# Funkcje konwersji

Komponentami wizualnymi organizującymi komunikowanie się między programem i użytkownikiem w aplikacji C++ Builder są obiekty manipulujące na tekstach. Najprostszym i najczęściej stosowanym jest pole tekstowe (***Edit***) . Transmisja odbywa się za pośrednictwem własności **Text**. Wartości które mogą być wprowadzane i wyprowadzane są typu *AnsiString* („łańcuch znaków”). Inne obiekty tekstowe wykorzystywane do transmisji też robią to za pośrednictwem danych typu *AnsiString* lub *String*.

 W przypadku pracy z danymi typu numerycznego istnieje więc potrzeba konwersji tych danych, tak aby możliwe było:

1. wyprowadzenie wyników numerycznych: konwersja ***liczba -> tekst***
2. pobieranie danych numerycznych: konwersja ***tekst -> liczba***.

C++ Builder oferuje bardzo duży zestaw funkcji, przy pomocy których taką konwersję można zrealizować.

#### Konwersja liczba -> tekst

FloatToStrF, FloatToStr, CurrToStrF, CurrtoStr, IntToStr (dodatkowo inne funkcje: itoa, ltoa, sprintf)

Poniżej omówione zostaną wybrane funkcje konwersji.

***AnsiString FloatToStrF(Extended Value, TfloatFormat Format, int Precision, int Digits);***

Funkcja dokonuje konwersji liczby zmiennoprzecinkowej (rzeczywistej) podanej w do jej postaci łańcuchowej; wartość ta to wynik zwracany prze funkcję.

*Value* – wartość liczbowa podlegająca konwersji.

*Format* – stała określająca postać łańcucha reprezentującego ww. liczbę. Wartości dozwolone dla parametru:

*ffGeneral*: ogólny format liczby, przedstawia liczbę w jak najkrótszej postaci bez nieznaczących zer i bez separatora części dziesiętnej jeśli jej nie ma. Format jest stałoprzecinkowy lub wykładniczy.

*ffExponent*; format naukowy (wykładniczy) postaci „–d.ddd...E+/-dddd”, przy czym znak minus występuje gdy liczba jest ujemna. Znak + lub – w środku łańcucha informuje o znaku wykładnika potęgi.

*ffFixed*: format stałoprzecinkowy postaci „–ddd.ddd...”, znak – dla liczb ujemnych. Co najmniej jedna cyfra zawsze poprzedza separator dziesiętny. Liczb cyfr po separatorze zależy od parametru Digits. Jeśli liczba cyfr przed separatorem jest większa niż parametr *Precision* wynikowa wartość ma postać wykładniczą. Przykład: *FloatToStrF(x,ffFixed,10,2);*

*ffNumber*: format liczby postaci „-d,ddd,ddd.ddd...” tzn. ma format *ffFixed* tylko z wprowadzonymi separatorami tysięcy.

*ffCurrency*: format walutowy odziedziczony w specjalnych zmiennych globalnych z panelu sterowania systemu Windows.

*Precision* – całkowita liczba znaków, jaka jest wykorzystywana do zapisu liczby. Dla typu *Single* co najwyżej 7 znaków dla typu *Double* co najwyżej 15, a dla typu *Extended* co najwyżej 18 znaków.

*Digits* – liczba cyfr po separatorze części dziesiętnej (< *Precision*).

Dla wszystkich formatów znaki użyte jako separatory tysięcy i części dziesiętnej są pobrane ze zmiennych globalnych *ThousandSeparator* i *DecimalSeparator*.

Jeśli parametr Value ma postać NAN (not-a-number) niezgodną z postacią liczby łańcuchem wynikowym jest wartość stała rózna ‘NAN’. Jeśli jest to liczba dodatnia lub ujemna nieskończona, wynikowym łańcuchem jest odpowiednio: ‘INF’ oraz ‘-INF’>

***AnsiString FloatToStr(Extended Value);***

Zamienia liczbę zmiennoprzecinkową zapisaną w parametrze *Value* na jej reprezentację tekstową. W trakcie konwersji użyty jest ogólny format liczby z 15 cyframi znaczącymi.

***AnsiString IntToStr(int Value);***

Funkcja dokonuje konwersji liczby całkowitej na jej reprezentację tekstową. Przykład: *a=IntToStr(12);*

#### Konwersja tekst -> liczba

StrToFloat, StrToCurr, TextToFloat, StrToInt (dodatkowo inne funkcje: atoi, atol, atof)

Poniżej omówione zostaną, chyba najlepsze do ww. zastosowań, funkcje konwersji.

***bool TextToFloat (char \* Buffer, void \*Value, TFloatValue ValueType);***

Funkcja zawarta w module *SysUnit*. Zamienia tekst zawarty w argumencie *Buffer* na liczbę zmiennoprzecinkową przekazaną do parametru *Value*. Trzeci parametr określa format wyniku. Funkcja zwraca wartość *true*, jeśli konwersja zakończy się powodzeniem. W przeciwnym przypadku zwróci *false* (np. jeśli pierwszy argument nie będzie miał reprezentacji liczby).

 ***Extended*** ***StrToFloat(const AnsiString S);***

Funkcja dokonuje konwersji łańcucha typu *AnsiString* zawartego w parametrze S do postaci liczby zmiennoprzecinkowej (wynik jest liczba typu *Extended*). Ta liczba jest wynikiem funkcji. Łańcuch S musi zawierać cyfry, wiodący znak minus dla liczby ujemnej (plus jest opcjonalny), opcjonalny znak kropki oddzielającej część dziesiętną oraz opcjonalnie literę E lub e w przypadku liczby zapisanej w formacie naukowym. Nieznaczące zera są pomijane. Globalna zmienna *DecimalSeparator* zawiera informację jaki znak pełni w liczbie rolę separatora części dziesiętnej. Niedopuszczalne są w argumencie znaki separatora tysięcy i waluty. Jeśli wejściowy łańcuch nie jest zgodny z formatem liczby, funkcja generuje wyjątek *EConvertError*.

  ***int StrToInt(const AnsiString S);***

Funkcja dokonuje konwersji łańcucha typu *AnsiString* zawartego w parametrze *S* do postaci liczby całkowitej. Argument *S* może mieć postać liczby w systemu dziesiętnego lub szesnastkowego. Jeśli wejściowy łańcuch nie jest zgodny z formatem liczby, funkcja generuje wyjątek *EConvertError*.

Jeśli w programie wystąpi błąd, jest generowana sytuacja wyjątkowa (wyjątek) sygnalizująca błąd. Sposób obsługi takich błędów zostanie omówiony na innym wykładzie.

Konwersja *AnsiString* null-terminated string

Zmienne, które mogą zawierać tekst nazywa się łańcuchami znakowymi (zmienne tekstowe). Łańcuchy są dostępne prawie w każdym komponencie C++ Builder w postaci właściwości *Caption*. Typ obsługujący łańcuch znaków w C++ Builder zaimplementowany jako klasa i nosi nazwę ***AnsiString***. Stąd przykładowa deklaracja zmiennej tego typu może mieć postać:

 *AnsiString Dana\_tekstowa;*

Na danych typu *AnsiString* można wykonywać operacje przypisania za pomocą operatora przypisania =

 *Dana\_tekstowa = ”To jest przykładowy tekst”;*

Można też wykonywać operacje złączenia (konkatencji tekstów) za pomocą operatora +

 *Dana\_tekstowa = Dana\_tekstowa + ” napisany przez programiste.”*

W C i C++ zmienne tekstowe można było deklarować tylko jako tablice znaków:

 *char tablica\_znakow[50];*

Taki typ określany jest jako *null-terminated string* (łańcuch zakończony zerem), dla zaakcentowania, że koniec

tekstu jest zaznaczony znakiem \0.

W BCB deklarowanie takich tablic jest dopuszczalne, ale typy *AnsiString* i tablica znaków nie są tożsame. Zmienne obu typów mogą wymieniać swoje dane, ale są do tego potrzebne operacje konwersji. Zostaną one omówione na osobnym wykładzie.