PROGRAMOWANIE W SYSTEMIE WINDOWS

Najważniejsze cechy:

→ Środowisko Windows umożliwia pracę wielozadaniową (z wieloma programami – aplikacjami jednocześnie). Występuje współdzielenie zasobów komputera oraz możliwość przełączania miedzy programami.

→ Aby umożliwić komunikację z użytkownikiem każda aplikacja powinna utworzyć swój interfejs → swoje okno (lub okna).

→ Programy w Windows sterowane są zdarzeniami:

- Rozmaite zbiory zdarzeń, takie jak: kliknięcie myszą, naciśnięcie klawisza, manipulowanie oknami, itp. generują komunikaty wysyłane do odpowiadających im okien (interfejsów aplikacji).
- Aplikacje oczekują na komunikaty, odpowiednio na nie reagują, a następnie wracają do stanu oczekiwania na kolejne komunikaty.
- Każda aplikacja również może wysyłać dowolne „swoje" komunikaty za pomocą funkcji SendMessage(...) .

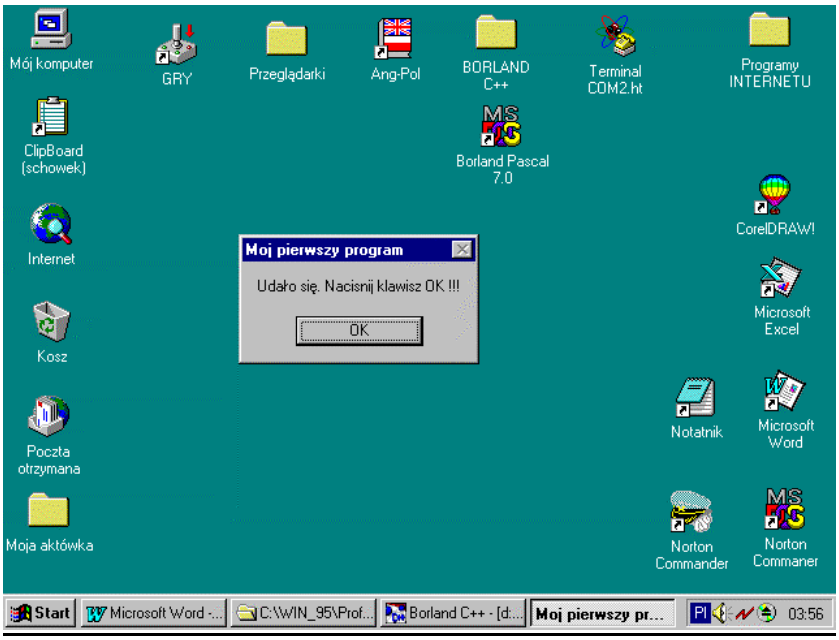
→ Dla programów wykorzystujących GUI (Graphical User Interface) systemu Windows stworzono interfejs programowania aplikacji API (Application Programming Interface). Zawiera on kilkaset funkcji udostępnianych przez Windows (np. MessageBeep, MessageBox, itd.).

→ Po uruchomienu aplikacji w środowisku Windows najpierw wywoływana jest funkcja **WinMain**, która otrzymuje cztery parametry:

```
//Program „0" → przykład bardzo prostego programu wykorzystującego API
#include "windows.h"
int PASCAL WinMain( HINSTANCE  identyfikator_aplikacji,
                    HINSTANCE  identyfikator_poprzedniej_aplikacji,
                    LPSTR      adres_tekstu_parametrow,
                    int        poczatkowy_stan_okna )
{
   MessageBeep( –1 );          // uruchamia generowanie standardowego dźwięku

   MessageBox( NULL,                          // identyfikator okna programu
               "Udało się. Nacisnij klawisz OK !!!",  // wyświetlany tekst
               "Moj pierwszy program",        // tytuł okienka
               MB_OK  );                      // styl okna komunikatu
                        // MB_OK. → Message Box zawierający tekst i klawisz OK
   return 0 ;
}
```

Widok ekranu podczas działania programu „0” :



Ogólna struktura funkcji **WinMain** jest zazwyczaj podobna:

1. Rejestracja wszystkich klas okien i przygotowanie innych zasobów wykorzystywanych przez aplikację (jeżeli nie było wcześniejszych kopii). Nowe klasy okien rejestruje się poprzez podanie ich opisu (za pomocą struktury typu **WNDCLASS** zdefiniowanej w <windows.h>) oraz wywołanie funkcji **RegisterClass**(...).

2. Utworzenie okna lub okien, które mają się pojawić na początku wykonywania programu. Utworzenie okna nastepuje w wyniku wywołania funkcji **CreateWindow**(...) oraz wyświetlenie go na ekranie poprzez wywołanie funkcji **ShowWindow**(...).

3. Oczekiwanie na komunikaty i ich rozsyłanie do odpowiednich okien. Charakterystycznym fragmentem programów w Windows jest pętla oczekiwania na komunikaty (ang. message loop)

```
• • •
MSG komunikat;
while( GetMessage( &komunikat, NULL, 0, 0 ) )
   {
      TranslateMessage( &komunikat );
      DispatchMessage( &komunikat );
   }
• • •
```

Realizacja pozostałych zadań należy w programie do funkcji obsługujących poszczególne okna.

PROJEKTOWANIE GRAFICZNEGO INTERFEJSU UŻYTKOWNIKA w środowisku C++ Builder z wykorzystaniem biblioteki VCL

Biblioteki wspomagające programowanie w środowisku MS Windows:

- **Windows API** (Application Programming Interface)
- **MFC** - Microsoft Foundation Class Library
- **OWL** –Object Windows Library (Borland)
- **VCL** – Visual Component Library (Borland, Delphi, 1995)

Nowe paradygmaty programowania:

- ⇨ obiektowe
- ⇨ komponentowe
- ⇨ zdarzeniowe
- ⇨ wizualne

C++ Builder – produkt firmy Borland Software Corporation:

- system „błyskawicznego" projektowania aplikacji (ang. RAD – Rapid Application Development)
- Umożliwia graficzne (wizualne) projektowanie interfejsu użytkownika (systemu okienek, menu, dialogów, itp.) poprzez rozmieszczanie odpowiednich komponentów za pomocą myszki

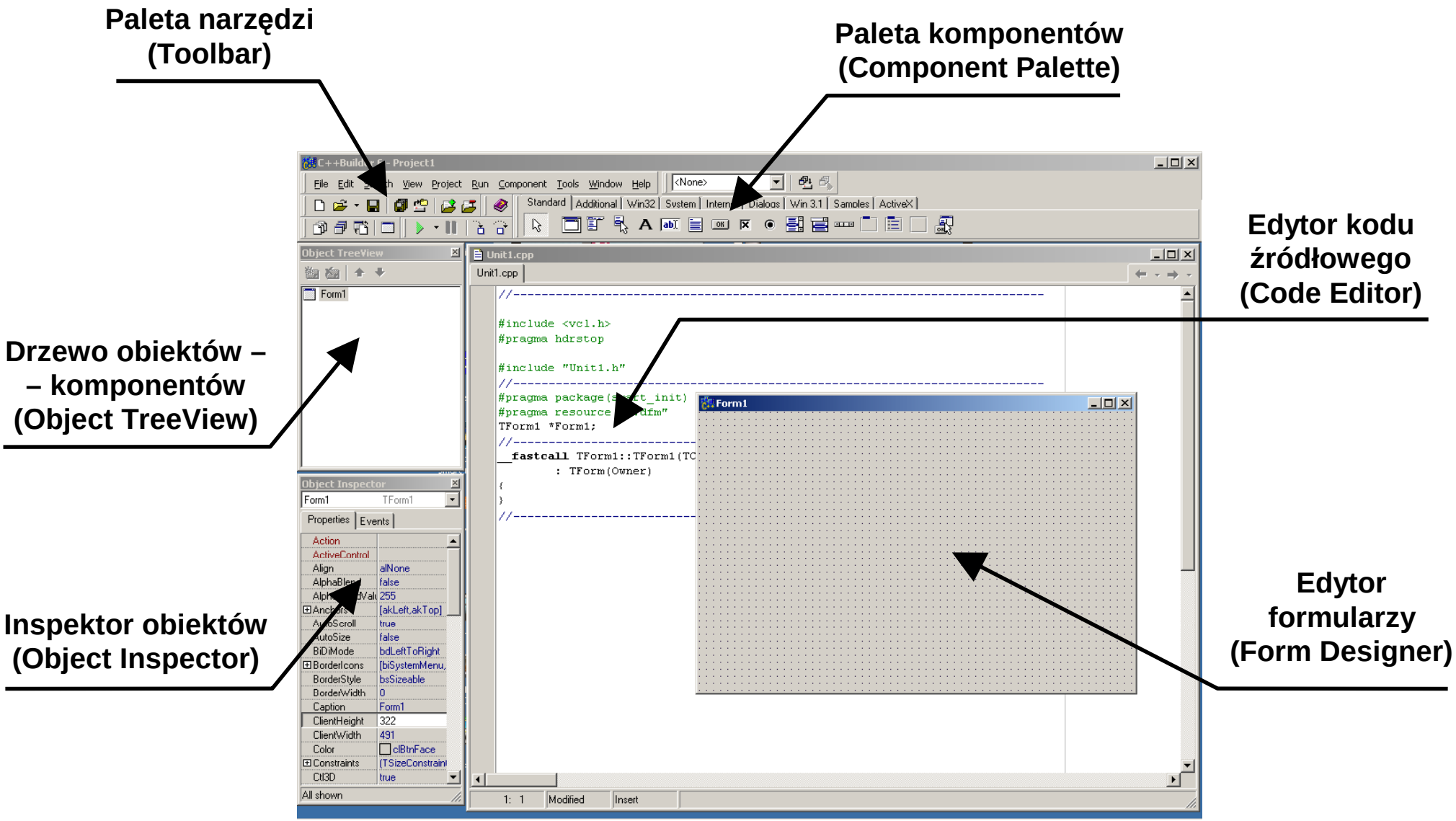
Komponent – (ang. ***component***) jest samodzielnym obiektem programowym wykonującym określone zadanie. Przykładem komponentu jest: przycisk, pole edycyjne. Mogą też być komponenty „niewidoczne" np Timer

Własciwość – (ang. ***property***) – określa parametry i zachowanie komponentu np. Color, Width, Height,

Zdarzenie – (ang. ***event***) – generowane jest w wyniku interakcji komponentu z użytkownikiem (np. kliknięcie myszką) lub systemem operacyjnym (sygnał od zegara)

Funkcja obsługi zdarzenia – (ang. ***event handler***) nazywamy metodę (funkcję własną) komponentu wywoływaną w momencie wystąpienia zdarzenia.

Formularz – formatka, okno (ang. form) jest podstawowym elementem konstrukcyjnym aplikacji tworzonych w systemie C++ Builder. Każda aplikacja musi zawierać przynajmniej jeden formularz pełniący rolę okna głównego.

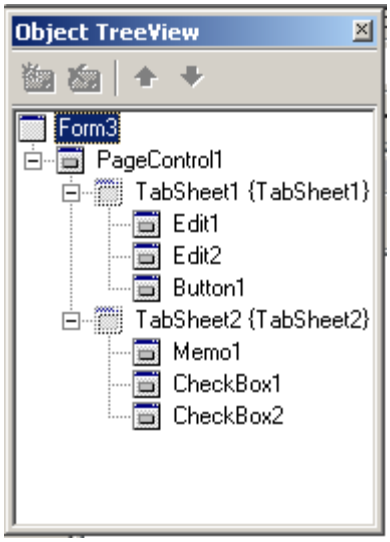


Typowy wygląd środowiska systemu C++ Builder

Paleta komponentów (Component Palette)

zawiera zestawy komponentów które będą umieszczane na formularzach: pola tekstowe, przyciski, znaczniki wyboru, listy rozwijane, itd. Dla większej czytelności zostały podzielone na grupy (zakładki)

Pod paletą narzędzi są dwa okna robocze:

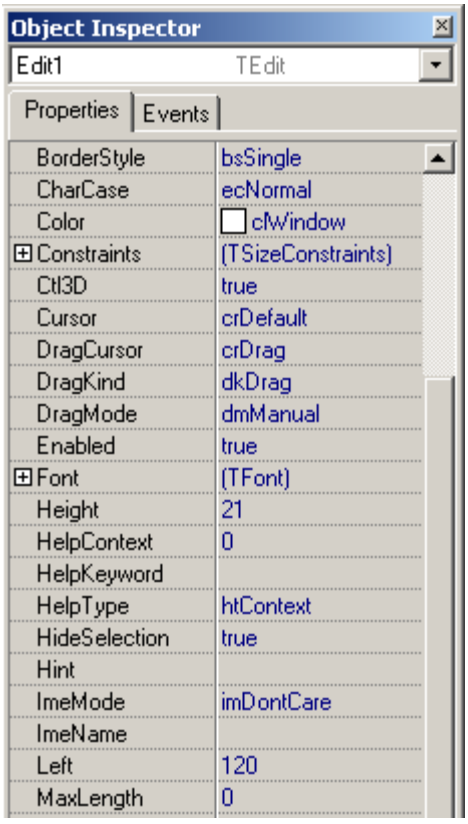


Object TreeView:

Okno podglądu drzewa obiektów wyświetla zawartość aktualnie edytowanej formatki (formularza) w postaci drzewa.

Każdy z węzłów tego drzewa reprezentuje konkretny komponent umieszczony na formatce. Niektóre z węzłów mogą mieć podwęzły (dzieci) reprezentujące komponenty zagnieżdżone.

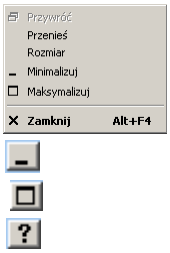
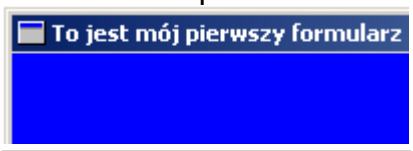
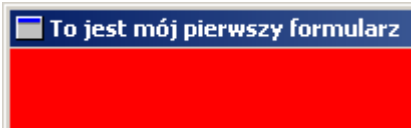
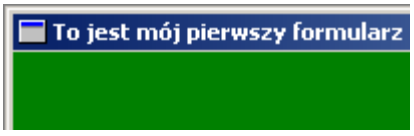
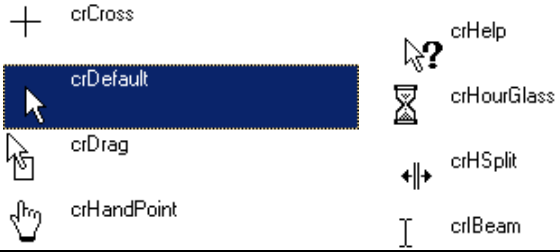
Np. formularz może mieć kilka „paneli”, panele mogą mieć kilka „zakładek”, a każda z zakładek może mieć wewnątrz kilka pól edycyjnych i przycisków.



Object Inspector:

Inspektor obiektów jest jednym z najczęściej wykorzystywanych elementów środowiska CBuilder.

Pozwala modyfikować własności wyglądu (**Properties**) oraz wydarzenia związane z komponentem (**Events**)

Wybrane właściwości (properties) formularzy	
NAZWA	**ZNACZENIE / ZASTOSOWANIE**
ActiveControl	wskazanie domyślnego elementu (komponentu) sterującego, który będzie aktywowany w chwili wybrania okna formularza
BorderIcons	ustala ikony wyświetlane na pasku tytułowym okna:  **biSystemMenu** – menu systemowe **biMinimize** – przycisk minimalizacji okna **biMaximize** – przycisk maksymalizacji okna **biHelp** – przycisk pomocy
BorderStyle	określenie rodzaju ramki okna. **bsSizeable** – pozwala na zmianę rozmiaru okna **bsDialog** – wymusza stały rozmiar okna (blokuje zmiany wielkości)
Caption	ustawia napis/nagłówek wyświetlany na pasku tytułowym okna 
Color	ustawia kolor powierzchni formularza:  Color = clBlue ;  Color = clRed ;  Color = clGreen ;
Cursor	ustawia domyślny kształt kursora myszki na formularzu: 
Enable	= true / false - określa stan aktywności okna
Font	parametry czcionki używanej domyślnie przez wszystkie komponenty umieszczone na tym formularzu
Height	**wysokość** – pionowy wymiar okna (w pikslach)
Hint	treść „podpowiedzi” wyświetlanej po zatrzymaniu kursora myszki
Icon	ustala **ikonę** wyświetlaną w lewym narożniku paska tytułowego
Visible	determinuje widzialność okna na ekranie. Po nadaniu wartości false okno przestaje być widoczne
WindowState	ustala bieżący stan okna: **fsMinimized** – okno jest zminimalizowane **fsNormal** – okno w stanie pośrednim **fsMaximized** – okno rozciągnięte na całym ekranie

Wybrane właściwości standardowych komponentów						
NAZWA	**TButton**	**TEdit**	**TLabel**	**TMemo**	**TListBox**	**ZNACZENIE / ZASTOSOWANIE**
Caption	+	--	+	--	--	ustawia **napis/nagłówek** wyświetlany na komponencie
Color	--	+	+	+	+	ustawia **kolor powierzchni** komponentu
Cursor	+	+	+	+	+	domyślny **kształt kursora** myszki na komponencie
Enabled	+	+	+	+	+	= true / false ← określa stan aktywności komponentu
Font	~	+	+	+	+	**parametry czcionki** używanej do wyświetlania zawartości pola Caption lub Text
Height	+	+	+	+	+	**wysokość** - pionowy wymiar okna (piksele)
Hint	+	+	+	+	+	treść „**podpowiedzi**" wyświetlanej po zatrzymaniu kursora myszki
Items	--	--	--	--	+	tablica tekstów wyświetlanych w kolejnych liniach wyświetlanej listy
Left	+	+	+	+	+	geometryczne **położenie lewej krawędzi** komponentu (względem lewej krawędzi komponentu nadrzędnego)
Lines	--	--	--	+	--	linie/wiersze ← zawartość kolejnych wierszy tekstu wielolinijkowego
MaxLength	--	+	--	+	--	określa maksymalną **pojemność pola** (w ilości znaków)
Name	+	+	+	+	+	nazwa wskaźnika komponentu w kodzie źródłowym
ReadOnly	--	+	--	+	--	= true / false; umożliwia zablokowanie modyfikacji zawartości
ShowHint	+	+	+	+	+	= true / false determinuje wyświetlanie podpowiedzi (Hint)
TabOrder	+	+	--	+	+	ustala kolejny numer komponentu w porządku wybierania klawiszem ***Tab***
TabStop	+	+	--	+	+	Sygnalizuje, że dany komponent może być wybierany za pomocą klawisza ***Tab***
Text	--	+	--	--	--	implementuje zawartość pola edycyjnego
Top	+	+	+	+	+	geometryczne **położenie górnej krawędzi** komponentu (względem górnej krawędzi komponentu nadrzędnego)
Visible	+	+	+	+	+	= true / false determinuje **widzialność komponentu** na ekranie. Po nadaniu wartości false komponent przestaje być widoczne
Width	+	+	+	+	+	**szerokość** - poziomy wymiar okna (piksele)
WordWrap	--	+	+	+	--	= true / false włącza funkcję dzielenia wierszy

Ważniejsze wydarzenia (zdefiniowane dla większości komponentów)	
NAZWA	**ZNACZENIE / ZASTOSOWANIE**
`OnChange`	źródłem tego zdarzenia jest zmiana zawartości lub postaci komponentu
`OnClick`	zdarzenie to generowane jest w chwili kliknięcia komponentu jednym z przycisków myszki
`OnDblClick`	zdarzenie dwukrotnego (podwójnego) kliknięcia komponentu myszką
`OnEnter`	zdarzenie to jest efektem wybrania komponentu (np. myszką lub klawiszem ***Tab***)
`OnExit`	zdarzenie generowane w momencie zakończenia stanu wybrania komponentu (w momencie przejścia do innego komponentu)
`OnKeyDown`	zdarzenie naciśnięcia dowolnego klawisza: klawisza alfanumerycznego, funkcyjnego lub specjalnego (np. klawisz strzałki, Home, Esc, Ctrl, itd)
`OnKeyPress`	Zdarzenie naciśnięcia klawisza: alfanumerycznego, spacji, tabulacji, Esc lub Enter
`OnKeyUp`	Zdarzenie zwolnienia dowolnego klawisza
`OnMouseDown`	Naciśnięcie przycisku myszki w chwili gdy jej kursor znajduje się nad komponentem. Jako parametry przekazywane jest: informacja o naciśniętym przycisku, naciśnietych klawiszach specjalnych (Shift, Ctrl) oraz współrzędne położenia kursora myszki w chwili naciśnięcia.
`OnMouseMove`	Zdarzenie przemieszczania kursora myszki w obszarze komponentu
`OnMouseUp`	Zdarzenie zwolnienia przycisku myszki, gdy kursor znajduje się w obszarze komponentu
`OnPaint`	Zdarzenie sygnalizuje konieczność ponownego wyrysowania (odświeżenia) komponentu.