Zadaća 2

Zadatak 1

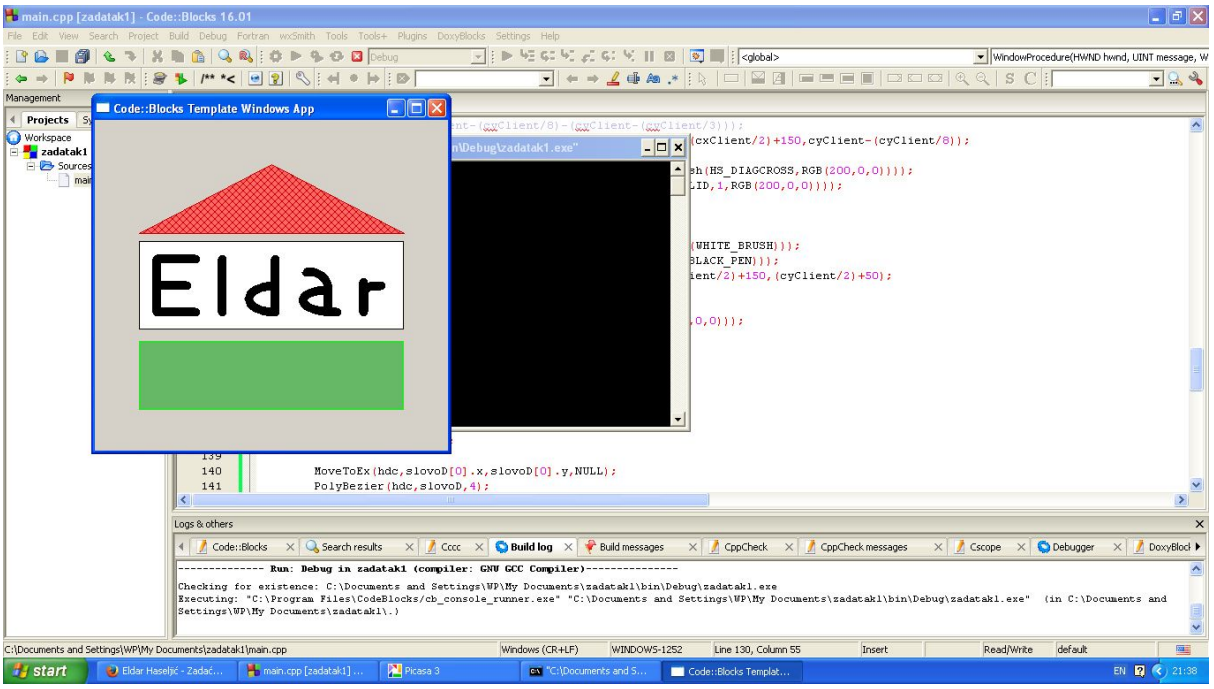
Napisati Windows program koji će u klijentskom području iscrtati Vaše ime sa zelenim pravougaonikom kao postoljem tog imena a šrafiranim trouglom boje crijepa kao krovom. Debljinu linije ispisa imenaprilagoditi veličini slova. Krajnja slika treba da bude širine 300 piksela i slična donjoj slici:

Napomena: Prvo slovo Vašeg imena je veliko štampano slovo, naredna slova su mala štampana. Ukoliko je Vaše ime duže od 5 slova, možete nacrtati samo prvih 5 slova. Ukoliko se Vaše ime sadrži od 4 slova onda nacrtajte i prvo slovo Vašeg prezimena.

Gdje god je moguće koristiti niz tačaka i Poly verzije funkcija.

U nastavku donjeg teksta, na odgovarajuća mjesta je potrebno ubaciti sliku dobijenog prozora, definiciju window klase, poziv funkcije CreateWindow te definiciju window procedure za odgovarajući prozor.

Slika prozora (screen shot)



Definicija window klase

```
      /* The Window structure */
      wincl.hInstance = hThisInstance;
      wincl.lpszClassName = szClassName;
      wincl.lpfnWndProc = WindowProcedure;       /* This function
is called by windows */
      wincl.style = CS_DBLCLKS;                  /* Catch
double-clicks */
      wincl.cbSize = sizeof (WNDCLASSEX);

      /* Use default icon and mouse-pointer */
      wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
      wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
      wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
      wincl.lpszMenuName = NULL;                 /* No menu */
      wincl.cbClsExtra = 0;                      /* No extra bytes
after the window class */
      wincl.cbWndExtra = 0;                      /* structure or
the window instance */
      /* Use Windows's default colour as the background of the
window */
      wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;
```

```
     /* Register the window class, and if it fails quit the
program */
     if (!RegisterClassEx (&wincl))
     return 0;
```

Poziv funkcije CreateWindow

```
     /* The class is registered, let's create the program*/
     hwnd = CreateWindowEx (
                0,                   /* Extended possibilites for
variation */
                szClassName,         /* Classname */
                _T("Code::Blocks Template Windows App"),        /*
Title Text */
                WS_OVERLAPPED | WS_MINIMIZEBOX | WS_SYSMENU, /*
default window */
                100,       /* Windows decides the position */
                100,       /* where the window ends up on the
screen */
                408,                 /* The programs width */
                408,                 /* and height in pixels */
                HWND_DESKTOP,        /* The window is a
child-window to desktop */
                NULL,                /* No menu */
                hThisInstance,       /* Program Instance handler
*/
                NULL                 /* No Window Creation data */
           );
```

Definicija window procedure

```
/*  This function is called by the Windows function
DispatchMessage()  */

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM
wParam, LPARAM lParam)
{
     static int cxClient, cyClient;
     PAINTSTRUCT ps;
     HDC hdc;

     switch (message)                /* handle the messages */
     {
     case WM_SIZE:
```

```
      cxClient = LOWORD(lParam);
      cyClient = HIWORD(lParam);

      break;
      case WM_PAINT:
      hdc = BeginPaint(hwnd,&ps);
      points[0] = {cxClient/2-150,128};
      points[1] = {cxClient/2,50};
      points[2] = {cxClient/2+150,128};

      SelectObject(hdc,
(HBRUSH)CreateSolidBrush(RGB(100,180,100)));
      SelectObject(hdc,(HPEN)CreatePen(PS_SOLID,1,RGB(0,255,0)));
      //printf("%d\n",(cxClient/2)+150 - ((cxClient/2)-150) );
      //printf("%d",cyClient-(cyClient/8)-(cyClient-(cyClient/3)));

Rectangle(hdc,(cxClient/2)-150,cyClient-(cyClient/3),(cxClient/2)+
150,cyClient-(cyClient/8));

      DeleteObject(SelectObject(hdc,(HBRUSH)
CreateHatchBrush(HS_DIAGCROSS,RGB(200,0,0))));
      DeleteObject(SelectObject(hdc, (HPEN)
CreatePen(PS_SOLID,1,RGB(200,0,0))));
      SetBkColor(hdc,RGB(255,100,100));
      Polygon(hdc,points,3);

      DeleteObject(SelectObject(hdc,(HBRUSH)
GetStockObject(WHITE_BRUSH)));
      DeleteObject(SelectObject(hdc, (HPEN)
GetStockObject(BLACK_PEN)));
      Rectangle(hdc,(cxClient/2)-150,
(cyClient/2)-50,(cxClient/2)+150,(cyClient/2)+50);
      // printf("%d",(cyClient/2)-50);

      SelectObject(hdc,(HPEN) CreatePen(PS_SOLID, 10, RGB(0,0,0)));
      MoveToEx(hdc,slovoE[1].x,slovoE[1].y,NULL);
      PolylineTo(hdc,slovoE,7);

      MoveToEx(hdc,144,slovoE[6].y,NULL);
      LineTo(hdc,144,157);

      MoveToEx(hdc,204,slovoE[6].y,NULL);
      LineTo(hdc,204,157);

      MoveToEx(hdc,slovoD[0].x,slovoD[0].y,NULL);
      PolyBezier(hdc,slovoD,4);
```

```
      for(auto i = 0; i<4;++i)
      {
            slovoD[i].x += 60;
            if(i<2)
            slovoD[i].y +=5;
      }
      slovoD[3].x +=10;

      //MoveToEx(hdc,slovoD[0].x,slovoD[0].y,NULL);
      PolyBezier(hdc,slovoD,4);

      slovoD[1].x = slovoD[0].x;
      slovoD[1].y = slovoD[2].y = 170;
      slovoD[2].x = slovoD[3].x = slovoD[0].x-25;
      slovoD[3].y = slovoD[0].y-20;

      //MoveToEx(hdc,slovoD[0].x,217,NULL);
      PolyBezier(hdc,slovoD,4);

      MoveToEx(hdc,slovoD[0].x,slovoD[0].y,NULL);
      LineTo(hdc,slovoD[0].x,217);

      MoveToEx(hdc,304,217,NULL);
      LineTo(hdc,304,183);

      PolyBezier(hdc,slovoR,4);

      slovoD[0].x = slovoD[3].x = 204;
      slovoD[0].y = slovoD[1].y = 187;
      slovoD[1].x = slovoD[2].x = 164;
      slovoD[2].y = slovoD[3].y = 217;
      DeleteObject(SelectObject(hdc, (HPEN)
GetStockObject(BLACK_PEN)));

      EndPaint(hwnd,&ps);
      break;
      case WM_DESTROY:
      PostQuitMessage (0);       /* send a WM_QUIT to the message
queue */
      break;
      default:                   /* for messages that we don't deal
with */
      return DefWindowProc (hwnd, message, wParam, lParam);
      }
```

```
	return 0;
}
```

Zadatak 2

U fajlu se nalaze definicije vektorskog crteža na sljedeći način:

Kružnica, koordinate centra 60, 70, r=50:
C 60 70 50

Linija od (50,60) do (70,80):
L 50 60 70 80

Promjena boje i debljine ispisa (ljubičasta, debljina 2): // olovka, koristiti samo dash stil
C S 2 100 0 100

Pravougaonik sa left, top, right i bottom granicama:
R 10 50 80 30

Elipsa sa left, top, right i bottom granicama:
E 10 50 80 30

Kvadrat , koordinate centra 50,50, dužina stranice 40:
S 50 50 40

Solid brush ljubičasta:
B 100 0 100

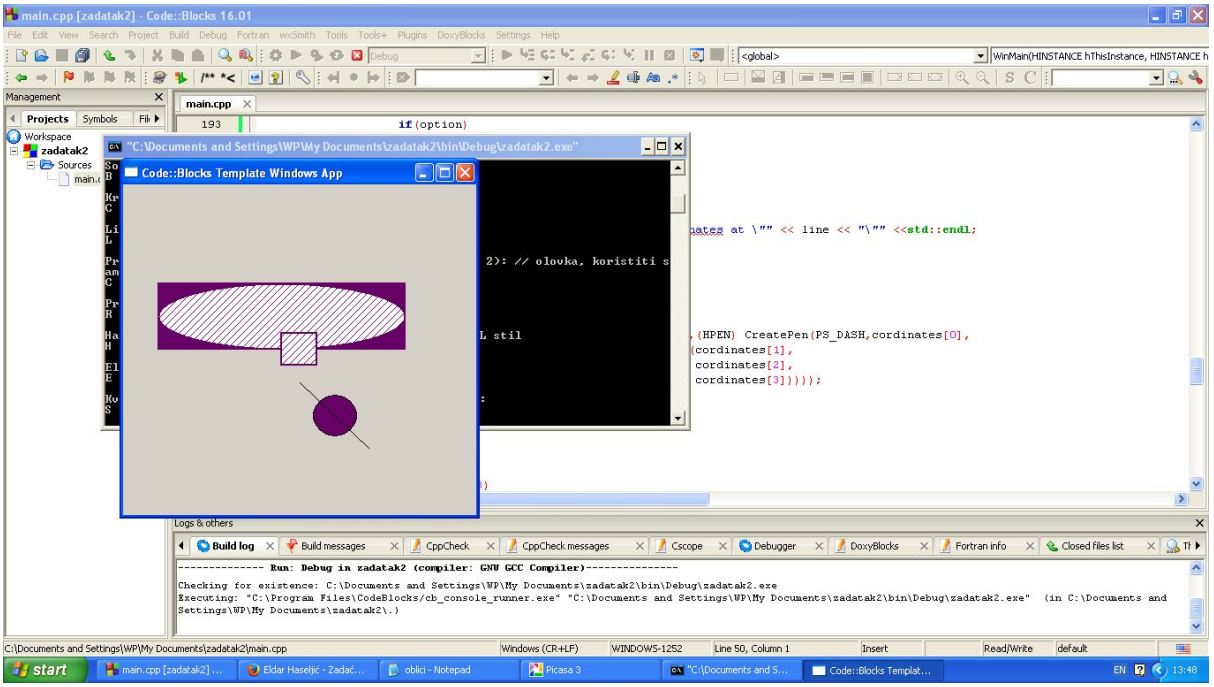
Hatch brush ljubičasta: // koristiti samo HS_BDIAGONAL stil
H B 100 0 100

Potrebno je napisati program koji će parsirati fajl i izvršavati parsirane komande. Sve koordinate su u procentima i sve se skalira kako se mijenjaju dimenzije prozora.

U nastavku donjeg teksta, na odgovarajuća mjesta je potrebno ubaciti sliku dobijenog prozora, definiciju window klase, poziv funkcije CreateWindow te definiciju window procedure za odgovarajući prozor.

Kad završite sa radom, kliknite na odgovarajuće dugme na interfejsu kako biste vratili Vaš rad na ocjenu.

Slika prozora (screen shot)



Definicija window klase

```
 /* The Window structure */
      wincl.hInstance = hThisInstance;
      wincl.lpszClassName = szClassName;
      wincl.lpfnWndProc = WindowProcedure;       /* This function is called
by windows */
      wincl.style = CS_DBLCLKS | CS_VREDRAW | CS_HREDRAW;                 /*
Catch double-clicks */
      wincl.cbSize = sizeof (WNDCLASSEX);

      /* Use default icon and mouse-pointer */
      wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
      wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
      wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
      wincl.lpszMenuName = NULL;                 /* No menu */
      wincl.cbClsExtra = 0;                      /* No extra bytes after the
window class */
      wincl.cbWndExtra = 0;                      /* structure or the window
instance */
      /* Use Windows's default colour as the background of the window */
      wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

      /* Register the window class, and if it fails quit the program */
      if (!RegisterClassEx (&wincl))
      return 0;
```

Poziv funkcije CreateWindow

```
/* The class is registered, let's create the program*/
hwnd = CreateWindowEx (
0,                  /* Extended possibilites for variation */
szClassName,        /* Classname */
_T("Code::Blocks Template Windows App"),    /* Title Text */
WS_OVERLAPPEDWINDOW, /* default window */
CW_USEDEFAULT,      /* Windows decides the position */
CW_USEDEFAULT,      /* where the window ends up on the screen */
408,                /* The programs width */
408,                /* and height in pixels */
HWND_DESKTOP,       /* The window is a child-window to desktop */
NULL,               /* No menu */
hThisInstance,      /* Program Instance handler */
NULL                /* No Window Creation data */
);
```

Definicija window procedure

```
/*  This function is called by the Windows function DispatchMessage()  */

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
      static int cxClient,cyClient;
      std::string line;
      PAINTSTRUCT ps;
      HDC hdc;

      switch (message)                /* handle the messages */
      {
      case WM_SIZE:
      cyClient = HIWORD(lParam);
      cxClient = LOWORD(lParam);
      break;
      case WM_PAINT:
      file.open("oblici.txt");
      if(!file.is_open())
      {
            perror("Error with opening");
            exit(0);
      }

      hdc = BeginPaint(hwnd,&ps);
      while(std::getline(file,line))
      {
            if(line.empty())
            {
```

```
            std::cout << line << std::endl;
            }
            else if (( line.at(0) == 'B' || line.at(0) == 'C' || line.at(0)
== 'L'
                        || line.at(0) == 'R' || line.at(0) == 'H' ||
line.at(0) == 'E'
                        || line.at(0) == 'S' ) && line.at(1)==' ')
            {
            std::cout << line << std::endl;
            int i = 1, start, len, num = 0;
            bool option = false;
            char op;

            while(i<line.size())
            {
                  while(line.at(i)==' ') ++i;

                  if((line.at(i)>'a' && line.at(i)<'z') || (line.at(i)>'A'
&& line.at(i)<'Z'))
                  {
                        option = true;
                        op = line.at(i);
                        ++i;
                  }

                  while(line.at(i)==' ') ++i;

                  start = i;
                  len = 0;
                  while(i<line.size())
                  {
                        ++len;
                        if(line.at(i)==' ') break;
                        ++i;
                  }
                  cordinates[num] = stof(line.substr(start,len)) /100.0;
                  // std::cout << cordinates[num] << std::endl;
                  ++num;
            }

            switch(line.at(0))
            {
            case 'E':
                  if(num<3)
                  {
                        std::cout << "Need more cordinates at \"" << line <<
"\"" <<std::endl;
                        exit(0);
                  }
                  else
                  {
```

```
                        // std::cout << cordinates[0]-cordinates[2] << " "
<< cordinates[1]-cordinates[2];
                        Ellipse(hdc,cxClient * (cordinates[0]),
                              cyClient * (cordinates[1]),
                              cxClient * (cordinates[2]),
                              cyClient * (cordinates[3]));
                  }
                  break;
            case 'B':
            case 'H':
                  if(num<3)
                  {
                        std::cout << "Need more cordinates at \"" << line <<
"\"" <<std::endl;
                        exit(0);
                  }
                  else
                  {
                        for(auto i =0; i<num; ++i)
                        cordinates[i] *= 100.0;
                        if(option && op == 'B')
                        {
                        DeleteObject(SelectObject(hdc,(HBRUSH)
CreateHatchBrush(HS_BDIAGONAL,RGB(cordinates[0],
                                                      cordinates[1],
                                                      cordinates[2])));
                        }
                        else
                        {
                        DeleteObject(SelectObject(hdc,(HBRUSH)
CreateSolidBrush(RGB(cordinates[0],
                                                      cordinates[1],
                                                      cordinates[2])));
                        }
                  }
                  break;
            case 'C':
                  if(option)
                  {
                        switch(op)
                        {
                        case 'S':
                        if(num<4)
                        {
                              std::cout << "Need more cordinates at \"" <<
line << "\"" <<std::endl;
                              exit(0);
                        }
                        else
                        {
```

```
                              for(auto i =0; i<num; ++i)
                                    cordinates[i] *= 100.0;
                              DeleteObject(SelectObject(hdc,(HPEN)
CreatePen(PS_DASH,cordinates[0],
                                                      RGB(cordinates[1],
                                                            cordinates[2],

cordinates[3])))));
                        }
                        break;
                        }
                  }
                  else
                  {
                        if(num<3)
                        {
                        std::cout << "Need more cordinates at \"" << line <<
"\"" <<std::endl;
                        exit(0);
                        }
                        else
                        {
                        cordinates[2] /= 8.0;
                        //std::cout << cordinates[0]-cordinates[2] << " " <<
cordinates[1]-cordinates[2];
                        Ellipse(hdc,cxClient *
(cordinates[0]-cordinates[2]),
                                    cyClient *
(cordinates[1]-cordinates[2]),
                                    cxClient *
(cordinates[0]+cordinates[2]),
                                    cyClient *
(cordinates[1]+cordinates[2]));
                        }
                  }
                  break;
            case 'L':
                  if(num<4)
                  {
                        std::cout << "Need more cordinates at \"" << line <<
"\"" <<std::endl;
                        exit(0);
                  }
                  else
                  {
MoveToEx(hdc,cxClient*cordinates[0],cyClient*cordinates[1],NULL);
LineTo(hdc, cxClient*cordinates[2],cyClient*cordinates[3]);
                  }
                  break;
            case 'R':
                  if(num<4)
```

```
                  {
                        std::cout << "Need more cordinates at \"" << line <<
"\"" <<std::endl;
                        exit(0);
                  }
                  else
                  {
                        Rectangle(hdc, cxClient * cordinates[0],
                              cyClient * cordinates[1],
                              cxClient * cordinates[2],
                              cyClient * cordinates[3]);
                  }
                  break;
            case 'S':
                  if(num<3)
                  {
  std::cout << "Need more cordinates at \"" << line << "\"" <<std::endl;
                        exit(0);
                  }
                  else
                  {
                        cordinates[2] /= 8.0;
            Rectangle(hdc,cxClient *(cordinates[0]-cordinates[2]),
                              cyClient * (cordinates[1]-cordinates[2]),
                              cxClient * (cordinates[0]+cordinates[2]),
                              cyClient * (cordinates[1]+cordinates[2]));
                  }
                  break;
            }
            }
            else
            {
            std::cout << line << std::endl;
            }
      }
      DeleteObject(SelectObject(hdc,(HPEN) GetStockObject(BLACK_PEN)));
      DeleteObject(SelectObject(hdc,(HBRUSH) GetStockObject(WHITE_BRUSH)));
      EndPaint(hwnd,&ps);
      file.close();
      break;
      case WM_DESTROY:
      PostQuitMessage (0);     /* send a WM_QUIT to the message queue */
      break;
      default:                      /* for messages that we don't deal with
*/
      return DefWindowProc (hwnd, message, wParam, lParam);
      }

      return 0;
}
```