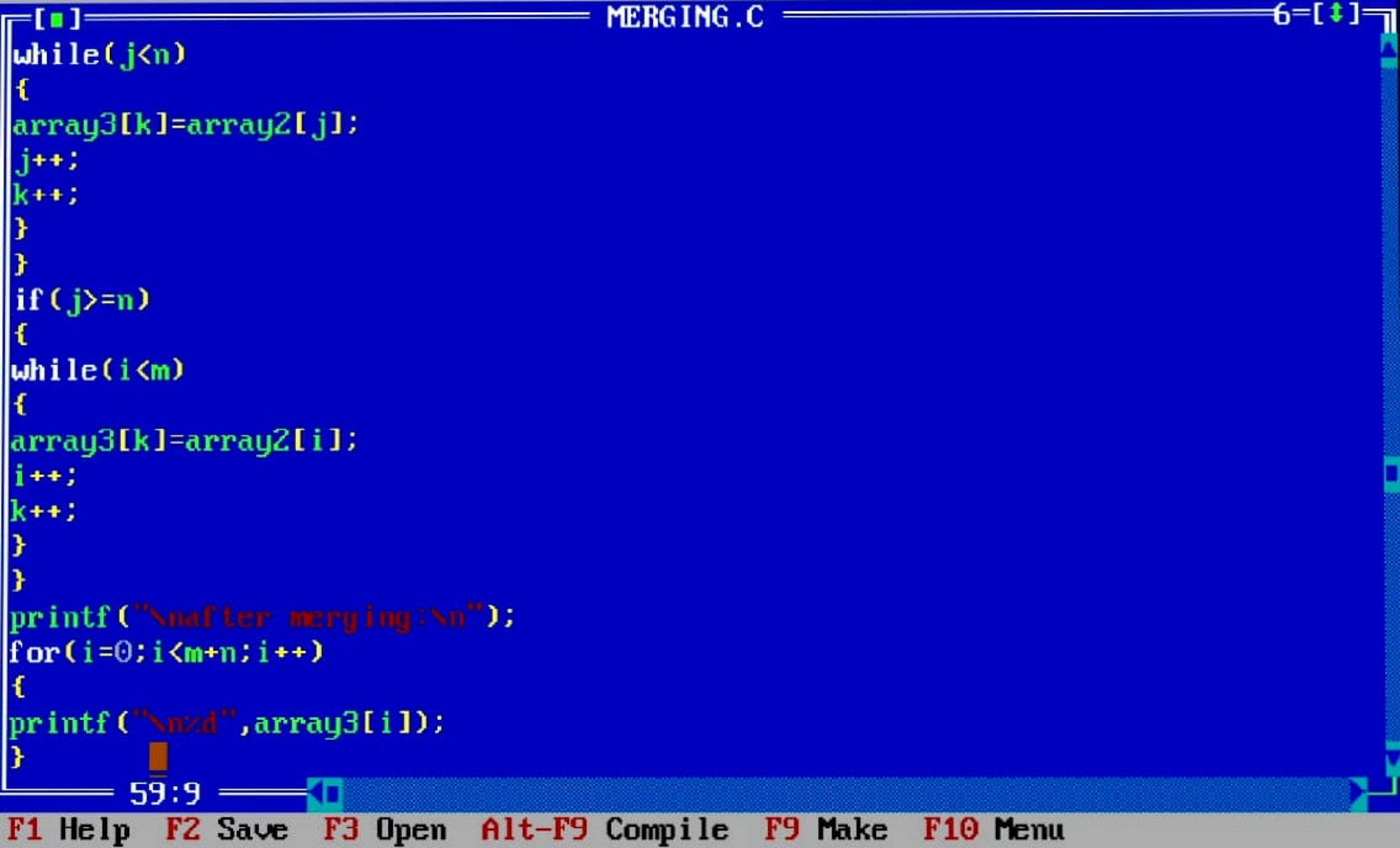
MERGING.C

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

1:1

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

```
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<m)
```


MERGING.C
6
while(j<n)
{
array3[k]=array2[j];
j++;
k++;
}
}
if(j>=n)
{
while(i<m)
{
array3[k]=array2[i];
i++;
k++;
}
}
printf("\nafter merging:\n");
for(i=0;i<m+n;i++)
{
printf("\n%d",array3[i]);
}
59:9
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
array3[k]=array2[j];
j++;
k++;
}
}
if(j>=n)
{
while(i<m)
{
array3[k]=array2[i];
i++;
k++;
}
}
printf("\nafter merging:\n");
for(i=0;i<m+n;i++)
{
printf("\n%d",array3[i]);
}
getch();
}
```

```
enter the size of first array3
enter the sorted elements of first array
1 2 3
enter the size of second array
3
enter the sorted elements of second array
4 5 6

after merging:

1
2
3
4
5
6
```

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 adjecency 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)
     {
```

1:19

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

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();
}
```

41:1

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

```
Enter the number of nodes:6

Enter the adjecency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
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:(1 2) cost:3
 Edge 3:(2 5) cost:3
 Edge 4:(3 6) cost:4
 Edge 5:(6 4) cost:2
 Minimum cost13_
```