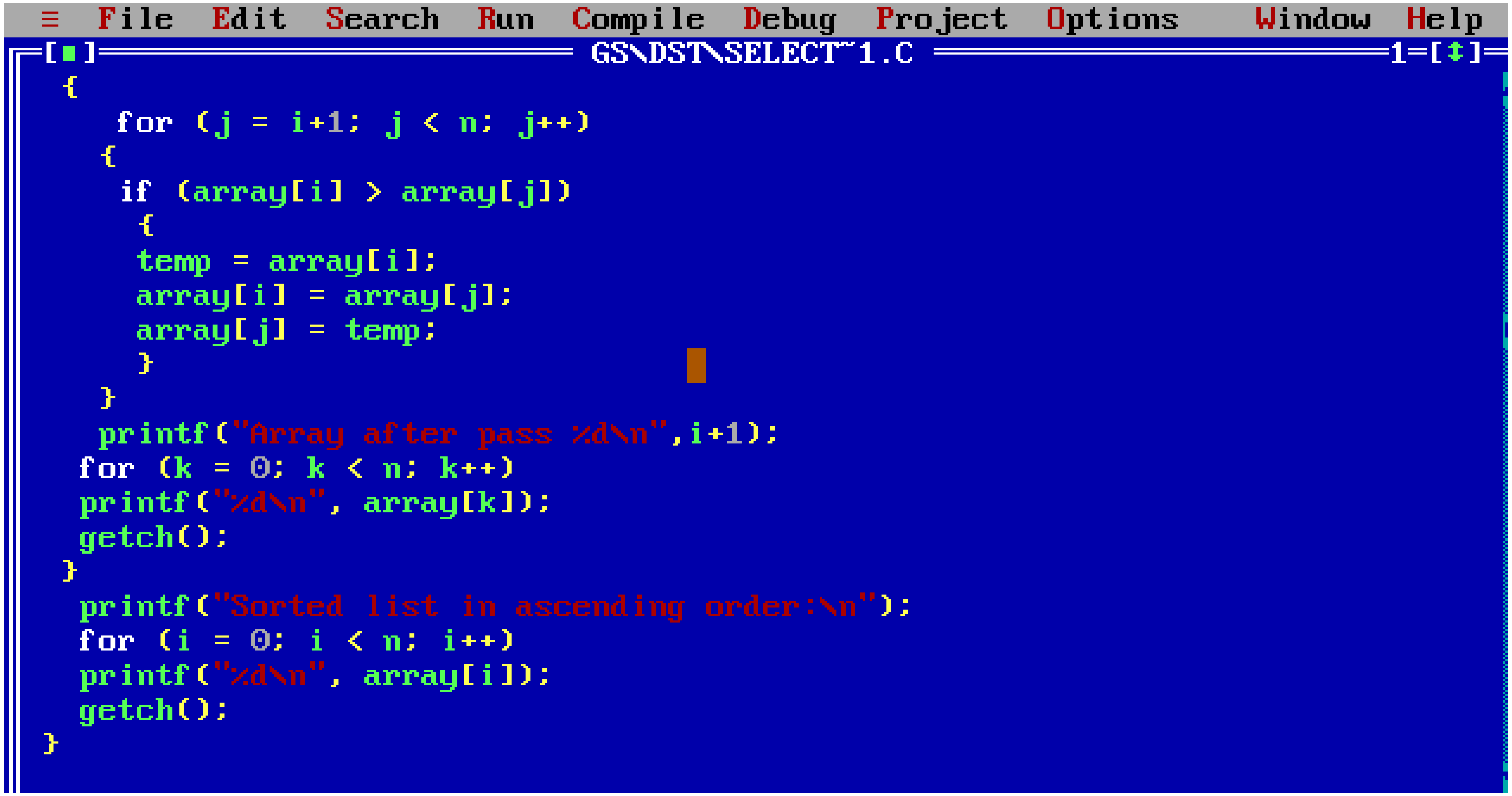
Selection Sort Algorithm

```
• selectionSort(array, size)
•   repeat (size - 1) times
•   set the first unsorted element as the minimum
•   for each of the unsorted elements
•     if element < currentMinimum
•       set element as new minimum
•   swap minimum with first unsorted position
• end selectionSort
```

Source code:

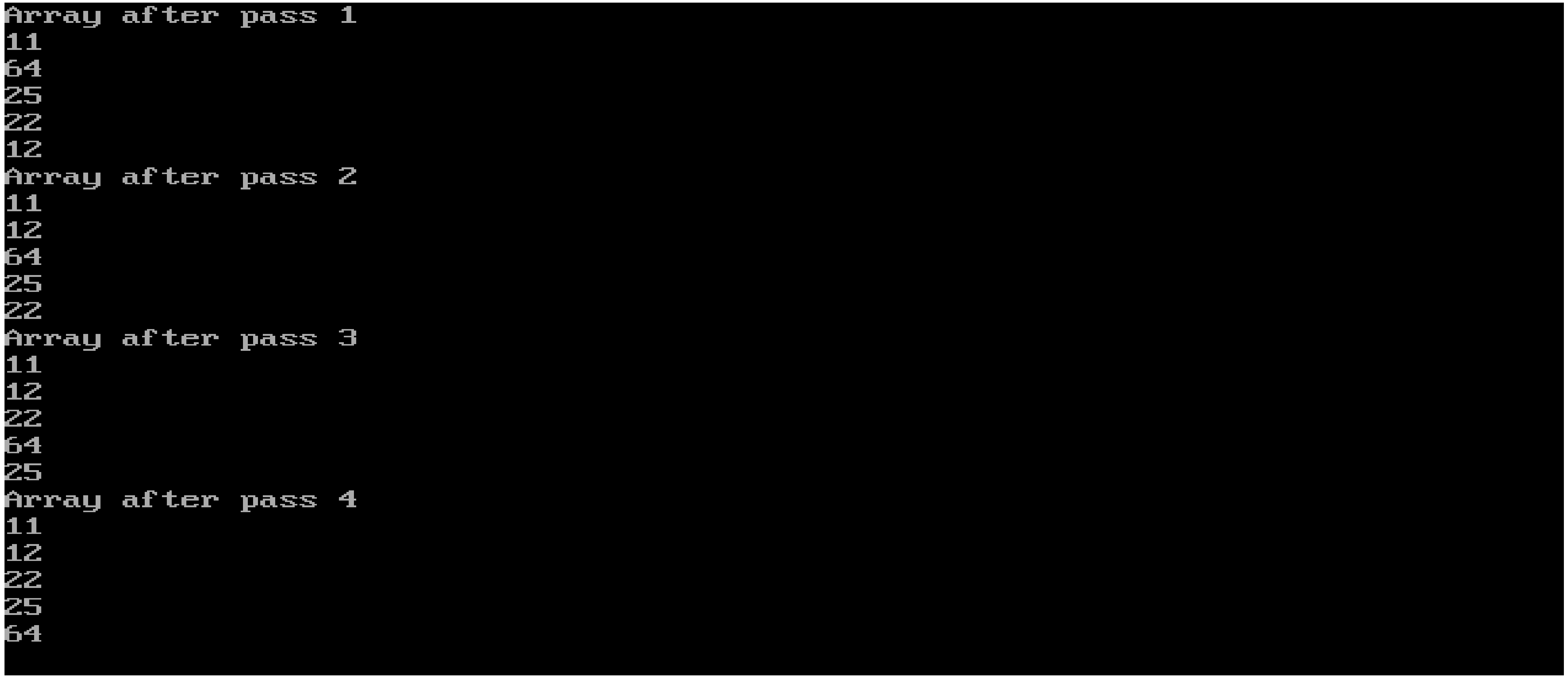

≡ File Edit Search Run Compile Debug Project Options Window Help
GS\DST\SELECT~1.C
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
//Selection sort program
#include <stdio.h>
void main()
{
  int array[10]={64,25,12,22,11};
  int i,j,n=5,temp,k;
   clrscr();
   for (i = 0; i< n - 1; i++)
  {
     for (j = i+1; j < n; j++)
    {
     if (array[i] > array[j])
      {
      temp = array[i];
      array[i] = array[j];
      array[j] = temp;
      }
    }
    printf("Array after pass %d\n",i+1);
   for (k = 0; k < n; k++)
   printf("%d\n", array[k]);
```

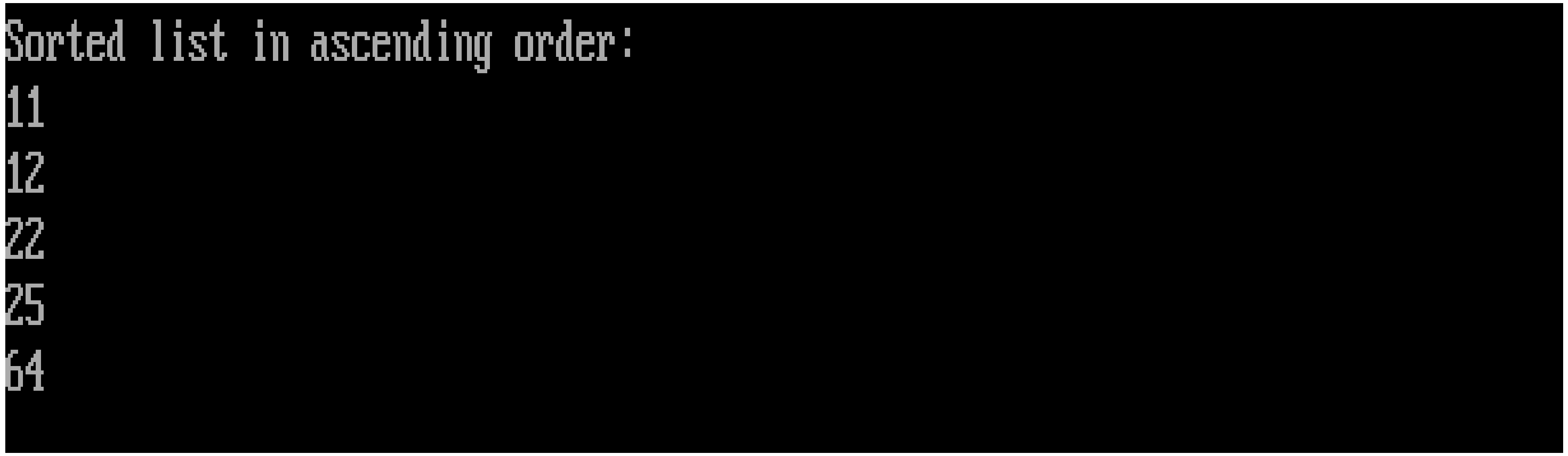

≡ File Edit Search Run Compile Debug Project Options Window Help
[■]
GS\DST\SELECT~1.C
1=[↕]

```
{
    for (j = i+1; j < n; j++)
   {
    if (array[i] > array[j])
     {
     temp = array[i];
     array[i] = array[j];
     array[j] = temp;
     }
   }
   printf("Array after pass %d\n",i+1);
  for (k = 0; k < n; k++)
  printf("%d\n", array[k]);
  getch();
 }
  printf("Sorted list in ascending order:\n");
  for (i = 0; i < n; i++)
  printf("%d\n", array[i]);
  getch();
}
```

output



```
Array after pass 1
11
64
25
22
12
Array after pass 2
11
12
64
25
22
Array after pass 3
11
12
22
64
25
Array after pass 4
11
12
22
25
64
```



```
Sorted list in ascending order:
11
12
22
25
64
```