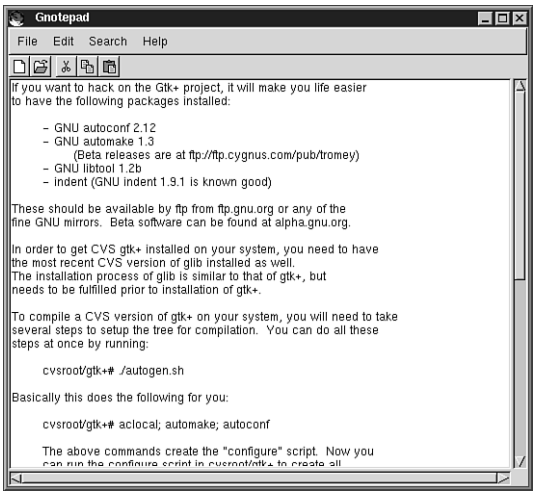
Rozdział 8

Tworzenie prostego edytora tekstu

W rozdziale tym stworzymy niewielki edytor tekstu, przypominający Notatnik z Windows, korzystając z wiedzy wyniesionej z lektury poprzednich rozdziałów. Oprzemy się na projekcie istniejącej aplikacji i spróbujemy naśladować jej funkcje, ponieważ może okazać się to łatwiejsze, niż projektowanie programu od podstaw. Notatnik Windows pozwala użytkownikom na edycję i zapisywanie prostych plików tekstowych, ale jego możliwości nie są imponujące. Napisanie zbliżonego programu będzie jednak dobrym ćwiczeniem w tworzeniu aplikacji przy użyciu kontrolek.

Aplikacja notatnika (patrz rysunek 8.1) ilustruje wiele zagadnień związanych z używaniem różnych kontrolek, jak menu, paski narzędziowe, okna dialogowe i obszar służący do edycji danych. Jest luźno oparta na programie Notatnik, dostarczanym wraz z Microsoft Windows. Można przystosować ją do swoich potrzeb, albo używać tak, jak jest, do szybkiej edycji niewielkich plików.

Menu najwyższego poziomu składa się z pozycji „Plik", „Edycja", „Szukaj" i „Pomoc". Menu „Plik" zawiera pozycję „Nowy", która tworzy nowe dokumenty, pozycję „Otwórz", która otwiera istniejące elementy, pozycję „Zapisz", która zapisuje bieżący dokument, pozycję „Zapisz jako...", która zapisuje bieżący dokument pod nową nazwą, oraz pozycję „Zakończ", która kończy pracę aplikacji. Menu „Edycja" zawiera polecenia operujące na schowku: „Wytnij", „Kopiuj" i „Wklej". Menu „Szukaj" zawiera tylko pozycję „Znajdź", a menu „Pomoc" - pozycję „O programie...", która wyświetla informacje o autorze i wersji programu.



Rysunek 8.1. Gnotatnik.

Oprócz menu aplikacja posiada także pasek narzędziowy, aby użytkownicy mieli szybki dostęp do większości poleceń. Oryginalny Notatnik nie posiada paska narzędziowego, ale dodanie go nie jest trudne, a zwiększy funkcjonalność programu.

main.c

Program zaczyna się podobnie, jak większość aplikacji GTK+ - tworzy okno i nadaje mu tytuł, a następnie tworzy pionowe pole pakujące, w którym zostanie umieszczone menu, pasek narzędziowy, oraz kontrolka tekstowa służąca do przeglądania i edycji tekstu. Kod tworzący menu i pasek narzędziowy umieścimy w funkcji UtworzMenu, a samo pole edycyjne zostanie utworzone w funkcji UtworzTekst.

W aplikacji pojawia się pewien problem, związany z tworzeniem paska narzędziowego. Utworzenie ikon paska wymaga przekształcenia tekstowych danych piksmapy na GdkPixmap, aby można było zamienić je w kontrolkę piksmapy. Funkcja konwertująca dane tekstowe wymaga podania okna X Window jako jednego z parametrów. Zamiast uwidaczniać okno (co spowodowałoby utworzenie okna X Window), a następnie ładować dane dla paska narzędziowego i wyświetlać pasek, można sprawić, aby okno było niewidoczne w trakcie konfigurowania, przy pomocy funkcji gtk_realize_widget. Funkcja ta tworzy okno X Window dla kontrolki, której można używać bez wyświetlania okna. Zazwyczaj „realizowanie" kontrolki odbywa się w trakcie działania funkcji gtk_widget_show, ale tutaj chcemy utworzyć okno X Windows dla aplikacji *zanim* wyświetlimy okno aplikacji. Można zrezygnować z użycia funkcji

gtk_realize_widget i przesunąć wywołanie funkcji UtworzMenu za funkcję uwidaczniającą okno, ale nie byłoby to najlepsze rozwiązanie, ponieważ wyświetlone okno nie byłoby jeszcze gotowe (nie zawierałoby menu i paska narzędziowego).

```
/*
 * --- main
 *
 * tutaj zaczyna się program
 */
int main(int argc, char *argv[])
{
   GtkWidget *okno;
   GtkWidget *ypole;

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

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

   /* --- tytuł i obramowanie --- */
   gtk_window_set_title (GTK_WINDOW (okno), "Gnotatnik");
   gtk_container_border_width (GTK_CONTAINER (okno), 0);

   /* --- sprawdzamy, czy główne okno nie jest zamykane --- */
   gtk_signal_connect (GTK_OBJECT(okno), "delete_event",
                       GTK_SIGNAL_FUNC (ZamknijOknoApl),
                       NULL);

   /* --- ustawiamy rozmiar okna --- */
   gtk_widget_set_usize (GTK_WIDGET(okno), 200, 200);

    /* --- tworzymy pionowe pole pakujące dla kontrolek --- */
   ypole = gtk_vbox_new (FALSE, 1);

   /* --- szerokość obramowania = 1 --- */
   gtk_container_border_width (GTK_CONTAINER(ypole), 1);

   /* --- Dodajemy pionowe pole pakujące do okna --- */
   gtk_container_add (GTK_CONTAINER(okno), ypole);

   /* --- Uwidaczniamy pole pakujące --- */
   gtk_widget_show (ypole);
```

```
    /* --- Wiążemy okno z oknem X, aby móc utworzyć piksmapy --- */
    gtk_widget_realize (okno);

    /* --- Tworzymy menu, pasek narzędziowy i kontrolkę tekstu --- */
    UtworzMenu (okno, ypole);
    UtworzTekst (okno, ypole);

    /* --- uwidaczniamy okno najwyższego poziomu --- */
    gtk_widget_show (okno);

    /* --- Pętla przetwarzania zdarzeń --- */
    gtk_main();

       return(0);
}
```

menu.c

Kod definiujący ikony paska narzędziowego i tworzący menu znajduje się w tym pliku. Duża część funkcji jest taka sama, jak w przykładowym interfejsie, który napisaliśmy wcześniej.

Pierwszy blok kodu w menu.c definiuje bitmapy dla paska narzędziowego. Powinny one być łatwo rozpoznawalne i jasno wyrażać swoje funkcje. Zazwyczaj powinniśmy ograniczyć liczbę używanych kolorów do najwyżej czterech. Zbyt wiele kolorów zaciemnia przeznaczenie bitmapy i utrudnia rozpoznanie jej funkcji. Ponieważ wiele aplikacji używa typowych ikon do przeprowadzania podobnych operacji, powinniśmy trzymać się standardów - nic tak nie mąci użytkownikom w głowie, jak zbiór zupełnie nowych ikon. Tutaj wykorzystamy ponownie ikony z wcześniejszych przykładów.

Funkcja UtworzMenu, wywoływana z main, konfiguruje menu i tablicę skrótów klawiszowych, a na koniec wywołuje funkcję UtworzPasekNarzedziowy, żeby - jak łatwo się domyślić - utworzyć pasek narzędziowy. Funkcje zwrotne dla elementów menu posiadają w nazwie przedrostek menu, aby odróżniały się od zwykłych funkcji. Na przykład funkcja menu_Nowy będzie wywoływana wtedy, kiedy użytkownik wybierze opcję menu Plik/Nowy.

```
/*
 * UtworzMenu
 *
 * Tworzy menu / pasek narzędziowy związane z głównym oknem
 */
void UtworzMenu (GtkWidget *okno, GtkWidget *ypole)
```

```
{
  GtkWidget *pasekmenu;
  GtkWidget *menu;
  GtkWidget *elmenu;

  glowne_okno = okno;

  /* --- tworzymy tablicę skrótów --- */
  grupa_skrotow = gtk_accel_group_new ();
  gtk_accel_group_attach (grupa_skrotow, GTK_OBJECT (okno));

  /* --- pasek menu --- */
  pasekmenu = gtk_menu_bar_new ();
  gtk_box_pack_start (GTK_BOX (ypole), pasekmenu, FALSE, TRUE, 0);
  gtk_widget_show (pasekmenu);

  /* --------------------
    --- menu Plik ---
    ------------------ */
  menu = UtworzPodmenuPaska (pasekmenu, "Plik");

  elmenu = UtworzElementMenu (menu, "Nowy", "^N",
                        "Tworzy nowy plik",
                        GTK_SIGNAL_FUNC (menu_Nowy), "nowy");

  elmenu = UtworzElementMenu (menu, "Otwórz", "^O",
                       "Otwiera istniejący plik",
                        GTK_SIGNAL_FUNC (menu_Otworz), "otwórz");

  elmenu = UtworzElementMenu (menu, "Importuj RTF", "",
                       "Importuje plik RTF",
                        GTK_SIGNAL_FUNC (menu_ImportujRTF), "importuj rtf");

  elmenu = UtworzElementMenu (menu, "Zapisz", "^S",
                       "Zapisuje bieżący plik",
                       GTK_SIGNAL_FUNC (menu_Zapisz), "zapisz");

  elmenu = UtworzElementMenu (menu, "Zapisz jako...", "",
                       "Zapisuje bieżący plik pod nową nazwą",
                       GTK_SIGNAL_FUNC (menu_ZapiszJako),  "zapisz jako");

  elmenu = UtworzElementMenu (menu, NULL, NULL,
                       NULL, NULL, NULL);
```

```
    elmenu = UtworzElementMenu (menu, "Zakończ", "",
                        "Czy istnieje bardziej wymowna opcja?",
                        GTK_SIGNAL_FUNC (menu_Zakoncz), "zakoncz");

    /* ----------------------
      --- menu Edycja ---
      ------------------- */
    menu = UtworzPodmenuPaska (pasekmenu, "Edycja");

    elmenu = UtworzElementMenu (menu, "Wytnij", "^X",
                         "Usuwa znaki i umieszcza je w schowku",
                          GTK_SIGNAL_FUNC (menu_Wytnij), "wytnij");

    elmenu = UtworzElementMenu (menu, "Kopiuj", "^C",
                         "Umieszcza kopię znaków w schowku",
                          GTK_SIGNAL_FUNC (menu_Kopiuj), "kopiuj");

    elmenu = UtworzElementMenu (menu, "Wklej", "^V",
                         "Wkleja znaki",
                          GTK_SIGNAL_FUNC (menu_Wklej), "wklej");

    /* ----------------------
      --- Menu Szukaj ---
      ------------------- */
    menu = UtworzPodmenuPaska (pasekmenu, "Szukaj");

    elmenu = UtworzElementMenu (menu, "Znajdź", "^F",
                          "Znajduje znaki ",
                          GTK_SIGNAL_FUNC (menu_Znajdz), "znajdz");

    /* ----------------------
      --- Menu Pomoc ---
      ------------------- */
    menu = UtworzPodmenuPaska (pasekmenu, "Pomoc");

    elmenu = UtworzElementMenu (menu, "O programie", "",
                "Informacje o programie",
                GTK_SIGNAL_FUNC (menu_OProgramie),
                          "o programie");

    /* --- tworzymy pasek narzędziowy --- */
    UtworzPasekNarzedziowy (ypole);
}
```

W końcowej części funkcji UtworzMenu wywoływana jest funkcja UtworzPasekNarzedziowy, która tworzy pasek narzędziowy. Większość bitmap paska narzędziowego weźmiemy z wcześniejszego przykładu, ilustrującego tworzenie graficznego interfejsu użytkownika. Pasek narzędziowy będzie zawierał zwykłe przyciski, do których przypiszemy te same funkcje, co do elementów menu. Na przykład przycisk „Nowy" będzie po kliknięciu wywoływał funkcję menu_Nowy, czyli tę samą, którą wywołuje opcja menu Plik/Nowy.

```
/*
 * UtworzPasekNarzedziowy
 *
 * Tworzy pasek narzędziowy, zawierający często używane polecenia.
 */
void UtworzPasekNarzedziowy (GtkWidget *ypole)
{

  /* --- Tworzymy pasek i dodajemy go do okna --- */
  pasek = gtk_toolbar_new (GTK_ORIENTATION_HORIZONTAL,
                           GTK_TOOLBAR_ICONS);
  gtk_box_pack_start (GTK_BOX (ypole), pasek, FALSE, TRUE, 0);
  gtk_widget_show (pasek);

  /*                        --------
   * --- Tworzymy przycisk | nowy |
   *                        --------
   */
  gtk_toolbar_append_item (GTK_TOOLBAR (pasek),
                           NULL, "Nowy plik", NULL,
          UtworzKontrolkeZXpm (glowne_okno, (gchar **) xpm_nowy),
                           (GtkSignalFunc) menu_Nowy, NULL);

  /*                        ----------
   * --- Tworzymy przycisk | otwórz |
   *                        ----------
   */
  gtk_toolbar_append_item (GTK_TOOLBAR (pasek),
                           "Otwórz plik ", "Otwórz plik", "",
          UtworzKontrolkeZXpm (glowne_okno, (gchar **) xpm_otworz),
                           (GtkSignalFunc) menu_Otworz, NULL);

  /* --- Mały odstęp --- */
  gtk_toolbar_append_space (GTK_TOOLBAR (pasek));
```

```
    /*                          ----------
     * --- Tworzymy przycisk | wytnij |
     *                          ----------
     */
    gtk_toolbar_append_item (GTK_TOOLBAR (pasek),
                             "Wytnij", "Wytnij", "",
                UtworzKontrolkeZXpm (glowne_okno, (gchar **) xpm_wytnij),
                             (GtkSignalFunc) menu_Wytnij, NULL);

    /*                          ----------
     * --- Tworzymy przycisk | kopiuj |
     *                          ----------
     */
    gtk_toolbar_append_item (GTK_TOOLBAR (pasek),
                             "Kopiuj", "Kopiuj", "",
                UtworzKontrolkeZXpm (glowne_okno, (gchar **) xpm_kopiuj),
                             (GtkSignalFunc) menu_Kopiuj, NULL);

     /*                          ---------
     * --- Tworzymy przycisk | wklej |
     *                          ---------
     */
    gtk_toolbar_append_item (GTK_TOOLBAR (pasek),
                             "Wklej", "Wklej", "",
                UtworzKontrolkeZXpm (glowne_okno, (gchar **) xpm_wklej),
                             (GtkSignalFunc) menu_Wklej, NULL);

}
```

rozne.c

Plik ten zawiera funkcje niższego poziomu, wywoływane z funkcji UtworzMenu i UtworzPasekNarzedziowy. Zaczerpnięty jest z poprzedniego przykładu, ilustrującego użycie menu i pasków narzędziowych i nie ma w nim żadnych nowych elementów. Ponownie wykorzystujemy ten sam kod do tworzenia bitmap, menu i podmenu.

komunikat.c

W kilku częściach aplikacji konieczne jest wyświetlenie komunikatu dla użytkownika. Aby nie powielać tego samego kodu, napiszemy funkcję PokazKomunikat, która będzie wyświetlała okno dialogowe. Funkcja ta wywoła gtk_grab_add, aby upewnić się, że komunikat nie umknie uwadze użytkownika.

Funkcja gtk_grab_add uniemożliwi użytkownikowi wykonanie jakiejkolwiek czynności, zanim nie zamknie on okna dialogowego. Po kliknięciu przycisku OK blokada zostanie zwolniona, dzięki wywołaniu funkcji gtk_grab_remove z funkcji zwrotnej dla sygnału destroy. Umieszczamy odpowiedni kod w funkcji zwrotnej dla sygnału destroy, ponieważ okno dialogowe może zostać zamknięte bez naciskania przycisku OK czy Anuluj.

Funkcja PokazKomunikat przyjmuje tytuł i komunikat, nadaje tytuł oknu dialogowemu i umieszcza w nim komunikat oraz przycisk OK.

```
/*
 * PokazKomunikat
 *
 * Wyświetla komunikat w oknie dialogowym
 */
void PokazKomunikat (char *szTytul, char *szKomunikat)
{
   GtkWidget *etykieta;
   GtkWidget *przycisk;
   GtkWidget *okno_dialogowe;

   /* --- tworzymy okno dialogowe --- */
   okno_dialogowe = gtk_dialog_new ();

   gtk_signal_connect (GTK_OBJECT (okno_dialogowe), "destroy",
                       GTK_SIGNAL_FUNC (UsunOknoKomunikatu),
         NULL);

   /* --- ustawiamy tytuł i obramowanie --- */
   gtk_window_set_title (GTK_WINDOW (okno_dialogowe), szTytul);
   gtk_container_border_width (GTK_CONTAINER (okno_dialogowe), 0);

   /* --- tworzymy przycisk OK z ogniskiem --- */
   przycisk = gtk_button_new_with_label ("OK");

   gtk_signal_connect (GTK_OBJECT (przycisk), "clicked",
         GTK_SIGNAL_FUNC (ZamknijOknoKomunikatu),
         okno_dialogowe);
```

```
    /* --- Ustawiamy przycisk OK jako domyślny --- */
    GTK_WIDGET_SET_FLAGS (przycisk, GTK_CAN_DEFAULT);
    gtk_box_pack_start (GTK_BOX (
                GTK_DIALOG (okno_dialogowe)->action_area),
                przycisk, TRUE, TRUE, 0);
    gtk_widget_grab_default (przycisk);
    gtk_widget_show (przycisk);

    /* --- tworzymy etykietę na komunikat --- */
    etykieta = gtk_label_new (szKomunikat);

    /* --- zostawiamy trochę miejsca wokół etykiety --- */
    gtk_misc_set_padding (GTK_MISC (etykieta), 10, 10);

    /* --- Dodajemy etykietę do odpowiedniego --- */
    /* --- obszaru okna dialogowego --- */
    gtk_box_pack_start (GTK_BOX (GTK_DIALOG (okno_dialogowe)->vbox),
                etykieta, TRUE, TRUE, 0);

    /* --- Uwidaczniamy etykietę --- */
    gtk_widget_show (etykieta);

    /* --- Wyświetlamy okno dialogowe --- */
    gtk_widget_show (okno_dialogowe);

    /* --- Można wykonywać czynności tylko w tym oknie --- */
    gtk_grab_add (okno_dialogowe);
}
```

Aby funkcja PokazKomunikat działała bezbłędnie, konieczne jest obsłużenie dwóch sytuacji przy pomocy odpowiednich sygnałów. Użytkownik może kliknąć przycisk OK, ale może także po prostu zamknąć okno dialogowe. Kliknięcie przycisku powoduje wygenerowanie sygnału clicked, ale nie robi nic poza tym. Zamknięcie okna powoduje wygenerowanie sygnału destroy, ale nie bierze pod uwagę faktu, że wywołano funkcję gtk_grab_add. Aby można było zamknąć okno po kliknięciu przycisku OK, w momencie jego tworzenia przypisaliśmy do sygnału clicked funkcję ZamknijOknoKomunikatu. Funkcja ta musi wywołać funkcję gtk_grab_remove i usunąć okno dialogowe. Zauważmy, że kontrolka okna dialogowego jest przekazywana do tej funkcji jako parametr danych, co ustawiliśmy podczas konfigurowaniu sygnału clicked w gtk_signal_connect.

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

```
 * Procedura zamykająca okno dialogowe z komunikatem.
 */
void ZamknijOknoKomunikatu (GtkWidget *kontrolka, gpointer dane)
{
   GtkWidget *kontrolka_dialogowa = (GtkWidget *) dane;

   /* --- usuwamy kontrolkę okna dialogowego --- */
   gtk_widget_destroy (kontrolka_dialogowa);
}
```

Jeśli użytkownik zamknie okno bez kliknięcia przycisku OK, wówczas okno zostanie usunięte z ekranu, ale dopiero po wysłaniu sygnału destroy. Funkcja UsunOknoKomunikatu ma przywrócić aplikację do stanu sprzed wyświetlenia okna dialogowego; tutaj wystarczy wywołanie funkcji gtk_grab_remove.

```
/*
 * UsunOknoKomunikatu
 *
 * Zwalnia blokadę okna
 */
void UsunOknoKomunikatu (GtkWidget *kontrolka, gpointer dane)
{
   gtk_grab_remove (kontrolka);
}
```

oprogramie.c

Możemy używać funkcji PokazKomunikat do tworzenia prostych okien dialogowych z wnętrza aplikacji. Na przykład opcja menu „O programie” powoduje wyświetlenie informacji o aplikacji w oknie dialogowym. Umieszcza się w nim zazwyczaj nazwę firmy, nazwisko autora i numer wersji programu. Cały kod dla opcji „O programie” składa się z wywołania funkcji PokazKomunikat, która wyświetli odpowiedni komunikat.

```
/*
 * PokazOProgramie
 *
 * Pokazuje okno dialogowe "O programie".
 * Wykorzystuje napisany już kod
 */
void PokazOProgramie ()
{
```

```
    PokazKomunikat ("O programie...",
                    "Gnotatnik v.07\n - "
                    "Eric Harlow\n");

}
```

wyborpliku.c

Okno dialogowe wyboru plików jest wyświetlane przez procedury wczytujące i zapisujące pliki. Tytuł okna dialogowego jest przekazywany do funkcji PobierzNazwePliku, aby przypominał użytkownikowi o rodzaju wykonywanej operacji. Przekazywana jest także funkcja, aby okno wyboru pliku miało uniwersalne zastosowanie. Jeśli użytkownik kliknie przycisk OK, wówczas funkcja ta zostanie wywołana wraz z wybraną nazwą pliku. Technika ta umożliwia wykorzystanie tego samego kodu wyboru pliku do wczytywania i zapisywania plików.

Podobnie jak poprzednio, funkcja gtk_grab_add uniemożliwia użytkownikowi wykonywanie innych czynności, zanim nie zamknie on okna dialogowego. Przechwytując sygnał destroy, możemy umieścić odpowiedzialny za czyszczenie kod w jednym miejscu.

Kod wybierający plik korzysta ze struktury przydzielonej przez funkcję PobierzNazwePliku, która jest przekazywana do wszystkich funkcji zwrotnych, związanych z oknem dialogowym. Kod ten zawiera informacje o oknie dialogowym oraz o działaniu, które należy podjąć, jeśli użytkownik wciśnie przycisk OK. Struktura wygląda następująco:

```
typedef struct {

   void (*funkcja) (gchar *);
   GtkWidget *wybpliku;

} typDaneWyboruPliku;

/*
 * PobierzNazwePliku
 *
 * Pokazuje okno wyboru pliku i jeśli użytkownik kliknie OK, wywołuje
 * funkcję wraz z nazwą wybranego pliku.
 */
void PobierzNazwePliku (char *sTytul, void (*fzwrotna) (char *))
{
   GtkWidget *kplik = NULL;
   typDaneWyboruPliku *dane;
```

```
    /* --- tworzymy nową kontrolkę wyboru pliku --- */
    kplik = gtk_file_selection_new (sTytul);

    dane = g_malloc (sizeof (typDaneWyboruPliku));
    dane->funkcja = fzwrotna;
    dane->wybpliku = kplik;

    gtk_signal_connect (GTK_OBJECT (kplik), "destroy",
                (GtkSignalFunc) usun, dane);

    /* --- podłączamy sygnał do przycisku OK --- */
    gtk_signal_connect (
                        GTK_OBJECT (GTK_FILE_SELECTION (kplik)->ok_button),
                        "clicked", (GtkSignalFunc) PlikOk, dane);

      /* --- podłączamy sygnał do przycisku Anuluj --- */
    gtk_signal_connect_object (
                        GTK_OBJECT          (GTK_FILE_SELECTION          (kplik)-
>cancel_button),
                        "clicked", (GtkSignalFunc) gtk_widget_destroy,
                        (gpointer) kplik);

      if (sNazwaPliku) {

      /* --- ustawiamy domyślną nazwę pliku --- */
      gtk_file_selection_set_filename (GTK_FILE_SELECTION (kplik),
                                       sNazwaPliku);
    }

      /* --- wyświetlamy okno dialogowe --- */
    gtk_widget_show (kplik);

    /* --- przechwytujemy ognisko --- */
    gtk_grab_add (kplik);
}
```

Jeśli użytkownik kliknie OK, wówczas wywoływana jest funkcja przekazana do PobierzNazwePliku wraz z nazwą pliku z okna dialogowego. W przypadku naszej aplikacji będzie to funkcja służąca albo do otworzenia, albo do zapisania pliku. Kod korzysta także ze statycznej zmiennej (sNazwaPliku), w której przechowuje nazwę pliku. Jeśli użytkownik chce otworzyć plik, albo korzysta z opcji „Zapisz jako...", aby zapisać plik, wówczas nazwa pliku jest umieszczana w oknie dialogowym, gdzie użytkownik będzie miał możliwość ją zmienić.

Funkcja PlikOK jest wywoływana po kliknięciu przycisku OK. Funkcja otrzymuje nazwę pliku z kontrolki, wywołuje kolejną funkcję, która wykona właściwą operację, a następnie usuwa okno dialogowe. Funkcja wykonująca operację jest tą samą, którą przekazaliśmy jako parametr funkcji PobierzNazwePliku. Funkcja jest przechowywana w globalnej zmiennej. Jest to możliwe, ponieważ w danym momencie może być widoczne tylko jedno okno dialogowe - gwarantuje to funkcja gtk_grab_add. Nie możemy otworzyć kolejnego okna wyboru pliku, nie zamykając uprzednio pierwszego.

```
/*
 * PlikOK
 *
 * Kliknięto przycisk "OK"
 * Wywołujemy funkcję (funkcja) aby przeprowadzić żądaną
 * operację na pliku.
 */
void PlikOk (GtkWidget *k, gpointer dane)
{
   char *sTmczNazwa;
   typDaneWyboruPliku *danelokalne;
   GtkWidget *kplik;

    danelokalne = (typDaneWyboruPliku *) data;
   kplik = danelokalne->filesel;

   /* --- Jaki plik? --- */
   sTmczNazwa = gtk_file_selection_get_filename (GTK_FILE_SELECTION
(kplik));

   /* --- Zwalniamy niepotrzebną pamięć --- */
   if (sNazwaPliku) g_free (sNazwaPliku);

   /* --- Powielamy łańcuch --- */
   sNazwaPliku = g_strdup (sTmczNazwa);

   /* --- Wywołujemy funkcję, która przeprowadzi --- */
   /* --- właściwą operację --- */
   (*(danelokalne->funkcja)) (sNazwaPliku);

   /* --- Zamykamy okno dialogowe --- */
   gtk_widget_destroy (kplik);
}
```

Funkcja usun wywołuje gtk_grab_remove. Funkcję tego typu spotkamy w każdym programie, posługującym się oknami dialogowymi.

```
/*
 * usun
 *
 * Funkcja obsługująca usuwanie okna dialogowego. Musimy zwolnić
 * przechwycone ognisko.
 */
static void usun (GtkWidget *kontrolka, gpointer *dane)
{
   /* --- Zwalniamy ognisko --- */
   gtk_grab_remove (kontrolka);

   g_free (dane);
}
```

notatnik.c

Funkcje w notatnik.c obsługują interakcje z kontrolką tekstu. Znajdują się tutaj funkcje służące do załadowania pliku do kontrolki i zapisania tekstu, oraz funkcje wycinające i wklejające tekst.

Funkcja UtworzTekst jest wywoływana z main.c, aby utworzyć kontrolkę tekstu i skonfigurować ją do wykorzystania w edytorze. Kontrolka znajduje się w tablicy pakującej 2 x 2, wraz z poziomym i pionowym paskiem przewijania. Wskaźnik do kontrolki tekstu jest przechowywany w statycznej zmiennej, aby funkcje z tego modułu miały dostęp do kontrolki. Bezpośredni dostęp do kontrolki spoza tego modułu jest niedozowolony.

```
/*
 * UtworzTekst
 *
 * Tworzy kontrolkę tekstu dla edytora
 *
 */
void UtworzTekst (GtkWidget *okno, GtkWidget *pojemnik)
{
   GtkWidget *tabela;
   GtkWidget *xpasek;
   GtkWidget *ypasek;

   /* --- tworzymy tabelę, która będzie przechowywać kontrolkę
```

```
    /* --- tekstu i paski przewijania --- */
    tabela = gtk_table_new (2, 2, FALSE);

    /* --- Dodajemy tabelę do pojemnika --- */
    gtk_container_add (GTK_CONTAINER (pojemnik), tabela);

    /* --- Nie stosujemy odstępów, aby paski przewijania
          wyglądały jak część kontrolki tekstu --- */
    gtk_table_set_row_spacing (GTK_TABLE (tabela), 0, 2);
    gtk_table_set_col_spacing (GTK_TABLE (tabela), 0, 2);

    /* --- uwidaczniamy tabelę --- */
    gtk_widget_show (tabela);

    /* --- tworzymy kontrolkę tekstu --- */
    tekst = gtk_text_new (NULL, NULL);

    /* --- pozwalamy na jej edycję --- */
    gtk_text_set_editable (GTK_TEXT (tekst), TRUE);

    /* --- wstawiamy kontrolkę tekstu do tabeli --- */
    gtk_table_attach (GTK_TABLE (tabela), tekst, 0, 1, 0, 1,
          GTK_EXPAND | GTK_SHRINK | GTK_FILL,
          GTK_EXPAND | GTK_SHRINK | GTK_FILL, 0, 0);

    /* --- uwidaczniamy ją --- */
    gtk_widget_show (tekst);

    /* --- dodajemy poziomy pasek przewijania --- */
    xpasek = gtk_hscrollbar_new (GTK_TEXT (tekst)->hadj);
    gtk_table_attach (GTK_TABLE (tabela), xpasek, 0, 1, 1, 2,
                      GTK_EXPAND | GTK_FILL | GTK_SHRINK, GTK_FILL, 0,
0);
    gtk_widget_show (xpasek);

    /* --- dodajemy pionowy pasek przewijania --- */
    ypasek = gtk_vscrollbar_new (GTK_TEXT (tekst)->vadj);
    gtk_table_attach (GTK_TABLE (tabela), ypasek, 1, 2, 0, 1,
                      GTK_FILL, GTK_EXPAND | GTK_SHRINK | GTK_FILL, 0,
0);
    gtk_widget_show (ypasek);

}
```

Wycinanie, kopiowanie i wklejanie

Kontrolka tekstu automatycznie obsługuje polecenia operujące na schowku, „Wytnij", „Kopiuj" i „Wklej", ale nie zmuszamy użytkownika, aby przeprowadzał te czynności za pomocą klawiszy (Ctrl+X, Ctrl+C, Ctrl+V). Opcje menu i przyciski paska narzędziowego powinny wydawać się równie zintegrowane, co skróty klawiszowe. Odwzorujemy je więc na funkcje, które przeprowadzają te same czynności, co polecenia wydawane z klawiatury. Na przykład opcja menu „Wytnij" będzie wywoływała funkcję gtk_editable_cut_clipboard, aby usunąć tekst z kontrolki i umieścić go w schowku.

```
/*
 * WytnijTekst
 *
 * Wycina zaznaczony tekst z kontrolki i umieszcza go w schowku
 */
void WytnijTekst (GtkWidget *kontrolka, gpointer dane)
{
   gtk_editable_cut_clipboard (GTK_EDITABLE (tekst));
}
```

Polecenia „Kopiuj" i „Wklej" są równie łatwe w implementacji:

```
/*
 * KopiujTekst
 *
 * Kopiuje zaznaczony tekst z kontrolki i umieszcza go w schowku
 */
void KopiujTekst (GtkWidget *kontrolka, gpointer dane)
{
   gtk_editable_copy_clipboard (GTK_EDITABLE (tekst));
}

/*
 * WklejTekst
 *
 * Wkleja tekst ze schowka do kontrolki
 */
void WklejTekst (GtkWidget *kontrolka, gpointer dane)
{
   gtk_editable_paste_clipboard (GTK_EDITABLE (tekst));
}
```

Czyszczenie kontrolki

Edytor obsługuje już operacje wycinania, kopiowania i wklejania. Musi także wykonywać kilka innych poleceń; opcja menu Plik/Nowy powinna usunąć tekst z kontrolki. Operację tę również można przeprowadzić w jednym kroku:

```
/*
 * WyczyscTekst
 *
 * Usuwa cały tekst z kontrolki
 */
void WyczyscTekst (GtkWidget *kontrolka, gpointer dane)
{
   gtk_editable_delete_text (GTK_EDITABLE (tekst), 0, -1);
}
```

Otwieranie plików

Wczytywanie pliku do kontrolki wymaga skorzystania z okna wyboru pliku. Funkcja OtworzPlik jest przekazywana do funkcji PobierzNazwePliku i wywoływana wtedy, kiedy użytkownik kliknie przycisk OK.

```
/*
 * menu_Otworz (main.c)
 * Wyświetlamy okno dialogowe, aby użytkownik mógł otworzyć plik
 */
void menu_Otworz (GtkWidget *kontrolka, gpointer dane)
{
   PobierzNazwePliku ("Otwórz", OtworzPlik);
}
```

Funkcja OtworzPlik wczytuje plik do kontrolki tekstu, wyświetlając stan operacji za pomocą paska postępów. Korzysta z trzech funkcji operujących na pasku postępu. Funkcja ZacznijPostep tworzy pasek postępów, UaktualnijPostep pokazuje aktualny postęp w ładowaniu pliku, a ZakonczPostep zamyka okno z paskiem postępów. Aby kontrolka nie traciła czasu na rysowanie ładowanego do niej tekstu, korzystamy z funkcji gtk_text_freeze. Plik jest wczytywany niewielkimi blokami, długości BUF_SIZE. Można ustawić tę stałą na większą wartość, aby zwiększyć szybkość wczytywania plików. Tutaj ustawiliśmy ją na małą wartość, ponieważ w innym przypadku pliki byłyby ładowane tak szybko, że pasek postępów nawet nie pojawiłby się na ekranie.

```
/*
 * OtworzPlik
```

```
 *
 * sNazwaPliku - plik do wczytania
 *
 * Wczytuje plik i umieszcza go w kontrolce tekstu
 */
void OtworzPlik (char *sNazwaPliku)
{
   char bufor[BUF_SIZE];
   int lznakow;
   FILE *otw_plik;
   struct stat stanPliku;
   long dlugPliku = 0;

   /* --- Zamrażamy kontrolkę tekstu --- */
   gtk_text_freeze (GTK_TEXT (tekst));

   /* --- Opróżniamy kontrolkę --- */
   gtk_editable_delete_text (GTK_EDITABLE (tekst), 0, -1);

   /* --- Pobieramy informację o pliku --- */
   stat (sNazwaPliku, &stanPliku);
   dlugPliku = stanPliku.st_size;

   ZacznijPostep ();

   /* --- Otwieramy plik --- */
   otw_plik = fopen (sNazwaPliku, "r");

    /* --- Jeśli udało się otworzyć plik... --- */
   if (otw_plik) {

         /* --- Odczytujemy blok--- */
         while ((lznakow = fread (bufor, 1, BUF_SIZE, otw_plik)) > 0) {

       /* --- Uaktualniamy pasek postępów --- */
       UaktualnijPostep (ftell (otw_plik), dlugPliku);

       /* --- Wstawiamy tekst --- */
       gtk_text_insert (GTK_TEXT (tekst), NULL, NULL,
                        NULL, bufor, lznakow);

            /* --- Czy to już koniec pliku? --- */
       if (lznakow < BUF_SIZE) break;
```

```
    }

    /* --- Zamykamy plik  --- */
    fclose (otw_plik);
  }
  ZakonczPostep ();

    /* --- Odmrażamy kontrolkę tekstu - teraz się przerysuje --- */
    gtk_text_thaw (GTK_TEXT (tekst));
}
```

Zapisywanie pliku

Zapisywanie pliku przypomina jego wczytywanie, ale można zapisać cały plik jako jeden, duży blok pamięci. Blok ten, pobierany z kontrolki tekstu przy pomocy funkcji gtk_editable_get_chars, powinien zostać zwolniony funkcją g_free.

```
/*
 * ZapiszPlik
 *
 * sNazwaPliku - nazwa zapisywanego pliku
 *
 * Funkcja zapisuje plik
 */

void ZapiszPlik (char *sNazwaPliku)
{
  FILE *zap_plik;
  char *bufor;
  int lznakow;

  gtk_text_freeze (GTK_TEXT (tekst));

  /* --- otwieramy plik --- */
  zap_plik = fopen (sNazwaPliku, "w");

    if (zap_plik) {
    /* --- pobieramy łańcuch z kontrolki --- */
    bufor = gtk_editable_get_chars (
                        GTK_EDITABLE (tekst),
                        (gint) 0,
                        (gint) gtk_text_get_length (GTK_TEXT (tekst)));
```

```
        /* --- zapisujemy bufor na dysku --- */
        lznakow = fwrite (bufor, sizeof (char),
                    strlen (bufor), zap_plik);

        /* --- zamykamy plik --- */
        fclose (zap_plik);

        if (lznakow != strlen (bufor)) {

          PokazKomunikat ("Zapisz",
                    "Błąd: Nie można zapisać pliku.");
        }

         /* --- Zwalniamy pamięć --- */
        g_free (bufor);

      } else {

        PokazKomunikat ("Zapisz", "Błąd: Nie można zapisać pliku.");
      }

      gtk_text_thaw (GTK_TEXT (tekst));
    }
```

Szukanie tekstu

Funkcję wyszukiwania zaimplementujemy za pomocą niemodalnego okna dialogowego. Okno „O programie” i okno wyboru pliku nie pozwalają na dostęp do aplikacji, dopóki są otwarte, natomiast okno wyszukiwania pozwala na otwieranie innych okien. Jednakże w danym momencie można korzystać tylko z jednej kopii tego okna.

Opcja menu „Znajdź” jest odwzorowana na funkcję menu_Znajdz, która tworzy okno dialogowe i pozwala użytkownikowi na szukanie łańcucha, od bieżącej pozycji kursora aż do końca pliku. Dwie funkcje związane z oknem dialogowym odpowiadają jego dwóm przyciskom. Jeśli użytkownik kliknie przycisk Znajdź następny, wówczas wywoływana jest funkcja FunkcjaZnajdz, która szuka w kontrolce określonego łańcucha. Jeśli zaś użytkownik kliknie przycisk Anuluj, wówczas wywoływana jest funkcja FunkcjaAnuluj, która zamyka okno dialogowe.

```
/*
 * menu_Znajdz
 *
```

```
 * Znajduje łańcuch w edytorze
 */
void menu_Znajdz (GtkWidget *kontrolka, gpointer dane)
{
   OknoDialogoweZnajdz ("Znajdź", FunkcjaZnajdz, FunkcjaAnuluj);
}
```

Funkcja OknoDialogoweZnajdz tworzy okno dialogowe i przypisuje funkcje FunkcjaZnajdz i FunkcjaAnuluj do przycisków Znajdź następny i Anuluj. Funkcja sprawdza globalną zmienną (okno_dialogowe), zanim utworzy okno dialogowe, aby nie otwierać drugiej kopii okna.

```
/*
 * OknoDialogoweZnajdz
 *
 * Funkcja wyświetlająca okno dialogowe "Znajdź"
 */
void OknoDialogoweZnajdz (char *szKomunikat, void (*FunkcjaTak)(),
                          void (*FunkcjaNie)())
{
   GtkWidget *etykieta;
   GtkWidget *przycisk;
   GtkWidget *xpole;

   /* --- wracamy, jeśli okno dialogowe jest już otwarte --- */
   if (okno_dialogowe) return;

   /* --- tworzymy okno dialogowe --- */
   okno_dialogowe = gtk_dialog_new ();

   gtk_signal_connect (GTK_OBJECT (okno_dialogowe), "destroy",
               GTK_SIGNAL_FUNC (ZamknijOknoZnajdz),
               okno_dialogowe);

   /* --- ustawiamy tytuł --- */
   gtk_window_set_title (GTK_WINDOW (okno_dialogowe), "Znajdź");

   /* --- Dodajemy niewielkie obramowanie --- */
   gtk_container_border_width (GTK_CONTAINER (okno_dialogowe), 5);

   /*
    * --- tworzymy komunikat
    */
```

```
    xpole = gtk_hbox_new (TRUE, TRUE);

    /* --- tworzymy etykietę z komunikatem --- */
    etykieta = gtk_label_new ("Znajdź:");
    gtk_widget_show (etykieta);

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

    /* --- Jeśli już czegoś szukaliśmy... --- */
    if (szIgla) {

       /* --- Ustawiamy wpis na ostatnio szukany łańcuch --- */
       gtk_entry_set_text (GTK_ENTRY (wpis), szIgla);
    }

    gtk_box_pack_start (GTK_BOX (xpole),
                  etykieta, TRUE, TRUE, 0);

    gtk_box_pack_start (GTK_BOX (xpole),
                  wpis, TRUE, TRUE, 0);
    gtk_widget_show (xpole);

    /* --- Dodajemy pole pakujące do pola dialogowego --- */
    gtk_box_pack_start (GTK_BOX (GTK_DIALOG (okno_dialogowe)->vbox),
                    xpole, TRUE, TRUE, 0);

  /* --- Tworzymy przycisk "Znajdź następny" --- */
    przycisk = gtk_button_new_with_label ("Znajdź następny");

    gtk_signal_connect (GTK_OBJECT (przycisk), "clicked",
                    GTK_SIGNAL_FUNC (FunkcjaTak),
                    okno_dialogowe);

    /* --- Dodajemy przycisk do okna dialogowego --- */
    gtk_box_pack_start (
                GTK_BOX (GTK_DIALOG (okno_dialogowe)->action_area),
                  przycisk, TRUE, TRUE, 0);

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

    /*
```

```
    * --- Przycisk "Anuluj"
    */

   /* --- tworzymy przycisk "Anuluj" --- */
   przycisk = gtk_button_new_with_label ("Anuluj");

   gtk_signal_connect (GTK_OBJECT (przycisk), "clicked",
                       GTK_SIGNAL_FUNC (FunkcjaNie),
                       okno_dialogowe);

   /* --- Pozwalamy, aby przycisk "Anuluj" był domyślnym --- */
   GTK_WIDGET_SET_FLAGS (przycisk, GTK_CAN_DEFAULT);

   /* --- Dodajemy przycisk "Anuluj" do okna dialogowego --- */
   gtk_box_pack_start (
              GTK_BOX (GTK_DIALOG (okno_dialogowe)->action_area),
                przycisk, TRUE, TRUE, 0);

   /* --- Czynimy przycisk "Anuluj" domyślnym --- */
   gtk_widget_grab_default (przycisk);

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

   /* --- Wyświetlamy okno dialogowe --- */
   gtk_widget_show (okno_dialogowe);
}
```

Funkcja FunkcjaAnuluj usuwa okno i zeruje globalną zmienną (okno_ dialogowe), aby można było ponownie otworzyć okno.

```
/*
 * FunkcjaAnuluj
 *
 * Zamyka okno dialogowe "Znajdź"
 */
void FunkcjaAnuluj (GtkWidget *kontrolka, gpointer dane)
{
   /* --- Zamykamy okno --- */
   gtk_widget_destroy (GTK_WIDGET (dane));

   okno_dialogowe = NULL;
}
```

Zamiast tworzyć i usuwać okno dialogowe za każdym razem, kiedy użytkownik zechce z niego skorzystać, możemy utworzyć je tylko raz i uwidaczniać je w razie potrzeby. Kiedy użytkownik zakończy wyszukiwanie, okno jest ukrywane, co wygląda dokładnie tak, jakby zostało zamknięte. Metoda taka zużywa więcej pamięci w czasie pracy aplikacji, ale okno dialogowe jest wyświetlane szybciej, ponieważ nie musi być tworzone od nowa.

Funkcja FunkcjaZnajdz szuka łańcucha w pliku. Wyszukiwanie rozpoczyna się od bieżącej pozycji kursora. Zamiast korzystać z funkcji gtk_text_get_point, aby pobrać pozycję kursora, użyjemy pola selection_end_pos w strukturze kontrolki. Robimy tak dlatego, że po przeprowadzeniu wyszukiwania znaleziony tekst zostanie podświetlony, a wyszukiwanie następnego wystąpienia tekstu musi rozpocząć się od końca zaznaczonego obszaru. Po odnalezieniu szukanego łańcucha musimy posłużyć się pewnym trikiem, aby kontrolka tekstu przewinęła się do zaznaczonego obszaru (samo zaznaczenie tekstu nie spowoduje jego ukazania się w kontrolce). Na szczęście wstawienie znaku do kontrolki powoduje przewinięcie wyświetlanego tekstu do punktu wstawienia. Ponieważ chcemy po prostu znaleźć łańcuch, możemy wstawić znak, pozwolić kontrolce na przewinięcie się do właściwej pozycji, a następnie usunąć znak.

```
/*
 * FunkcjaZnajdz
 *
 * Szuka łańcucha w notatniku
 */
void FunkcjaZnajdz (GtkWidget *kontrolka, gpointer dane)
{
   int nIndeks;
   GtkWidget *tekst = PobierzKontrolkeTekstu ();
   char *szStogSiana;

   /* --- Pobieramy tekst z kontrolki --- */
   szStogSiana = PobierzTekst ();

   /* --- Zwalniamy poprzednią "igłę" (tekst) --- */
   if (szIgla) {
      g_free (szIgla);
   }

   /* --- Pobieramy tekst, który należy znaleźć  --- */
   szIgla = gtk_editable_get_chars (
               GTK_EDITABLE (wpis), 0, -1);

   /* --- Pobieramy pozycję kursora --- */
   nIndeks = GTK_EDITABLE (tekst)->selection_end_pos;
```

```
    /* --- Szukamy tekstu --- */
    nIndeks = SzukajLancucha (szStogSiana, szIgla, nIndeks);

    if (nIndeks >= 0) {

      /* --- Przesuwamy kursor do właściwej pozycji --- */
      gtk_text_set_point (GTK_TEXT (tekst), nIndeks);

      /* --- Te dwie linie powodują przewinięcie kontrolki do --- */
      /*     miejsca, w którym znajduje się znaleziony tekst --- */
      gtk_text_insert (GTK_TEXT (tekst), NULL, NULL, NULL, " ", 1);
      gtk_text_backward_delete (GTK_TEXT (tekst), 1);

      /* --- Zaznaczamy znaleziony tekst --- */
      gtk_editable_select_region (GTK_EDITABLE (tekst),
                        nIndeks, nIndeks + strlen (szIgla));

      /* --- Pozwalamy ma ponowne utworzenie okna --- */
      okno_dialogowe = NULL;
    } else {

      PokazKomunikat ("Znajdź...", "Nie znaleziono tekstu w pliku");
    }

    /* --- Zwalniamy pamięć --- */
    g_free (szStogSiana);
}
```

Nie zapominajmy, że musimy zwolnić pamięć, zajmowaną przez łańcuch zwrócony z funkcji PobierzTekst, ponieważ funkcja ta wywołuje funkcję gtk_editable_get_chars, która zwraca blok pamięci przydzielony łańcuchowi. Funkcja SzukajLancucha przeprowadza „siłowe” wyszukiwanie łańcucha, zaczynając od przekazanej jej pozycji, i zwraca indeks pozycji odnalezionego tekstu albo -1, jeśli nie odnajdzie tekstu w kontrolce.

```
/*
 * SzukajLancucha
 *
 * Szukamy łańcucha (szIgla) w dłuższym łańcuchu
 * (szStogSiana) zaczynając od określonej pozycji (nStart)
 * w dłuższym łańcuchu.
 *
 * Algorytm ten znany jest jako "algorytm siłowy"
 */
```

```
int SzukajLancucha (char *szStogSiana, char *szIgla, int nStart)
{
   int nDlugoscStoguSiana;
   int nDlugoscIgly;
   int nPoz;

   /* --- ustalamy długość łańcuchów --- */
   nDlugoscStoguSiana = strlen (szStogSiana);
   nDlugoscIgly = strlen (szIgla);

   /* --- sprawdzamy każdy łańcuch --- */
   for (nPoz = nStart; nPoz < nDlugoscStoguSiana; nPoz++) {

      /* --- Czy znaleźliśmy go w tym miejscu? --- */
      if (strncmp (&szStogSiana[nPoz], szIgla, nDlugoscIgly) == 0) {

         /* --- tak, zwracamy indeks --- */
         return (nPoz);
      }
   }

   /* Nie znaleźliśmy łańcucha, zwracamy -1 --- */
   return (-1);
}

/*
 * PobierzTekst
 *
 * Zwraca tekst, znajdujący się w kontrolce
 */
char *PobierzTekst ()
{
   char *bufor;

   /* --- pobieramy łańcuch z kontrolki --- */
   bufor = gtk_editable_get_chars (
            GTK_EDITABLE (tekst),
            (gint) 0,
            (gint) gtk_text_get_length (GTK_TEXT (tekst)));

    return (bufor);
}
```

```
/*
 * PobierzKontrolkeTekstu
 *
 * Zwraca kontrolkę, która służy do wyświetlania tekstu. Funkcja
 * ta po prostu kapsułkuje globalną zmienną.
 */
GtkWidget *PobierzKontrolkeTekstu ()
{
   return (tekst);
}
```

Pasek postępów

Funkcje paska postępów omówiliśmy w rozdziale 6, „Dalsze kontrolki: ramki, tekst, okna dialogowe, okno wyboru pliku, pasek postępów", a tutaj korzystamy z nich ponownie. Paska postępów używamy po to, aby poinformować użytkownika o stanie operacji, która może trwać przez dłuższy czas. Wczytywanie pliku jest celowo opóźniane, ponieważ w pętli korzystamy z niewielkiego bufora. Jeśli wczytywanie odbywałoby się zbyt szybko, pasek w ogóle nie zostałby wyświetlony. Możemy oczywiście zmienić rozmiar bufora, aby przyspieszyć wczytywanie pliku.

Podsumowanie

GTK+ jest doskonałym narzędziem do szybkiego tworzenia aplikacji. W programie edytora, który napisaliśmy w tym rozdziale, większość pracy wykonywana jest przez kontrolki GTK+; wystarczyło połączyć ze sobą kontrolki menu, paska narzędziowego i tekstu, aby otrzymać gotową aplikację.