GOVERNMENT COLLEGE OF TECHNOLOGY
(*An autonomous institution*)
Thadagam Road, Coimbatore - 641013

IMPLEMENTATION OF STEGANOGRAPHY IN

C

- ✓ *An image creation application*
- ✓ *Run Length Encoding*
- ✓ *Huffman Encoding*
- ✓ *Data hiding*
- ✓ *Data Retrieving*
- ✓ *Image Retrieving*

Sri Rama Prasanna P
9914143
ECE @ GCT

Balaji S
9914103
ECE @ GCT

SYNOPSIS

Data Security has been an indispensable object of concern ever since computer communication commenced. The vulnerability of the encryption techniques posed a great threat to clandestine transmission of data. This led to the invention of Steganography, an ingenious phenomenon which hides data within an image. The robustness of Steganography is due to the fact that there is no direct way to determine whether data is present in an image or not.

We have developed a C application for data hiding and retrieving. Our project basically includes an image creation application, Run Length Encoding (RLE), Huffman coding in addition to the data hiding and retrieving programs. The Run Length Encoding and Huffman coding aid in compressing the image size. For ease of handing images, we have designed a new image format (.srp).

The data hiding is accomplished by first encrypting the data and splitting it into several fragments. These chunks of encrypted data are inserted into the image file at random locations following a particular series. The image containing the encrypted chunks of data is transmitted across the network. While retrieving, the image file is

traversed and the locations of encrypted data are identified. The encrypted data chunks are grouped together and a decryption algorithm is applied to get back the original data.

IMAGE CREATION APPLICATION:

The Image Creation (Paint) program aids in creating an image and saving it in a novel format called .srp. This program incorporates the following features.

- Variable Brush Size.
- Colour Management.
- Variable Style
- Erasing
- Clearing
- Saving the constructed image.

As soon as the program is invoked, it asks for the colour, brush size and style from the user. These three parameters can also be changed during the course of the painting process. The program also provides an option to save the picture drawn and prompts for the path details if the user decides to save the image.

The Erase and Clear option adds up to the user friendliness of the application. After the construction of the picture, the user can save it by pressing the return key.

The entire screen is scanned column by column and each pixel value is thrown into a File.

IMAGE RETRIEVING APPLICATION:

This is a simple program which retrieves the image (stored by the image creation application) from the file and displays it. The program prompts for the path details when

invoked. The required file is opened and its details are used to reconstruct the original image. The image remains on the screen until a key is pressed.

RUN LENGTH ENCODING (RLE):

The technique of RLE exploits the high interpixel redundancy in relatively simple images. A run of consecutive pixels whose colour values are identical are replaced with two values namely the length of the run and the colour attribute.

If the length of the run exceeds the maximum value of an integer in C, the program splits up the number into two tangible portions and saves it down. A very high degree of compression ratio can be achieved for uniformly illuminated images. RLE is a lossless compression technique. This means that the original image and the decompressed image are identical.

HUFFMAN'S CODING:

Huffman Coding works on the phenomenon that in any file the probability of occurrence of all characters are not same. This means that some characters occur often and some other characters are sparsely distributed. In Huffman Coding scheme, a short code is associated with an item value which has a high probability of occurrence and a long code is assigned to an item value with a low probability of occurrence. This results in an appreciable reduction in file size.

The codes for characters are assigned after determining the probability of occurrence of each character in the file. As this statistical coding algorithm is lossless in nature, the entire details of the image are preserved.

DATA HIDING:

This program hides a user defined data in the encoded image file. The program prompts for the data to be hidden. The entered data is broken up into numerous fragments. These chunks of data are then encrypted to an intangible format.

The encrypted chunks of data are then inserted into the encoded image file at locations following a particular series. This image containing the encrypted data is transmitted to the destination.

DATA RETRIEVING:

This program gets back the information from the transmitted image. The transmitted image file is traversed through to identify the valid encrypted data blocks (chunks). The encrypted data chunks are regrouped and a decryption algorithm is applied to obtain the original data.

C CODES:

IMAGE CREATION APPLICATION:

```
#include<dos.h>
#include<graphics.h>
#include<string.h>
#include<stdio.h>
union REGS i,o;

void main()
{
int erase = 1,gd=DETECT,gm,c=0,e,f,bs,col,ef=1,x=0,y=0,col1;
char str[40],t;
FILE *fp;
clrscr();
initgraph(&gd,&gm,"d:\\tc\\tcc\\bgi");

setcolor(10);
settextstyle(4,HORIZ_DIR,7);
outtextxy(200,10,"PAINT");
printf("\n\n\n\n\n\n\n\n\n\nEnter the brush size : ");
scanf("%d",&bs);
printf("\n\nEnter the color :");
scanf("%d",&col);
```

```
printf("\n\nEnter the Style :");
scanf("%d",&ef);
fflush(stdin);
printf("\n\nDo you want to save the picture?");
t = getchar();

if((t=='y')||(t=='Y'))
{
fflush(stdin);
printf("\n\nEnter the output filename :");
fflush(stdin);
scanf("%s",str);
}
 ef = ef*2;
 closegraph();
  initgraph(&gd,&gm,"d:\\tc\\tcc\\bgi");

 i.x.ax=0;
 int86(0x33,&i,&o);
 setcolor(7);
 rectangle(0,0,getmaxx(),getmaxy());
rectangle(2,2,getmaxx()-2,getmaxy()-2);

 setfillstyle(1,col);
 fillellipse(0,0,70,70);
 fillellipse(640,475,60,60);
 fillellipse(0,475,60,60);
 fillellipse(640,0,68,68);
 setcolor(0);
 settextstyle(1,HORIZ_DIR,1);
 outtextxy(580,450,"CLEAR");
 outtextxy(1,450,"ERASE");
 outtextxy(580,5,"BALAJI");
 outtextxy(5,5,"SRP");

while(1)
 {
  if(erase == 23)
  {
   setfillstyle(1,4);
 setcolor(7);
 fillellipse(0,0,70,70);
 fillellipse(640,475,60,60);
 fillellipse(0,475,60,60);
 fillellipse(640,0,68,68);
 setcolor(0);
 settextstyle(1,HORIZ_DIR,1);
 outtextxy(580,450,"CLEAR");
 outtextxy(1,450,"ERASE");
 outtextxy(580,5,"BALAJI");
 outtextxy(5,5,"SRP");
 }
 settextstyle(4,HORIZ_DIR,7);
```

```
 if(kbhit())
 {
  setcolor(7);
 rectangle(0,0,getmaxx(),getmaxy());
 setcolor(0);
 rectangle(1,1,getmaxx()-1,getmaxy()-1);
 setcolor(7);

 rectangle(2,2,getmaxx()-2,getmaxy()-2);
   setfillstyle(1,col);
 fillellipse(0,0,70,70);
 fillellipse(640,475,60,60);
 fillellipse(0,475,60,60);
 fillellipse(640,0,68,68);
 setcolor(0);
 settextstyle(1,HORIZ_DIR,1);
 outtextxy(585,450,"CLEAR");
 outtextxy(1,450,"ERASE");
 outtextxy(580,5,"BALAJI");
 outtextxy(5,5,"SRP");

  if((t=='y')||(t=='Y'))
  {
     fp = fopen(strcat(str,".srp"),"w");
     for(x=0;x<640;x++)
     {
      for(y=0;y<480;y++)
       {
            c = '0' + getpixel(x,y);
            fputc(c,fp);
       }
     }
     fclose(fp);
 }
  break;
 }

 setcolor(10);
settextstyle(4,HORIZ_DIR,7);
outtextxy(200,10,"PAINT");
 c++;
 if(c==16)
 {
 c=0;
 }
 i.x.ax = 1;
 int86(0x33,&i,&o);
 i.x.ax=3;
  int86(0x33,&i,&o);
 if((o.x.bx & 2)==2)
 {
    setcolor(10);
settextstyle(4,HORIZ_DIR,7);
```

```
outtextxy(200,10,"PAINT");

setcolor(0);
setfillstyle(1,0);
fillellipse(0,0,100,100);
gotoxy(65,1);
gotoxy(1,1);
printf("Brush:");

scanf("%d",&bs);

printf("Color:");
scanf("%d",&col);
printf("Style:");
scanf("%d",&ef);
ef = ef*2;
 setcolor(7);
 rectangle(0,0,getmaxx(),getmaxy());
 rectangle(2,2,getmaxx()-2,getmaxy()-2);

 if(erase==23)
 {
  erase = 1;
 }
 setfillstyle(1,col);
 fillellipse(0,0,70,70);
 fillellipse(640,475,60,60);
 fillellipse(0,475,60,60);
 fillellipse(640,0,68,68);
 setcolor(0);
 settextstyle(1,HORIZ_DIR,1);
 outtextxy(585,450,"CLEAR");
 outtextxy(1,450,"ERASE");
 outtextxy(580,5,"BALAJI");
 outtextxy(5,5,"SRP");
  }
 if(col==23)
 {
  setfillstyle(1,c);
 setcolor(c);
 }
 else
 {
 setfillstyle(1,col);
 setcolor(col);
 }
 if((o.x.bx & 1)==1)
 {
   if((o.x.cx >= 600)&&(o.x.dx >= 440))
   {
      closegraph();
      initgraph(&gd,&gm,"d:\\tc\\tcc\\bgi");
      setcolor(7);
```

```
 rectangle(0,0,getmaxx(),getmaxy());
 rectangle(2,2,getmaxx()-2,getmaxy()-2);

 setfillstyle(1,col);
 fillellipse(0,0,70,70);
 fillellipse(640,475,60,60);
 fillellipse(0,475,60,60);
 fillellipse(640,0,68,68);
 setcolor(0);
 settextstyle(1,HORIZ_DIR,1);
 outtextxy(585,450,"CLEAR");
 outtextxy(1,450,"ERASE");
 outtextxy(580,5,"BALAJI");
 outtextxy(5,5,"SRP");
   }
  if((o.x.cx >= 600)&&(o.x.dx <= 30))
   {
     setfillstyle(1,0);
     setcolor(0);
   }
   if((o.x.cx <= 30)&&(o.x.dx >= 430))
   {
     col1 = col;
     col=0;
     bs=20;
     erase = 23;
     gotoxy(65,1);
   }
 for(e=o.x.cx;e<o.x.cx+ef-1;e++)
 {
 for(f=o.x.dx;f<o.x.dx+ef-1;f++)
 {
    fillellipse(e,f,bs,bs);
}
 }
 }
 }
   cleardevice();
}
```

IMAGE RETRIEVAL PROGRAM:

```
include<stdio.h>
#include<string.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm,x=0,y=0,p;
char str[40];
FILE *fp;
printf("Enter the Filename :");
fflush(stdin);
```

```
gets(str);
initgraph(&gd,&gm,"d:\\tc\\tcc\\bgi");
fp = fopen(strcat(str,".srp"),"r");
for(x=0;x<640;x++)
{
for(y=0;y<480;y++)
{
  p = fgetc(fp);
  putpixel(x,y,p);
}
}
fclose(fp);
getch();
cleardevice();
}
```

RUN LENGTH ENCODING (RLE):

```
#include<stdio.h>
void main()
{
int col,d=0,q,s,p=0,r=0;
 long int n=1,m=0,a=0;
char c;
FILE *fp,*fq,*fr;
clrscr();
if((fp=fopen("c:\\fin.srp","r"))==NULL)
{
  printf("Error in fin.srp");
  getch();
  exit(0);
}
if((fr=fopen("c:\\fin.srp","r"))==NULL)
{
  printf("Error in fin.srp");
  getch();
  exit(0);
}
if((fq=fopen("c:\\sai.srp","w"))==NULL)
{
  printf("Error in sai.srp");
  getch();
  exit(0);
}
r = fgetc(fr);
while(1)
{
n=1;
while(1)
{
    p = fgetc(fp);
    if(p == EOF)
```

```
    {
     d = 23;
     break;
    }
  r = fgetc(fr);
  if(r == EOF)
  {
    d = 23;
    break;
  }
  if(p==r)
  {
    n++;
  }
  else
  {
     break;
  }
}
c = p ;
fputc(c,fq);
m = n/200;
c ='0'+m;
fputc(c,fq);
a = n%200;
c='0'+a;
fputc(c,fq);
if(d==23)
{
 fcloseall();
 exit(0);
}
}
}
```

RUN LENGTH DECODING :

```
#include<stdio.h>
#include<conio.h>
void main()
{
 long int n=0,i=0;
 int r=0,a=0,m=0;
FILE *fq,*fr;
clrscr();

if((fr=fopen("c:\\sai.srp","r"))==NULL)
{
  printf("Error in sai.srp");
  getch();
  exit(0);
}
if((fq=fopen("c:\\FF.SRP","w"))==NULL)
```

```
{
  printf("Error in ff.srp");
  getch();
  exit(0);
}
while(1)
{
n=0;
if((r = fgetc(fr))==EOF)
{
  exit(0);
}
m = fgetc(fr);
a = fgetc(fr);
m=m-'0';
a=a-'0';
n = m*200 + a;
i=0;
while(i<n)
{
  fputc(r,fq);
   i++;
}
}
}
```

HUFFMAN'S CODING :

```
#include<stdio.h>
#include<conio.h>
#include<dir.h>
#include<math.h>
#include<string.h>
#include<alloc.h>
struct collec
{
  char col[128];
  long int count;
}be[256];
int formbyte(int *extr,char **bs,int n);
void main()
{
FILE *in,*out;
int n=0,cot=0,i,j,p,g=0,tb=0,ab=0,extr[256];
static char puf[20];
static long int cc[256];
char fname[30],chr,chs,fn[MAXPATH],asb,ch[256],t,sr;
static char *bs[256],*bsm[256];
clrscr();
printf("Enter the source file name:");
scanf("%s",fn);
in=fopen(fn,"rb");
```

```
if(!in)
 {
 printf("Error opening file");
 exit(0);
 }
 printf("Enter the dest file name :");
fflush(stdin);
gets(fname);
if((out = fopen(fname,"w")) == NULL)
{
  printf("Error in file creation for Zip !! ");
}
while(!feof(in))
  cc[getc(in)]++;
for(i=0;i<256;i++)
{
          if(cc[i])
          {
    if ((bs[n] = (char *) malloc(32)) == NULL)
           {
                    printf("Not enough memory to allocate buffer\n");
                    exit(1);
           }
    strcpy(bs[n],"");
          ch[n++]=i;
          }
}
          cot=n;
  for(i=0;i<(n-1);i++)
          {
           for(j=i+1;j<n;j++)
           {
                    if(cc[ch[i]]<cc[ch[j]])
                    {
                              t=ch[i];
                              ch[i]=ch[j];
                              ch[j]=t;
                    }
           }
          }
for(i=0;i<n;i++)
          {

                    be[i].col[0]=ch[i];
                    be[i].count=cc[ch[i]];
       extr[ch[i]]=i;
          }
while(n>1)
{
          n=formbyte(extr,bs,n);
}
for(i=0;i<cot;i++)
  {
```

```
          asb=ch[i];
      if ((bsm[asb] = (char *) malloc(32)) == NULL)
            {
                  printf("Not enough memory to allocate buffer\n");
                  exit(1);
            }
      strcpy(bsm[asb],bs[i]);
      free(bs[i]);
  }
rewind(in);
/*******storing to file**********/
p=i=0;
while(!feof(in))
{
          chr=getc(in);
    while(bsm[chr][p]!='\0')
          {
                  puf[i]=bsm[chr][p];
        chs=getc(in);
        if(feof(in))
          ab=3;
        ungetc(chs,in);
        if(i==7||((ab==3)&&(bsm[chr][p+1]=='\0')))
                  {
                   while(i<7)
                   {
                   puf[++i]='0';
                   }
                   for(g=0;g<8;g++)
                           {
                                   tb = tb +(puf[g] - '0')*pow(2,(7-g));
                           }
                   fputc(tb,out);
         tb=0;
                   for(i=0;i<8;i++)
                   puf[i]='\0';
                   i=-1;
                  }
                  p++;
                  i++;
          }
          p=0;
}
fcloseall();
i=0;
while(i<cot)
free(bsm[ch[i++]]);
}
int formbyte(int *extr,char **bs,int n)
{
int i,j,p,x=0;
static char t[260];
          for(i=0;i<n-1;i++)
```

```
        {
          for(j=i+1;j<n;j++)
          {
                if(be[i].count<be[j].count)
                {
                        strcpy(t,be[i].col);
                        strcpy(be[i].col,be[j].col);
                        strcpy(be[j].col,t);
                                p=be[i].count;
                                be[i].count=be[j].count;
                                be[j].count=p;
                }
          }
        }
while(be[n-1].col[x])
        {
                        strcpy(t,"0");
                        strcat(t,bs[extr[be[n-1].col[x]]]);
                        strcpy(bs[extr[be[n-1].col[x]]],t);
                        x++;
         }
         n--;
         x=0;
while(be[n-1].col[x])
         {
                        strcpy(t,"1");
                        strcat(t,bs[extr[be[n-1].col[x]]]);
                        strcpy(bs[extr[be[n-1].col[x]]],t);
                        x++;
         }
if(n>0)
        strcat(be[n-1].col,be[n].col);
be[n-1].count += be[n].count;
return(n);
}
```

DATA HIDING:

```
#include<stdio.h>
#include<string.h>
void main()
{
  int n=0,p,i=0,t=0,d=1;
  char str[100];
  FILE *fp,*fq;
  if((fp = fopen("c:\\sai.srp","r")) == NULL)
  {
    printf("sai.srp");
    exit(0);
  }
  if((fq = fopen("c:\\sri.srp","w")) == NULL)
  {
   printf("sri.srp");
```

```
 exit(0);
 }
 printf("Enter the string :");
 gets(str);
 while(1)
 {
  n++;
  if((p = fgetc(fp))==EOF)
  {
   exit(0);
       }
  if((d == 1) &&(n%30 == 0))
  {
   fputc(p,fq);
   t=0;
   while(t<3)
   {
    fputc(str[i],fq);
    t++;
    if(i==(strlen(str)-1))
    {
      if((strlen(str)%3)==1)
      {
       fputc(' ',fq);
       fputc(' ',fq);
       t += 2;
                  }
      if((strlen(str)%3)==2)
      {
       fputc(' ',fq);
       t++;
                 }
      d = 23;
               }
   i++;
   }

       }
  else
  {
   fputc(p,fq);
       }
 }
}
```

DATA RETRIEVING:

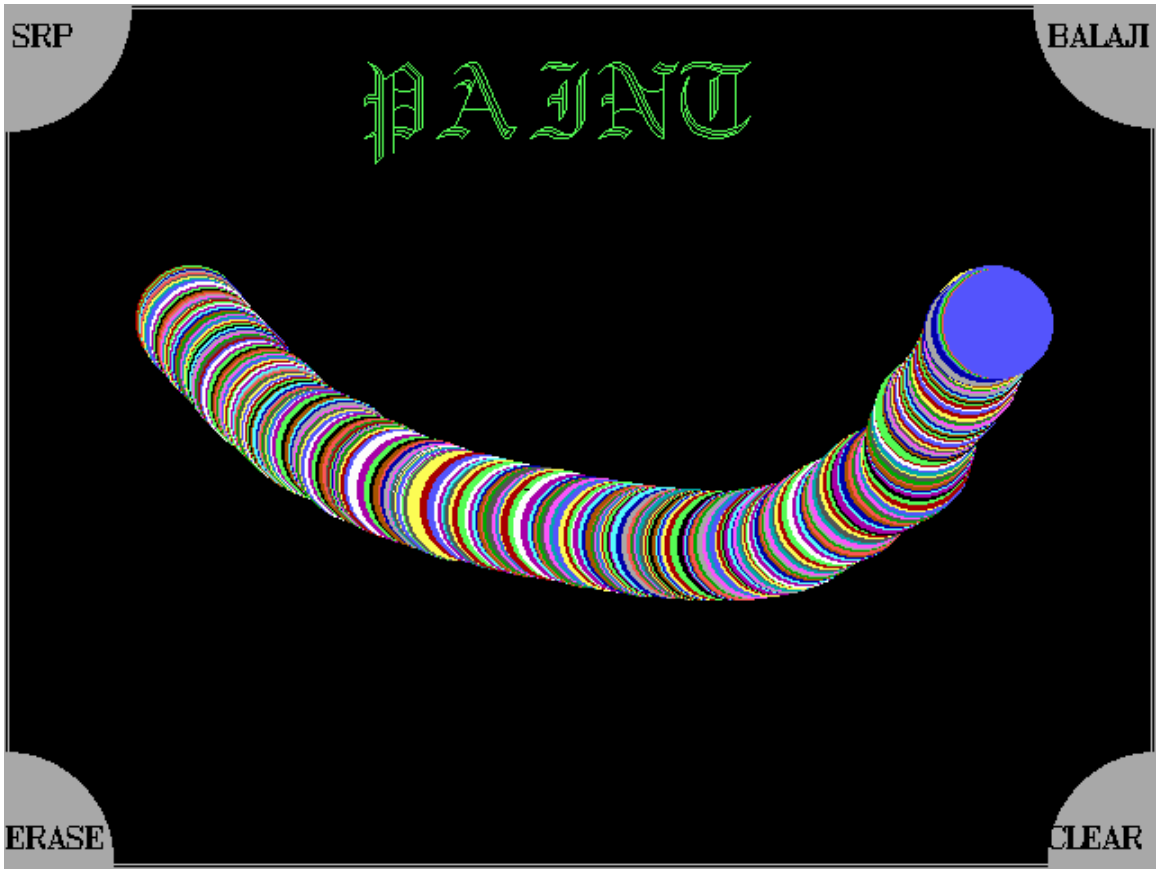
```
#include<stdio.h>
#include<conio.h>
void main()
{
 long int n=0,i=0;
 int r=0,a=0,m=0,g=0,z=0;
```

```
char str[100];
FILE *fq,*fr;
clrscr();

if((fr=fopen("c:\\sri.srp","r"))==NULL)
{
  printf("Error in sri.srp");
  getch();
  exit(0);
}

if((fq=fopen("c:\\F.SRP","w"))==NULL)
{
  printf("Error in f.srp");
  getch();
  exit(0);
}
while(1)
{
n=0;
if((r = fgetc(fr))==EOF)
{

  str[g] = '\0';
  puts(str);
  fcloseall();
  getch();
  exit(0);
}
if(((r-'0') > 15) && ((r-'0') != 23))
{

   str[g] = r-50;
   for(z=0;z<2;z++)
   {
     g++;
     str[g] = fgetc(fr)-50;

   }
   g++;
   r = fgetc(fr);

}

m = fgetc(fr);
a = fgetc(fr);
m=m-'0';
a=a-'0';
n = m*200 + a;
i=0;
while(i<n)
{
  fputc(r,fq);
```

```
  i++;
}
}
}
```

SAMPLE OUTPUTS :

IMAGE CREATION APPLICATION :



The above image is obtained by setting the colour value = 23 (multi colour), brush size = 30 and style =1. The required pattern is drawn by maneuvering the mouse across the screen.

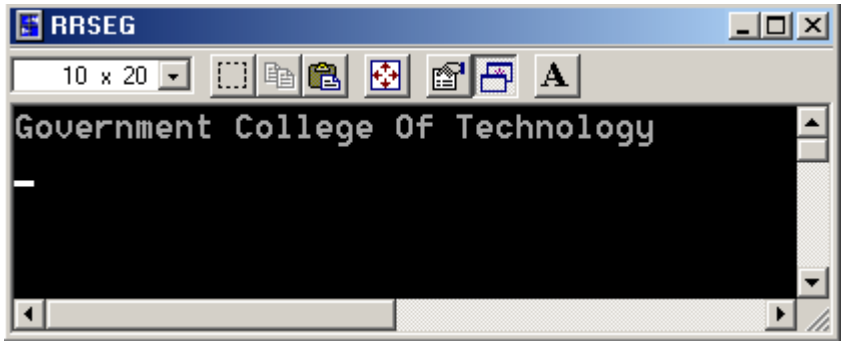
RLE ENCODED IMAGE :

DATA HIDING :

The data to be hidden in the image is entered as shown in the above window.

RLE ENCODED IMAGE CONTAINING HIDDEN DATA:

DATA RETRIEVAL :



The data from the transmitted image is shown in the above window.

FUTURE WORK:

We are planning to make our project compatible to vogue image formats like JPEG and BMP . The encoding of JPEG is a formidable task as it internally uses a lossy compression algorithm based on discrete cosine transforms (DCT).

BIBLIOGRAPHY:

1. Dwayne Phillips, “Image Processing in C”, BPB Publications, 1995
2. Nick Efford, “Digital Image Processing”, Pearson Education,2000
3. Jean-Paul Tremblay Paul G. Sorenson, “An Introduction to data structures with applications”, Tata McGraw-Hill ltd,1991