Opis i przykłady zastosowań wybranych komponentów VCL

W bibliotece VCL obiekt klasy **AnsiString** (o alternatywnej skrótowej nazwie **String**) służy do przechowywania w pamięci komputera tekstu (napisu, łańcucha znaków) o dowolnej długości.

Obiekty klasy **String** zastępują standardową reprezentację napisów z języka C jaką była zwykła tablica znaków.

```
char   tekst_C[50];                ← standardowy tekst C / C++
String tekst_VCL;                  ← reprezentacja tekstów w bibl. VCL
String tekst = "Kowalski" ;
```

Wybrane metody klasy String	
Nazwa metody	**Znaczenie – opis działania**
konstruktor **String**	klasa **String** posiada bardzo wiele wersji konstruktora, które pozwalają utworzyć nowy tekst/napis na podstawie istniejących danych różnego typu: String(const **String**& src); np. String nowy_napis = **String(tekst_VCL)**; String(const **char*** src); np. String nowy_napis = **String("Nowak")**; String(const **char*** src, unsigned int len); np. String nowy_napis = **String("Nowak", 3)**; String(const **char** src); np. String nowy_napis = **String('N')**; String(**long** src); String(**int** src); np. String nowy_napis = **String(123)**; String(**double** src); np. String nowy_napis = **String(345.67)**;
c_str	**char* c_str() const;** Zwraca standardowy wskaźnik (typu **char***) do łańcucha znaków zawartego w obiekcie **String** np. printf("%s", **tekst.c_str( )**);
Delete	**void Delete(int index, int count);** Usuwa fragment łańcucha poczynając od pozycji **index** (Uwaga: pozycje są numerowane od 1). Ilość usuwanych znaków (długość usuwanego fragmentu) jest zadawana wartością parametru **count**. np. **tekst.Delete(1,3)**; ← usuwa pierwsze trzy litery

Insert	**void Insert(const String& str, int index);** Wstawia nowy fragment tekstu na zadanej pozycji (**index**) łańcucha np. **tekst.Insert(”xyz”, 2)**; ← wstawia trzy litery **tekst.Insert(String(”xyz”) , 2)**;
IsEmpty	**bool IsEmpty( ) const;** Zwraca wartość **true** jeżęli obiekt String zawiera pusty łańcuch (o zerowej długości. np. if(**tekst.IsEmpty( )**) tekst = “wstawiam coś”;
Length	**int Length( ) const;** Zwraca długość (ilość znaków) łancucha bez kończącego znaku NULL
LowerCase	**String LowerCase( ) const;** Zwraca nowy łańcuch, w którym wszystkie duże litery są zamienione na małe. String po_zamianie = **tekst.LowerCase( )**;
operator [ ]	**char& operator [ ](const int idx);** Operator indeksu pozwala modyfikować wartości pojedynczych znaków łańcucha, podobnie jak to było w przypadku standardowej tablicy znaków. **UWAGA:** w klasie String pozycje są indeksowane od **1** np. **tekst[3]** = ‘X’; ← zamienia trzecią literę na ‘X’
operator **!=** operator **+ +=** operator **<** operator **<=** operator **=** operator **==** operator **>** operator **>=**	Klasa **String** posiada zaiplementowane większość operatorów “arytmetycznych” , co pozwala operować na łańcuchach równie naturalnie jak na liczbach. String napis1 = “Kowalski”; String napis2 = “Jan”; String napis3 = “Wrocław”; String nowy **= napis1 + napis2 + napis3**; if(**napis1 !=** ”Nowak”) **napis1 =** ”Nowak”;
Pos	**int Pos(const String& subStr) const;** Wyszukuje pierwsze wystąpienie zadanego ciągu znaków i zwraca jego pozycję. Jeżeli szukany ciąg nie występuje w łańcuchu, to zwracana jest wartość **0**. if(**tekst.Pos(** “Jan” **)** == 0) tekst+=”Jan”;
printf	**int printf(const char* format, . . .);** Nadanie wartości łańcucha poprzez “wydruk” w stylu standardowej funkcji printf z języka **C**. float x=12.5; **tekst.printf(** ”Wynik = %f”, x **)**;

SubString	**String SubString(int index, int count) const;** Zwraca nowy łańcuch zawierający kopię wskazanego fragmentu łańcucha o długości **count** poczynając od zadanej pozycji **index**. String kopia_fragmentu = **tekst.SubString(** 2,3 **)**;
ToDouble	**double ToDouble() const;** Zamienia (konweruje) łańcuch na odpowiadającą mu liczbę zmiennoprzecinkową. W przypadku niepowodzenia konwersji generuje wyjątek (błąd). String napis = “234.67”; double x = **napis.ToDouble( )**;

ToInt	**int ToInt() const;**
	Zamienia (konweruje) łańcuch na odpowiadającą mu liczbę całkowitą. W przypadku niepowodzenia konwersji generuje wyjątek EConvertError.
	try {
	x = **napis.ToInt( )**;
	} catch (**EConvertError** &e) {
	ShowMessage(“Wystąpił błąd konwersji”);
	}

Trim	**String Trim() const;** Zwraca nowy łańcuch powstający poprzez usunięcie znaków spacji z początku i końca łańcucha. String napis = “ abcd xyz “; String napis_bez_spacji = **napis.Trim( )**;
TrimLeft	Zwraca łańcuch bez początkowych spacji.
TrimRight	Zwraca łańcuch bez końcowych spacji.
UpperCase	Zwraca nowy łańcuch, w którym wszystkie małe litery są zamienione na duże.

Przykład wykorzystania:

```
String  nazwa = "wroCŁAw";                    // utworzenie nowego łańcucha
nazwa = nazwa.LowerCase( );                   // zamiana wszystkich liter na małe
nazwa[1] = toupper( nazwa[1] );               // zamiana pierwszej litery na dużą
if( nazwa == "Wrocław" )                      // porównanie łańcuchów
  ShowMessage("Formatowanie OK!");

nazwa += " główny PKP";                       // dopisanie tekstu do końca łańcucha

int pozycja = nazwa.Pos("główny");            // szukanie wystąpienia tekstu
if( pozycja >0 )
 {
  nazwa.Delete(pozycja,6);                    // usunięcie znalezionego fragmentu
  nazwa.Insert("GŁ.", pozycja);               // wstawienie nowego fragmentu
 }
```

TStrings

Zdefiniowana w bibiotece VCL klasa TStrings obsługuje listy łańcuchów i jest wykorzystywana przez komponenty wykorzystujące kilkuwierszowe dane tekstowe np.

 TMemo->**Lines**,  TListBox->**Items**,  TComboBox->**Items**,

Wybrane właściwości klasy TStrings	
Nazwa	**Znaczenie – opis działania**
Count	typu **int** – licznik ilości łańcuchów/wierszy w obiekcie np. for(int i=0; **Memo1–>Lines–>Count**; i++) if(Memo1–>Lines–>Strings[i] == “Koniec”) ...
Strings[ ]	typu **String** – udostępnia wskazany wiersz obiektu pozycja zwracanego elementu (tekstu) zadawana jest za pomocą operatora indeksu (Uwaga: indeksowanie jest od zera!) np. String pierwszy = **Memo1–>Lines–>Strings[0]**;
Text	typu **String** – udostępnia całą zawartość obiektu wszystkich wierszy obiektu w postaci jednego jednego łańcucha (zawierającego wewnątzr znaki końca wierszy CR NL)

Wybrane metody klasy TStrings	
Nazwa	**Znaczenie – opis działania**
Append **Add**	**void Append(const String S);** wywołanie metody dopisuje nowy łańcuch S na końcu listy np. Memo1–>Lines–>**Append(“Nowy tekst”)**;
Delete	**void Delete(int Index);** usuwa z listy łańcuch (wiersz) znajdujący się na pozycji I**ndex** np. Memo1–>Lines–>**Delete(0)**;
Insert	**void Insert(int Index, const String S);** wstawia nowy łańcuch na zadanej pozycji I**ndex** np. Memo1–>Lines–>**Insert(0, ”Nowy pierwszy wiersz”)**;
LoadFromFile	**void LoadFromFile(const AnsiString FileName);** wczytuje nową zawartość listy tekstów z pliku tekstowego o nazwie zadanej parametrem **FileName** np. Memo1–>Lines–>**LoadFromFile (”Unit1.cpp”)**;

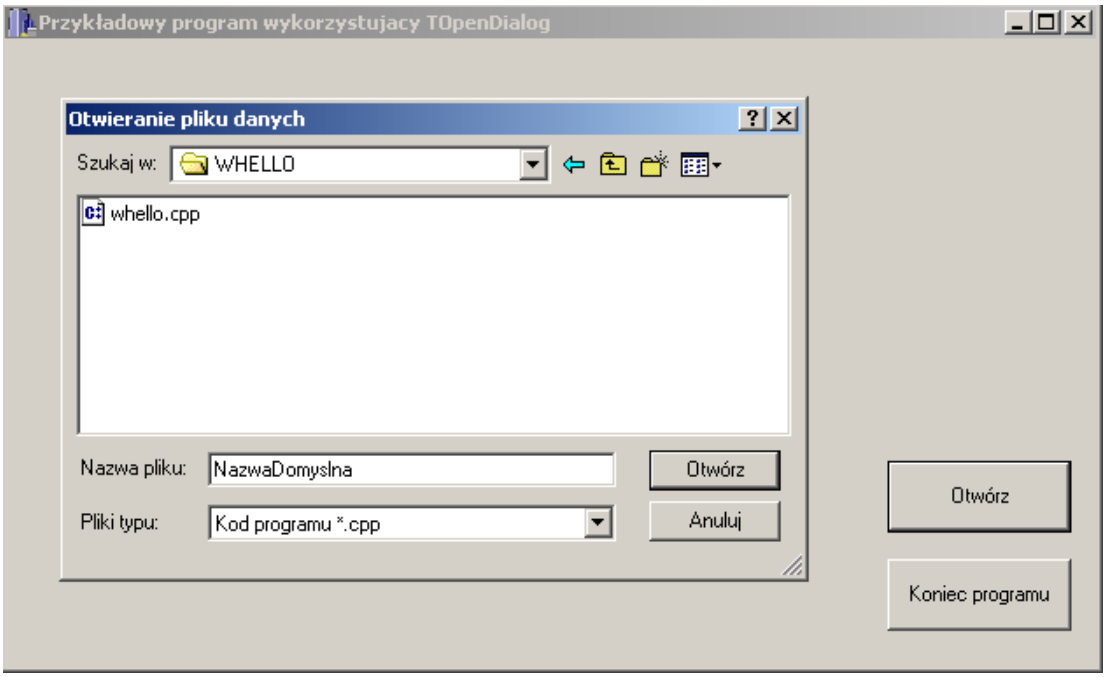
TMemo

Obiekt klasy TMemo  jest wielowierszowym polem edycyjnym. Zawartość tego pola jest zadawana i modyfikowana za pomocą właściwości **Lines**. Pozostałe właściwości pozwalają sterować wyglądem okna edycyjnego.

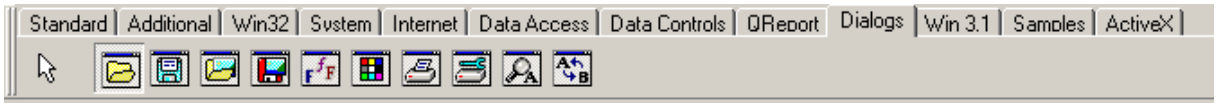
Wybrane właściwości TMemo	
Nazwa	**Znaczenie - zastosowanie / przykładowe wartości**
Aligment	Ustawia wyrównanie tekstu w oknie: **taLeftJustify** – wyrównanie tekstów do lewej **taCenter** – wycentrowanie **taRightJustify** – wyrównanie do prawej strony
Lines	**TStrings** – lista/tablica edytowanych wierszy
ReadOnly	true / **false** – blokowanie edycji zawartości
ScrollBars	Steruje wyświetlaniem suwaków do przewijania okna **ssNone** – bez suwaków **ssHorizontal** – tylko przesuwanie w poziomie **ssVertical** – tylko przesuwanie w pionie **ssBoth** – wyświetlanie obu suwaków
WantTabs	**true** / false – sposób reagowania na **tabulacje**
WantReturns	**true** / false – sposób reagowania na **Enter**
WordWrap	**true** / false steruje włączaniem dzielenia (łamania) linii o długości większej niż szerokość okna

TOpenDialog

Obiekt klasy **TOpenDialog** służy do wyświetlania (modalnego) okna dialogowego umożliwiającego wybór/wskazanie pliku oraz jego lokalizacji (ścieżki dostępu). Na rysunku poniżej okno tego dialogu jest zatytułowane „Otwieranie pliku danych”



Ikona  obiektu **TOpenDialog** znajduje się na zakładce DIALOGS:



Inne podobne klasy:

TSaveDialog (następna ikona obok TOpenDialog)
TOpenPictureDialog (trzecia ikona, wywodzi się z **TOpenDialog**)
TSavePictureDialog (czwarta ikona, wywodzi się z **TSaveDialog**)

Wybrane właściwości **TOpenDialog**	
Nazwa	**Znaczenie - zastosowanie / przykładowe wartości**
FileName	typu **AnsiString** – zawiera nazwę i pełną ścieżkę dostępu do ostatnio wybranego pliku, Umożliwia odczyt wybranego pliku po zakończeniu dialogu oraz zadawanie domyślnej nazwy pliku wyświetlanej w momencie otwarcia dialogu.
Title	typu **AnsiString** – zawiera tytuł wyświetlany w nagłówku okna dialogu. Domyslnie zawiera tekst ”Open”

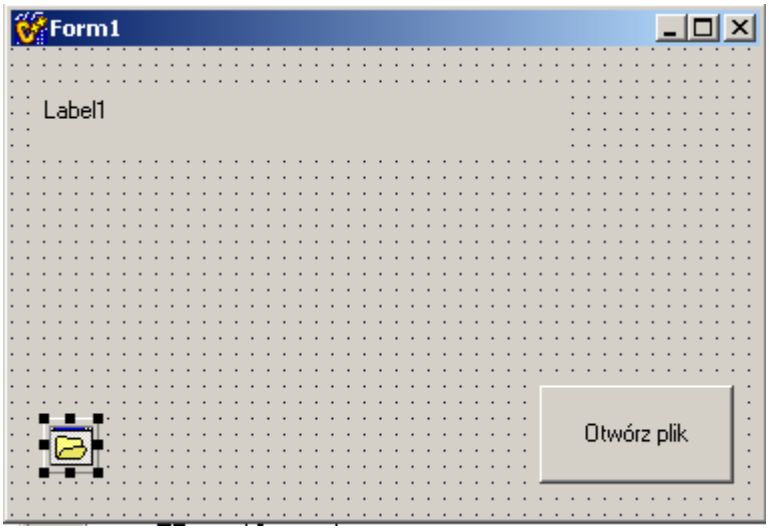
DefaultExt	typu **AnsiString** – zawiera rozszerzenie (np. txt, cpp) dodawane automatycznie do nazwy pliku wpisanej przez użytkownika (jeżeli plik jeszcze nie zawiera zarejestrowanego rozszerzenia)
InitialDir	typu **AnsiString** – umożliwia zadawanie katalogu wyświetlanego w momencie otwarcia dialogu np. **OpenDialog-> InitialDir= "C:\WINDOWS\SYSTEM"** spowoduje wyświetlenie katalogu systemowego.
Files	typu **TStrings*** – wskaźnik na obiekt TStrings zawierający nazwy wszystkich wybranych plików. (w przypadku umożliwienia wyboru wielokrotnego poprzez ustawienie flagi ofAllowMultiSelect=true)
Filter	typu **AnsiString** – pozwala zadawać filtry wyświetlanych plików. Pierwszy tekst (do pionowej kreski) jest wyświetlany w oknie a drugi tekst (po pionowej kresce) zadaje filtr wykorzystywany podczas wyświetlania plików. Np. ustawienie wartości: **OpenDialog->Filter = "Text files (*.txt)|*.TXT"**; umożliwi wyświetlanie tylko plików tekstowych. Aby zadać kilka różnych filtrów, teksty kolejnych definicji powinny być pooddzielane dodatkowymi kreskami np: **"Text files (*.txt)|*.TXT|Pascal files (*.pas)|*.PAS"**
FilterIndex	typu **int** – pozwala wskazać numer filtru aktywnego w momencie otwarcia dialogu. np. polecenie **OpenDialog->FilterIndex = 2**; ustawi filtr drugi z w/w tzn. "Pascal files (*.pas)|*.PAS"

Podstawowa metoda TOpenDialog	
Nazwa	**Znaczenie - zastosowanie / przykładowe wartości**

Execute( )	Ta bezparametrowa funkcja powoduje otwarcie / wyświetlenie okna dialogowego na ekranie. Kończy swoje działanie po naciśnięciu klawisza Otwórz (wówczas zwraca wartość true) lub po naciśnięciu klawisza Anuluj (wówczas zwraca wartość false), Rezultat działania okna dialogowego tzn. nazwę i ściezkę dostępu do wskazanego pliku można pobrać z pola FileName.
	Najczęściej wywołanie tej metody umieszczane jest w instrukcji if np.
	`if( OpenDialog->Execute( ) )`
	`{`
	`// tu są umieszczane operacje wykonywane`
	`// po prawidłowym wybraniu pliku przyciskiem „Otwórz”`
	`}`

Przepis na zbudowanie prostej aplikacji wykorzystującej obiekt TOpenDialog

1. Utwórz nowy projekt poprzez wybranie z menu głównego
   FILE → NEW APPLICATION
2. Z zakładki „Standard” wybierz narzędzie dodawania przycisków  i umieść w prawym dolnym rogu formularza przycisk zatytułowany „Otwórz plik”
3. Z zakładki „Standard” wybierz narzędzie dodawania tekstów  i umieść w lewym górnym rogu formularza tekst o standardowej nazwie Label1
4. Z zakładki „Dialogs” wybierz narzędzie dodawania dialogu TOpenDialog  i umieść go w dowolnym miejscu formularza (ikona tego elementu nie będzie widoczna w czasie działania programu)
5. Po wykonaniu w/w kroków formularz powinien mieć następujący wygląd:



6. Kliknij dwa razy na przycisku „Otwórz plik” i oprogramuj zdarzenie „OnClick” wpisując nastepującą treść wewnątrz funkcji „Button1Click”

```
void __fastcall TForm1::Button1Click(TObject *Sender)
  {
    OpenDialog1->Title = "Otwieranie pliku danych";
    OpenDialog1->Filter =
                "Pliki tekstowe *.txt|*.txt|Kod programu *.cpp|*.cpp";
    OpenDialog1->FilterIndex=2;
    OpenDialog1->FileName="NazwaDomyslna.txt";
    OpenDialog1->DefaultExt="txt";
    if ( OpenDialog1->Execute() )
       Label1->Caption=OpenDialog1->FileName;
  }
```

7. Od tego momentu, kliknięcie przycisku „Otwórz plik” będzie powodować otwarcie okna dialogowego TOpenDialog. Okno to będzie:
   - zatytułowane „Otwieranie pliku danych”,
   - będzie miało ustawione dwa rodzaje filtrów: *.txt dla plików tekstowych i *.cpp dla plików zawierających kod programu,
   - domyślnie wybranym filtrem będzie filtr o numerze „2” tzn. „*.cpp”
   - domyślna nazwa pliku wpisana po otwarciu do okienka edycyjnego „Nazwa pliku” będzie miała treść „NazwaDomyslna.txt”
   - domyślne rozszerzenie dopisywane do plików (których nazwy nie zawierają zarejestrowanego rozszerzenia) będzie „.txt”
   - po poprawnym zakończeniu dialogu (poprzez kliknięcie przycisku „Otwórz”), treść statycznego tekstu „Label1” zmieni się na nazwę wybranego pliku wraz z pełną ścieżką dostępu.

TCanvas

Obiekt klasy **Canvas** (z bibioteki VCL Borlanda) służy do ułatwienia rysowania grafiki (punktów, figur geometrycznych, tekstów) w oknie komponentów **TForm** lub **TImage**.

Wybrane właściwości **TCanvas**	
Nazwa	**Znaczenie - zastosowanie / przykładowe wartości**
Pixels	dwuwymiarowa tablica (macierz): TColor Pixels[int X][int Y] umożliwia dostęp (zapis i odczyt) koloru pojedynczych piksli w oknie
Pen	Definiuje właściwości konturu (linii) rysowanych obiektów → **Color** = kolor linii np. **clBlue**, **clRed**, **clGreen**, **clBlack**, ... → **Width** = grubość linii w pikselach np. **3** → **Style** = styl linii np. **psSolid**, **psDot**, **psDash**, **psDashDot**
Brush	Definiuje właściwości wypełnienia rysowanych obiektów → **Color** = kolor wypełnienia np. **clBlue**, **clRed**, ... → **Style** = styl wypełnienia np. **psSolid**, **psDot**, **psDash**, **psDashDot**

Wybrane metody **TCanvas**	
Nazwa	**Znaczenie - zastosowanie / przykładowe wartości**
Arc	Wykreśla łuk okręgu, używając aktualnie wybranego pióra. Łuk jest fragmentem elipsy ograniczonej zadanym prostokątem
Ellipse	Rysuje elipsę wpisaną w prostokąt o zadanych wierzchołkach.
LineTo	Rysuje linię od aktualnej pozycji kursora graficznego do punkyu o zadanych współrzędnych (X,Y)
MoveTo	Przesuwa kursor graficzny do punktu o zadanych współrzędnych (bez rysowania linii)
Pie	Rysuje wycinek koła
Rectangle	Wykreśla prostokąt którego przeciwległymi wierzchołkami są zadane punkty o współrzędnych (X1, Y1) oraz (X2, Y2).
RoundRect	Wykreśla prostokąt o zaokrąglonych narożnikach.
TextOut	Wypisuje na powierzchni „płótna” zadany tekst, rozpoczynając od punktu o zadanych współrzędnych (X,Y).