GTK+
Uporabniški vmesniki

Vsebina

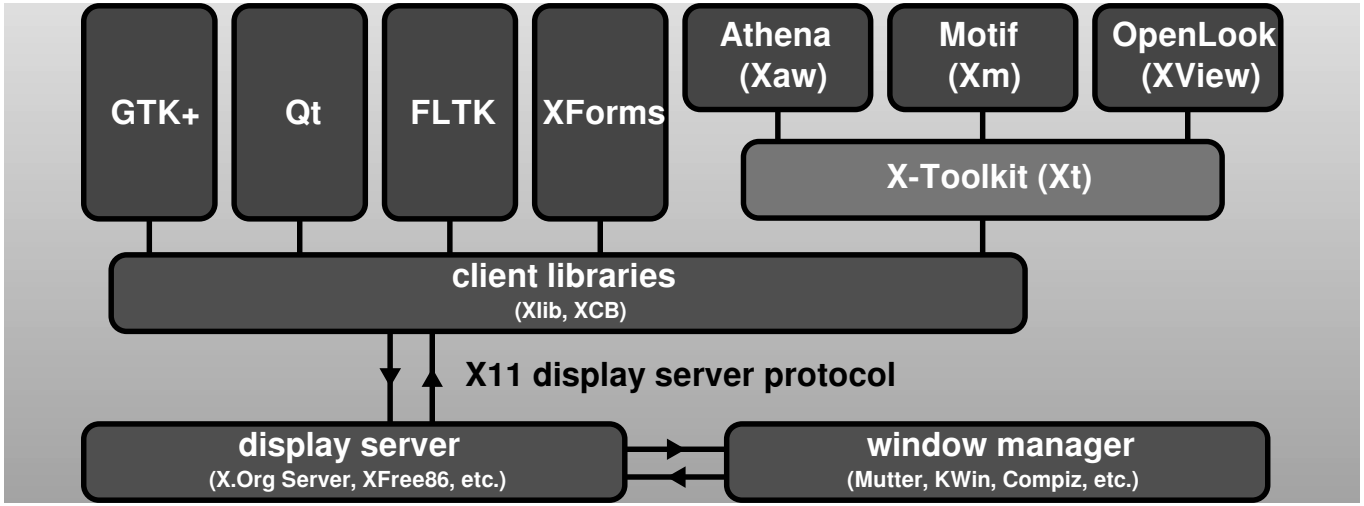
- hierarhija orodij za razvoj uporabniških vmesnikov
- knjižnice potrebne za GTK+ uporabniške vmesnike
- podatkovni tipi
- prevajanje aplikacije GTK+
- primer aplikacije GTK+

Uvod

- na Unix/Linux operacijskih sistemih je pogosto osnova grafičnega sistema okenski sistem X
- okenski sistem X ponuja osnovno funkcionalnost
- GTK+ omogoča enostavnejši razvoj aplikacij
- je prosto-dostopen in odprt sistem
- GTK+ omogoča razvoj in izvajanje na različnih platformah

Hierarhija orodij za načrtovanje uporabniških vmesnikov

- GTK+ je nadgradnja knjižnice Xlib
- omogoča pozicijsko in strojno neodvisnost
- implementirana tudi za MS Windows platforme (GTK#)
- alternativa je knjižnica Qt



Knjižnice

- GLib → GIMP Library
- GTK+ → GTK + GDK
  - GTK → GIMP ToolKit
  - GDK → GIMP Drawing Kit
- GNOME → GNU Network Object Model Environment
- GIMP → GNU Image Manipulation Program
- GNU → GNU's Not Unix (rekurzivni akronim)

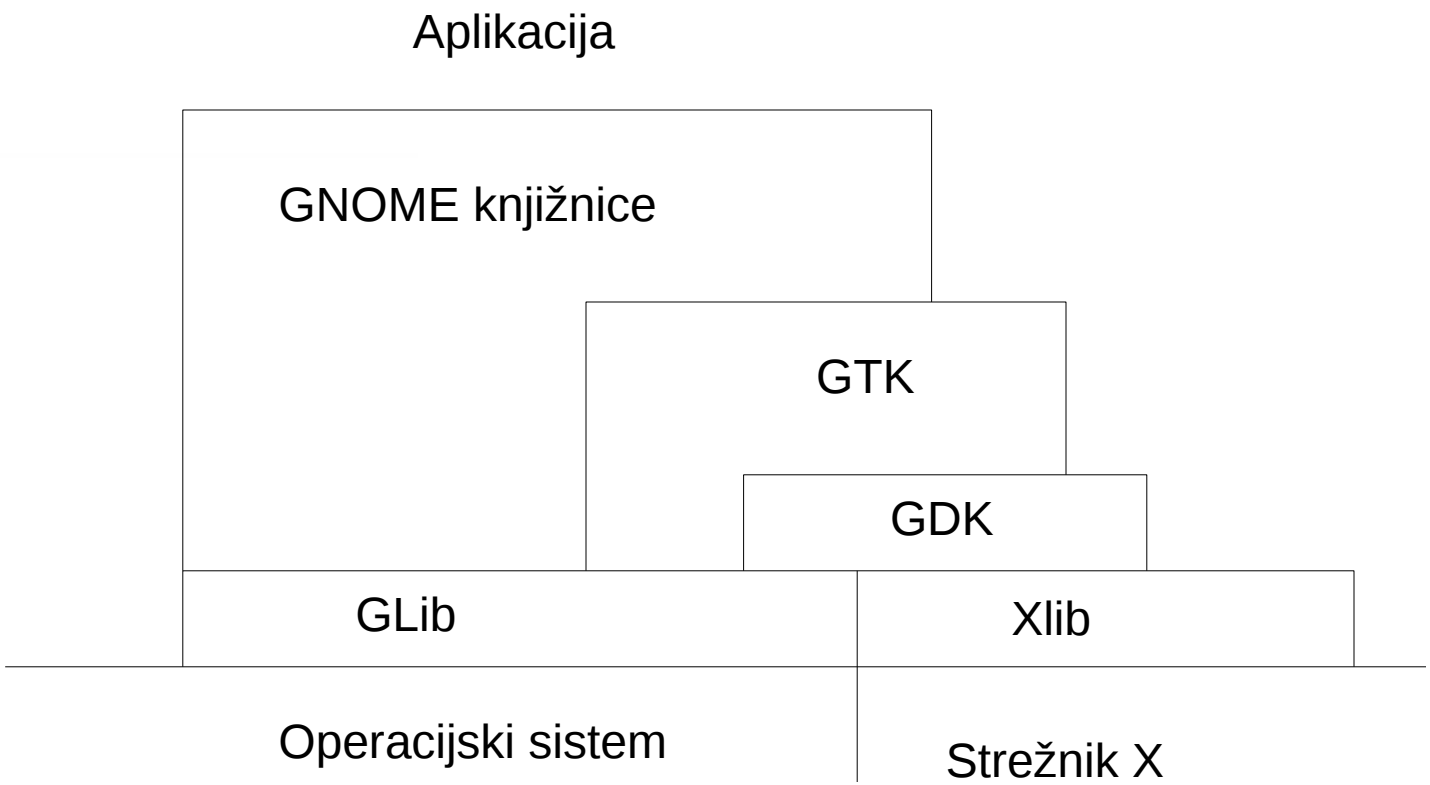
Knjižnice

- GLib
  - knjižnica C z definiranimi konstrukti in rutinami
  - novi, platformno neodvisni tipi
  - zagotavlja prenosljivost, višjenivojska uporaba C
- GTK+
  - platformno neodvisna
  - objektno usmerjena
  - razvoj aplikacij v X (načeloma tudi druge platforme)
  - GTK → osnovne podobe, vsebovalniki
  - GDK → vmesnik do Xlib, lažje delo s klici v Xlib

Knjižnice

- libGnome
  - osnovna knjižnica okenskega sistema GNOME
  - zagotavlja storitve za povezavo z drugimi knjižnicami → delo z besedili, internacionalizacija, ...
- libGnomeUI
  - knjižnica za gradnjo uporabniških vmesnikov
  - sestavljene podobe, ki nadgrajujejo GTK+

Hierarhija knjižnic za aplikacije



Podatkovni tipi v GLib

- »standardni« podatkovni tipi
  - zagotavljajo prenosljivost
  - poenostavitev kode
  - konsistentnost

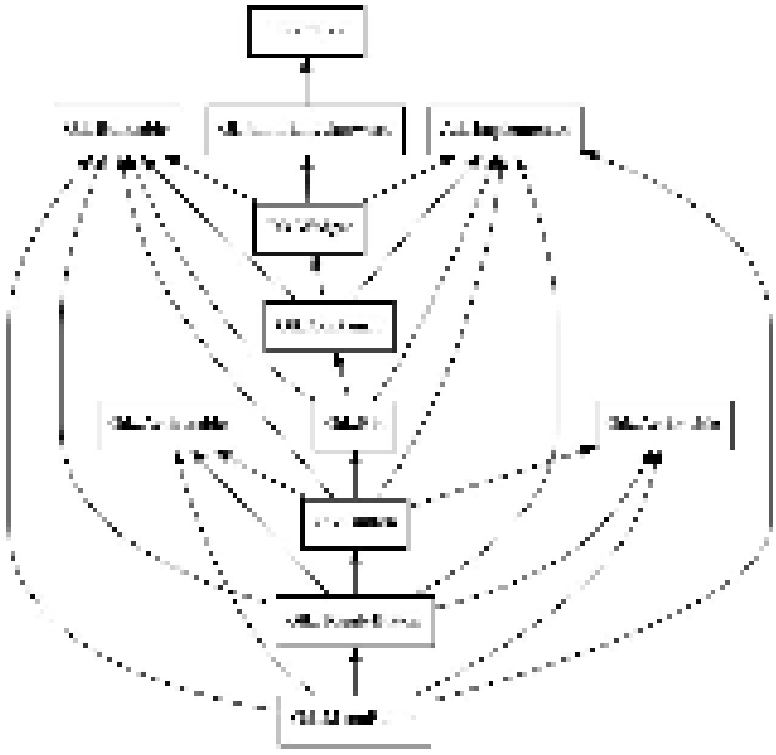
GLib	standardni C
gshort	short
glong	long
gint	int
gint8, gint16,...	8-bitni, 16-bitni int
gfloat	float
gdouble	double
gboolean	boolean
gpointer	void*

Prevajanje

- uporaba GNU c prevajalnika → gcc
  - gcc -o programcek programcek.c `pkg-config --cflags --libs gtk+-3.0`
- izvajanje
  - ./programcek
- pkg-config → vrne meta informacijo o nameščenih knjižnicah
    - pkg-config --cflags <knjižnica>
    - pkg-config --libs <knjižnica>
- skripta com_gtk je tekstovna datoteka, ki omogoča lažje in enostavnejše prevajanje

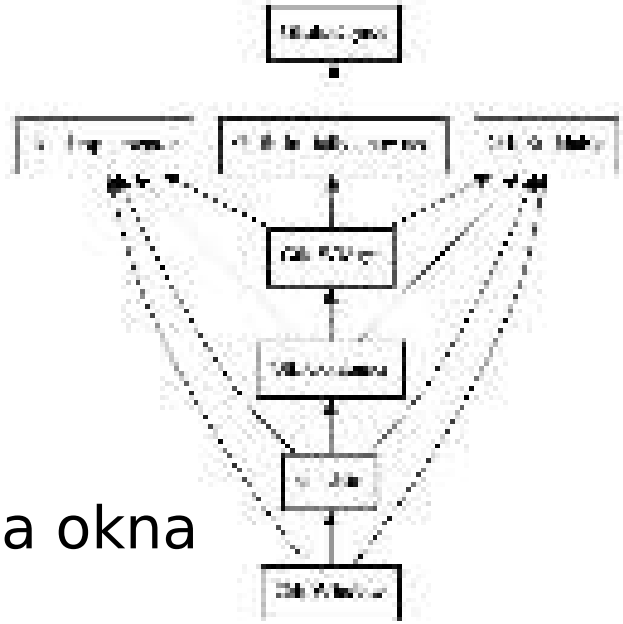
Programiranje GTK+ v C

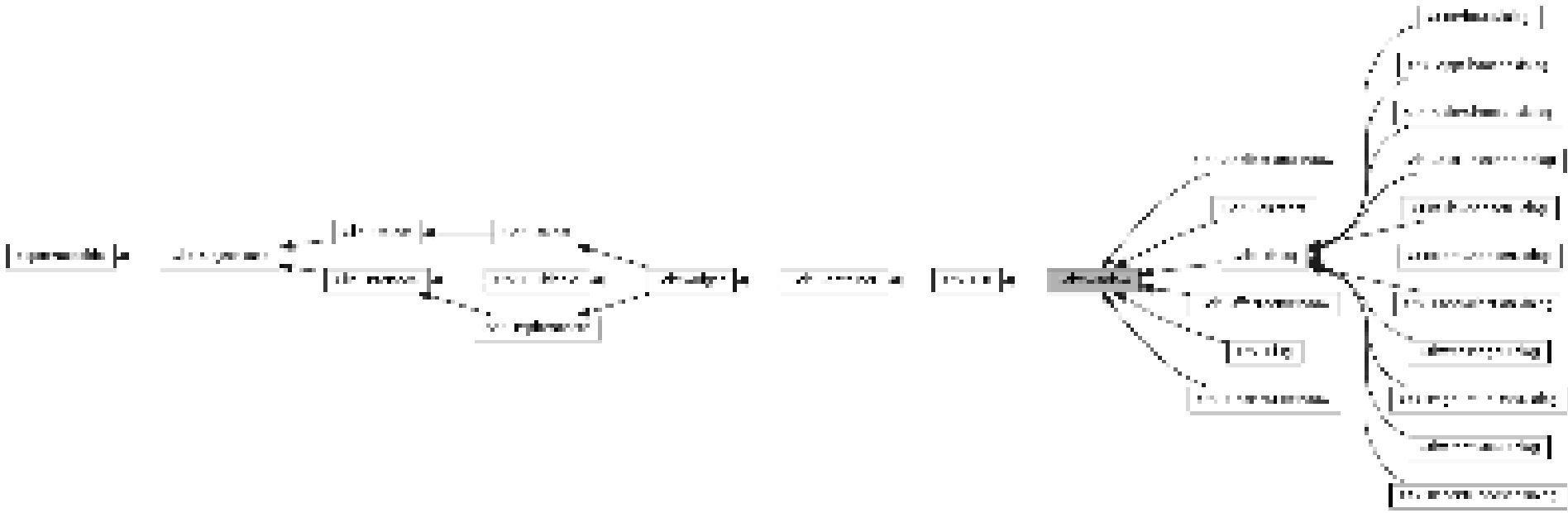
- objektno usmerjena knjižnica napisana v C
- prvenstveno za X11 in Wayland, deluje tudi na drugih platformah
- temelj vseh podobe je razred GObject
- GObject omogoča transparentno kompatibilnost med jeziki
  - je del knjižnice GLib



GTK+ okna

- osnovna komponenta je GObject
- osnovno okno je GtkWindow
- iz GtkWindow so izpeljana vsa druga okna





Primeri podob v GTK+

Oznaka:
GObject→GtkWidget→GtkMisc→GtkLabel

Polje za vnos:
GObject→GtkWidget→GtkEditable→GtkEntry

Gumb:
GObject→GtkWidget→GtkContainer→GtkBin→GtkButton

Vrstični menu:
GObject→GtkWidget→GtkContainer→GtkMenuShell→GtkMenuBar

Mrežni vsebovalnik:
GObject→GtkWidget→GtkContainer→GtkBox

Polje za risanje:
GObject→GtkWidget→GtkDrawingArea

Okno z drsniki:
GObject→GtkWidget→GtkContainer→GtkBin→GtkScrolledWindow

Dialog za izbiro datoteke:
GObject→GtkWidget→GtkContainer→GtkBin→GtkWindow→GtkFileSelection

Primer

```
#include <gtk/gtk.h>

/* Podobi doda gumb z doloceno oznako */
GtkWidget * AddButton ( GtkWidget * theWindow, const gchar * buttonText ){
   GtkWidget * button;
   button = gtk_button_new_with_label ( buttonText );
   gtk_container_add ( GTK_CONTAINER ( theWindow ), button );
   gtk_widget_show ( button );
   return button;
}

/* Odzivne funkcije */
void ButtonClicked ( GtkButton * button, gpointer data ){
   g_print ( "Hello world!\n" );
}

void StopTheApp ( GObject * object, gpointer data ){
   gtk_main_quit ();
}
```

Primer

```
gint main ( gint argc, gchar * argv[] ){
   GtkWidget * window;
   GtkWidget * button;

   gtk_init ( &argc, &argv );
                    /* Zgradi okno z ustreznimi dimenzijami */
   window = gtk_window_new ( GTK_WINDOW_TOPLEVEL );
   gtk_window_set_default_size ( GTK_WINDOW ( window ), 160, 100 );
                    /* Nastavi sirino roba na 5 pikslov */
   gtk_container_set_border_width ( GTK_CONTAINER ( window ), 5 );
                    /* Uporabi funkcijo AddButton za gradnjo novega gumba */
   button = AddButton ( window, "Pritisni za pozdrav!" );
                    /* Instaliraj signal za zaprtje aplikacije */
   g_signal_connect ( G_OBJECT ( window ),  "destroy",
                    G_CALLBACK ( StopTheApp ),   NULL );
                    /* Povezi signal gumba z odzivno funkcijo ButtonClicked */
   g_signal_connect ( G_OBJECT ( button ), "pressed",
                      G_CALLBACK ( ButtonClicked ),  NULL );
                    /* Prikazi okno */
   gtk_widget_show ( window );
                    /*  GTK+ zanka dogodkov */
   gtk_main ();
   return 0;
}
```

Signali

- signali so način za klicanje seznama poimenovanih funkcij (metod)
- ko neka podoba odda »signal« se pregledajo in kličejo vse odzivne funkcije, ki so povezane s tem signalom
- signali so lahko povezani z GTK+ podobami (»pressed«, »clicked«, »released«, ...) ali z objekti v GLib knjižnici (GObject, »destroy«)
- tip parametrov se v tem primeru razlikujejo
  - GObject oziroma GtkButton
  - lahko pa tudi posplošimo na GObject

Dogodki

```
#include <gtk/gtk.h>

/* Zaprtje aplikacija */
void CloseTheApp ( GObject * object, gpointer data ){
  gtk_main_quit ();
}

/* Odzivna funkcija za delo z dogodki */
gboolean EventHandler ( GtkWidget * widget, GdkEvent *event, gpointer data){
  /* Ugotovi vrsto dogodka */
  switch ( event->type ){
  case GDK_EXPOSE:
    g_print ("The window contents were redrawn\n"); break;
  case GDK_ENTER_NOTIFY:
    g_print ("The mouse entered the window\n"); break;
  case GDK_LEAVE_NOTIFY:
    g_print ("The mouse left the window\n"); break;
  case GDK_DELETE:
    g_print ("The user killed the window\n"); break;
  default:
    g_print ("\n"); break;
  }
  //return TRUE;
  return FALSE;
}
```

Dogodki

```
gint main ( gint argc, gchar *argv[] ){
  GtkWidget *window;

  gtk_init ( &argc, &argv );
  window = gtk_window_new ( GTK_WINDOW_TOPLEVEL );
  g_signal_connect (G_OBJECT (window), "event",
                    G_CALLBACK (EventHandler), NULL);
  g_signal_connect (G_OBJECT (window), "destroy",
                    G_CALLBACK (CloseTheApp), NULL);
  gtk_widget_show (window);

  gtk_main ();
  return 0;
}
```

Dogodki

- dogodki v GDK so strukture, ki so povezane z dogodki okenskega sistema platforme
- lahko so podobni signalom, npr. dogodek ob pritisku gumba
  - gumb sprejme dogodek okenskega sistema
  - ob sprejemu mora nekaj narediti, ponavadi je to klic notranje funkcije (v primeru gumba poskrbi za spremembo izgleda)
  - včasih to zadostuje lahko pa se kliče še odzivna funkcija za signal, ki se sproži ob dogodku

Signali in dogodki

- Signali
  - GObject→ destroy
  - GtkWidget → show, hide, draw, ...
  - GtkButton → pressed, released, ...
  - ...
  - parametri so podoba in uporabniški podatki
- Dogodki
  - event, button press event,button release event, key press event, key release event, configure event, expose event, enter notify event, delete event, ...
  - parametri so podoba, dogodek in uporabniški podatki

Signali in dogodki

```
#include <gtk/gtk.h>
      /* Zaprtje aplikacija */
void CloseTheApp ( GObject * object, gpointer data ){
      gtk_main_quit ();
}
      /* Odzivna funkcija za delo z dogodki */
gboolean EventHandler ( GtkWidget * widget, GdkEvent *event, gpointer data){
      /* Ugotovi vrsto dogodka */
  switch ( event->type ){
  case GDK_EXPOSE:
      g_print ("The window contents were redrawn\n");
      break;
  case GDK_BUTTON_PRESS:
      g_print ("Button pressed\n");
      //return TRUE;
      break;
  default:
      g_print ("\n");
      break;
  }
  return FALSE; //return TRUE;
}
/* Odzivna funkcija za signal */
void ButtonClicked ( GtkButton * button, gpointer data ){
  g_print ( "Hello world!\n" );
}
```

Signali in dogodki

```
gint main ( gint argc, gchar * argv[] ){
        GtkWidget * window;
        GtkWidget * button;

        gtk_init ( &argc, &argv );

        window = gtk_window_new ( GTK_WINDOW_TOPLEVEL );
        gtk_window_set_default_size ( GTK_WINDOW ( window ), 160, 100 );
        gtk_container_set_border_width ( GTK_CONTAINER ( window ), 5 );
        button = gtk_button_new_with_label ( "Pritisni za pozdrav!" );
        gtk_container_add ( GTK_CONTAINER ( window ), button );

        g_signal_connect (G_OBJECT ( button ), "event",
                        G_CALLBACK ( EventHandler ), NULL);
        g_signal_connect (G_OBJECT ( button ),
                        "pressed",
                        G_CALLBACK ( ButtonClicked ), NULL );
        g_signal_connect (G_OBJECT (window), "destroy",
                        G_CALLBACK (CloseTheApp), NULL);
        gtk_widget_show_all ( window );

        gtk_main ();
        return 0;
}
```

Vsebovalniki

```
GtkWidget * MakeEntryBox (){
    GtkWidget * box;
    GtkWidget * widget;
    box = gtk_box_new ( GTK_ORIENTATION_HORIZONTAL, 2 );
    widget = gtk_label_new ( "Vpisite vase  ime:" );
    gtk_label_set_justify(GTK_LABEL(widget), GTK_JUSTIFY_LEFT);
    gtk_label_set_xalign(GTK_LABEL(widget), 0.0);
    gtk_box_pack_start ( GTK_BOX (box ), widget, FALSE, TRUE, 0 );
    widget = gtk_entry_new ();
    gtk_box_pack_start ( GTK_BOX (box ), widget, FALSE, TRUE, 0 );
    gtk_box_set_homogeneous(GTK_BOX(box), TRUE);
    return box;
}
GtkWidget * MakeButtons (){
    GtkWidget * box;
    GtkWidget * button;
    box = gtk_box_new ( GTK_ORIENTATION_HORIZONTAL, 2 );
    button = gtk_button_new_with_label ( "OK" );
    gtk_box_pack_start ( GTK_BOX (box ), button, FALSE, TRUE, 0 );
    button = gtk_button_new_with_label ( "Cancel" );
    gtk_box_pack_start ( GTK_BOX (box ), button, FALSE, TRUE, 0 );
    gtk_box_set_homogeneous(GTK_BOX(box), TRUE);
    return box;
}
```

Vsebovalniki

```
#include <gtk/gtk.h>
/* Zaprtje aplikacija */
void StopTheApp ( GtkWidget * window, gpointer data ){
      gtk_main_quit ();
}
GtkWidget * MakeWindow (){
     GtkWidget * window;
     window = gtk_window_new ( GTK_WINDOW_TOPLEVEL );
     gtk_window_set_default_size ( GTK_WINDOW ( window ), 320, 80 );
     gtk_container_set_border_width ( GTK_CONTAINER ( window ), 10 );
     gtk_window_set_title ( GTK_WINDOW ( window ), "Upor. vmesnik" );
     g_signal_connect ( G_OBJECT ( window ), "destroy",
                        G_CALLBACK ( StopTheApp ),  NULL );
     return window;
}
```

```
gint main ( gint argc, gchar * argv[] ){
     GtkWidget * window;
     GtkWidget * mainbox;
     GtkWidget * entrybox;
     GtkWidget * buttonbox;
     gtk_init( &argc, &argv );
     window = MakeWindow ();
     entrybox = MakeEntryBox ();
     buttonbox = MakeButtons ();
     mainbox = gtk_box_new ( GTK_ORIENTATION_VERTICAL, 10 );
     gtk_box_pack_start ( GTK_BOX (mainbox ), entrybox, FALSE, FALSE, 0 );
     gtk_box_pack_start ( GTK_BOX (mainbox ), buttonbox, FALSE, FALSE, 0 );
     gtk_container_add ( GTK_CONTAINER ( window ), mainbox );
     gtk_widget_show_all ( window );
     gtk_main ();
     return 0;
}
```

Generator vmesnikov