Engineering Graphics

Project Based Learning

Topic: Projection of Points using C code

Name: Ayush Saini
Class: CS A2
PRN: 19070122042
Batch: 2019-23

Contents

Introduction:

- In this project, I have implemented a C code to illustrate the topic ‘Projection of Points’ of Engineering Graphics.
- C code as a text file has been typed at the end of this text file.
- C code file is also attached in .c format in the same folder as this text file.
- Screenshots of the projection of points in each quadrant have been attached with the function definition itself.
- These screenshots are also attached in the same folder.

Points to note:

1. Single digit non-zero dimensions are considered for drawing the projection throughout the project.
2. Top views are represented by only small letters eg. p.
3. Their front views are conventionally represented by small letters with dashes eg. p´
4. The line of intersection of HP and VP is denoted as XY.
5. Each single vertical line denotes 1cm.
6. Unidirectional method of dimensioning has been used throughout the project.
7. Dimensions have been marked next to the projections with lines ending with arrowheads and only numbers without units denote the distance of that point from the XY line.

Illustration:

Header files used:

- Stdio.h
- Process.h

Functions used:

- void quad1() [for projection of the point in 1st quadrant]
- void quad2() [for projection of the point in 2nd quadrant]
- void quad3() [for projection of the point in 3rd quadrant]
- void quad4() [for projection of the point in 4th quadrant]

(These are all user defined functions. Each function is used to draw the projection of a point in that particular quadrant.)

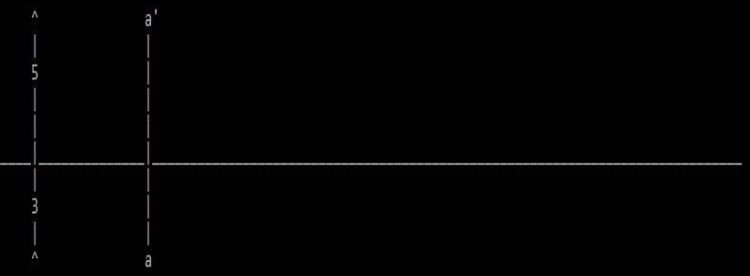
'*Main*' function:

- Main function asks the user the location of the point, i.e. in which quadrant it is located or the point is above/below the HP and in front/behind VP.
- One of the numbers against the following choices is accepted as input in the integer variable '***ch***'.
  1. Point above HP and in front of VP (1st Quadrant) :
  2. Point above HP and behind VP (2nd Quadrant)
  3. Point below HP and behind VP (3rd Quadrant)
  4. Point below HP and in front of VP (4th Quadrant)
- Switch-case is implemented which calls the respective function(as mentioned in the above topic). If any number other than 1,2,3 or 4 is entered, the program gets terminated, displaying to the user, "ERROR! Input Mismatch!"
- If the user enters 1, quad1() is called.
- If the user enters 2, quad2() is called.
- If the user enters 3, quad3() is called.
- If the user enters 4, quad4() is called.

Function '*quad1()*':

- This function projects the point if it is above HP and in front of VP i.e. in the first quadrant.
- The user is prompted to enter the dimensions above HP and in front of VP.
- Dimensions of HP are stored in an integer variable '*h*' and dimensions of VP are stored in an integer variable '*v*'.
- A 2D character array of size (h+v+2)*(150) is created.
- Blank Spaces are created in the complete array manually so as to avoid any garbage value to be stored in that array.
- At the 149th column and row 0 to (h+v+1), null character is introduced at the end to mark the end of each string.(strings are 1D character arrays)
- XY line is drawn at the h^{th} row.
- Projection of both HP and VP are drawn at 70th column.
- At the 55^{th} column, the dimension line is drawn.
- Using the unidirectional method of dimensioning, the distance of projection from XY line is printed at the dimension line.
- Now the 2D character array has the complete projection as well as dimension.
- The 2D character array with the name '*arr*' is displayed on the screen and thus we have the projection on the screen.

```
Choose:
1. Point above HP and in front of VP (1st Quadrant) :
2. Point above HP and behind VP (2nd Quadrant) :
3. Point below HP and behind VP (3rd Quadrant) :
4. Point below HP and in front of VP (4th Quadrant) :
Enter your choice: 1



ALL DIMENSIONS IN CM AND SINGLE DIGIT DIMENSIONS ONLY!



 Projections of a point lying in first quadrant:
 All dimensions in cm(enter only integer values!
 Enter dimensions above HP: 5

 Enter dimensions in front of VP: 3
```



Function '*quad2()*':

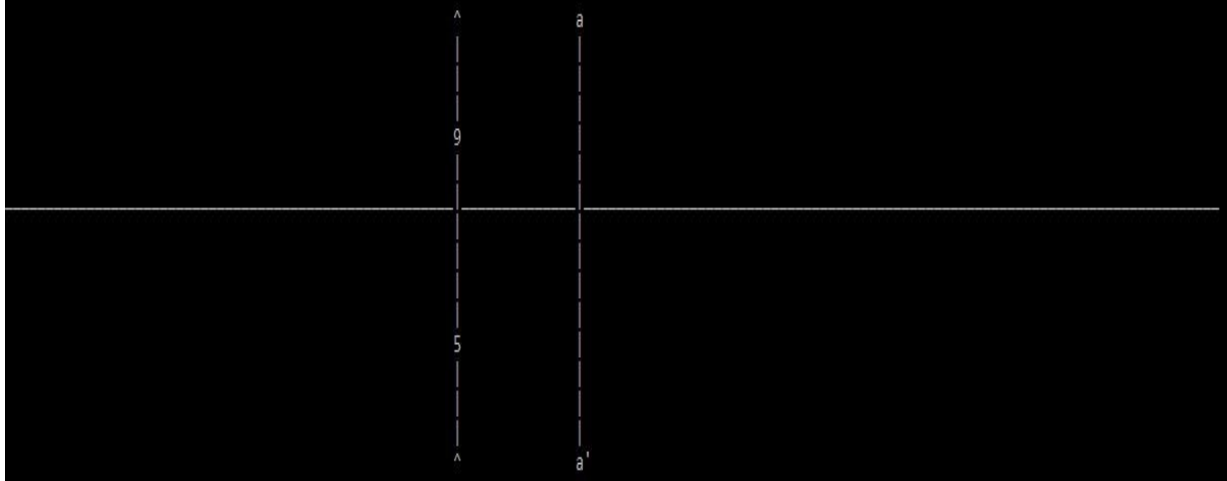
- The function projects the point if it is above HP and behind VP i.e. in the 2nd quadrant.
- The user is prompted to enter the dimensions above HP and behind VP.
- Dimensions of HP are stored in an integer variable '*h*' and dimensions of VP are stored in an integer variable '*v*'.
- '*k'* is a variable created to store the size of the array, whose value is equal to two plus the magnitude of '*h'* or '*v',* whichever is greater.
- A 2D character array of size (k)*(150) is created.
- Blank Spaces are created in the complete array manually so as to avoid any garbage value to be stored in that array.
- At the 149th column and row 0 to (h+v+1), null character is introduced at the end to mark the end of each string.(strings are 1D character arrays)
- XY line is drawn at the k^{th} row.
- Projection of VP', it is drawn from $(k\text{-}v)^{th}$ row at the 50^{th} column.
- Projection of HP, it is drawn from $(k\text{-}h)^{th}$ row at the 100^{th} column.
- At the 40^{th} and 110^{th} column, dimension line VP and HP are drawn respectively.
- Using the unidirectional method of dimensioning, the distance of projection from XY line is printed at the dimension line for both the projections.
- Now the 2D character array has the complete projection as well as dimension.
- The 2D character array with the name '*arr*' is displayed on the screen and thus we have the projection on the screen.

```
Choose:
1. Point above HP and in front of VP (1st Quadrant) :
2. Point above HP and behind VP (2nd Quadrant) :
3. Point below HP and behind VP (3rd Quadrant) :
4. Point below HP and in front of VP (4th Quadrant) :
Enter your choice: 2



ALL DIMENSIONS IN CM AND SINGLE DIGIT DIMENSIONS ONLY!



 Projections of a point lying in second quadrant:
 All dimensions in cm(enter only integer values!
 Enter dimensions above HP: 7

 Enter dimensions behind VP: 5
```

Function '*quad3()*':

- The function projects the point if it is below HP and behind VP i.e. in the 3rd quadrant.
- The user is prompted to enter the dimensions below HP and behind VP.
- Dimensions of HP are stored in an integer variable '*h*' and dimensions of VP are stored in an integer variable '*v*'.
- A 2D character array of size (h+v+2)*(150) is created.
- Blank Spaces are created in the complete array manually so as to avoid any garbage value to be stored in that array.
- At the 149th column and row 0 to (h+v+1), null character is introduced at the end to mark the end of each string.(strings are 1D character arrays)
- If h is greater than or equal to v, then XY line is drawn at the v^{th} row, and if v is greater than h, then the XY line is drawn at the h^{th} row.
- In both cases, the projection line is drawn at 70th column.
- ' *a' '* is assigned to the projection below XY.
- '*a'* is assigned to the projection above XY.
- At the 55^{th} column, the dimension line is drawn.
- Using the unidirectional method of dimensioning, the distance of projection from XY line is printed at the dimension line for both the projections.
- Now the 2D character array has the complete projection as well as dimension.
- The 2D character array with the name '*arr*' is displayed on the screen and thus we have the projection on the screen.

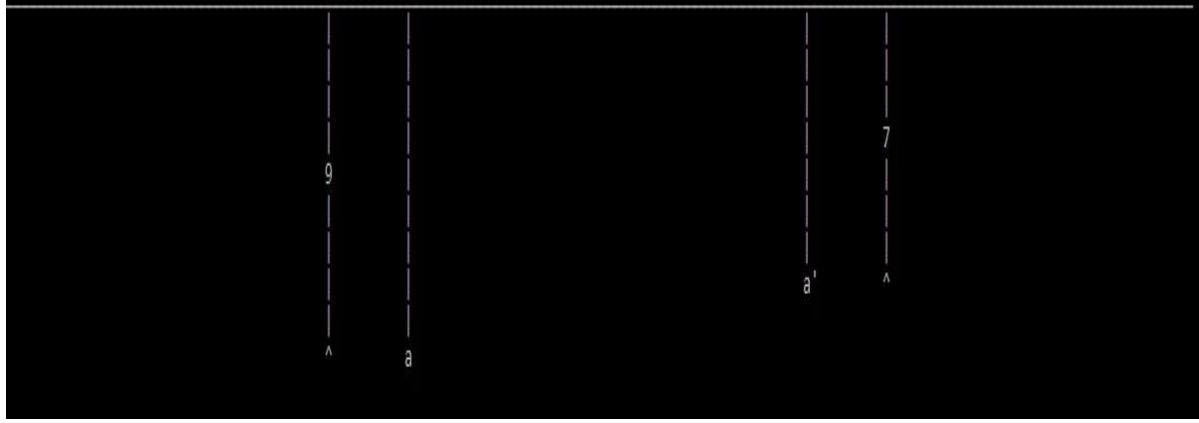
```
Choose:
1. Point above HP and in front of VP (1st Quadrant) :
2. Point above HP and behind VP (2nd Quadrant) :
3. Point below HP and behind VP (3rd Quadrant) :
4. Point below HP and in front of VP (4th Quadrant) :
Enter your choice: 3



ALL DIMENSIONS IN CM AND SINGLE DIGIT DIMENSIONS ONLY!



 Projections of a point lying in third quadrant:
 All dimensions in cm(enter only integer values!
 Enter dimensions below HP: 5

 Enter dimensions behind VP: 9
```



Function '*quad4()*':

- The function projects the point if it is below HP and in front of VP i.e. in the 4th quadrant.
- The user is prompted to enter the dimensions below HP and in front of VP.
- Dimensions of HP are stored in an integer variable '*h*' and dimensions of VP are stored in an integer variable '*v*'.
- '*k'* is a variable created to store the size of the array, whose value is equal to two plus the magnitude of '*h'* or '*v',* whichever is greater.
- A 2D character array of size (k)*(150) is created.
- Blank Spaces are created in the complete array manually so as to avoid any garbage value to be stored in that array.
- At the 149th column and row 0 to (h+v+1), null character is introduced at the end to mark the end of each string.(strings are 1D character arrays)
- XY line is drawn at the 0th row.
- HP and VP are drawn at the 50th and 100th column.
- At the 40th and 110th column, the dimension line is drawn.
- Top view and front view are properly assigned.
- Using the unidirectional method of dimensioning, the distance of projection from XY line is printed at the dimension line for both the projections.
- Now the 2D character array has the complete projection as well as dimension.
- The 2D character array with the name '*arr*' is displayed on the screen and thus we have the projection on the screen.

```
Choose:
1. Point above HP and in front of VP (1st Quadrant) :
2. Point above HP and behind VP (2nd Quadrant) :
3. Point below HP and behind VP (3rd Quadrant) :
4. Point below HP and in front of VP (4th Quadrant) :
Enter your choice: 4



ALL DIMENSIONS IN CM AND SINGLE DIGIT DIMENSIONS ONLY!



 Projections of a point lying in fourth quadrant:
 All dimensions in cm(enter only integer values!
 Enter dimensions below HP: 7

 Enter dimensions in front of VP: 9
```



Text file of C code:

```
#include<stdio.h>
#include<process.h>

void quad1();
void quad2();
void quad3();
void quad4();

int main()
{
	int ch;
	printf("\nChoose: ");
	printf("\n1. Point above HP and in front of VP (1st Quadrant) : ");
	printf("\n2. Point above HP and behind VP (2nd Quadrant) :");
	printf("\n3. Point below HP and behind VP (3rd Quadrant) :");
	printf("\n4. Point below HP and in front of VP (4th Quadrant) :");
	printf("\nEnter your choice: ");
	scanf("%d",&ch);
	printf("\n\n\n\nALL DIMENSIONS IN CM AND SINGLE DIGIT DIMENSIONS
ONLY!\n\n\n\n");
	switch(ch)
	 {
		case 1:
			quad1();
		   break;
		case 2:
			quad2();
			break;
		case 3:
			quad3();
			break;
		case 4:
			quad4();
			break;
		default:
```

```
                        printf("\n Error! Input Mismatch!");
                        exit(1);
        }
}

void quad1()
{
        int h,v;
 printf("\n Projections of a point lying in first quadrant: ");
 printf("\n All dimensions in cm(enter only integer values!");
 printf("\n Enter dimensions above HP: ");
 scanf("%d",&h);
 printf("\n Enter dimensions in front of VP: ");
 scanf("%d",&v);
 printf("\n\n\n");
 char arr[h+v+2][150];              //creation of array to store projections
 int i,j;
 for(i=0;i<(h+v+2);i++)              //allotting blank spaces to remove garbage input
  for(j=0;j<150;j++)
    arr[i][j]=' ';
 for(i=0;i<(h+v+1);i++)             //putting null character at the end of each line of array of
strings
   arr[i][149]='\0';
 for(i=0;i<149;i++)                //drawing XY line
    arr[h][i]='_';
 for(i=1;i<(h+v+1);i++)              //drawing projection
    arr[i][70]='|';


 //dimensioning above xy

 int temp;
 for(i=1;i<=h;i++)
    arr[i][55]='|';
 temp=h/2;
 if(temp%2!=0)
    temp++;
```

```
 arr[temp][55]=h+48;        //assigning ascii value of integer dimensions
 arr[0][55]=94;             //assigning arrowhead ascii value

//dimensioning below xy

 temp=h;
 for(i=h+1;i<=(h+v);i++)
   arr[i][55]='|';
 temp+=v/2;
 if((v/2)%2!=0)
   temp++;
 arr[temp][55]=v+48;       //assigning ascii value of integer dimensions
 arr[h+v+1][55]=94;        //assigning arrowhead ascii value

 //assigning top view and front view
 arr[0][70]='a';
 arr[0][71]=39;
 arr[h+v+1][70]='a';
 for(i=0;i<(h+v+2);i++)
  {
   for(j=0;j<150;j++)                  //printing the projection
    {
     printf("%c",arr[i][j]);
    }
   printf("\n");
  }
}

void quad2()
{
        int h,v,k,i,j;
        printf("\n Projections of a point lying in second quadrant: ");
   printf("\n All dimensions in cm(enter only integer values!");
   printf("\n Enter dimensions above HP: ");
   scanf("%d",&h);
   printf("\n Enter dimensions behind VP: ");
   scanf("%d",&v);
```

```
   printf("\n\n\n");
   if(h>=v)
     k=h+2;                    //alloting size of array according to maximum dimensions
   else
     k=v+2;
   char arr[k][150];           //creation of array to store projection
   for(i=0;i<k;i++)
     for(j=0;j<149;j++)        //allotting blank spaces to remove garbage value
        arr[i][j]=' ';
   for(i=0;i<k;i++)            //putting null character at the end of each line of array of
strings
       arr[i][149]='\0';
   for(i=0;i<149;i++)              //drawing XY line
      arr[k-1][i]='_';
   for(i=(k-v);i<k;i++)
    {
        arr[i][50]='|';
    }
   for(i=(k-h);i<k;i++)
     {
         arr[i][100]='|';
     }

         int temp;
   //dimensioning above xy line
   for(i=k-v;i<k;i++)
           arr[i][40]='|';
        arr[k-v-1][40]=94;

        for(i=k-h;i<k;i++)
           arr[i][110]='|';
        arr[k-h-1][110]=94;


   //allotting top view and front view
   arr[k-v-1][50]='a';
   if(h>=v)
```

```
    {
     arr[k-v-2][100]='a';
     arr[k-v-2][101]=39;
     temp=k-(v/2);
     if(v%2!=0)
       temp--;
     arr[temp][40]=v+48;
     temp=k-(h/2);
     if(h%2!=0)
       temp--;
     arr[temp][110]=h+48;
    }
   else
    {
        arr[k-v][100]='a';
        arr[k-v][101]=39;
        temp=k-(h/2);
        if(v%2!=0)
          temp--;
        arr[temp][40]=v+48;
        temp=k-(v/2);
        if(h%2!=0)
                temp--;
        arr[temp][110]=h+48;
    }
   for(i=0;i<k;i++)
   {
     for(j=0;j<150;j++)                    //printing the projection
      {
        printf("%c",arr[i][j]);
      }
     printf("\n");
   }
}

void quad3()
{
```

```
        int h,v,i,j;
        printf("\n Projections of a point lying in third quadrant: ");
   printf("\n All dimensions in cm(enter only integer values!");
   printf("\n Enter dimensions below HP: ");
   scanf("%d",&h);
   printf("\n Enter dimensions behind VP: ");
   scanf("%d",&v);
   printf("\n\n\n");
   char arr[h+v+2][150];
   for(i=0;i<(h+v+2);i++)                //allotting blank spaces to remove garbage input
   for(j=0;j<150;j++)
     arr[i][j]=' ';
   for(i=0;i<(h+v+2);i++)            //putting null character at the end of each line of array of
strings
     arr[i][149]='\0';
   if(h>=v)
   {
   for(i=0;i<149;i++)                //drawing XY line
     arr[v][i]='_';
   for(i=1;i<(h+v+1);i++)                //drawing projection
     arr[i][70]='|';

   //assigning " a' " to the projection

    arr[i][70]='a';
   arr[i][71]=39;
   }

   else
   {
    for(i=0;i<149;i++)                //drawing XY line
       arr[h+1][i]='_';
    for(i=0;i<(h+v+1);i++)                //drawing projection
       arr[i][70]='|';

   //assigning " a' " to the projection
```

```
    arr[i][70]='a';
    arr[i][71]=39;
   }
   arr[0][70]='a';                          //assigning 'a' to the projection

   //dimensioning above xy line
   int temp;
   for(i=1;i<=h;i++)
     arr[i][55]='|';
   temp=v/2;
   if(temp%2!=0)
    temp++;
   arr[temp][55]=v+48;         //assigning ascii value of integer dimensions
   arr[0][55]=94;              //assigning arrowhead ascii value

   //dimensioning below xy

   temp=v;
   for(i=h+1;i<=(h+v);i++)
   arr[i][55]='|';
   temp+=h/2;
   if((v/2)%2!=0)
     temp++;
   arr[temp][55]=h+48;       //assigning ascii value of integer dimensions
   arr[h+v+1][55]=94;        //assigning arrowhead ascii value
   for(i=0;i<(h+v+2);i++)
    {
      for(j=0;j<150;j++)                    //printing the projection
        {
          printf("%c",arr[i][j]);
        }
      printf("\n");
    }
}

void quad4()
{
```

```
        int h,v,k,i,j,temp;
        printf("\n Projections of a point lying in fourth quadrant: ");
   printf("\n All dimensions in cm(enter only integer values!");
   printf("\n Enter dimensions below HP: ");
   scanf("%d",&h);
   printf("\n Enter dimensions in front of VP: ");
   scanf("%d",&v);
   printf("\n\n\n");
   if(h>=v)
     k=h+2;                    //alloting size of array according to maximum dimensions
   else
     k=v+2;
   char arr[k][150];             //creation of array to store projection
        for(i=0;i<k;i++)
     for(j=0;j<149;j++)          //allotting blank spaces to remove garbage value
        arr[i][j]=' ';
   for(i=0;i<k;i++)              //putting null character at the end of each line of array of
strings
      arr[i][149]='\0';
   for(i=0;i<149;i++)                //drawing XY line
      arr[0][i]='_';
        for(i=1;i<(k-1);i++)
    {
        arr[i][50]='|';
    }
   if(h==v)
     {
        for(i=1;i<(k-1);i++)
       {
          arr[i][100]='|';
       }

      //allotting front and top view if h and v are equal
      arr[k-1][50]='a';
      arr[k-1][100]='a';
      arr[k-1][101]=39;
```

```
   //dimensioning front and top views

   for(i=1;i<k-1;i++)
    {
      arr[i][40]='|';
      arr[i][110]='|';
    }
   arr[i][40]=94;
   arr[i][110]=94;
   temp=h/2;
   if(h%2!=0)
     temp++;
   arr[temp][40]=h+48;
   arr[temp][110]=h+48;

  }
else if(h>v)
  {
    for(i=1;i<=v;i++)
     {
         arr[i][100]='|';
     }
  }
else if(v>h)
 {
      for(i=1;i<=h;i++)
     {
         arr[i][100]='|';
     }
 }

 //allotting top view and front view if v and h are different
 if(h!=v)
  {
       if(h>v)
        {
               arr[v+1][100]='a';
```

```
            arr[k-1][50]='a';
arr[k-1][51]=39;
for(i=1;i<k-1;i++)
   arr[i][40]='|';
arr[i][40]=94;
for(i=1;i<=v;i++)
   arr[i][110]='|';
arr[i][110]=94;

temp=h/2;
if(h%2!=0)
   temp++;
arr[temp][40]=h+48;

                  temp=v/2;
if(v%2!=0)
  temp++;
arr[temp][110]=v+48;
 }
 else if(v>h)
  {
        arr[h+1][100]='a';
arr[h+1][101]=39;
arr[k-1][50]='a';
for(i=1;i<k-1;i++)
   arr[i][40]='|';
arr[i][40]=94;
for(i=1;i<=h;i++)
   arr[i][110]='|';
arr[i][110]=94;

temp=v/2;
if(v%2!=0)
   temp++;
arr[temp][40]=v+48;

                  temp=h/2;
```

```
            if(h%2!=0)
              temp++;
            arr[temp][110]=h+48;
              }

     }
    for(i=0;i<k;i++)
    {
      for(j=0;j<150;j++)                    //printing the projection
       {
         printf("%c",arr[i][j]);
       }
     printf("\n");
    }
}
```