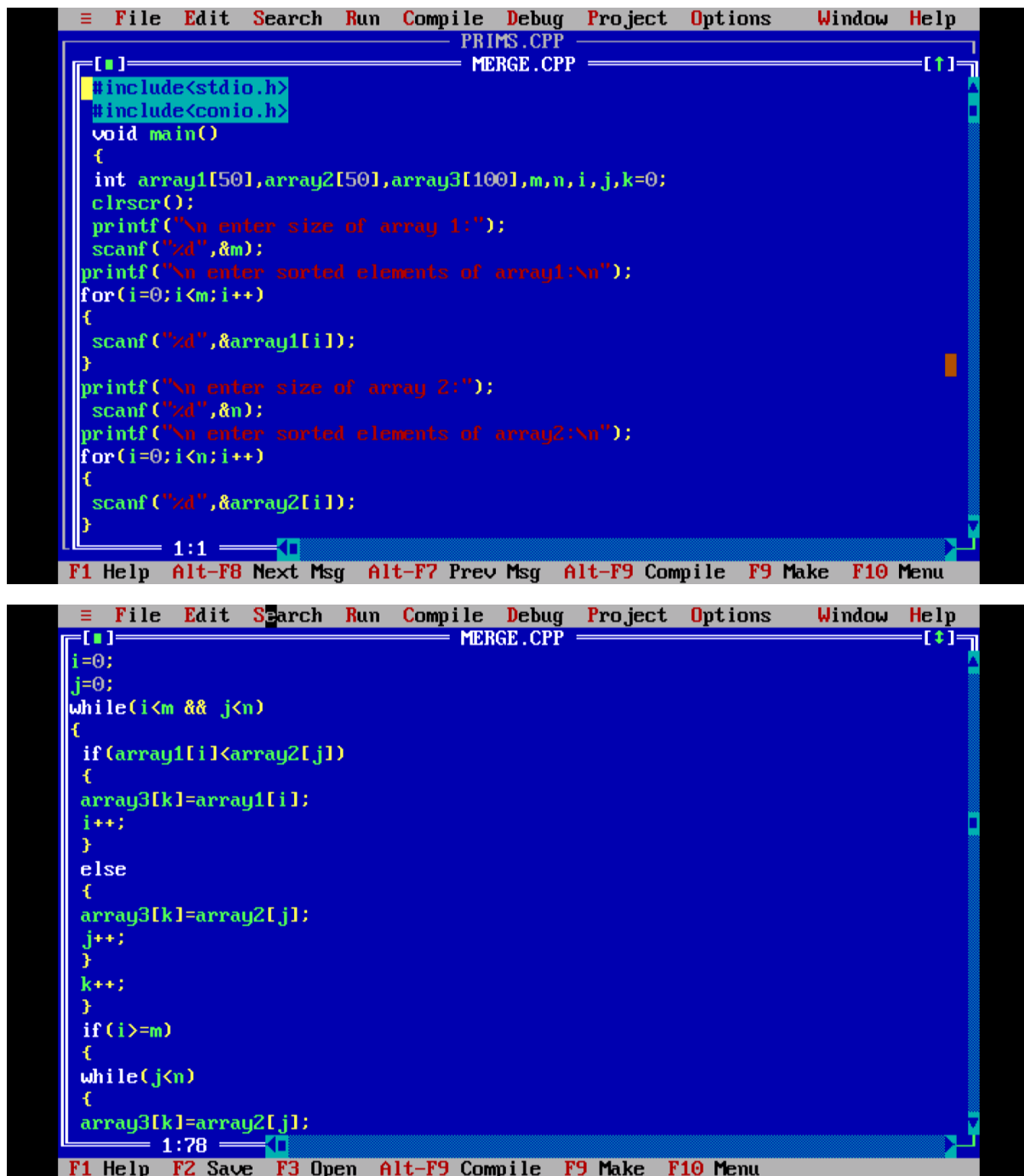


1.Merging of 2 Arrays.



```
PRIMS.CPP
MERGE.CPP

#include<stdio.h>
#include<conio.h>
void main()
{
    int array1[50],array2[50],array3[100],m,n,i,j,k=0;
    clrscr();
    printf("\n enter size of array 1:");
    scanf("%d",&m);
    printf("\n enter sorted elements of array1:\n");
    for(i=0;i<m;i++)
    {
        scanf("%d",&array1[i]);
    }
    printf("\n enter size of array 2:");
    scanf("%d",&n);
    printf("\n enter sorted elements of array2:\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&array2[i]);
    }

    i=0;
    j=0;
    while(i<m && j<n)
    {
        if(array1[i]<array2[j])
        {
            array3[k]=array1[i];
            i++;
        }
        else
        {
            array3[k]=array2[j];
            j++;
        }
        k++;
    }
    if(i>=m)
    {
        while(j<n)
        {
            array3[k]=array2[j];
            j++;
            k++;
        }
    }
    if(j>=n)
    {
        while(i<m)
        {
            array3[k]=array1[i];
            i++;
            k++;
        }
    }
}
```

1:1

F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
MERGE.CPP

i=0;
j=0;
while(i<m && j<n)
{
    if(array1[i]<array2[j])
    {
        array3[k]=array1[i];
        i++;
    }
    else
    {
        array3[k]=array2[j];
        j++;
    }
    k++;
}
if(i>=m)
{
    while(j<n)
    {
        array3[k]=array2[j];
        j++;
        k++;
    }
}
if(j>=n)
{
    while(i<m)
    {
        array3[k]=array1[i];
        i++;
        k++;
    }
}
}
```

1:78

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

```
enter size of array 1:3

enter sorted elements of array1:
2
3
4

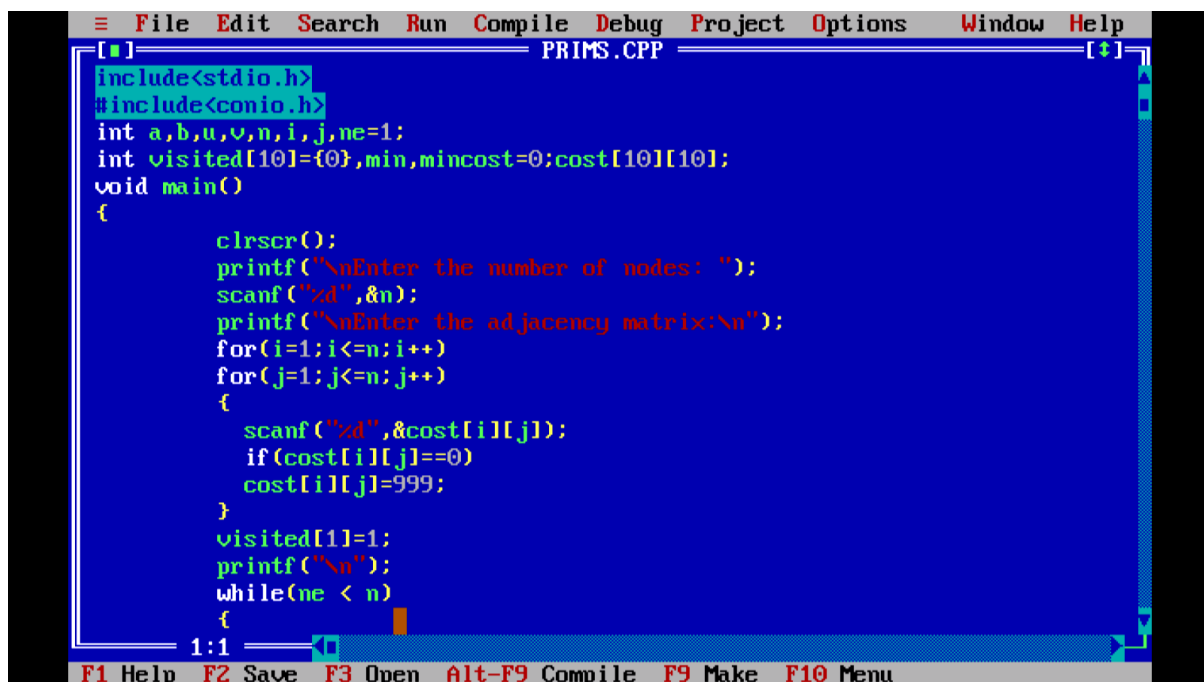
enter size of array 2:2

enter sorted elements of array2:
5
6

after merging :

2
3
4
5
6_
```

2.Implement Prim's Algorithm

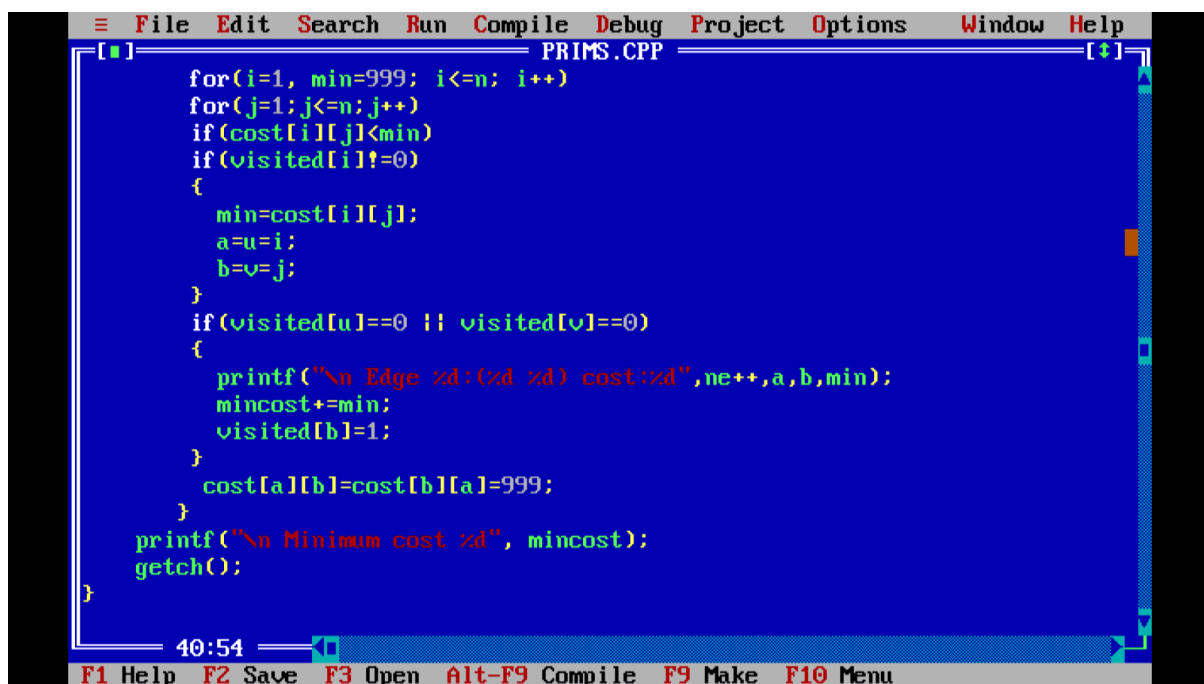


This screenshot shows the first part of the C++ code for Prim's Algorithm. The code includes headers for `stdio.h` and `conio.h`, and declares variables for node counts, visited status, and edge costs. The `main` function starts by clearing the screen and prompting the user for the number of nodes and the adjacency matrix.

```
PRIMS.CPP
#include<stdio.h>
#include<conio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[101]={0},min,mincost=0;cost[101][101];
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)
    {
```

1:1

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



This screenshot shows the second part of the C++ code for Prim's Algorithm. It continues the loop from the previous screenshot, finding the minimum cost edge that can be added to the tree without creating a cycle. Once found, it updates the minimum cost and marks the new node as visited.

```
PRIMS.CPP
    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();
}
```

40:54

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

Enter the number of nodes: 3

Enter the adjacency matrix:

1
2
3
4
7
1
4
5
8

Edge 1:(1 2) cost:2

Edge 2:(2 3) cost:1

Minimum cost 3