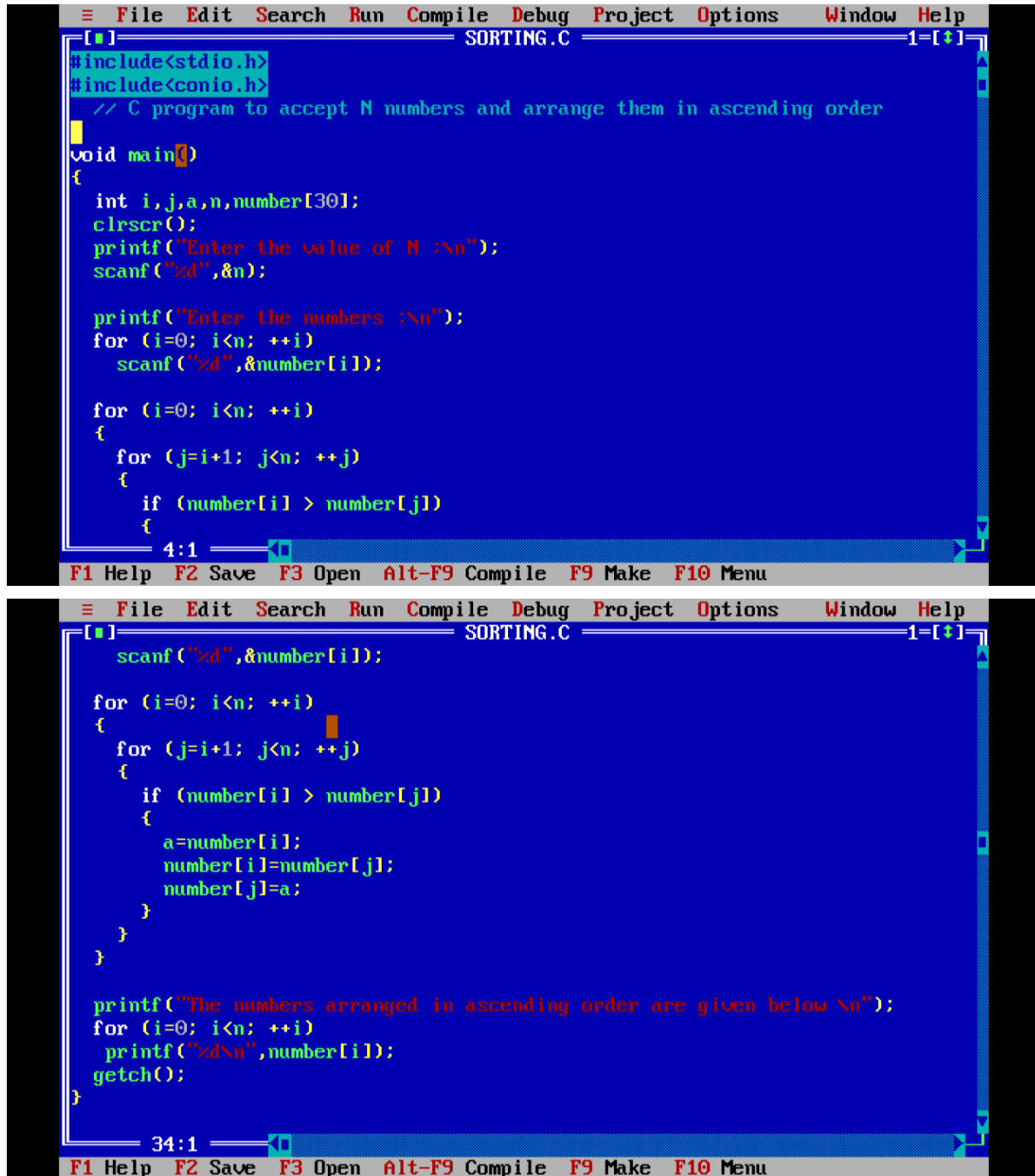


1. Sorting of an integer array

Program



```
#include<stdio.h>
#include<conio.h>
// C program to accept N numbers and arrange them in ascending order

void main()
{
    int i,j,a,n,number[30];
    clrscr();
    printf("Enter the value of N :\n");
    scanf("%d",&n);

    printf("Enter the numbers :\n");
    for (i=0; i<n; ++i)
        scanf("%d",&number[i]);

    for (i=0; i<n; ++i)
    {
        for (j=i+1; j<n; ++j)
        {
            if (number[i] > number[j])
            {
                a=number[i];
                number[i]=number[j];
                number[j]=a;
            }
        }
    }

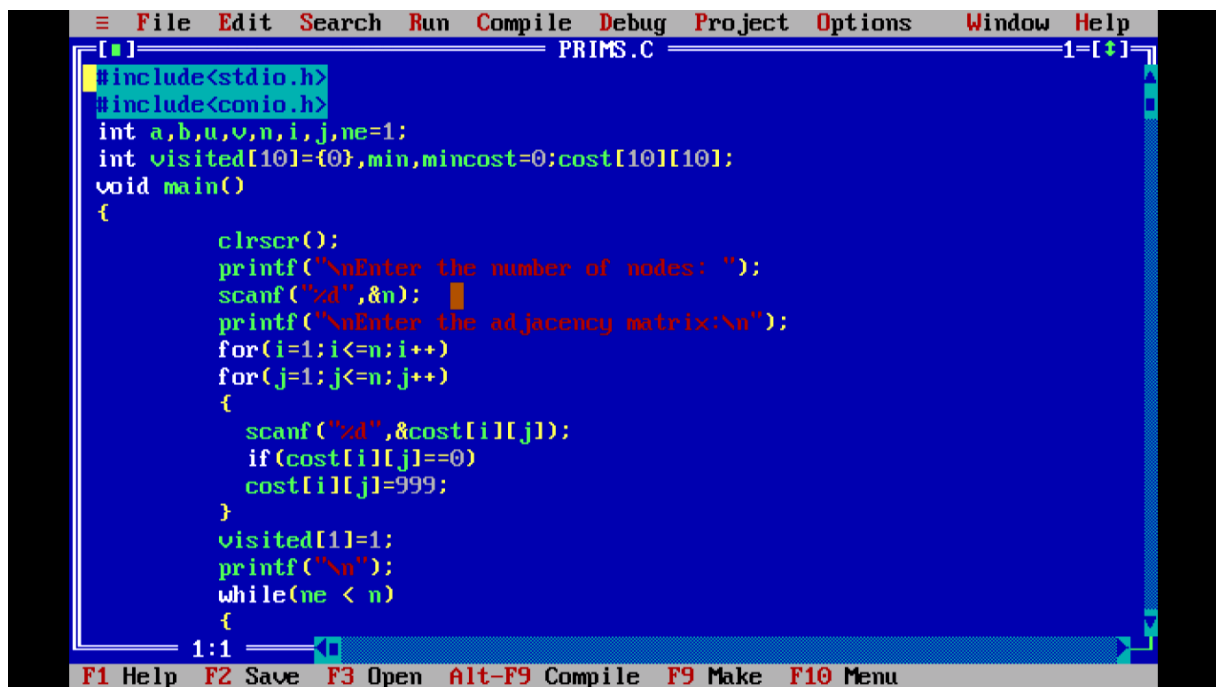
    printf("The numbers arranged in ascending order are given below \n");
    for (i=0; i<n; ++i)
        printf("%d\n",number[i]);
    getch();
}
```

Output

```
Enter the value of N :
4
Enter the numbers :
6
4
10
3
The numbers arranged in ascending order are given below
3
4
6
10
-
```

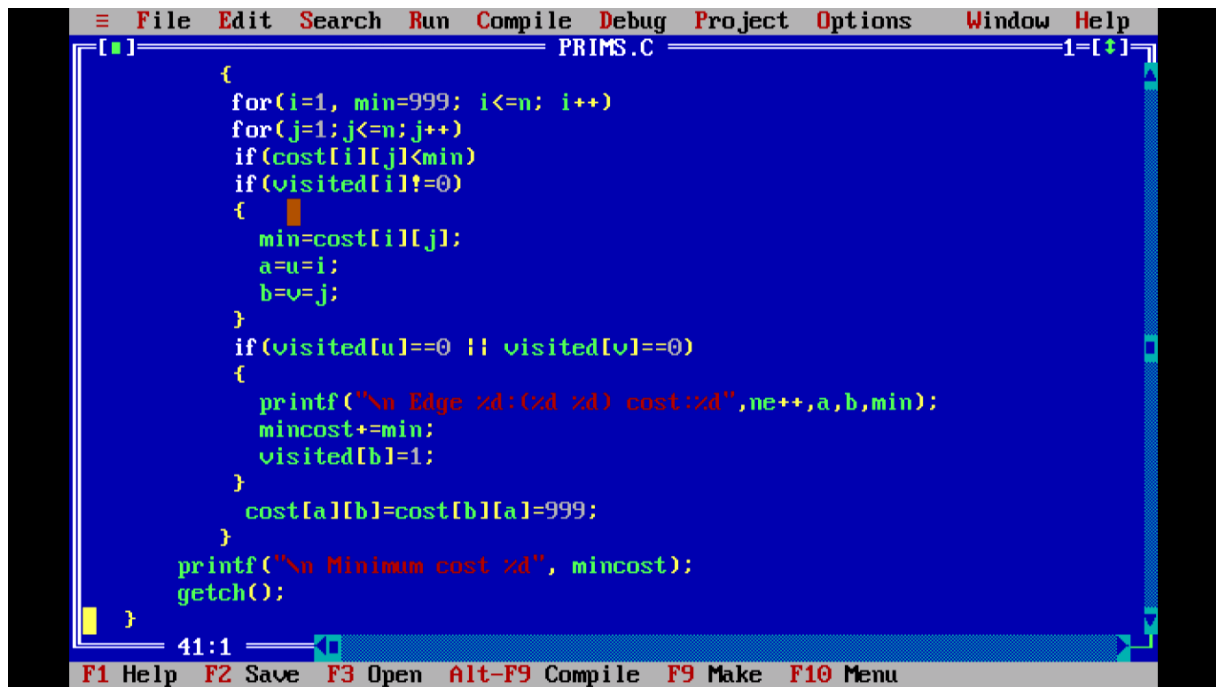
2. Implementation of Prim's algorithm

Program



```
File Edit Search Run Compile Debug Project Options Window Help
PRIMS.C
#include<stdio.h>
#include<conio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10]={0},min,mincost=0;cost[10][10];
void main()
{
    clrscr();
    printf("\nEnter the number of nodes: ");
    scanf("%d",&n);
    printf("\nEnter the adjacency matrix:\n");
    for(i=1;i<=n;i++)
    for(j=1;j<=n;j++)
    {
        scanf("%d",&cost[i][j]);
        if(cost[i][j]==0)
            cost[i][j]=999;
    }
    visited[1]=1;
    printf("\n");
    while(ne < n)
    {
```

The screenshot shows a Turbo C++ IDE window titled "PRIMS.C". The code implements Prim's algorithm. It starts by including `stdio.h` and `conio.h`. It declares variables `a, b, u, v, n, i, j, ne=1` and arrays `visited[10]` and `cost[10][10]`. The `main` function clears the screen, prompts for the number of nodes `n`, and then prompts for the adjacency matrix. It initializes the `cost` matrix with 999 for zero values. It then starts the algorithm by marking node 1 as visited and printing a newline. The `while` loop continues until `ne` is less than `n`.



```
File Edit Search Run Compile Debug Project Options Window Help
PRIMS.C
{
    for(i=1, min=999; i<=n; i++)
    for(j=1; j<=n; j++)
    if(cost[i][j]<min)
    if(visited[i]!=0)
    {
        min=cost[i][j];
        a=u=i;
        b=v=j;
    }
    if(visited[u]==0 || visited[v]==0)
    {
        printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);
        mincost+=min;
        visited[b]=1;
    }
    cost[a][b]=cost[b][a]=999;
}
printf("\n Minimum cost %d", mincost);
getch();
}
```

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Output

```
Enter the number of nodes: 6

Enter the adjacency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 1
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0

Edge 1:(1 3) cost:1
Edge 2:(3 6) cost:1
Edge 3:(6 4) cost:2
Edge 4:(1 2) cost:3
Edge 5:(2 5) cost:3
Minimum cost 10
```