FLTK é uma biblioteca para criar GUI (graphical user interface). Cekeikon5 vem com FLTK versão 1.3.3.

FLTK foi desenvolvida usando o conceito de programação orientada a objetos.

Site oficial de FLTK:
http://www.fltk.org

Manual de FLTK:
http://www.fltk.org/doc-1.3

Tutoriais sobre FLTK:
http://www.fltk.org/links.php?SD455
http://seriss.com/people/erco/fltk/
http://www.fltk.org/doc-1.3/examples.html
(Estes exemplos estão em C:\cekeikon5\fltk\share\doc\fltk\examples)

Os exemplos desta apostila estão em:
c:\cekeikon5\cekeikon\samples\cek-fltk

1) MostraC: Mostra uma imagem colorida.

```
//mostrac.cpp
//>compila -f -c mostrac
//>compila -f -c -w mostrac
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Shared_Image.H>
#include <FL/Fl_Box.H>
#include <cekeikon.h>

Fl_Window* win=0;
  Fl_Box* box=0;
Fl_RGB_Image* rgb=0;
Mat_<COR> ent; // armazenado internamente como RGB

int main(int argc, char** argv) {
  if (argc!=2) {
    printf("MostraC: Mostra imagem colorida\n");
    printf("MostraC ent.ppm\n");
    printf("  Aceita varios tipos de imagens: ppm, jpg, png, tga, bmp, etc.\n");
    erro("Erro: Numero de argumentos invalido");
  } else {
    le(ent,argv[1]);
    cvtColor(ent,ent,CV_BGR2RGB);

    rgb = new Fl_RGB_Image(ent.data, ent.cols, ent.rows, 3);

    win = new Fl_Window(ent.cols,ent.rows, argv[1]);
    win->begin();
      box = new Fl_Box(0,0,ent.cols,ent.rows);
      box->image(rgb);
    win->end();
    win->show();
    return Fl::run();
  }
}
```

Nota: Um programa típico de FLTK não obedece a metodologia RAII (Resource Acquisition Is Initialization), recomendada para ser usada em C++. A memória usada pelos objetos alocados com “new” é desalocada com o fim do programa.

Nota: As imagens coloridas de FLTK são armazenadas no formato RGB, enquanto que as imagens coloridas de OpenCV são armazenadas no formato BGR. Para poder trabalhar com as duas bibliotecas, vamos convencionar que armazenaremos todas as imagens internamente no formato RGB.

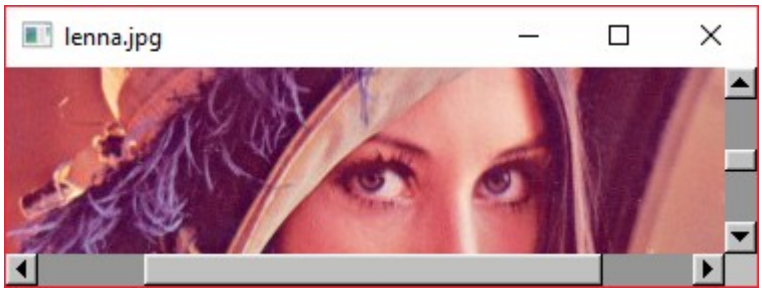
Nota: O comando “`rgb = new Fl_RGB_Image(ent.data, ent.cols, ent.rows, 3);`” não copia a imagem ent para rgb, mas faz o apontador de rgb apontar para os dados de ent. Porém, uma alteração no ent não será visível pois Windows faz uma cópia interna (cache). Para que uma alteração de ent seja visível, você deve dar o comando box->uncache(); win->redraw().

Nota:

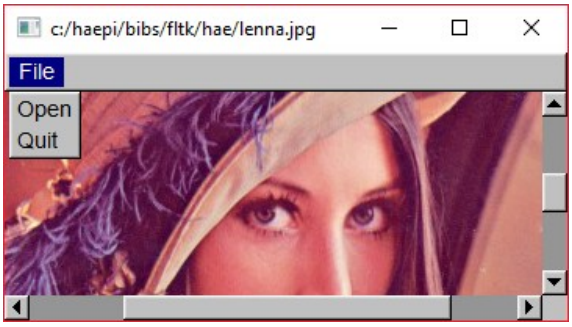
```
>compila -f -c mostrac      → permite visualizar saída de printf
>compila -f -c -w mostrac   → programa fica independente de prompt.
```

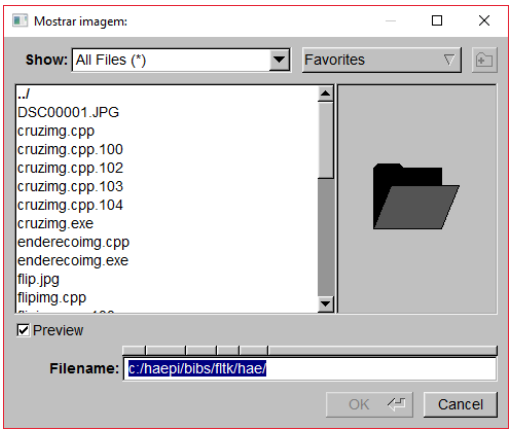
2) ScrollC: Mostra uma imagem colorida, com scroll.
O programa anterior não consegue mostrar imagens maiores que a tela do computador. É possível resolver isto usando scroll.

```
//scrollc.cpp
//>compila -f -c mostrac
//>compila -f -c -w mostrac
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Shared_Image.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_Scroll.H>
#include <cekeikon.h>

Fl_Window* win=0;
  Fl_Scroll* scr=0;
    Fl_Box* box=0;
Fl_RGB_Image* rgb=0;
Mat_<COR> ent; // armazenado internamente como RGB

int main(int argc, char** argv) {
  if (argc!=2) {
    printf("ScrollC: Mostra imagem colorida com scroll\n");
    printf("ScrollC ent.ppm\n");
    printf("  Aceita varios tipos de imagens: ppm, jpg, png, tga, bmp, etc.\n");
    erro("Erro: Numero de argumentos invalido");
  } else {
    le(ent,argv[1]);
    cvtColor(ent,ent,CV_BGR2RGB);

    rgb = new Fl_RGB_Image(ent.data, ent.cols, ent.rows, 3);

    win = new Fl_Window(ent.cols,ent.rows, argv[1]);
    win->begin();
      scr = new Fl_Scroll(0,0,ent.cols,ent.rows);
      scr->begin();
        box = new Fl_Box(0,0,ent.cols,ent.rows);
        box->image(rgb);
      scr->end();
    win->end();
    win->resizable(scr);
    win->show();
    return Fl::run();
  }
}
```



3) ShowImg: Mostra uma imagem colorida, com scroll, podendo escolher a imagem através de menu_bar e caixa de diálogo.

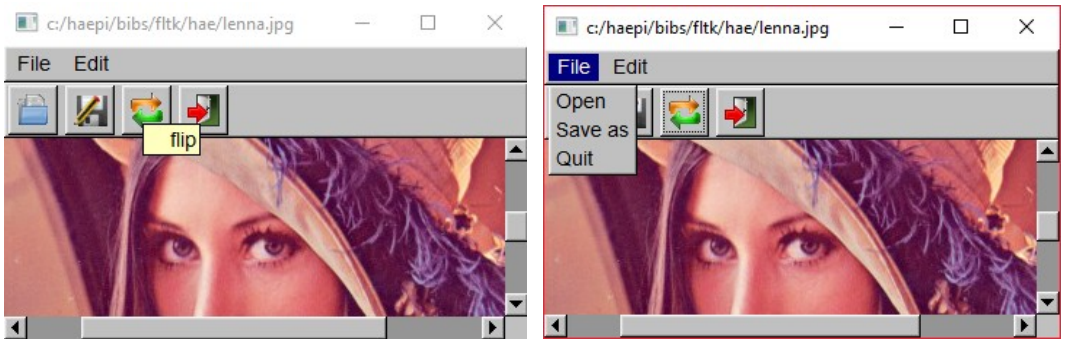





```
//showimg.cpp
//>compila -f -c showimg
//>compila -f -c -w showimg
#include <FL/Fl.H>
#include <FL/Fl_Menu_Bar.H>
#include <FL/Fl_File_Chooser.H>
#include <FL/Fl_Shared_Image.H>
#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Scroll.H>
#include <FL/Fl_Box.H>
#include <cekeikon.h>

Fl_Double_Window* win=0;
  Fl_Menu_Bar* menubar=0;
  Fl_Scroll* scr=0;
    Fl_Box* box=0;
Fl_RGB_Image* rgb=0;
Mat_<COR> ent; // armazenado internamente como RGB

void cb_open(Fl_Widget* cb, void*) {
  Fl_File_Chooser chooser(".",                        // directory
                          "*",                        // filter
                          Fl_File_Chooser::SINGLE,     // chooser type
                          "Mostrar imagem:");        // title
  chooser.show();
  while ( chooser.shown() ) Fl::wait();
  if ( chooser.value() == NULL ) return;

  le(ent,chooser.value());
  cvtColor(ent,ent,CV_BGR2RGB);

  if (rgb!=0) delete rgb;
  rgb = new Fl_RGB_Image(ent.data, ent.cols, ent.rows, 3);

  box->size(rgb->w(),rgb->h());
  box->image(rgb);

  win->label(chooser.value());
  win->redraw();
}
```

```
void cb_quit(Fl_Widget*, void*) {
  exit(0);
}

int main() {
  Fl::visual(FL_DOUBLE|FL_RGB8);
  int nl=240; int nc=360;
  win = new Fl_Double_Window(nc,nl+25, "Show image");
  win->begin();
    menubar = new Fl_Menu_Bar(0,0,nc,25);
    menubar->add("File/Open", 0, cb_open);
    menubar->add("File/Quit", 0, cb_quit);

    scr = new Fl_Scroll(0,25,nc,nl);
    scr->begin();
      box = new Fl_Box(0,25,nc,nl);
      box->color(FL_GRAY);
    scr->end();
  win->end();
  win->resizable(scr);
  win->show();
  return Fl::run();
}
```

4) FlipImg: Acrescenta botões (toolbar), permite salvar imagem, e alterar imagem (reflexão).




```
//flipimg.cpp
//>compila -f -c flipimg
//>compila -f -c -w flipimg

//<<<<<<<<<<<<<<<<<< include .h <<<<<<<<<<<<<<<<<<<<
#include <FL/Fl.H>
#include <FL/Fl_Menu_Bar.H>
#include <FL/Fl_File_Chooser.H>
#include <FL/Fl_Shared_Image.H>
#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Scroll.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_Pixmap.H>
#include <cekeikon.h>

//<<<<<<<<<<< icones <<<<<<<<<<<<<<<<<<<<
#include "document-open-5.xpm"
#include "document-save-as-5.xpm"
#include "view-refresh-2.xpm"
#include "application-exit-2.xpm"

//<<<<<<<<<<< headers <<<<<<<<<<<<<<<<<<<<
void open_cb(Fl_Widget* cb, void*);
void saveAs_cb(Fl_Widget* cb, void*);
void quit_cb(Fl_Widget*, void*);

class Toolbar : public Fl_Group {
public:
  Fl_Button *b_load=0;
  Fl_Button *b_saveAs=0;
  Fl_Button *b_flip=0;
  Fl_Button *b_quit=0;
  Fl_Pixmap *p_load=0;
  Fl_Pixmap *p_saveAs=0;
  Fl_Pixmap *p_flip=0;
  Fl_Pixmap *p_quit=0;
  Toolbar(int Xpos, int Ypos, int Width, int Height);
};

//<<<<<<<<<<<<<<<< program .cpp <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

//<<<<<<<<<<<<<<<< variaveis globais <<<<<<<<<<<<<<<<<<
Fl_Double_Window* win=0;
  Fl_Menu_Bar* menubar=0;
  Toolbar* tool=0;
```

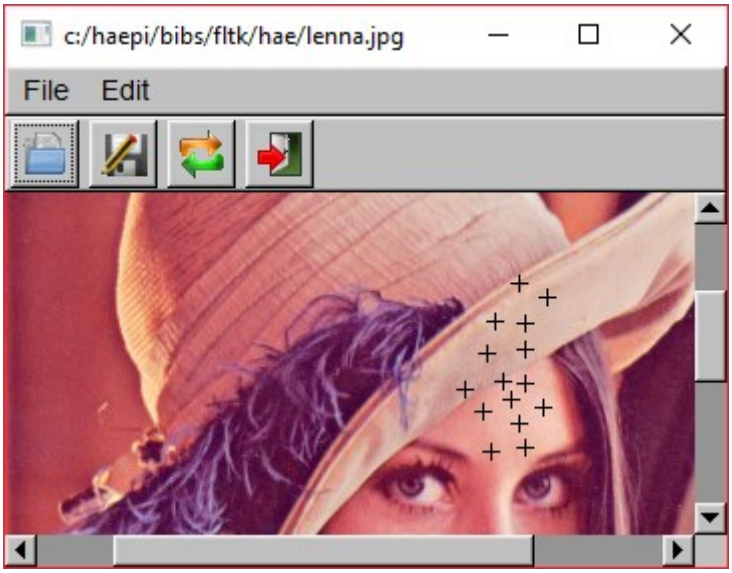
```
  Fl_Scroll* scr=0;
    Fl_Box* box=0;
Fl_RGB_Image* rgb=0;
Mat_<COR> ent;

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
void cb_open(Fl_Widget* cb, void*) {
  Fl_File_Chooser chooser(".",                        // directory
                          "*",                        // filter
                          Fl_File_Chooser::SINGLE,     // chooser type
                          "Mostrar imagem:");        // title
  chooser.show();
  while ( chooser.shown() ) Fl::wait();
  if ( chooser.value() == NULL ) return;

  le(ent,chooser.value());
  cvtColor(ent,ent,CV_BGR2RGB);

  if (rgb!=0) delete rgb;
  rgb = new Fl_RGB_Image(ent.data, ent.cols, ent.rows, 3);

  box->size(rgb->w(),rgb->h());
  box->image(rgb);

  win->label(chooser.value());
  win->redraw();
}

void cb_saveAs(Fl_Widget* cb, void*) {
  if (rgb==0) return;
  Fl_File_Chooser chooser(".",                        // directory
                          "*",                        // filter
                          Fl_File_Chooser::CREATE,     // chooser type
                          "Gravar como:");        // title
  chooser.show();
  while ( chooser.shown() ) Fl::wait();
  if ( chooser.value() == NULL ) return;

  Mat_<COR> temp;
  cvtColor(ent,temp,CV_RGB2BGR);
  imp(temp,chooser.value());
  win->label(chooser.value());
  win->redraw();
}

void cb_flip(Fl_Widget* cb, void*) {
  if (rgb==0) return;
  flip(ent,ent,1);
  rgb ->uncache();
  win->redraw();
}

void cb_quit(Fl_Widget*, void*) {
  exit(0);
}

Toolbar::Toolbar(int Xpos, int Ypos, int Width, int Height) :
  Fl_Group(Xpos, Ypos, Width, Height)
{
```

```
  box(FL_UP_BOX);
  Ypos += 2; Height -= 4; Xpos += 3; Width = Height;
  int i;
  begin();
    b_load   = new Fl_Button(Xpos, Ypos, Width, Height); Xpos += Width + 5;
    b_saveAs = new Fl_Button(Xpos, Ypos, Width, Height); Xpos += Width + 5;
    b_flip   = new Fl_Button(Xpos, Ypos, Width, Height); Xpos += Width + 5;
    b_quit   = new Fl_Button(Xpos, Ypos, Width, Height); Xpos += Width + 5;
  end();

  p_load   = new Fl_Pixmap(document_open_5);
  p_saveAs = new Fl_Pixmap(document_save_as_5);
  p_flip   = new Fl_Pixmap(view_refresh_2);
  p_quit   = new Fl_Pixmap(application_exit_2);

  b_load->  image(p_load);
  b_saveAs->image(p_saveAs);
  b_flip->  image(p_flip);
  b_quit->  image(p_quit);

  b_load->  callback(cb_open);
  b_saveAs->callback(cb_saveAs);
  b_flip->  callback(cb_flip);
  b_quit->  callback(cb_quit);

  b_load->  tooltip("    open");
  b_saveAs->tooltip("    save as");
  b_flip->  tooltip("    flip");
  b_quit->  tooltip("    quit");
}

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
int main() {
  Fl::visual(FL_DOUBLE|FL_RGB8);
  int nl=240; int nc=360;
  win = new Fl_Double_Window(nc,nl+25+38, "Flip image");
  win->begin();
    menubar = new Fl_Menu_Bar(0,0,nc,25);
    menubar->add("File/Open", 0, cb_open);
    menubar->add("File/Save as", 0, cb_saveAs);
    menubar->add("File/Quit", 0, cb_quit);
    menubar->add("Edit/Flip", 0, cb_flip);

    tool = new Toolbar(0,25,nc,38);

    scr = new Fl_Scroll(0,25+38,nc,nl);
    scr->begin();
      box = new Fl_Box(0,25+38,nc,nl);
      box->color(FL_GRAY);
    scr->end();
  win->end();
  win->resizable(scr);
  win->show();
  return Fl::run();
}
```

Nota: Os ícones são de “open icon library” http://sourceforge.net/projects/openiconlibrary/.

Nota: Os ícones foram convertidos de PNG para XPM usando o programa “Image Magick Display”. Também pode-se usar o programa GIMP para esta tarefa.

5) CruzImg: Marca o pixel clicado com uma cruz.




```
//cruzimg.cpp
//>compila -f -c cruzimg
//>compila -f -c -w cruzimg

//<<<<<<<<<<<<<<<<<< include .h <<<<<<<<<<<<<<<<<<<<
#include <FL/Fl.H>
#include <FL/Fl_Menu_Bar.H>
#include <FL/Fl_Pixmap.H>
#include <FL/Fl_File_Chooser.H>
#include <FL/Fl_Shared_Image.H>
#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Scroll.H>
#include <FL/Fl_Box.H>
#include <cekeikon.h>

//<<<<<<<<<< icones <<<<<<<<<<<<<<<<<<<<
#include "document-open-5.xpm"
#include "document-save-as-5.xpm"
#include "view-refresh-2.xpm"
#include "application-exit-2.xpm"

//<<<<<<<<<< headers <<<<<<<<<<<<<<<<<<<<
void open_cb(Fl_Widget* cb, void*);
void saveAs_cb(Fl_Widget* cb, void*);
void quit_cb(Fl_Widget*, void*);

class Toolbar : public Fl_Group {
public:
  Fl_Button *b_load=0;
  Fl_Button *b_saveAs=0;
  Fl_Button *b_flip=0;
  Fl_Button *b_quit=0;
  Fl_Pixmap *p_load=0;
  Fl_Pixmap *p_saveAs=0;
  Fl_Pixmap *p_flip=0;
  Fl_Pixmap *p_quit=0;
  Toolbar(int Xpos, int Ypos, int Width, int Height);
};
```

```
class ScrollBox : public Fl_Box {
public:
  int handle(int e);
  using Fl_Box::Fl_Box;
};

//<<<<<<<<<<<<<<<< program .cpp <<<<<<<<<<<<<<<<<<<<<<<<<<<<<

//<<<<<<<<<<<<<< variaveis globais <<<<<<<<<<<<<<<<<<<<
Fl_Double_Window* win=0;
  Fl_Menu_Bar* menubar=0;
  Toolbar* tool=0;
  Fl_Scroll* scr=0;
    ScrollBox* box=0;
Fl_RGB_Image* rgb=0;
Mat_<COR> ent; // armazenado internamente como RGB

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
void cb_open(Fl_Widget* cb, void*) {
  Fl_File_Chooser chooser(".",                        // directory
                          "*",                        // filter
                          Fl_File_Chooser::SINGLE,     // chooser type
                          "Mostrar imagem:");        // title
  chooser.show();
  while ( chooser.shown() ) Fl::wait();
  if ( chooser.value() == NULL ) return;

  le(ent,chooser.value());
  cvtColor(ent,ent,CV_BGR2RGB);

  if (rgb!=0) delete rgb;
  rgb = new Fl_RGB_Image(ent.data, ent.cols, ent.rows, 3);

  box->size(rgb->w(),rgb->h());
  box->image(rgb);

  win->label(chooser.value());
  win->redraw();
}

void cb_saveAs(Fl_Widget* cb, void*) {
  if (rgb==0) return;
  Fl_File_Chooser chooser(".",                        // directory
                          "*",                        // filter
                          Fl_File_Chooser::CREATE,     // chooser type
                          "Gravar como:");        // title
  chooser.show();
  while ( chooser.shown() ) Fl::wait();
  if ( chooser.value() == NULL ) return;

  Mat_<COR> temp;
  cvtColor(ent,temp,CV_RGB2BGR);
  imp(temp,chooser.value());
  win->label(chooser.value());
  win->redraw();
}

void cb_flip(Fl_Widget* cb, void*) {
```

```
  if (rgb==0) return;
  flip(ent,ent,1);
  rgb->uncache();
  win->redraw();
}

void cb_quit(Fl_Widget*, void*) {
  exit(0);
}

Toolbar::Toolbar(int Xpos, int Ypos, int Width, int Height) :
  Fl_Group(Xpos, Ypos, Width, Height)
{
  box(FL_UP_BOX);
  Ypos += 2; Height -= 4; Xpos += 3; Width = Height;
  int i;
  begin();
    b_load   = new Fl_Button(Xpos, Ypos, Width, Height); Xpos += Width + 5;
    b_saveAs = new Fl_Button(Xpos, Ypos, Width, Height); Xpos += Width + 5;
    b_flip   = new Fl_Button(Xpos, Ypos, Width, Height); Xpos += Width + 5;
    b_quit   = new Fl_Button(Xpos, Ypos, Width, Height); Xpos += Width + 5;
  end();

  p_load   = new Fl_Pixmap(document_open_5);
  p_saveAs = new Fl_Pixmap(document_save_as_5);
  p_flip   = new Fl_Pixmap(view_refresh_2);
  p_quit   = new Fl_Pixmap(application_exit_2);

  b_load->  image(p_load);
  b_saveAs->image(p_saveAs);
  b_flip->  image(p_flip);
  b_quit->  image(p_quit);

  b_load->  callback(cb_open);
  b_saveAs->callback(cb_saveAs);
  b_flip->  callback(cb_flip);
  b_quit->  callback(cb_quit);

  b_load->  tooltip("    open");
  b_saveAs->tooltip("    save as");
  b_flip->  tooltip("    flip");
  b_quit->  tooltip("    quit");
}

int ScrollBox::handle(int e) {
  if (e == FL_PUSH) {
    int c=Fl::event_x() - scr->x() + scr->hscrollbar.value();
    int l=Fl::event_y() - scr->y() + scr->scrollbar.value();
    cruz(ent,l,c,4);
    rgb->uncache();
    win->redraw();
  }
  return Fl_Box::handle(e);
}

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
int main() {
  Fl::visual(FL_DOUBLE|FL_RGB8);
  int nl=240; int nc=360;
```

```
  win = new Fl_Double_Window(nc,nl+25+38, "Cruz image");
  win->begin();
    menubar = new Fl_Menu_Bar(0,0,nc,25);
    menubar->add("File/Open", 0, cb_open);
    menubar->add("File/Save as", 0, cb_saveAs);
    menubar->add("File/Quit", 0, cb_quit);
    menubar->add("Edit/Flip", 0, cb_flip);

    tool = new Toolbar(0,25,nc,38);

    scr = new Fl_Scroll(0,25+38,nc,nl);
    scr->begin();
      box = new ScrollBox(0,25+38,nc,nl);
      box->color(FL_GRAY);
    scr->end();
  win->end();
  win->resizable(scr);
  win->show();
  return Fl::run();
}
```

6) ShowVid: Mostra vídeo na tela usando OpenCV para leitura e FLTK para mostrar.

```
#include <FL/Fl.H>
#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Shared_Image.H>
#include <FL/Fl_JPEG_Image.H>
#include <FL/Fl_Box.H>
#include <cekeikon.h>

Fl_Window     *win = 0;
  Fl_Box      *box = 0;
Fl_RGB_Image  *rgb=0;
Mat_<COR> ent;
VideoCapture vi;
double rate;

void ShowNextImage_CB(void*) {
  vi >> ent;
  if (!ent.data) exit(0);
  cvtColor(ent,ent,CV_BGR2RGB);
  rgb->uncache();
  win->redraw();
  Fl::repeat_timeout(rate, ShowNextImage_CB);
}

int main(int argc, char** argv) {
  if (argc!=2) {
    printf("ShowVid: Le video com OpenCV e mostra com FLTK\n");
    printf("ShowVid video.ext\n");
    erro("Erro: Numero de argumentos invalido");
  }

  vi.open(argv[1]);
  if (!vi.isOpened()) erro("Erro: Abertura de video de entrada ",argv[1]);
  vi >> ent;
  if (!ent.data) exit(0);
  cvtColor(ent,ent,CV_BGR2RGB);
  double fps=vi.get(CV_CAP_PROP_FPS);
  if (fps<=0) fps=30;
  rate=1.0/fps;

  win = new Fl_Double_Window(ent.cols, ent.rows, "ShowVid");
  win->begin();
    rgb = new Fl_RGB_Image(ent.data, ent.cols, ent.rows, 3);
    box = new Fl_Box(0, 0, ent.cols, ent.rows);
    box->image(rgb);
  win->end();
  win->show();
  Fl::add_timeout(rate, ShowNextImage_CB);
  return Fl::run();
}
```

7) ShowCam: Mostra webcam na tela usando OpenCV para captura e FLTK para mostrar.

```
#include <FL/Fl.H>
#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Shared_Image.H>
#include <FL/Fl_Box.H>
#include <cekeikon.h>

Fl_Window     *win = 0;
  Fl_Box      *box = 0;
Fl_RGB_Image  *rgb=0;
Mat_<COR> ent;
VideoCapture vi;
double rate;

void ShowNextImage_CB(void*) {
  vi >> ent;
  cvtColor(ent,ent,CV_BGR2RGB);
  rgb->uncache();
  win->redraw();
  Fl::repeat_timeout(rate, ShowNextImage_CB);
}

int main(int argc, char** argv) {
  if (argc!=2) {
    printf("ShowCam: Captura webcam com OpenCV e mostra com FLTK\n");
    printf("ShowCam camId\n");
    erro("Erro: Numero de argumentos invalido");
  }

  int id;
  if (sscanf(argv[1],"%d",&id)!=1) erro("Erro: Numero de camId invalido");

  vi.open(id);
  if (!vi.isOpened()) erro("Erro: Abertura de camera ",argv[1]);
  vi >> ent;
  cvtColor(ent,ent,CV_BGR2RGB);
  double fps=vi.get(CV_CAP_PROP_FPS);
  if (fps<=0) fps=30;
  rate=1.0/fps;

  win = new Fl_Double_Window(ent.cols, ent.rows, "ShowCam");
  win->begin();
    rgb = new Fl_RGB_Image(ent.data, ent.cols, ent.rows, 3);
    box = new Fl_Box(0, 0, ent.cols, ent.rows);
    box->image(rgb);
  win->end();
  win->show();
  Fl::add_timeout(rate, ShowNextImage_CB);
  return Fl::run();
}
```