Rozdział 4

Podstawowe kontrolki

Tworzenie aplikacji wymaga użycia wielu różnych kontrolek. Najczęściej używanymi są kontrolki przycisku, wpisu, listy i listy kombinowanej. Kontrolki te mają procentowo największy udział w interfejsie aplikacji, i zrozumienie ich działania jest niezbędne do opanowania bardziej skomplikowanych kontrolek. Większość kontrolek jest do siebie bardzo podobna. Wiele spośród nich to po prostu pojemniki na inne kontrolki ze specyficznymi właściwościami; kontrolki posiadają także wspólne właściwości i funkcje, przy pomocy których można nimi manipulować.

Funkcje wspólne dla wszystkich kontrolek

Wszystkie funkcje tworzące kontrolki zwracają wskaźnik do GtkWidget. Wskaźnika tego można używać w uniwersalnych funkcjach, które operują na dowolnej kontrolce. Przykładem takiej uniwersalnej funkcji może być gtk_widget_show. Funkcja ta uwidacznia kontrolkę, albo przynajmniej ustawia jej właściwości tak, że można zobaczyć ją na ekranie, o ile znajduje się wewnątrz widocznego rodzica. Istnieje wiele innych funkcji, operujących na wszystkich kontrolkach. Niektóre, jak gtk_widget_show, używane są częściej od innych.

Rzutowanie kontrolek

Ponieważ utworzona kontrolka jest kontrolką uniwersalną, musi zostać przekształcona na właściwy typ, zanim użyjemy jej w funkcji operującej na specyficznym typie kontrolki. Utworzenie przycisku zwraca wskaźnik GtkWidget, ale procedury specyficzne dla przycisku potrzebują wskaźnika GtkButton. Uniwersalny wskaźnik GtkWidget musi więc być przekształcony na GtkButton przy pomocy makra GTK_BUTTON, zanim zostanie wywołana funkcja dla przycisku. Wymaganie, aby przekazywane typy były zgodne z oczekiwanymi przez procedury, pomaga programistom w pisaniu lepszych aplikacji.

Złe rzutowanie

Tylko kiepsko napisany program przekazuje kontrolkę tekstową do funkcji, która spodziewa się kontrolki przycisku. Wymuszenie konwersji sprawia, że zostaną zgłoszone wszelkie błędy w przekształcaniu typu kontrolki. Poniżej znajduje się przykładowy fragment kodu, który tworzy kontrolkę przycisku i przekształca ją na kontrolkę tekstową:

```
/* --- tworzymy kontrolkę przycisku --- */
przycisk = gtk_button_new_with_label ("Przycisk");

/* --- próbujemy przekształcić przycisk na kontrolkę tekstową --- */
tekst = GTK_TEXT (przycisk);
```

Jeśli umieścimy powyższy fragment w kodzie programu, kompilacja przebiegnie prawidłowo, ponieważ sprawdzanie konwersji odbywa się w czasie wykonania. Jednak po uruchomieniu programu zobaczymy następujący komunikat o błędzie:

```
** WARNING **: invalid cast form "GtkButton" to "GtkText"
```

Komunikaty takie nie powinny się pojawiać. Jeśli tak się dzieje, zazwyczaj wynika to z błędu programisty.

Dobre rzutowanie

Kontrolki zazwyczaj wywodzą się od innych kontrolek. Kontrolka przycisku wywodzi się od kontrolki pojemnika (GtkContainer), która z kolei wywodzi się od kontrolki uniwersalnej (GtkWidget). Ze względu na te zależności, kontrolki przycisku (GtkButton) mogą być rzutowane na kontrolki pojemnika (GtkContainer) w celu użycia ich w funkcjach, które operują na pojemnikach. Używaliśmy już funkcji gtk_container_add, która dodawała kontrolki do pojemników. Można użyć tej funkcji, aby dodać etykietę do przycisku. Oczywiście, w celu użycia w funkcji gtk_container_add przycisk musi być rzutowany na typ GtkContainer, co jest możliwe, ponieważ przycisk jest także pojemnikiem. Kontrolka może być rzutowana na dowolną inną kontrolkę, położoną nad nią w hierarchii klas.

```
/* --- tworzymy przycisk bez etykiety --- */
przycisk = gtk_button_new ();

/* --- tworzymy etykietę --- */
etykieta = gtk_label_new ("Bla bla");

/* --- dodajemy etykietę do przycisku --- */
gtk_container_add (GTK_CONTAINER (przycisk), etykieta);
```

Każda kontrolka posiada własny zbiór sygnałów, dla których można napisać funkcje zwrotne. Liczba sygnałów, z którymi związana jest dana kontrolka, zależy od jej złożoności, ale zwykle w aplikacji potrzebna jest obsługa tylko nielicznych sygnałów. Dobrym przykładem jest tu kontrolka przycisku. Chociaż związane z nią jest wiele sygnałów, to jedynym zdarzeniem, dla którego zazwyczaj warto pisać procedurę obsługi, jest sygnał "clicked". Nie oznacza to, że nigdy nie używa się innych zdarzeń; należy raczej powiedzieć, że *zazwyczaj* ich się nie używa. Zdarzenia również mogą być dziedziczone; przyjrzymy się bliżej temu zagadnieniu, kiedy zajmiemy się kontrolkami wywiedzionymi od przycisku.

Zwykły przycisk

Zwykły przycisk (GtkButton) jest jedną z najprostszych kontrolek, ponieważ jego jedyna funkcja polega na tym, że użytkownik może na nim kliknąć. W oknach dialogowych przyciski zwykle zaopatrzone są w teksty OK albo Anuluj. Najważniejszym zdarzeniem, związanym z przyciskiem, jest "clicked". Zdarzenie "clicked" oznacza kombinację wciśnięcia i zwolnienia przycisku w pojedynczym ruchu. Zazwyczaj kliknięcie przycisku powoduje jakąś reakcję, na przykład zapisanie pliku albo zamknięcie okna dialogowego. Przykładowy przycisk przedstawiony jest na rysunku 4.1.

Przyciski pochodzą od pojemników (GtkContainer), więc dzielą z nimi wiele cech. Jedną z najważniejszych jest możliwość przechowywania innych kontrolek. Tekst wewnątrz przycisku GtkButton jest w istocie kontrolką etykiety, umieszczoną wewnątrz przycisku. Można utworzyć pasek narzędziowy, składający się z rzędu przycisków, z których każdy zawiera rysunek, obrazujący jego funkcję. To, co początkowo wydawało się dość nieciekawą kontrolką, teraz jawi się jako kontrolka bardzo elastyczna - na tyle elastyczna, że wywodzą się od niej także inne kontrolki.

Przycisk można utworzyć albo z tekstem, albo bez niego. Funkcja gtk_button_new_with_label tworzy przycisk z podpisem, natomiast funkcja gtk_button_new tworzy przycisk bez kontrolki potomnej. Funkcja ta przydaje się w sytuacjach, kiedy chcemy utworzyć przycisk zawierający rysunek. Ponieważ nie potrzebujemy etykiety, tworzymy pusty przycisk przy pomocy gtk_button_new, a następnie dodajemy rysunek do przycisku. Funkcja zwraca wskaźnik typu GtkWidget, który można przekształcić na typ GtkButton przy pomocy makra GTK_BUTTON.

Rysunek 4.1. Zwykły przycisk.

Kontrolki przycisku mogą generować różne sygnały, ale programiści zazwyczaj ignorują większość z nich, oprócz sygnału "clicked". Sygnały dla przycisku to:

0Sygnał	**0Czynność**
1pressed	1Oqyxbhrj ynrs`ġ vbhŇmhĵsx.
2Released	2Oqyxbhrj ynrs`ġ yvnkmhnmx.
3Clicked	**3**Oqyxbhrj ynrs`ġ jkhjmhĵsx. **Jest to kombinacja "pressed" i "released".**
4Enter	4Vrj`őmhj lxryx oqydrtmiġ rhĵm`c oqyxbhrj.
5leave	5Vrj`őmhj lxryx ncrtmiġ rhĵym`c oqyxbhrjt.

Zdarzenie mogą być spowodowane przez działania użytkownika albo zasymulowane przy pomocy jednej z funkcji, generujących sygnały. Funkcje sygnałowe są rzadko stosowane.

Sygnał	**0Funkcja generująca sygnał**
pressed	**gtk_button_pressed (przycisk)**
released	**gtk_button_released (przycisk)**
clicked	**gtk_button_clicked (przycisk)**
enter	**gtk_button_enter (przycisk)**
leave	**gtk_button_leave (przycisk)**

Poniższy program tworzy okno, zawierające przycisk i wyświetla na konsoli każdy występujący sygnał.

```
/*
 * przycisk.c
 *
 * Przykład ilustrujący działanie przycisku
 */

#include <gtk/gtk.h.

/*
 * Usuwanie okna
 *
```

```
 * Program kończy pracę; trzeba zakończyć działanie gtk.
 */
gint ZamknijOknoAplikacji (GtkWidget *kontrolka, gpointer *dane)
{
    gtk_main_quit();

    /* --- Kontynuujemy zamykanie --- */
    return (FALSE);
}
/*
 * przycisk_zdarzenie
 *
 * Wystąpiło zdarzenie - jego nazwa przekazywana jest
 * w parametrze "dane"
 */
void przycisk_zdarzenie (GtkWidget *kontrolka, gpointer *dane)
{
    g_print ("Zdarzenie: %s\n", dane);
}

int main (int argc, char *argv[])
{
    GtkWidget *okno
    GtkWidget *przycisk

    /* --- Inicjacja GTK --- */
    gtk_init (&argc, &argv);

    /* --- tworzymy okno najwyższego poziomu --- */
    okno = gtk_window_new (GTK_WINDOW_TOPLEVEL);

    gtk_window_set_title (GTK_WINDOW (okno), "Zwykły przycisk");

    /* Należy zawsze pamiętać o podłączeniu zdarzenia
    * delete_event do głównego okna
    */
    gtk_signal_connect (GTK_OBJECT (okno), "delete_event",
    GTK_SIGNAL_FUNC (ZamknijOknoAplikacji), NULL);

    /* --- tworzymy obramowanie okna --- */
    gtk_container_border_width (GTK_CONTAINER (okno), 50);

    /* ---------------------------- */
```

```
    /* --- Tworzymy przycisk --- */
    /* ------------------------------ */

    /* --- Tworzymy nowy przycisk --- */
    przycisk = gtk_button_new_with_label ("Przycisk");

    /* --- Wyświetlamy przycisk (nie jest jeszcze jednak widoczny) */
    gdk_widget_show (przycisk);

    /* ------------------------------------------- */
    /* --- Rejestrujemy procedury obsługi  --- */
    /* ------------------------------------------- */
    gtk_signal_connect (GTK_OBJECT (przycisk), "pressed",
                        GTK_SIGNAL_FUNC    (przycisk_zdarzenie),    "wciśnię-
cie");
    gtk_signal_connect (GTK_OBJECT (przycisk), "released",
                        GTK_SIGNAL_FUNC    (przycisk_zdarzenie),    "zwolnie-
nie");
    gtk_signal_connect (GTK_OBJECT (przycisk), "clicked",
                        GTK_SIGNAL_FUNC (przycisk_zdarzenie), "kliknięcie");
    gtk_signal_connect (GTK_OBJECT (przycisk), "enter",
                        GTK_SIGNAL_FUNC    (przycisk_zdarzenie),    "wejście
myszy");
    gtk_signal_connect (GTK_OBJECT (przycisk), "leave",
    GTK_SIGNAL_FUNC (przycisk_zdarzenie), "wyjście myszy");

    /* --- Dodajemy przycisk do okna --- */
    gtk_container_add (GTK_CONTAINER (okno), przycisk);

    /*
    * Uwidaczniamy główne okno, teraz przycisk stanie się widoczny
    */
    gtk_widget_show (okno);

    gtk_main ();
    return (0);
}
```

Przełącznik

Przełączniki (GtkToggleButton) wywodzą się od przycisków GtkButton i przypominają je wyglądem, ale zachowują się w nieco odmienny sposób. Przełączniki posiadają określony stan (włączone/wyłączone), który odzwierciedlają swoim wyglądem. Początkowo przełącznik wygląda jak

zwykły przycisk, ale po wciśnięciu pozostaje włączony. Aby powrócił do pierwotnego stanu, konieczne jest ponowne kliknięcie. Rysunek 4.2 przedstawia przełączniki w stanie włączonym i wyłączonym.

Przełączniki można tworzyć wraz z etykietą, przy pomocy funkcji gtk_toggle_button_new_with_label, albo bez etykiety, przy pomocy gtk_toggle_button_new. Ponieważ przełącznik wywodzi się od zwykłego przycisku, wszystkie zdarzenia i funkcje operujące na zwykłym przycisku mogą być stosowane także do przełącznika, jeśli użyjemy makra GTK_BUTTON w celu przekształcenia typu przełącznika.

Przełącznik posiada dodatkowy sygnał, oprócz odziedziczonych z GtkButton. Sygnał "toggled" jest wysyłany przy każdej zmianie stanu przycisku. Stan można ustawić przy pomocy funkcji gtk_toggle_button_set_state, której należy przekazać stan, w jakim powinien znaleźć się przełącznik. Funkcja gtk_toggle_button_toggled zmienia stan przełącznika na przeciwny. Funkcje zmieniające stan przełącznika mogą spowodować wysłanie sygnałów do aplikacji; jeśli na przykład funkcja gtk_toggle_button_set_state ustawia stan przełącznika na TRUE, a obecnym stanem jest FALSE, wówczas do aplikacji zostaną wysłane sygnały "clicked" oraz "toggled".

Rysunek 4.2. Przełączniki.

```
/*
 * przelacznik.c
 * Autor: Eric Harlow
 *
 * Przykład ilustrujący działanie przełącznika
 */

#include <gtk/gtk.h>

/*
 * ZamknijOknoAplikacji
 *
 * zamyka się główne okno, kończymy pracę gtk
```

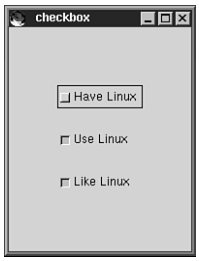
```
 */
gint ZamknijOknoAplikacji (GtkWidget *kontrolka, gpointer *dane)
{
    gtk_main_quit ();

    return (FALSE);
}

/*
 * Wystąpiło zdarzenie.
 */
void PrzyciskZdarzenie (GtkWidget *kontrolka, gpointer *dane)
{
    g_print ("Zdarzenie: %s\n", data);
}

int main (int argc, char *argv[])
{
    GtkWidget *okno;
    GtkWidget *przycisk;
    GtkWidget *ypole;

    /* --- Inicjacja GTK --- */
    gtk_init (&argc, &argv);

    /* --- Tworzymy okno najwyższego poziomu --- */
    okno = gtk_window_new (GTK_WINDOW_TOPLEVEL);

    gtk_window_set_title (GTK_WINDOW (okno), "Przełącznik");

    /* --- Należy zawsze pamiętać o podłączeniu zdarzenia delete_event
    *     do głównego okna.
    */
    gtk_signal_connect (GTK_OBJECT (okno), "delete_event",
                        GTK_SIGNAL_FUNC (ZamknijOknoAplikacji), NULL);
    /* --- Tworzymy obramowanie okna --- */
    gtk_container_border_width (GTK_CONTAINER (okno), 50);

    /* --- Tworzymy pionowe pole pakujące, które będzie przechowywać
    *     przełączniki.
    */
    ypole = gtk_vbox_new (FALSE, 0);

    /* ------------------------ */
```

```
    * --- Tworzymy przycisk --- */
    * ------------------------------ */

    /* --- Tworzymy nowy przycisk. --- */
    przycisk = gtk_toggle_button_new_with_label ("Górny przełącznik");
    /* --- Upakowujemy przycisk w pionowym polu (ypole) --- */
    gtk_box_pack_start (GTK_BOX (ypole), przycisk, FALSE, FALSE, 0);

    /* --- Uwidaczniamy przycisk --- */
    gtk_widget_show (przycisk);
    gtk_signal_connect (GTK_OBJECT (przycisk), "toggled",
                        GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "górny prze-
łączony");
    gtk_signal_connect (GTK_OBJECT (przycisk), "pressed",
                        GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "górny naci-
śnięty");
    gtk_signal_connect (GTK_OBJECT (przycisk), "released",
                        GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "górny zwol-
niony");
    gtk_signal_connect (GTK_OBJECT (przycisk), "clicked",
                        GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "górny klik-
nięty");
    gtk_signal_connect (GTK_OBJECT (przycisk), "enter",
                        GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "mysz
w górnym");
    gtk_signal_connect (GTK_OBJECT (przycisk), "leave",
    GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "mysz poza górnym");

    /* ---------------------------------------- */
    /* --- Tworzymy następny przycisk --- */
    /* ---------------------------------------- */
    przycisk = gtk_toggle_button_new_with_label ("Dolny przełącznik");

    /* --- Upakowujemy przycisk w pionowym polu (ypole) --- */
    gtk_box_pack_start (GTK_BOX (ypole), przycisk, FALSE, FALSE, 0);

    /* --- Uwidaczniamy przycisk --- */
    gtk_widget_show (przycisk);

    gtk_signal_connect (GTK_OBJECT (przycisk), "toggled",
                        GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "dolny prze-
łączony");
    gtk_signal_connect (GTK_OBJECT (przycisk), "pressed",
```

```
                    GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "dolny naci-
śnięty");
    gtk_signal_connect (GTK_OBJECT (przycisk), "released",
                    GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "dolny zwol-
niony");
    gtk_signal_connect (GTK_OBJECT (przycisk), "clicked",
                    GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "dolny klik-
nięty");
    gtk_signal_connect (GTK_OBJECT (przycisk), "enter",
                    GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "mysz
w dolnym");
    gtk_signal_connect (GTK_OBJECT (przycisk), "leave",
                    GTK_SIGNAL_FUNC (PrzyciskZdarzenie), "mysz poza
dolnym");
    /* ---------------------------------------- */
    /* --- Uwidaczniamy główne okno --- */
    /* ---------------------------------------- */
    gtk_container_add (GTK_CONTAINER (okno), ypole);
    gtk_widget_show (ypole);
    gtk_widget_show (okno);
    gtk_main ();
    return (0);
}
```

Przycisk wyboru

Przyciski wyboru (GtkCheckButton) wywodzą się od przełączników GtkToggleButton, więc ich działanie jest zbliżone. Jedyną różnicą pomiędzy GtkCheckButton i GtkToggleButton jest sposób wyświetlania kontrolki na ekranie (patrz rysunek 4.3). Obie kontrolki mogą wyświetlać opisujący je tekst, i obie przedstawiają graficznie swój bieżący stan, ale czynią to w nieco odmienny sposób. GtkToggleButton wyświetla tekst wewnątrz przycisku, a GtkCheckButton po prawej stronie niewielkiego przycisku. Oba przyciski spełniają dokładnie te same funkcje, więc wybór jednego z nich jest zwykle uzależniony od preferencji programisty.



Rysunek 4.3. Przycisk wyboru GtkCheckButton.

Przycisk GtkCheckButton można utworzyć wraz z etykietą, przy pomocy funkcji gtk_check_button_new_with_label, albo bez niej, przy pomocy gtk_check_button_new. Ponieważ GtkCheckButton wywodzi się od GtkToggleButton, wszystkie funkcje i zdarzenia stosowane z GtkToggle-Button można stosować także w przypadku GtkCheckButton.

Przykładowy program dla przełącznika można przekształcić na przykład dla przycisku wyboru, przepisując kod tworzący przyciski. Kod dla przełącznika

```
/* --- Tworzymy nowy przełącznik --- */
przycisk = gtk_toggle_button_new_with_label ("Górny przełącznik");
```

można zastąpić następującym kodem dla GtkCheckButton:

```
/* --- Tworzymy nowy przycisk wyboru --- */
przycisk = gtk_check_button_new_with_label (szEtykieta);
```

Poprzedni program będzie teraz przykładem użycia przycisku wyboru.

Przycisk opcji

Na pierwszy rzut oka przyciski opcji (GtkRadioButton) wyglądają podobnie to przyciski wyboru, GtkCheckButton. Ich wygląd sugeruje, że wywodzą się od GtkCheckButton (patrz rysunek 4.4). GtkRadioButton posiada jednak dodatkową charakterystykę, która odróżnia go od GtkCheckButton. Kliknięcie jednego z przycisków wyłącza poprzednio wybrany przycisk i powoduje zaznaczenie klikniętego przycisku.

Przyciski GtkRadioButton można utworzyć wraz z etykietą, przy pomocy funkcji gtk_radio_button_new_with_label, albo bez niej, przy pomocy gtk_radio_button_new. Utworzenie przycisku to dopiero połowa pracy, ponieważ należy go jeszcze związać z grupą, aby jednocześnie mógł być wybrany tylko jeden przycisk. Po utworzeniu każdego z przycisków

GtkRadioButton należy wywołać funkcję gtk_radio_button_group, z argumentami w postaci przycisku i grupy, aby dodać przycisk do grupy. Kiedy dodajemy pierwszy przycisk, musimy jako wskaźnika do grupy użyć NULL, ponieważ grupa jeszcze nie istnieje - w takiej sytuacji grupa zostanie utworzona (grupa jest w rzeczywistości jednokierunkową listą [GSList], przechowującą dodane do grupy przyciski).

Rysunek 4.4. Przyciski opcji.

Zaniechanie wywołania gtk_radio_button_group po dodaniu każdego przycisku opcji prowadzi do nieprzewidywalnych rezultatów; jeśli zaś grupa nie zostanie zainicjowana na NULL przed pierwszym wywołaniem gtk_radio_button_group, zazwyczaj spowoduje to wyjątek w programie.

Poniższy przykład pokazuje tworzenie i dodawanie przycisku do grupy. Po utworzeniu przycisku, pobierana jest grupa, do której ma należy przycisk.

```
GtkWidget *opcja;
GSList *grupa = NULL;

/* --- tworzymy przycisk opcji i dodajemy go do grupy --- */
/* --- jeśli grupa=NULL, tworzona jest nowa grupa --- */
opcja = gtk_radio_button_new_with_label (grupa, szEtykieta);

/* --- Pobieramy wskaźnik do grupy, do której należy przycisk, --- */
/* --- Grupa jest łączoną listą przycisków, która zmienia się, --- */
/* --- ponieważ elementy są dodawane do czoła listy. Zawsze --- */
/* --- należy pobrać grupę po dodaniu przycisku! --- */
grupa = gtk_radio_button_group (GTK_RADIO_BUTTON (opcja));
```

Ponieważ przycisk opcji wywodzi się od przycisku wyboru, który z kolei wywodzi się od przełącznika, przyciski opcji mogą korzystać ze zdarzeń i funkcji właściwych dla przełącznika, aby ustawić swoje parametry.

Zdarzenia używane przez przełącznik są używane także przez przycisk opcji.

Etykieta

Etykiety (GtkLabel) są statycznymi, nieedytowalnymi polami, używanymi zazwyczaj do opisania innych pól na ekranie. Etykiety mogą opisywać przycisk, jeśli zostaną umieszczone wewnątrz przycisku, mogą też znajdować się w pobliżu innych pól, na przykład pól edycyjnych, zaopatrując je w odpowiedni opis. Etykiety tworzy się przy pomocy funkcji gtk_label_new. Pobranie tekstu etykiety umożliwia funkcja gtk_label_get, a zmianę tekstu - funkcja gtk_label_set.

W przeciwieństwie do większości innych kontrolek, etykiety nie pochodzą od GtkWidget, ale od GtkMisc. Nie mogą więc posiadać przypisanych im zdarzeń.

Etykiety są kontrolkami „wagi lekkiej", ponieważ w czasie ich tworzenia nie jest tworzone nowe okno. Etykiety są po prostu rysowane na kontrolce macierzystej. Mechanizm ten ma tę zaletę, że GTK+ nie musi przechowywać informacji o kolejnym oknie. Wadą są natomiast ograniczone możliwości tych kontrolek.

```
GtkWidget *etykieta;
char *bufor;

/* --- tworzymy nową etykietę --- */
etykieta = gtk_label_new(_To jest etykieta_);

/* --- ustawiamy nowy tekst etykiety --- */
gtk_label_set (etykieta, _To jest nowa etykieta_);

/* --- Pobieramy tekst etykiety --- */
gtk_label_get (etykieta, &bufor);
```

Funkcja gtk_label_get pobiera referencję do tekstu, wyświetlanego przez etykietę. Nie należy bezpośrednio edytować tego łańcucha; jeśli chcemy zmodyfikować tekst etykiety, powinniśmy skorzystać z funkcji gtk_label_set. Referencja do tekstu etykiety uniemożliwia zwolnienie pamięci przydzielonej na tekst, więc należy zawsze ją usunąć, kiedy nie jest już potrzebna.

Ponieważ etykieta wywodzi się od GtkMisc, można użyć funkcji gtk_misc_set_alignment, aby zmodyfikować sposób wyświetlania tekstu wewnątrz etykiety. Funkcja przyjmuje dwie wartości, opisujące wyrów-

nanie tekstu w poziomie (x) i w pionie (y). W przypadku wyrównania x, zero (0) wskazuje, że tekst powinien być wyrównany do lewej strony, .5 wskazuje, że tekst powinien być wyśrodkowany w poziomie, a 1, że powinien być wyrównany do prawej strony. W przypadku wyrównania y, 0 wskazuje, że tekst powinien być wyrównany do góry, .5, że powinien być wyśrodkowany w pionie, a 1, że powinien być wyrównany do dołu.

```
/* --- tekst wyrównany do prawej, ale wyśrodkowany w pionie --- */
gtk_misc_set_alignment (GTK_MISC (etykieta), 1.0, .5);
```

Kontrolka wpisu

Kontrolka wpisu (GtkEntry) jest jednoliniowym polem edycyjnym, używanym do wprowadzania i wyświetlania danych tekstowych. Kontrolka wpisu wywodzi się od kontrolki edycyjnej, i jest odchudzoną i uproszczoną wersją kontrolki tekstowej. Mimo to jest dużo bardziej skomplikowana od kontrolek przycisku czy etykiety, ponieważ posiada znacznie więcej funkcji.

Rysunek 4.5. Lista kombinowana, etykieta, wpis i przycisk opcji.

Kontrolkę wpisu (GtkEntry) można utworzyć przy pomocy funkcji gtk_entry_new albo gtk_entry_new_with_max_length. Funkcja gtk_entry_new_with_max_length ustawia maksymalną liczbę znaków, którą można wprowadzić do kontrolki. Tekst w kontrolce można pobrać przy pomocy funkcji gtk_entry_get_text, zwracającej wskaźnik do danych tekstowych, których nie należy jednak modyfikować poprzez ten wskaźnik. Tekst w kontrolce można ustawić w jeden z trzech sposobów:

- Można wstawić tekst na początku pola, przy pomocy funkcji gtk_entry_prepend_text.
- Można dołączyć tekst na końcu pola, przy pomocy funkcji gtk_entry_append_text.
- Można także ustawić tekst przy pomocy funkcji gtk_entry_set_text, która usuwa poprzednią zawartość pola.

Można również sterować położeniem kursora wewnątrz pola. Bieżącą pozycję kursora (i zaznaczenia) można sprawdzić przy pomocy funkcji gtk_entry_get_region. Kursor można przesunąć w obrębie pola przy pomocy funkcji gtk_entry_set_position. Funkcja gtk_entry_select_region pozwala na zaznaczenie zakresu tekstu w kontrolce.

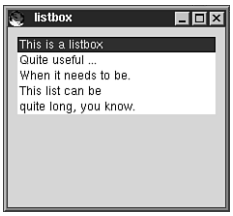
Dodatkowe właściwości kontrolki wpisu pozwalają na określenie, czy użytkownik może zmieniać tekst w kontrolce. Właściwość tę można ustawić przy pomocy funkcji gtk_entry_set_editable. Funkcja gtk_entry_set_visibility określa natomiast, czy wpisywany w polu tekst jest widoczny dla użytkownika. Zazwyczaj powinien być on widoczny, ale na przykład w polach, służących do wpisywania haseł, wyświetlanie wpisywanych przez użytkownika znaków byłoby niewłaściwe.

```
/* --- tworzymy pole wpisu --- */
wpis = gtk_entry_new ();

/* --- ustawiamy domyślną wartość --- */
gtk_entry_set_text (GTK_ENTRY (wpis), "12:00");

/* --- dodajemy jakiś tekst --- */
gtk_entry_append_text (GTK_ENTRY (wpis), " w południe");

/* --- nie pozwalamy na edycję pola --- */
gtk_entry_set_editable (GTK_ENTRY (wpis), FALSE);
```

Kontrolka wpisu może oczekiwać na wystąpienie sygnału "changed", który wskazuje, że tekst wewnątrz kontrolki uległ zmianie.

Pole listy

Pola listy (GtkList) pokazują listę elementów, z których użytkownik może wybrać jeden lub więcej elementów, w zależności od konfiguracji listy (patrz rysunek 4.6). Kontrolkę GtkList tworzymy przy pomocy funkcji gtk_list_new, natomiast dodawanie elementów do listy może przebiegać na kilka różnych sposobów. Najprostszym z nich jest użycie kombinacji gtk_list_item_new_with_labeloraz gtk_container_add. Funkcja gtk_container_add może dodawać wiele elementów do GtkList, ponieważ GtkList przeciąża funkcję dodawania z GtkContainer, umożliwiając obsługę wielu elementów w pojemniku.



Rysunek 4.6. Lista GtkList.

Poniższy fragment kodu ilustruje tworzenie listy i dodawanie do niej kilku elementów.

```
/* --- tworzymy pole listy --- */
lista = gtk_list_new ();

/* --- pozwalamy na jednoczesny wybór wielu elementów listy --- */
gtk_list_selection_mode (GTK_LIST (lista), GTK_SELECTION_MULTIPLE);

/* ------------------------------------ */
/* --- dodajemy jeden element --- */
/* ------------------------------------ */

/* --- tworzymy element --- */
element = gtk_list_item_new_with_label ("Samochód");

/* --- dodajemy go do listy --- */
gtk_container_add (GTK_CONTAINER (lista), element);

/* --- uwidaczniamy element --- */
gtk_widget_show (element);

/* ------------------------------------- */
/* --- dodajemy kolejny element --- */
/* ------------------------------------- */

/* --- tworzymy element --- */
element = gtk_list_item_new_with_label ("Dom");

/* --- dodajemy go do listy --- */
gtk_container_add (GTK_CONTAINER (lista), element);

/* --- uwidaczniamy element --- */
gtk_widget_show (element);
```

Oczywiście w sytuacjach, kiedy wielokrotnie korzystamy z powyższej procedury w celu dodania elementu do listy, najlepiej jest przesunąć odpowiedzialny za to kod do funkcji. Podejście takie upraszcza program i ułatwia jego czytanie. Możemy uprościć nasz kod w następujący sposób:

```
/* ------------------------------ */
/* --- dodajemy element --- */
/* ------------------------------ */

DodajElementListy (lista, _Samochód_);
DodajElementListy (lista, _Dom_);
```

Oto funkcja DodajElementListy:

```
void DodajElementListy (GtkWidget *lista, char *sTekst)
{
    GtkWidget *element;

    /* --- tworzymy element listy na podstawie danych --- */
    element = gtk_list_item_new_with_label (sTekst);

    /* --- dodajemy element do listy --- */
    gtk_container_add (GTK_CONTAINER (lista), element);

    /* --- uwidaczniamy element --- */
    gtk_widget_show (element);
}
```

Funkcja ta nie być może nie oszczędza zbyt wiele pracy w naszym przykładzie, ale jeśli kod dodawałby wiele więcej elementów, o wiele większe byłyby też oszczędności. Dodatkowe korzyści odnieślibyśmy, wypełniając inne kontrolki GtkList przy pomocy tej samej funkcji.

Można także skorzystać z funkcji, która dodaje do GtkList całą grupę elementów jednocześnie. Grupa ta musi mieć postać łączonej listy (GList *) kontrolek, utworzonych przy pomocy funkcji gtk_list_item_new_with_label. Można wstawić ją na koniec GtkList przy pomocy funkcji gtk_list_append_items, na początek przy pomocy gtk_list_prepend_items, albo w określonym punkcie, przy pomocy gtk_list_insert_items. W każdym przypadku, w polu danych GList muszą znajdować się kontrolki GtkListItem.

Poniższy przykład dodaje grupę kontrolek na koniec GtkList. Funkcja UtworzElement (zamieszczona niżej) służy do uproszczenia programu. Funkcja ta tworzy i uwidacznia element listy oraz zwraca wskaźnik do kontrolki, aby można było ją dodać do łączonej listy.

```
/*
 * Utwórz Element
```

```
 *
 * Twory element listy na podstawie przekazanego tekstu i zwraca
 * wskaźnik do elementu
 */
GtkWidget *UtworzElement (char *sTekst)
{
    GtkWidget *element;

    /* --- tworzymy element listy na podstawie danych --- */
    element = gtk_list_item_new_with_label (sTekst);

    /* --- uwidaczniamy element --- */
    gtk_widget_show (element);

    /* --- zwracamy wartość --- */
    return (element);
}
```

Możemy teraz wykorzystać funkcję UtworzElement, aby stworzyć element dodawany do łączonej listy. Listę tę można wypełnić przy pomocy następującego kodu:

```
elementy = NULL;
elementy = g_list_append (elementy, UtworzElement ("różowy"));
elementy = g_list_append (elementy, UtworzElement ("niebieski"));
elementy = g_list_append (elementy, UtworzElement ("czerwony"));
elementy = g_list_append (elementy, UtworzElement ("żółty"));
```

Po wypełnieniu łączonej listy, można dołączyć ją na koniec GtkList w następujący sposób:

```
gtk_list_append_items (GTK_LIST (lista), elementy);
```

Można także wstawić elementy na początku GtkList, wywołując funkcję:

```
gtk_list_prepend_items (GTK_LIST (lista), elementy);
```

Kontrolka GtkList może zostać opróżniona przy pomocy funkcji gtk_list_clear_items. Aby wyczyścić listę, trzeba znać indeks pierwszego i ostatniego elementu. Całą listę można wyczyścić przy pomocy

```
gtk_list_clear_items (lista, 0, -1);
```

ponieważ -1 oznacza ostatni element listy. Linia ta usuwa wszystkie elementy z GtkList.

Do zaznaczania elementów służy funkcja gtk_list_select_item, a do ich odznaczania - funkcja gtk_list_unselect_item. Element, który należy zazna-

czyć/odznaczyć, określamy przy pomocy przekazywanego do funkcji indeksu. Funkcje gtk_list_select_child i gtk_list_unselect_child przeprowadzają podobne operacje, z tym, że zamiast indeksu przyjmują kontrolkę GtkListItem.

```
/* --- zaznaczamy potomka w polu listy --- */
gtk_list_select_child (lista, potomek);

/* --- odznaczamy potomka w polu listy --- */
gtk_list_usselect_child (lista, potomek);
```

GtkList może pozwalać na jednoczesne zaznaczanie wielu elementów, albo tylko jednego, poprzez odpowiednie ustawienie trybu GtkList. Służy do tego funkcja gtk_list_set_selection_mode. Najczęściej używanymi trybami są GTK_SELECTION_SINGLE, który pozwala na zaznaczanie tylko jednego elementu GtkList, oraz GTK_SELECTION_MULTIPLE, który pozwala na jednoczesne zaznaczanie wielu elementów.

W przypadku GtkList najczęściej wykorzystywanym sygnałem jest "selection_changed". Sygnał ten jest generowany za każdym razem, kiedy zmienia się zaznaczenie, zarówno podczas zaznaczania kolejnego elementu, jak i odznaczania elementu. Ponieważ elementy listy GtkList także są kontrolkami, mogą również otrzymywać sygnały, w tym sygnał "select", który wskazuje, że element został zaznaczony. Czasem wykorzystanie tego sygnału jest wygodne, ponieważ poszczególne elementy listy mogą przekazywać dodatkowe informacje do procedury obsługi. Funkcję DodajElementListy można zmodyfikować tak, aby wraz z tekstem przyjmowała pewien kod elementu, który będzie przekazywany do procedury obsługi zdarzenia.

```
/*
 * Dodaj Element Listy
 *
 * Dodaje tekst do listy
 */
void DodajElementListy (GtkWidget *lista, char *sTekst, char *sKod)
{
    GtkWidget *element;

    /* --- tworzymy element listy na podstawie danych --- */
    element = gtk_list_item_new_with_label (sTekst);

    /* --- dodajemy element do listy --- */
    gtk_container_add (GTK_CONTAINER (lista), element);

    /* --- ustawiamy procedurę obsługi sygnału. Zauważmy, że --- */
```

```
    /* --- będzie do niej przekazywany kod elementu. --- */
    gtk_signal_connect (GTK_OBJECT (element), "select",
    GTK_SIGNAL_FUNC (wybrano_element), sKod);

}

/*
 * wybrano_element
 *
 * Funkcja ta zostanie wywołana po zaznaczeniu elementu na liście.
 * Kod elementu zostanie przekazany jako parametr "dane"
 */
void wybrano_element (GtkWidget *kontrolka, gpointer *dane)
{
    g_print ("kod wybranego elementu - %s\n", (char *) dane);
}
```

Listy kombinowane

Lista kombinowana (GtkCombo) jest połączeniem pola edycyjnego i pola listy (patrz rysunek 4.5). Pole edycyjne może pozwalać na ręczne wprowadzenie wartości, bądź też może być ograniczone do wartości zawartych w rozwijanej części GtkCombo. Kontrolkę GtkCombo można utworzyć przy pomocy funkcji gtk_combo_new i wypełnić przy pomocy gtk_combo_set_popdown_strings.

```
/* ---------------------------------------------------- */
/* --- najpierw tworzymy listę elementów --- */
/* ---------------------------------------------------- */

lk_elementy = NULL;
lk_elementy = g_list_append (lk_elementy, "Samochód");
lk_elementy = g_list_append (lk_elementy, "Dom");
lk_elementy = g_list_append (lk_elementy, "Praca");
lk_elementy = g_list_append (lk_elementy, "Komputer");

/* --- tworzymy listę kombinowaną --- */
combo = gtk_combo_new ();

/* --- tworzymy rozwijaną część listy kombinowanej --- */
gtk_combo_set_popdown_strings (GTK_COMBO (combo), lk_elementy);
```

GtkCombo można ograniczyć do przyjmowania tylko tych wartości, które znajdują się na rozwijanej liście, przy pomocy gtk_combo_set_value_in_list, ale nie jest to najlepsze rozwiązanie. Jeśli chcemy, aby użytkownik nie mógł wpisywać danych w części edycyjnej GtkCombo, możemy ustawić ją jako „tylko do odczytu" - wówczas nie będzie można wprowadzać do niej znaków, ale użytkownik wciąż będzie mógł wybrać wartość z rozwijanej części. Aplikacja może uzyskać dostęp do pola edycyjnego poprzez pobranie kontrolki wpisu, będącej częścią GtkCombo, w identyczny sposób, jak w przypadku standardowej kontrolki wpisu. Poniższy kod zapobiega wpisywaniu znaków w polu edycyjnym:

```
/* --- pobieramy część edycyjną kombinowanej listy --- */
wpis = GTK_ENTRY (GTK_COMBO (combo)->entry);

/* --- nie pozwalamy na edycję --- */
gtk_entry_set_editable (wpis, FALSE);
```

Pole edycyjne w GtkCombo może przechwytywać sygnał "changed", który wskazuje, że zawartość pola edycyjnego uległa zmianie.

```
gtk_signal_connect (GTK_OBJECT (GTK_COMBO (combo)->entry),
                    "changed",
                    GTK_SIGNAL_FUNC (funkcja_combo),
                    NULL);
```

Kontrolka GtkCombo nie posiada żadnej funkcji do czyszczenia zawartości, ale zawiera kontrolkę GtkList, która przechowuje elementy rozwijanej listy. Możemy skorzystać z tego faktu, aby manipulować elementami GtkCombo. Wyczyszczenie rozwijanej listy polega po prostu na pobraniu wewnętrznego pola listy i skorzystaniu z funkcji gtk_list_clear_items.

```
GtkList *lista;

/* --- pobieramy listę z listy kombinowanej --- */
lista = GTK_LIST (GTK_COMBO (combo)->list);

/* --- czyścimy listę wewnątrz listy kombinowanej --- */
gtk_list_clear_items (lista, 0, -1);
```

Możemy także wyczyścić listę pojedynczą instrukcją:

```
gtk_list_clear_items (GTK_LIST (GTK_COMBO (combo)->list), 0, -1);
```

Przycisk menu

Przyciski menu (GtkOptionMenu) przypominają GtkList bez możliwości edycji (patrz rysunek 4.5). Chociaż ich działanie jest podobne, to wygląd jest zupełnie odmienny. Przyciski menu są czasem używane w sposób zbliżony do GtkCombo.

Po kliknięciu przycisku GtkOptionMenu pojawia się menu, przedstawiające użytkownikowi dopuszczalne opcje. Wybranie którejś opcji powoduje, że po zniknięciu menu pojawia się ona na przycisku.

Należy najpierw utworzyć GtkOptionMenu przy pomocy funkcji gtk_option_menu_new. Następnie trzeba utworzyć kontrolki GtkRadioMenuItem, dodać je do grupy (tak, jak przyciski opcji), a następnie dodać je do GtkMenu. Po wypełnieniu GtkMenu można związać je z GtkOptionMenu.

Inicjacja GtkOptionMenu wygląda mniej więcej w ten sposób:

```
/* --- tworzymy przycisk menu --- */
pmenu = gtk_option_menu_new ();

/* --- tworzymy menu --- */
menu = gtk_menu_new ();

/* --- na razie nie ma żadnej grupy --- */
grupa = NULL;
```

Każdy element dodawany do przycisku menu musi przejść przez następujący proces:

```
/* --- wyświetlana wartość --- */
sTekst = "Średni";

/* --- tworzymy element menu z etykietą --- */
elmenu = gtk_radio_menu_item_new_with_label (grupa, sTekst);

/* --- pobieramy grupę, w której jest element menu --- */
grupa = gtk_radio_menu_item_group (GTK_RADIO_MENU_ITEM (elmenu));

/* --- dodajemy element do menu --- */
gtk_menu_append (GTK_MENU (menu), elmenu);

/* --- uwidaczniamy element --- */
gtk_widget_show (elmenu);

/* --- trzeba powiadamiać o wybraniu elementu i przekazywać --- */
/* --- tekst do funkcji zwrotnej --- */
```

```
gtk_signal_connect_object (GTK_OBJECT (elmenu),
                           "activate",
                           GTK_SIGNAL_FUNC (wybrano_el_menu),
                           (gpointer) sTekst);
```

Sygnał "activate" wskazuje, że wybrano kontrolkę GtkRadioMenuItem. Po dodaniu wszystkich elementów do menu, można związać je z przyciskiem menu w taki oto sposób:

```
/* --- wiążemy menu z przyciskiem menu --- */
gtk_option_menu_set_menu (GTK_OPTION_MENU (pmenu), menu);

/* --- uwidaczniamy przycisk menu --- */
gtk_widget_show (pmenu);
```

GtkOptionMenu przydaje się w sytuacji, kiedy chcemy dać użytkownikowi listę opcji do wyboru i wyświetlać aktualnie wybraną opcję.

Pojemniki

Pojemniki (GtkContainer) są kontrolkami, które mogą być rodzicami dla innych kontrolek, umieszczanych w kontrolce pojemnika. Nie można stworzyć kontrolki pojemnika jako takiej, ponieważ nie istnieje funkcja gtk_container_new, ale kontrolek pojemnika używa się w programach aż nadto często. Przykładami kontrolek pojemnika mogą być główne okna z wcześniejszych przykładów, przyciski i pola list. Głównym zastosowaniem kontrolki pojemnika jest udostępnienie uniwersalnych funkcji, których można używać do operacji na wszelkich typach kontrolek pojemnika, bez potrzeby pisania specyficznego zbioru funkcji dla każdej kontrolki.

Kontrolki można dodawać do pojemników przy pomocy funkcji gtk_container_add i usuwać przy pomocy gtk_container_remove. Większość kontrolek pojemnika pozwala na dodanie do nich tylko jednej kontrolki. Istnieje kilka wyjątków, na przykład GtkList, ale pojemniki te służą specjalnie do przechowywania wielu elementów. Jeśli zwykły pojemnik musi zawierać więcej niż jedną kontrolkę, wówczas trzeba skorzystać z pól albo tabel pakujących.

Iterację listy kontrolek potomnych, przechowywanych w pojemniku, umożliwia funkcja gtk_container_children, która zwraca łączoną listę (GList *) kontrolek potomnych. Można także wykorzystać funkcję gtk_container_foreach, która wywołuje określoną funkcję dla każdej z kontrolek potomnych i przekazuje do niej tę kontrolkę jako parametr.

Funkcja gtk_container_border_width kontroluje szerokość obramowania wokół kontrolki, przechowywanej w pojemniku. Ustawienie szerokości obramowania na 0 powoduje, że kontrolka całkowicie wypełnia pojemnik.

Podsumowanie

Podczas tworzenia kontrolek konieczne jest ustawienie ich właściwości, zanim zostaną wyświetlone na ekranie. Kontrolki posiadają rozmaite funkcje, które ustawiają ich atrybuty, a większość z nich sygnalizuje zdarzenia, które mogą zostać przechwycone przez aplikację. W rozdziale opisano etykiety, przyciski, pola list, pola wpisu i listy kombinowane. Są to kontrolki najczęściej używane w aplikacjach, a uzmysłowienie sobie ich działania jest konieczne do zrozumienia bardziej skomplikowanych kontrolek.