Controls

ZÁKLADNÍ KOMPONENTY VIZUALIZAČNÍHO SYSTÉMU

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.00

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 21.01.2004
Datum posledního uložení dokumentu: 23.01.2004
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.00	Cr	21.01.2004	První vydání

1.2. Účel dokumentu

Tento dokument slouží jako popis knihovny Controls, která je součástí balíku vizualizačních knihoven pro jednotku KIT.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu je potřeba seznámit se s manuálem IoDrv, Bitmaps a Fonts.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

2. Termíny a definice

Používané termíny a definice jsou popsány v samostatném dokumentu Termíny a definice.

3. Úvod

3.1. Účel knihovny Controls

4. Komponenty

4.1. Komponenta a skupiny komponent

Komponenta je základní stavební prvek vizualizačního systému. Komponentou je například editační řádka, posuvná lišta, tlačítko apod. Obecně lze říct, že komponenta je třída, která popisuje chování definované obdélníkové oblasti na displeji. Komponenta předepisuje co a jak bude v dané oblasti vykresleno a jak bude daná oblast reagovat na vstupní události přicházející od uživatele.

Všechny komponenty vycházejí z abstraktní třídy **TControl** (viz. kapitola 5.3.4). Třída **TControl** definuje standardní chování komponent a poskytuje rozhraní pro snadný návrh komponent a snadnou manipulaci s komponentami z uživatelské aplikace.

```
TControl = object( TObject )
public
  Id           : Word;
  Owner        : PGroup;
  Next         : PControl;
  Bounds       : TRect;
  State        : Word;
  Options      : Word;

  ...


  constructor Init( const ABounds: TRect );
  destructor Done; virtual;

  procedure HandleEvent( var AEvent: TEvent ); virtual;
  procedure Paint( ACanvas: PCanvas ); virtual;

  function  Focus: Boolean;
  procedure Select;
  procedure Show;
  procedure Hide;
  procedure Disable;
  procedure Enable;

  ...

end;
```

Každá komponenta má jednoznačné identifikační číslo (položka **Id**), pozici a rozměry (položka **Bounds**), svého vlastníka (položka **Owner**), svého následníka (položka **Next**), stav (položka **State**) a nastavení (položka **Options**).

Potomci třídy **TControl** měnící chování komponenty, musí minimálně předefinovat metodu **Paint**, tak aby správně vykreslovala aktuální stav komponenty (viz. kapitola 4.5) a metodu **HandleEvent**, tak aby správně obsluhovala vstupní události (viz. kapitola 4.6).

Např. komponenta pro editační řádku je definována přibližně takto:

```
TEdit = object( TControl )
public
  Data      : PString;      { Editovaný řetězec                  }
  MaxLen    : Integer;      { Maximální délka editovaného řetězce }
  CurPos    : Integer;      { Aktuální pozice textového kurzoru   }

  ...

  constructor Init( const ABounds: TRect; AMaxLen: Integer );
  destructor Done; virtual;

  procedure Paint( ACanvas: PCanvas ); virtual;
  procedure HandleEvent( var AEvent: TEvent ); virtual;

  procedure SetText( const AText: string );
  procedure GetText( var AText: string );

  ...
end;
```

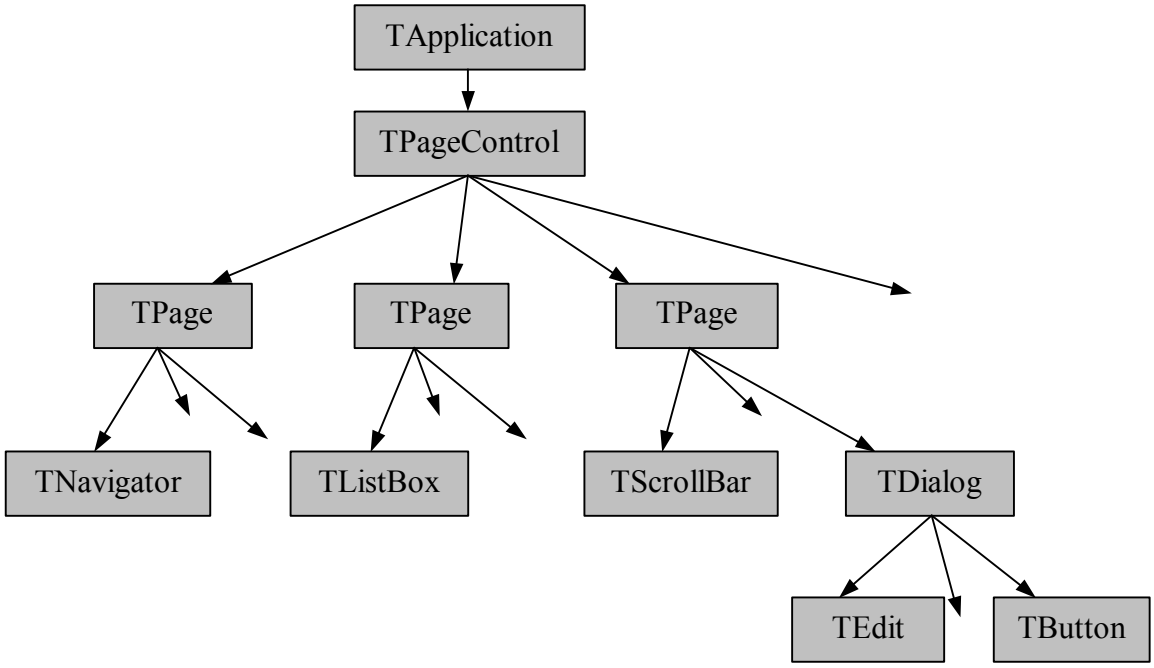
Komponenty nemohou existovat izolovaně. Vždy jsou vloženy do seznamu vlastníka. Vlastník komponent je speciální komponenta – skupina – **TGroup** (viz. kapitola 5.3.5) odvozená z komponenty **TControl**.

```
TGroup = object( TControl )
public
  Current   : PControl;
  Last      : PControl;

  ...

  constructor Init( const ABounds: TRect );
  destructor Done; virtual;

  procedure HandleEvent( var AEvent: TEvent ); virtual;

  procedure Insert( AControl: PControl );
  procedure Delete( AControl: PControl );

  function  FocusNext( AForwards: Boolean ): Boolean;
  procedure SelectNext( AForwards: Boolean );

  function  Execute: Word; virtual;

  ...

end;
```

Třída **TGroup** a její potomci obsahují odkaz na poslední komponentu (komponenta na pozadí) ve skupině (položka **Last**) a vybranou komponentu ve skupině (položka **Current**). Dále umožňuje vložení komponenty do skupiny (metoda **Insert**), odstranění komponenty ze skupiny (metoda **Delete**).

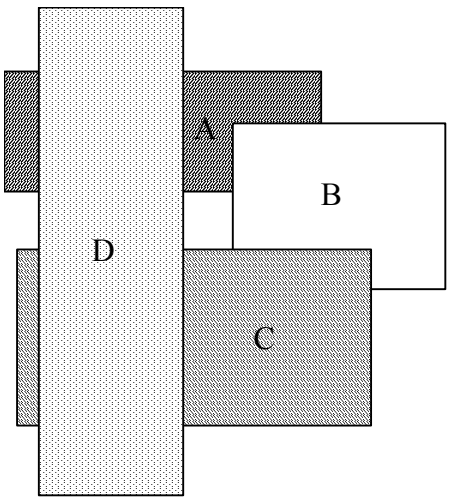
Ze třídy **TGroup** jsou odvozeny např. komponenty typu okno (**TWindow**), stránka (**TPage**) nebo třída zastřešující celou aplikaci (**TApplication**).

Komponenty aplikace jsou organizovány do stromové struktury. Viz. např. následující obrázek:



Na vrcholu stromu je vždy komponenta **TApplication.** Konkrétní propojení ostatních komponent si podle potřeby aplikace zvolí sám programátor. Výše uvedený obrázek je pouze příklad.

Pořadí komponent v seznamu vlastníka určuje jejich pořadí v ose kolmé na rovinu displeje (tzv. Z-order). Komponenty na konci seznamu jsou v případě překrytí komponent zakryty komponentami na začátku seznamu. Viz. následující obrázek, kde pořadí komponent je A (konec) – B – C – D (začátek):



4.2. Vytvoření a inicializace komponent

Předpokládejme, že někde v programu jsou definovány následující proměnné:

```
var
  R     : TRect;
  Text  : PStaticText;
  Group : PGroup;
```

a proměnná Group je již inicializována. Pro vytvoření komponenty a její začlenění do skupiny Group jsou nutné následující tři operace:

```
R.Assign( 32, 32, 160, 48 );
Text := New( PStaticText, Init( R, 'Text' ) );
Group^.Insert( Text );
```

Na první řádce se definuje umístění a rozměr komponenty, tj. inicializuje se struktura R typu **TRect** (viz. dokumentace k standardní knihovně Objects), která se předá jako první parametr konstruktoru komponenty.

Na druhém řádku se vytváří instance komponenty (v tomto případě komponenty **TStaticText** jejíž konstruktor má dva parametry). Prvním parametrem je výše zmíněný obdélník udávající umístění a rozměry komponenty a druhým je zobrazovaný text.

Na třetím řádku se vytvořená komponenta vloží do komponenty Group pomocí metody **Insert**. Metoda **Insert** vloží komponentu vždy na začátek seznamu, tedy do popředí.

4.2.1. Přizpůsobení vlastností komponent

V předchozí kapitole byla vytvořena komponenta a začleněna do skupiny komponent. Takto vytvořená komponenta má všechny parametry nastavené na implicitní hodnoty nastavené v konstruktoru komponenty. Implicitní nastavení parametrů komponenty je uvedeno v dokumentaci k jednotlivým komponentám. Obvykle však potřebujeme některé z parametrů změnit. Nejvhodnější okamžik ke změně parametrů je místo mezi inicializací instance komponenty a jejím vložením do skupiny.

Např.

```
R.Assign( 32, 32, 160, 48 );
Text := New( PStaticText, Init( R, 'Text' ) );
Text^.Flags := Text^.Flags or stfBorder;
Group^.Insert( Text );
```

Třetí řádka nastaví příznak stfBorder v položce Flags instance komponenty. Takto modifikovaná komponenta vykreslí kolem zobrazovaného textu obdélníkový rámeček.

Další z možností jak modifikovat vlastnosti komponenty je použití metody **Customize**.

```
R.Assign( 32, 32, 160, 48 );
Text := New( PStaticText, Init( R, 'Text' ) );
Text^.Customize( ccStaticText );
Group^.Insert( Text );
```

Metoda **Customize** provede automatické nastavení parametrů komponenty podle identifikátoru třídy nastavení komponenty.

Identifikátor třídy nastavení komponenty má vždy prefix cc_ (Component Class), za kterým následuje název třídy komponenty bez úvodního znaku T. Např. ccStaticText, ccScrollBar, ccEdit, ccButton apod (viz. kapitola 5.1.6).

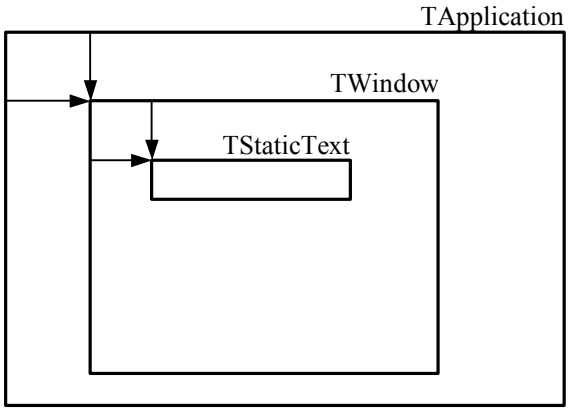
Implicitní nastavení parametrů jednotlivých tříd je součástí knihoven ovladačů konkrétních terminálů – přesněji knihovne z xxxCus.pas (např. T10Cus, T11MCus apod).. knihovny s. Metoda **Customize** obvykle nastavuje barevnou paletu komponenty, mapování kláves a výjimečně další parametry komponenty na implicitní hodnoty vhodné pro konkrétní terminál.

Pokud nestačí implicitní nastavení komponenty pro konkrétní terminál, je možné parametry modifikovat buď vytvořením nové podtřídy nastavení (viz. kapitola 4.8) nebo ruční modifikací parametrů jak bylo ukázáno výše (ale až po volání metody **Customize**). Např.

```
R.Assign( 32, 32, 160, 48 );
Text := New( PStaticText, Init( R, 'Text' ) );
Text^.Customize( ccStaticText );
Text^.Id := cidMyText;
Text^.Flags := Text^.Flags or stfBorder;
Group^.Insert( Text );
```

4.3. Umístění a velikost komponenty

Počáteční rozměry a umístění komponenty se zadává prvním parametrem konstruktoru komponenty. Umístění komponenty je relativní k levému hornímu rohu vlastníka komponenty (tedy skupině, do které je komponenta vložena). Viz. následující obrázek:



Konstruktor všech komponent je definován následovně:

```
constructor Init( const ABounds: TRect; ... );
```

Parametr **ABounds** udává rozměry obdélníku posunutého relativně k levému rohu komponenty skupiny, do které bude vytvořená komponenta vložena.

Parametr ABounds lze inicializovat např. následovně:

```
var
  R : TRect;

R.Assign( 0, 0,  64, 20 );
R.Move( 200, 100 );
HScrollBar := New( PScrollBar, Init( R ) );
```

Pro změnu pozice nebo velikosti komponenty za běhu aplikace slouží tři metody:

```
procedure TControl.Locate( var ABounds: TRect );
procedure TControl.MoveTo( AX, AY: Integer);
procedure TControl.GrowTo( AX, AY: Integer);
```

Metoda **Locate** umožňuje změnit zároveň rozměry a umístění komponenty. Metoda **MoveTo** umožňuje přesunout levý horní roh komponenty na jiné místo při zachování rozměrů komponenty a metoda **GrowTo** umožňuje změnit rozměry komponenty při zachování umístění levého horního rohu komponenty.

4.4. Změna základních stavů komponenty

Základní stav komponenty je uložen v položce TControl.State (viz. kapitola 5.3.4.8). Položka State obsahuje několik nezávislých příznaků např.

Identifikátor	Kód	Popis
sfVisible	$0001	Pokud je nastaven příznak sfVisible, pak je komponenta viditelná, pokud není zcela překryta jinou komponentou.
sfFocused	$0002	Pokud je nastaven příznak sfFocused, pak se komponenta nachází v ohnisku (má fokus) a přijímá události od klávesnice.
sfDisabled	$0008	Pokud je nastaven příznak sfDisabled, pak je komponenta zakázaná a nepřijímá žádné vstupní události.

Interně se modifikace jednotlivých bitů položky State provádí pomocí metody **TControl.SetState** (viz. kapitola 5.3.4.32). Z aplikace je možné měnit stav komponenty jen pomocí speciálních funkcí popsaných v následujících kapitolách.

4.4.1. Zobrazení a skrytí komponenty

Za běhu aplikace lze libovolnou komponentu skrýt a opětovně zobrazit. K tomu slouží dvě metody:

```
procedure TControl.Show;
procedure TControl.Hide;
```

Metoda **Hide** skryje komponentu. Pokud komponenta je vlastník jiných komponent, pak jsou skryty zároveň i všechny komponenty, které jsou umístěny v této komponentě. Metoda **Show** zobrazí komponentu, která byla dříve skryta pomocí metody **Hide**.

Metody **Show** a **Hide** zároveň generují oznámení nmShow a nmHide (viz. kapitola 5.1.3).

4.4.2. Zákaz a povolení komponenty

Za běhu aplikace lze libovolnou komponentu zakázat a opětovně povolit. Tzv. zakázaná komponenta nemůže přijímat žádné vstupní události, takže se vůči uživateli chová, jako kdyby byla nefunkční. Komponenta může zakázaný stav odlišit od povoleného např. jiným zobrazením komponenty. Pro zákaz a povolení komponenty se používají dvě metody:

```
procedure TControl.Enable;
procedure TControl.Disable;
```

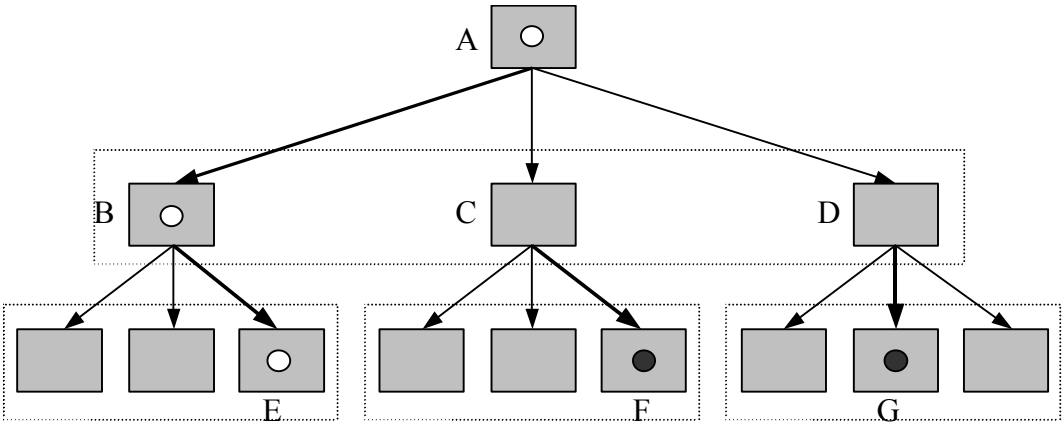
Metoda **Enable** komponentu povolí a metoda **Disable** komponentu zakáže.

Metody **Enable** a **Disable** zároveň generují oznámení nmEnable a nmDisable (viz. kapitola 5.1.3).

4.4.3. Změna ohniska (fokus)

Komponenta, která se nachází v ohnisku přednostně zpracována události od klávesnice. Obvykle je komponenta v ohnisku nějak graficky označena, tak aby bylo zřejmé, že stisk klávesy způsobí změnu právě v této komponentě.

Ve skupině komponent se může nacházet v ohnisku maximálně jedna komponenta, viz. následující obrázek:



Na obrázku výše je schématicky znázorněn strom komponent. Komponenta označená písmenem A na vrcholu stromu je vždy v ohnisku. Pouze jedna z komponent

vložených do komponenty A může být vybraná. V tomto případě se jedná o komponentu B. V rámci komponenty B je vybraná komponenta označená písmenem E. **Komponenty A, B, E se nacházejí v ohnisku**. V rámci komponenty C resp. D jsou vybrány komponenty F resp. G. Komponenty F, G se nenacházejí v ohnisku, protože jejich vlastníci, tj. komponenty C, D se nenacházejí v ohnisku.

U komponent lze tedy rozlišit tři možné stavy:

- Komponenta není vybraná
- Komponenta je vybraná
- Komponenta je vybraná a nachází se v ohnisku

Tyto tři stavy jsou odlišeny příznaky sfSelected a sfFocused v položce State třídy **TControl** (viz. kapitola 5.3.4.8).

K přepnutí ohniska nebo výběru komponenty lze použít následující metody.

```
function  TControl.Focus: Boolean;
procedure TControl.Select;
function  TGroup.FocusNext( AForwards: Boolean ): Boolean;
procedure TGroup.SelectNext( AForwards: Boolean);
```

Metoda **Select** provede výběr komponenty v rámci skupiny. Pokud je vlastník komponenty v ohnisku, dostane se do ohniska i tato komponenta.

Metoda **Focus** provede výběr komponenty v rámci skupiny a zároveň rekurzivně zavolá metodu **Focus** vlastníka komponenty. V podstatě dojde k výběru všech komponent na cestě ke kořeni stromu komponent. V případě, že se podařilo všechny komponenty na cestě ke kořeni stromu vybrat, vrátí metoda hodnotu True. V opačném případě, kdy např. jedna z komponent nepovolí přepnutí (viz. kapitola 5.3.4.67), metoda vrací hodnotu False.

Metoda **SelectNext** provádí výběr následující nebo předchozí komponenty v seznamu vlastníka a metoda **FocusNext** přepíná ohnisko na následující nebo předchozí komponentu vlastníka.

Tyto metody je možné ručně volat z aplikace, jinak jsou obvykle automaticky volány např. při stisku klávesy pro přesun na následující příp. předchozí komponentu, při přepnutí stránky nebo okna apod.

4.5. Vykreslování komponent

Každá komponenta si pamatuje svůj stav, tak aby mohla být v libovolném okamžiku automaticky vykreslena. Komponenty jsou vykreslovány např. tehdy pokud je zavolána metoda **Show**, nebo v případě, že byla komponenta zakryta jinou komponentou a tento stav se změnil, nebo v případě, že se změnil stav komponenty.

K překreslení komponenty slouží metoda

```
procedure TControl.Repaint;
```

Po zavolání této metody se provedou následující akce:

1) Inicializuje se instance třídy **TCanvas**, tzv. kreslící plátno (viz. dále)
2) Zavolá se funkce **BeforePaint**
3) Zavolá se metoda **TControl.Paint**
4) Zavolá se metoda **AfterPaint**
5) Instance třídy **TCanvas** se uvolní

Virtuální metoda **Paint** je deklarována následovně:

```
procedure TControl.Paint( ACanvas: PCanvas ); virtual;
```

Metodu **Paint** musí potomci třídy **TControl** tj. prakticky všechny komponenty předefinovat. Jejím úkolem je vykreslit komponentu do obdélníkové oblasti dané rozměry a umístěním komponenty. Např. následující implementace procedury **Paint**, vyplní celý obdélník komponenty černou barvou.

```
procedure TControlXXX.Paint( ACanvas: PCanvas );
var
  R : TRect;
begin
  { Ulozeni rozmeru komponenty do promenne R }
  GetExtent( R );

  with ACanvas^ do
  begin
     { Nastaveni cerneho stetce }
     Brush.Color := clBlack;
     { Vyplneni obdelniku }
     FillRect( R.A.X, R.A.Y, R.B.X – 1, R.B.Y – 1 );
  end;
end;
```

Třída **TCanvas** implementuje funkce pro kreslení do oblasti komponenty, např. metody pro vykreslení bodu (**DrawPoint**), čáry (**DrawLine**), obdélníku (**DrawRect**), kružnice (**DrawCircle**), textu (**DrawText**) apod. Není potřeba se starat o to, zda je komponenta do níž se kreslí např. z částí překrytá jinou komponentou, toto vše řeší třída **TCanvas** a její metody. Detailní popis třídy **TCanvas** je uveden v kapitole 5.3.2).

Třída **TControl** obsahuje položky nazvané **BeforePaint** a **AfterPaint** definované následovně:

```
BeforePaint  : TPaintHandler;
AfterPaint   : TPaintHandler;
```

Typ **TPaintHandler** je procedurální typ:

```
TPaintHandler = procedure( AControl: PControl; ACanvas: PCanvas );
```

Položky **BeforePaint** a **AfterPaint** slouží k vyvolání aplikačních funkcí pro dodatečné kreslení do komponenty. Tento mechanismus umožňuje drobné změny ve vykreslování komponent bez toho, aby bylo potřeba vytvářet potomky tříd komponent.

Jedná se např. o vykreslení rámečku nebo různých doplňujících značek do komponent.

Funkce **AfterPaint** se volá po návratu metody **Paint**, tj. vždy poté, co se komponenta sama vykreslila. Naproti tomu funkce **BeforePaint** se volá před voláním metody **Paint,** a proto její použití má význam pouze u některých komponent, např. pro vykreslení uživatelského pozadí komponenty.

4.5.1. Paleta komponenty

Každá komponenta používá pro vlastní vykreslování alespoň dvě barvy – barvu pozadí a barvu popředí. Složitější komponenty si obvykle se dvěmi barvami nevystačí. Např. i tak jednoduchá komponenta jakou je statický text používá čtyři barvy a to:

- barvu pozadí
- barvu rámečku vlevo a nahoře
- barvu rámečku vpravo a dole
- barvu textu

Aby bylo možné, zobrazovat různé instance komponent v odlišných barvách, obsahuje každá instance komponenty tzv. paletu. Paleta je jednoduchá tabulka barevných konstant. Index do této tabulky označuje určitou část komponenty a hodnota na dané pozici určuje barvu této částí. Délka tabulky je dána potřebami konkrétní komponenty (např. zmíněná komponenta statický text má v této tabulce 4 hodnoty). Při vykreslování komponenty se barvy čtou přímo z této tabulky pomocí metody **TControl.GetColor**:

```
function TControl.GetColor( AIndex: Byte ): TColorRef;
```

Paleta je implementovaná jako řetězec znaků o určité délce (typ **string**). Ukazatel na tento řetězec je uložen v položce TControl.Palette (viz. kapitola 5.3.4.10). Paleta se tedy definuje jako řetězec znaků s identifikátorem s prefixem C, např.

```
const
  CStaticText = #$01#$08#$07#$0F;
```

Celou Paletu komponenty lze nastavit pomocí metody **TControl.SetPalette** (viz. kapitola 5.3.4.55).

```
Text^.SetPalette( CStaticText, True );
```

Jednotlivé barvy palety lze modifikovat pomocí metody **TControl.SetColor** (viz. kapitola 5.3.4.56). Např. následující příklad nastaví u komponenty **TStaticText** pozadí textu na černou barvu.

```
Text^.SetColor( 1, clBlack );
```

Každá komponenta má vlastní implicitní paletu nastavovanou v konstruktoru instance. Tato paleta je uzpůsobena pro displeje se dvěmi barvami. Ovladače jednotlivých terminálů definují vlastní třídy nastavení komponent (viz. kapitola 4.2.1), jejichž součástí je i nastavení barevných palet. To znamená, že pokud při vytváření

komponenty použijeme metodu **Customize**, bude paleta barev nastavena na implicitní hodnoty pro konkrétní typ terminálu či displeje.

4.5.2. Font komponenty

Mnohé komponenty vykreslují také nějaký text. Tento text mohou vykreslit libovolným fontem a nebo mohou použít tzv. font komponenty.

Pro nastavení a zjištění fontu komponenty slouží metody **SetFont** a **GetFont**:

```
procedure SetFont( AFont: Word );
function  GetFont: Word;
```

Metoda **GetFont** vrací identifikátoru fontu, tj. konstantu s prefix fid_ (font id), viz. dokumentace ke knihovně Fonts.

Implicitně mají všechny komponenty nastaven font na hodnotu fidDefault. Což znamená, že pro vykreslování textu mají použít font svého vlastníka. Pokud má i vlastník font nastaven na tuto hodnotu, použije se font vlastníka této komponenty atd. rekurzivně. V případě, že je implicitní font nastaven na hodnotu jinou než fidDefault, jedná se o konkrétní identifikátor fontu a tento mechanismus se neuplatní.

Příklad. metoda **Paint** využívající metodu **GetFont**.

```
procedure TControlXXX.Paint( ACanvas: PCanvas );
var
  R : TRect;
begin
  { Ulozeni rozmeru komponenty do promenne R }
  GetExtent( R );

  with ACanvas^ do
  begin
    { Nastaveni barvy pozadi }
    Brush.Color := GetColor( 1 );
    { Vyplneni obdelniku }
    FillRect( R.A.X, R.A.Y, R.B.X - 1, R.B.Y - 1 );
    { Nastaveni barvy pisma }
    Font.Color := GetColor( 2 );
    { Nastaveni fontu }
    Font.Id := GetFont;
    { Vykresleni textu fontu }
    DrawTextRect( R, Text, tfCenterX + tfCenterY );
  end;
end;
```

4.6. Vstupní události (Event)

Každé vstupní zařízení (klávesnice, myš apod.) generuje tzv. vstupní události, které jsou obsluhovány systémem komponent. Vstupní událost je popsána obecnou strukturou **TEvent** definovanou níže:

```
TEvent = record
  Code : Word;                 { Konstanta evXXX              }
  case Integer of
    0: ( WParam   : Word;      { Obecný parametr Word         }
         LParam   : Longint    { Obecný parametr Longint      }
       );
    1: ( Buttons  : Word;      { Stisknutá tlačítka           }
         Pos      : TPoint     { Pozice ukazatele             }
       );
    2: ( KeyCode  : Word;      { Kód klávesy                  }
         VirtKey  : Word;      { Kód virtuální klávesy        }
         CharCode : Char       { ASCII znak                   }
       );
    3: ( TimerId  : Word;      { Identifikátor časovače       }
         Control  : Pointer    { Odkaz na komponentu          }
       );
    4: ( Command  : Word;      { Příkaz oběžníku              }
         Param    : Pointer ); { Parametr uživatelské zprávy  }
end;
```

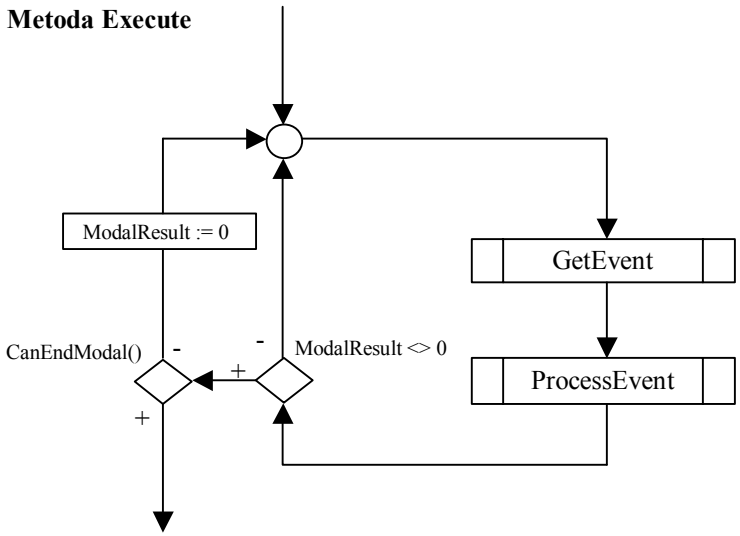
Struktura **TEvent** obsahuje položku **Code**, která jednoznačně určuje typ události. Tato položka může obsahovat jednu z konstant s prefixem ev_. Jejich úplný seznam je uveden v následující tabulce:

Identifikátor	Kód	Popis
evNothing	$0000	Prázdná (neplatná) událost
evKeyDown	$0001	Stisk klávesy
evMouseMove	$0002	Pohyb ukazatele myši
evMouseDown	$0004	Stisk tlačitka myši
evMouseUp	$0008	Uvolnění tlačítka myši
evMouseDbl	$0010	Dvojklik myši
evMouseRep	$0020	Držení tlačítka myši (automatické opakování)
evTimer	$0040	Vypršení limitu časovače
evBroadcast	$0080	Oběžník
evMessage	$0100	Zpráva

Ostatní položky struktury **TEvent** upřesňují další vlastnosti události. Úplný popis struktury **TEvent** je uveden v dokumentaci ke knihovně IoDrv.

4.6.1. Zpracování vstupních událostí

Události vstupních zařízení jsou obsluhovány ve smyčce metody **TGroup.Execute**, která je schématicky znázorněna na následujícím obrázku.



Metodu **Execute** poskytují všechny komponenty (až na výjimky pouze potomci třídy TGroup však tuto metodu implementují). Metoda provádí zpracování událostí v tzv. modální stavu. Po spuštění aplikace je zavolána metoda **Execute** instance třídy **TApplication**. Při otevření dialogového okna je zavolána metoda **Execute** třídy **TDialog**.

Vstupní události jsou vybírány z front ovladačů vstupních zařízení pomocí metody **TControl.GetEvent**. Implicitně tato metoda volá metodu **GetEvent** svého vlastníka, až je nakonec zavolána metoda **TApplication.GetEvent**. Tato metoda zajišťuje mimo jiné volání metody **Tick** ovladače displeje, ovladače vstupních zařízení a zajišťuje výběr vstupní události.

Metoda **ProcessEvent** provádí zpracování vstupní události. Postupně předá vstupní událost proceduře **BeforeHandle**, metodě **HandleEvent** a nakonec proceduře **AfterHandle** (viz kapitola 4.6.2).

4.6.2. Obsluha vstupních událostí

Vstupní události přicházející do komponenty jsou obsluhovány virtuální metodou **TControl.HandleEvent**.

```
procedure HandleEvent( var AEvent: TEvent ); virtual;
```

Typický kód metody **HandleEvent** vypadá následovně:

```
procedure TControlXXX.HandleEvent( var AEvent: TEvent );
begin
  inherited HandleEvent( AEvent );

  case Event.Code of
    evKeyDown:
    begin
      ...

      if událost_obsloužena then ClearEvent( AEvent );
    end;

    evMouseDown:
```

```
    begin
      ...

      if událost_obsloužena then ClearEvent( AEvent );
    end;

    ...

  end;
end;
```

Pokud metoda **HandleEvent** obslouží událost, pak by měla zavolat metodu **ClearEvent** (viz. kapitola 5.3.4.37), aby se událost nešířila do dalších komponent.

Aplikace může změnit chování obsluhy událostí pomocí dvou položek třídy **TControl**, položek **BeforeHandle** a **AfterHandle**. Tyto položky jsou definovány následovně:

```
BeforeHandle : TEventHandler;
AfterHandle  : TEventHandler;
```

Typ **TEventHandler** je procedurální typ:

```
TEventHandler = procedure( AControl: PControl; var AEvent: TEvent );
```

Procedura **BeforeHandle** je volána před obsluhou událostí komponentou, tj. před metodou **HandleEvent** a funkce **AfterHandle** je volána po této obsluze. Pokud funkce **BeforeHandle** nebo **AfterHandle** obslouží událost, měla by zavolat funkci **ClearEvent** (viz. kapitola 5.4.3), která zabrání šíření události do dalších obslužných rutin.

4.6.3. Předávání vstupních událostí komponentám

Vstupní události jsou vyzvedávány z front vstupních zařízení komponentou v modálním stavu, obvykle se jedná o komponentu **TApplication**, případně se může jednat např. o dialog v modálním stavu (komponenta **TDialog**). Všechny tyto komponenty jsou potomky třídy **TGroup**. Vstupní událost je nejprve předána jejich metodě **HandleEvent**.

Metoda **TGroup.HandleEvent** provádí roztřídění vstupních události podle typu a jejich správné předání vloženým komponentám. Události můžeme rozdělit do třech skupin: ohniskové události, poziční události a ostatní události.

4.6.3.1. Ohniskové události

Ohniskové události jsou takové události, které jsou předávány komponentám, jenž jsou v ohnisku. V současné době se jedná prakticky pouze o událost **evKeyDown**.

Ohniskové události jsou zpracovávány ve třech fázích ve třech fázích. V první fázi je komponenta **TGroup** rozešle všem vlastněným komponentám, které mají nastaven příznak **ofPreProcess** v položce **Options**. Poté se událost předá vybrané komponentě (položka **Current**) a nakonec se rozešle všem vlastněným komponentám, které mají

nastaven příznak **ofPostProcess** v položce **Options**. Tento mechanismus umožňuje i komponentám, které nejsou v ohnisku zpracovat události od klávesnice (např. obsluha hot-keys apod.). Fází ve které se zpracování události nachází lze rozlišit v metodě **HandleEvent** pomocí položky **Phase** vlastníka, tj. Owner^.Phase, která může obsahovat jednu ze tří hodnot **phPreProcess**, **phFocused**, **phPostProcess**. Např.

```
procedure TControlXXX.HandleEvent( var AEvent: TEvent );
begin
  inherited HandleEvent( AEvent );

  case Event.Code of
    evKeyDown:
    begin
      if Owner^.Phase = phPostProcess then
      begin
         ...
      end;
    end;

    ...

  end;
end;
```

4.6.3.2. Poziční události

Poziční události jsou takové události, které nesou informaci o souřadnicích vzniku události a jsou předávány pouze těm komponentám, které pokrývají oblast, ve které událost vznikla. Jedná se o události **evMouseDown**, **evMouseUp**, **evMouseMove**, **evMouseDbl** a **evMouseRep**.

Ohniskové události jsou předávány pouze jedné komponentě ve skupině, a to té která pokrývá oblast, ve které událost vznikla. Pokud ve skupině nachází více takových komponent, pak je událost předána té, která je nejvíce na vrchu (viz. kapitola 4.1).

4.6.3.3. Ostatní události

Události, které nelze zařadit ani do jedné z předchozích skupin jsou **evMessage**, **evBroadcast** a **evTimer**. Tyto události jsou předány všem komponentám ve skupině.

Výjimkou je událost evTimer. Pokud událost evTimer je cílena na konkrétní komponentu (položka Control struktury **TEvent** obsahuje ukazatel na komponentu), pak není obsluhována standardním způsobem, ale již v metodě **TApplication.GetEvent** je zavolána metoda **ProcessEvent** konkrétní komponenty. Pokud položka Control struktury **TEvent** obsahuje hodnotu **nil**, pak je událost zpracována stejně jako oběžník.

4.6.4. Mapování kláves

Komponenty reagující na vstup z klávesnice (prakticky se jedná o téměř všechny komponenty) obsluhují vstupní událost o stisku klávesy. Tyto události jsou generovány ovladačem klávesnice **TKeyboardDriver** (viz. manuál ke knihovně IoDrv). Ovladač klávesnice vytvoří událost (struktura **TEvent**) a nastaví její položky

KeyCode, CharCode. Komponenta může při obsluze této události vycházet přímo z těchto položek. Protože jsou však komponenty implementovány tak, aby je bylo možné použít na různých typech terminálu, které nemusí vždy obsahovat potřebné klávesy, vychází se při obsluze z kódu tzv. virtuální klávesy. Kód virtuální klávesy je uložen v položce VirtKey struktury události.

Každá komponenta obsahuje položky **KeyMapper**, jedná se o ukazatel na funkci provádějící překlad kódu kláves kb_ na kódy virtuálních kláves vk_.

```
KeyMapper : TKeyMapper;
```

Typ **TKeyMapper** je procedurální typ definovaný následovně:

```
TKeyMapper = function( KeyCode: Word ): Word;
```

Při překladu kódu klávesy se postupuje tak, že nejprve je zavolána funkce **KeyMapper** komponenty umístěné v kořenu stromu komponent (tj. třídy TApplication), a poté se postupuje dolů až ke komponentě, pro kterou se překlad provádí. Lze tedy provádět překlad kláves na libovolné úrovni stromu komponent.

V knihovně Controls je definována funkce **DefaultKeyMapper**, která provádí standardní překlad kláves a je implicitně přiřazena položce KeyMapper instance třídy **TApplication**.

```
function DefaultKeyMapper( KeyCode: Word ): Word; far;
begin
  case KeyCode of
    kbEnter      : DefaultKeyMapper := vkEnter;
    kbEsc        : DefaultKeyMapper := vkEsc;
    kbTab        : DefaultKeyMapper := vkNext;
    kbShiftTab   : DefaultKeyMapper := vkPrev;
    kbLeft       : DefaultKeyMapper := vkLeft;
    kbRight      : DefaultKeyMapper := vkRight;
    kbUp         : DefaultKeyMapper := vkUp;
    kbDown       : DefaultKeyMapper := vkDown;
    kbInsert     : DefaultKeyMapper := vkInsert;
    kbDelete     : DefaultKeyMapper := vkDelete;
    kbBackSpace  : DefaultKeyMapper := vkBackSpace;
    kbHome       : DefaultKeyMapper := vkHome;
    kbEnd        : DefaultKeyMapper := vkEnd;
    kbPageUp     : DefaultKeyMapper := vkPageUp;
    kbPageDown   : DefaultKeyMapper := vkPageDown;
    kbF1         : DefaultKeyMapper := vkHelp;
    kbAltX       : DefaultKeyMapper := vkAppExit;
    kbF6         : DefaultKeyMapper := vkNextWindow;
    kbShiftF6    : DefaultKeyMapper := vkPrevWIndow;
    kbClear      : DefaultKeyMapper := vkClear;
    kbF10        : DefaultKeyMapper := vkMenu;
  else
    DefaultKeyMapper := 0;
  end;
end;
```

4.7. Oznámení (Notification)

Oznámení jsou krátké zprávy komponent určené uživatelské aplikaci. Oznámení

slouží k informování aplikace např. obvykle o změně stavu komponenty. Oznámení je uloženo ve struktuře **TNotification** (viz. kapitola 5.2.3):

```
TNotification = record
  Control : PControl;
  Code    : Word;
  case Integer of
    0: ( WParam  : Word;
         LParam  : Longint;
         LParam2 : Longint );

    1: ( __Res1  : Word;
         WParam2 : Word;
         WParam3 : Word;
         WParam4 : Word;
         WParam5 : Word;
       );

    2: ( Result  : Word );

    3: ( ItemId  : Word );

    4: ( Index   : Integer );

    5: ( Row     : Integer;
         Col     : Integer );

    6: ( __Res2  : Word;
         __Res3  : Word;
         Text    : PString );

    7: ( __Res4  : array[0..7] of Byte;
         Accept  : Boolean;
       );
    8: ( ErrCode : Integer );
end;
```

Položka Control obsahuje odkaz na komponentu, která oznámení vygenerovala. Položka Code obsahuje identifikátor oznámení, tj. konstantu s prefixem nm_ (viz. kapitola 5.1.3). Ostatní položky struktury upřesňují parametry oznámení.

4.7.1. Generování a obsluha oznámení

Pro vygenerování oznámení slouží metody **TControl.Notify** příp. **TControl.NotifyEx** (viz. kapitola 5.3.4.36, 5.3.4.35). Metoda **Notify** umožňuje odeslat jednoduché oznámení bez parametrů:

```
procedure Notify( ACode: Word );
```

Metoda **NotifyEx** umožňuje odeslat libovolné oznámení, volající musí však kompletně vyplnit parametr ANotification.

```
procedure NotifyEx( var ANotification: TNotification );
```

4.7.2. Obsluha oznámení

V okamžiku kdy komponenta vygeneruje oznámení (tj. když zavolá metodu Notify nebo NotifyEx), dochází k volání obsluhy oznámení. K tomu obsahuje každá

komponenta metodu **HandleNotification**:

```
procedure HandleNotification( var ANotification: TNotification );
            virtual;
```

Metoda **HandleNotification** je u většiny komponent prázdná a potomek komponenty může tuto metodu předefinovat. Oznámení jsou určené převážně uživatelské aplikaci a proto samotné komponenty obsluhují oznámení jen ve speciálních případech.

Aby mohla uživatelská aplikace obsluhovat oznámení, obsahuje každá komponenta položky **BeforeNotify** a **AfterNotify**.

```
BeforeNotify : TNotificationHandler;
AfterNotify  : TNotificationHandler;
```

Typ **TNotificationHandler** je procedurální typ definovaný následovně:

```
TNotificationHandler = procedure( var ANotification: TNotification );
```

Funkce **BeforeNotify** resp. **AfterNotify** je volána těsně před resp. za voláním metody **HandleNotification**.

V okamžiku kdy jedna funkcí **BeforeNotify**, **AfterNotify** nebo **HandleNotification** obslouží oznámení měla by zavolat funkci **ClearNotification** (viz. kapitola 5.4.4), která zabrání dalšímu šíření oznámení. Pokud toto obsluha neučiní, pak je oznámení automaticky předáno vlastníkovi komponenty, u něhož se provede opět volání funkcí **BeforeNotify**, **HandleNotification** a **AfterNotify**. Neobsloužené oznámení tedy jakoby putuje od komponenty, která jej vygenerovala ke kořenu stromu komponent. Obvykle se této vlastnosti využívá, viz. následující příklad:

Předpokládejme, že aplikace obsahuje okno (**TWindow**) se dvěmi tlačítky ‘OK‘a ‘Cancel‘ a jednou komponentou **TListView** (seznam položek). Tyto komponenty mají identifikátory postupně cidOkButton, cidCancelButton a cidListView. Následující funkce ukazuje jak provádět obsluhu oznámení všech třech komponent zároveň. Ukazatel na tuto funkci musí být přiřazen položce **AfterNotify** (příp. **BeforeNotify**) instance komponenty okna, ve které jsou obsluhované komponenty vloženy.

```
procedure MyWindowNotify( var N: TNotification ); far;
begin
  case N.Control^.Id of

    cidOkButton: { Obsluha oznámení tlacitka OK }
    begin
      if N.Code = nmClick then
      begin
        ... (obsluha)

        ClearNotification( N );
      end;
    end;

    cidCancelButton: { Obsluha oznámení tlacitka Cancel }
    begin
       ... (obsluha)
    end;
```

```
    cidListView: { Obsluha oznámení komponenty TListView }
    begin
      case N.Code of

        nmGetData: { Obsluha oznámení nmGetData }
        begin
          ... (obsluha)
          ClearNotification( N );
        end;

        nmChange: { Obsluha oznámení nmChange }
        begin
          ... (obsluha)
          ClearNotification( N );
        end;
      end;
    end;

  end;
end;
```

4.8. Vytvoření vlastní třídy nastavení komponenty

V případě, že se v aplikaci používají často komponenty s identickým nastavením parametrů, je možné vytvořit vlastní třídu nastavení komponenty. Nejprve je nutné deklarovat identifikátory tříd (viz. kapitola 5.1.6).

```
  const

    ccStaticTextRed  = ccStaticText + 16;
    ccEditBlue       = ccEdit       + 16;
```

V dalším kroku je potřeba vytvořit konfigurační proceduru, která bude provádět nastavení parametrů komponent podle zadané třídy nastavení. Viz. následující příklad:

```
  procedure MyCustomizeProc( AControl: PControl; AClass: Word ); far;
  const
    { Paleta pro TStaticText se zlutym pismem na cervenem pozadi }
    CStaticTextRed : string[4]  = #$04#$08#$0C#$0E;
    { Paleta pro TEdit s bilym pismem na modrem pozadi }
    CEditBlue      : string[4]  = #$01#$00#$0B#$0F;
  begin

    { Vyber podle tridy }
    case AClass and ccGeneralMask of

      ccStaticText: { Komponenta TStaticText }
      begin
        { Vyber podle podtridy }
        case AClass of

          ccStaticTextRed:
          begin
            { Zmena implicitni palety }
            AControl^.SetPalette( CStaticTextRed, True );
          end;

        end;
      end;
```

```
      ccEdit: { Komponenta TEdit }
      begin
        { Vyber podle podtridy }
        case AClass of
          ccEditBlue:
          begin
            { Zmena implicitni palety }
            AControl^.SetPalette( CEditBlue, True );
          end;
        end;
      end;

    end;
  end;
```

Konfigurační proceduru je potřeba registrovat dříve než se uživatelské třídy budou používat, nejlépe ihned po startu aplikace pomocí procedury **RegisterCustomizeProc** (viz. kapitola 5.4.1).

Např.

```
RegisterCustomizeProc( MyCustomizeProc, cplUser );
```

První parametr procedury je ukazatel na konfigurační proceduru, druhý parametr je typu Integer a udává pořadí v jakém se budou konfigurační procedury volat v případě, že jich je registrováno více. Standardně jsou definovány následující úrovně:

Identifikátor	Hodnota	Popis
cplDefault	1000	Konfigurační procedury komponent pro implicitní nastavení dané typem terminálu. Na této urovní registruje ovladač terminálu své implicitní konfigurační procedury.
cplUser	2000	Uživatelské konfigurační procedura.

Konfigurační procedury registrované s nižší hodnotou úrovně jsou vyvolány dříve, než procedury s vyšší úrovní.

5. Reference

5.1. Konstanty

5.1.1. Konstanty tf_

Konstanty s prefixem tf_ řídí vykreslování textu pomocí metodu **DrawTextRect** třídy **TCanvas** (viz. kapitola 5.3.2.23).

Identifikátor	Kód	Popis

tfLeft	$00	Zarovnání textu doleva
tfRight	$01	Zarovnání textu doprava
tfCenterX	$02	Zarovnání textu na střed v ose X
tfTop	$00	Zarovnání textu nahoru
tfBottom	$04	Zarovnání textu dolů
tfCenterY	$08	Zarovnání textu na střed v ose Y
tfMultiline	$10	Výpis textu na více řádků

Konstanty lze kombinovat pomocí logického součinu. Nelze však navzájem kombinovat konstanty tfLeft, tfRight a tfCenterX (resp. tfTop, tfBottom a tfCenterY).

5.1.2. Konstanty cm_

Konstanty s prefixem cm_ jsou identifikátory oběžníků a zpráv posílaných komponentám. Oběžník resp. zpráva je událost typu evBroadcast resp. evMessage. Kód zprávy je uložen v položce Command struktury TEvent. Pro další upřesnění zprávy lze použít parametr Param struktury TEvent.

V knihovně Controls jsou nadefinovány tyto konstanty:

Identifikátor	Kód	Popis
cmScrollBarChanged	1	Zprávu cmScrollBarChanged generuje komponenta TScrollBar v případě změny posuvníku. Odkaz na komponentu, která vygenerovala tuto událost je uveden v parametru Param.
cmPageCall	2	Na zprávu cmPageCall reaguje komponenta TPageControl s identifikátorem uvedeným v horním slově parametru Param přepnutím na stránku uvedenou v dolním slově parametru Param.
cmSetText	3	Na zprávu cmSetText reagují komponenty TStaticText a TEdit nastavením textu komponenty. Odkaz na text je uveden v parametru Param
cmCalibPageCall		

5.1.3. Konstanty nm_

Konstanty s prefixem nm_ identifikují typ oznámení (Notification).

První skupina oznámení uvedená v nasledující tabulce je generována všemi typy komponent.

Identifikátor	Kód	Popis
nmNothing	0	Identifikátor pro neplatné oznámení.
nmEnable	1	Oznámení generované poté, co byla komponenta povolena (voláním metody Enable třídy TControl). Toto oznámení nemá žádné parametry.
nmDisable	2	Oznámení generované poté, co byla komponenta zakázána

		(voláním metody Disable třídy TControl). Toto oznámení nemá žádné parametry.
nmShow	3	Oznámení generované poté, co byla komponenta zobrazena (voláním metody Show třídy TControl). Toto oznámení nemá žádné parametry.
nmHide	4	Oznámení generované poté, co byla komponenta skryta (voláním metody Hide třídy TControl). Toto oznámení nemá žádné parametry.
nmEnter	5	Oznámení generované poté, co se komponenta dostane do ohniska (získá fokus). Toto oznámení nemá žádné parametry.
nmExit	6	Oznámení generované poté, co se komponenta dostane mimo ohnisko (ztratí fokus). Toto oznámení nemá žádné parametry.
nmEndModal	7	Oznámení generované poté, co se uživatel pokusil ukončit modální stav dialogu. Tuto událost generuje třída TGroup a její potomci. Způsob ukončení modálního stavu je uložen v parametru Result (viz. kapitola 5.1.7). Obsluha oznámení může zakázat ukončení modálního stavu nastavení parametru Accept na hodnotu False.
nmCanExit	8	Oznámení generované těsně předtím, než se komponenta dostane mimo ohnisko (ztratí fokus). Obsluha oznámení může zakázat přechod komponenty mimo ohnisko tak, že nastaví položku Accept na hodnotu False.
nmError	9	Oznámení generováné při chybě validátoru komponenty metodou TControl.Error (viz. kapitola 5.3.4.66). Chybový kód validátoru je uložen v parametru ErrCode oznámení. Pokud obsluha oznámení může opravit hodnotu komponenty a nastavit položku Accept na hodnotu True, validace projde úspěšně.
nmTimer	10	Oznámení generované při vypršení časovače naplánovaného metodou TControl.ScheduleTimer (viz. kapitola 5.3.4.82). Identifikátor časovače je uložen v parametru TimerId.
nmHelp	11	Oznámení generované komponentou v ohnisku při stisku virtuální klávesy vkHelp.

Druhá skupina oznámení je generována jen některými komponentami. Oznámení se mohou lišit v parametrech. Přesný popis parametrů těchto oznámení je vždy uveden v popisu konkrétní komponenty.

Identifikátor	Kód	Popis
nmChange	16	Toto oznámení generují komponenty v případě, že se změnil jejich stav (TScrollBar, TListBox, TListView, TEdit apod.)
nmSelect	17	Toto oznámení generují komponenty TListBox a TListView při výběru položky buď stiskem klávesy Enter nebo dvojklikem na položku. Oznámení nemá žádné parametry.
nmGetData	18	Toto oznámení generují komponent jako žádost o uživatelská data, která se mají komponentou zobrazit

		(TListBox, TListView apod.)
nmClick	19	Toto oznámení generují některé komponenty (např. TButton, TPaintBox apod.) v okamžiku kdy uživatel klikne na oblast, kterou pokrývají. Obvykle toto oznámení nemá žádný parametr.

5.1.4. Konstanty vk_

Konstanty s prefixem vk_ identifikují tzv. virtuální klávesy.

Komponenty reagující na vstup z klávesnice (prakticky se jedná o téměř všechny komponenty) obsluhují vstupní událost o stisku klávesy. Tyto události jsou generovány ovladačem klávesnice **TKeyboardDriver** (viz. manuál ke knihovně IoDrv). Ovladač klávesnice vytvoří událost (struktura **TEvent**) a nastaví její položky KeyCode, CharCode. Komponenta může při obsluze této události vycházet přímo z těchto položek. Protože jsou však komponenty implementovány tak, aby je bylo možné použít na různých typech terminálu, které nemusí vždy obsahovat potřebné klávesy, vychází se při obsluze z kódu tzv. virtuální klávesy. Kód virtuální klávesy je uložen v položce VirtKey struktury události.

Identifikátor	Kód	Popis
vkEnter	$200	Potvrzení výběru.
vkEsc	$201	Opuštění komponenty, ukončení modálního stavu.
vkNext	$202	Přesun kurzoru na následující položkua.
vkPrev	$203	Přesun kurzoru na předchozí položku.
vkLeft	$204	Přesun kurzoru na znak nebo položku vlevo.
vkRight	$205	Přesun kurzoru na znak nebo položku vpravo.
vkUp	$206	Přesun kurzoru na řádek nebo položku nahoru.
vkDown	$207	Přesun kurzoru na řádek nebo položku dolu.
vkInsert	$208	Vložení znaku, položky, řádku.
vkDelete	$209	Vymazání znaku, položky, řádku na pozici kurzoru.
vkBackSpace	$20A	Vymazání znaku, polozky, řádku před kurzorem.
vkHome	$20B	Přesun kurzoru na první znak, řádek nebo položku.
vkEnd	$20C	Přesun kurzoru na poslední znak, řádek nebo položku.
vkPageUp	$20D	Přesun kurzoru o stránku výše.
vkPageDown	$20E	Přesun kurzoru o stránku níže.
vkHelp	$20F	Vyvolání nápovědy.
vkNextWindow	$210	Výběr následující okna nebo stránky.
vkPrevWindow	$211	Výběr předchozího okna nebo stránky.
vkAppExit	$212	Ukončení aplikace.
vkYes	$213	Kladné potvrzení.
vkNo	$214	Záporné potvrzení.
vkClear	$215	Vymazání položky, bez odstranění
vkMenu	$216	Vyvolání hlavního menu.

5.1.5. Konstanty cid_

Konstanty s prefixem cid_ slouží jako identifikátory instancí komponent. Názvy těchto identifikátorů si navrhuje programátor aplikace. Identifikátory komponent obvykle slouží k identifikaci komponent při zpracování oznámení, příp. jako identifikátory stránek komponenty **TPageControl** (viz. kapitola 0).

Knihovna Controls definuje pouze několik standardních identifikátorů konfiguračních stránek terminálů.

Identifikátor	Kód	Popis
cidTermDispSetupPage	$FF00	Stránka s konfigurací displeje terminálu
cidTermMouseSetupPage	$FF01	Stránka s konfigurací polohovacího zařízení terminálu (myš, dotykový panel, klávesnice).
cidTermCalibPage	$FF02	Kalibrační stránka pro dotykové panely.
cidTermSetupPagev	$FF00	Úvodní stránka konfigurace terminálu.
cidInlineEdit	$FF10	Identifikátor komponenty TEdit vytvořené funkcí InlineEdit (viz. dokumentace ke knihovně GrCtrls)

Programátor aplikace může vytvořit libovolný počet identifikátorů cid_ podle potřeb aplikace. Hodnoty identifikátorů však může volit pouze v rozsahu 1 až 61439 ($EFFF). Ostatní hodnoty jsou rezervovány pro vizualizační knihovny.

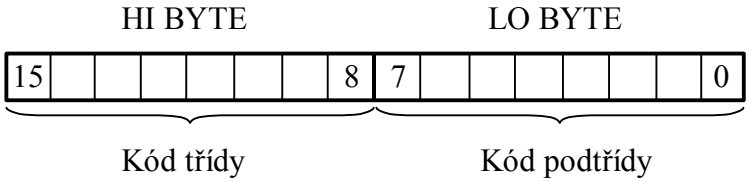
5.1.6. Konstanty cc_

Konstanty s prefixem cc_ definují tzv. třídy nastavení komponent. Identifikátory se tvoří podle následujícího pravidla cc + třída (název komponenty) + podtřída, např. ccStaticTextTitle (třída StaticText, podtřída Title – statický text používaný pro titulek stránky), ccEditBlue (třída Edit, podtřída Blue – komponenta edit s modrým pozadím). Identifikátor neobsahující název podtřídy, tj. např. ccStaticText, ccWindow reprezentuje implicitní nastavení komponenty, dané obvykle typem použitého terminálu.

Přímo v knihovně Controls jsou definovány pouze konstanty implicitních tříd nastavení komponent definovaných v této knihovně. Viz. následující tabulka:

Identifikátor	Kód	Komponta
ccApplication	$0100	TApplication
ccDesktop	$0200	TDesktop
ccPageControl	$0300	TPageControl
ccPage	$0400	TPage

Hodnota konstanty cc_ má následující strukturu:



Při vytvoření nové komponenty je nutné vytvořit novou třídu nastavení komponenty. Kódy 0 až 127 jsou rezervovány. Uživatel může použít kódy 128 až 255. Při vytváření nové podtřídy nastavení je možné použít kódy 16 až 255, kódy 0 až 15 jsou rezervovány.

Základní třída nastavení (tj. identifikátor třídy, který neobsahuje název podtřídy) má vždy v nižší slabice kódu hodnotu 0. Kódy podtříd je tedy možné odvozovat pomocí jednoduché operace např.

```
const
   ccEditBlue =  ccEdit + 16;
```

5.1.7. Konstanty mr_

Konstanty s prefixem mr_ určují důvod ukončení modálního stavu komponenty (většinou dialogového okna). V knihovně Controls je předdefinováno několik konstant mr_, které jsou uvedeny v následující tabuce.

Identifikátor	Kód	Popis
mrNone	0	Vyhrazená konstanta
mrOk	1	Modální stav ukončen tlačítkem ’OK ’
mrCancel	2	Modální stav ukončen tlačítkem ’Cancel ’
mrYes	3	Modální stav ukončen tlačítkem ’Yes’
mrNo	4	Modální stav ukončen tlačítkem ’No ’
mrYesToAll	5	Modální stav ukončen tlačítkem ’Yes to All’
mrNoToAll	6	Modální stav ukončen tlačítkem ‘No to All’

Tyto konstanty vrací metoda TControl.Execute (viz. kapitola 5.3.4.61), příp. TGroup.ExecControl (viz. kapitola 5.3.5.14).

5.1.8. Konstanty sf_

Konstanty s prefixem sf_ jsou příznaky stavu komponenty uložené v položce TControl.State (viz. kapitola 5.3.4.8).

Identifikátor	Kód	Popis
sfVisible	$0001	Pokud je nastaven příznak sfVisible, pak je komponenta viditelná, pokud není zcela překryta jinou komponentou. Modifikace tohoto příznaku se provádí pomocí metod TControl.Show a TControl.Hide (viz. kapitoly 5.3.4.71 a 5.3.4.72)

sfFocused	$0002	Pokud je nastaven příznak sfFocused, pak se komponenta nachází v ohnisku (má fokus) a přijímá události od klávesnice. Pokud je nastaven příznak sfFocused, pak je zároveň nastaven příznak sfSelected, opačně to však neplatí.
sfSelected	$0004	Příznak sfSelected má nastavena ta komponenta, která je v seznamu vlastíka označena jako vybraná (viz. položka TGroup.Current v kapitole 5.3.5.2). V seznamu vlastníka může být vybrána pouze jedna komponenta.
sfDisabled	$0008	Pokud je nastaven příznak sfDisabled, pak je komponenta zakázaná a nepřijímá žádné vstupní události. Modifikace tohoto příznaku se provádí pomocí metod TControl.Enable a TControl.Disable (viz. kapitoly 5.3.4.74 a 5.3.4.73).
sfModal	$0010	Příznak sfModal je nastaven u komponenty v modálním stavu, tj. u komponenty u níž byla zavolána metoda **Execute** (viz. kapitola 5.3.4.61). Po ukončení modálního stavu je tento příznak vynulován.
sfCaretVis	$0020	Příznak sfCaretVis je nastaven, pokud má komponenta zobrazovat textový kurzor v případě, že se nachází v ohnisku.
sfActive	$0100	Příznak je nastaven u komponent, které se nacházejí v vybraném okně.
sfExposed	$0400	Interní příznak označující komponenty, které jsou viditelné a zároveň jsou umístěny ve viditelných komponentách.
sfOverlapped	$0800	Interní příznak označující komponenty, které jsou v rámci seznamu vlastníka překryty jinými komponentami. Příznak slouží k urychlení vykreslování komponent. Příznak sfOverlapped je nastavován automaticky.

5.1.9. Konstanty of_

Konstanty s prefixem of_ definují základní chování (nastavení) komponenty. Jednotlivé konstanty jsou příznaky položky TControl.Options (viz. kapitola 5.3.4.9).

Identifikátor	Kód	Popis
ofSelectable	$0001	Komponentu lze vybrat pomocí metod Select příp. Focus.
ofTopSelect	$0002	Při výběru komponenty dojde k jejímu přesunutí do popředí. Příznak ofTopSelect se uplatní pouze tehdy, jestliže je zároveň nastaven příznak ofSelectable.
ofPreProcess	$0004	Komponenta zpracovává ohniskové události před komponentou v ohnisku (viz. kapitola 4.6.3.1).
ofPostProcess	$0008	Komponenta zpracovává ohniskové události po komponentě v ohnisku (viz. kapitola 4.6.3.1).
ofFirstClick	$0010	Pokud komponenta není vybraná, první kliknutí do oblasti komponenty způsobí její výběr. Pokud je nastaven tento příznak událost o stisku tlačítka myší je zároveň předána metodě **HandleEvent** komponenty. V opačném případě je tato první událost zahozena.
ofValidate	$0040	

ofSharedPalette	$0100	Komponenta sdílí paletu. Pokud tento příznak není nastaven, pak komponenta má alokovanou vlastní paletu a je zodpovědná za její uvolnění. Viz. metoda **SetPalette** v kapitole 5.3.4.55.
ofBackground	$0200	Pokud je nastaven tento příznak, dojde před voláním metody **Paint** k vyplnění oblasti komponenty první barvou z palety (tj. barvou s indexem 1).
ofPaintControl	$0400	U komponent, které nemají nastaven tento příznak, se nevolá metoda **Paint**. Příznak má význam pouze u komponent odvozených od třídy **TGroup**.
ofFlickSafe	$0800	U displejů, jejichž videopaměť je mapovaná přímo do adresového prostoru procesoru, může docházet při překreslování komponenty k "probliknutí", tj.efektu způsobeného tím, že se nejdříve vyplní oblast komponentou barvou pozadí a až poté se vykresluje popředí. Tento příznak informuje metodu **Paint** komponenty, aby překreslení provedla, tak aby k tomuto efektu nedocházelo, příp. aby byl minimální. Takové překreslení však může být komplikovanější a tedy i pomalejší.
ofShowFocus	$1000	Příznak, který informuje metodu **Paint** komponenty o tom, že by měla viditelně označit komponentu, jestliže je v ohnisku. Tento příznak má význam především u terminálu, které nemají k dispozici polohovací zařízení (myš nebo dotykový panel). Pokud je tento příznak nastaven, pak je při každé změně příznaku sfFocused v položce State automaticky volána metoda **Repaint**.

5.1.10. Konstanty gm_

Konstanty s prefixem gm_ určují chování komponenty v případě změny rozměrů vlastníka komponenty. Určují způsob, jakým se bude měnit pozice a velikost komponenty při změně rozměrů vlastníka komponenty Kombinace těchto konstant je uložena v položce GrowMode třídy **TControl** (viz. kapitola 5.3.4.13).

Identifikátor	Kód	Popis
gmGrowLoX	$01	Souřadnice X levé strany komponenty udržuje vzdálenost od pravé strany vlastníka.
gmGrowLoY	$02	Souřadnice Y horní strany komponenty udržuje vzdálenost od spodní strany vlastníka.
gmGrowHiX	$04	Souřadnice X pravé strany komponenty udržuje vzdálenost od pravé strany vlastníka
gmGrowHiY	$08	Souřadnice Y spodní strany komponenty udržuje vzdálenost od pravé strany vlastníka.
gmGrowAll	$0F	Všechny souřadnice komponenty udržují stejnou vzdálenost od pravého dolního rohu vlastníla.
gmGrowRel	$10	Souřadnice komponenty se mění relativně s velikostí komponenty vlastníka.

Pokud není nastaven ani jeden z příznaků gmGrowLoXX nebo gmGrowHiXX, pak souřadnice a rozměry komponenty se při změně rozměrů vlastníka nemění.

5.1.11. Konstanty btm_

Konstanty s prefixem btm_ upřesňují chování komponenty **TCustomButton** a jejich potomků. Kombinace těchto konstant je uložena v položce Mode komponenty **TCustomButton**.

Identifikátor	Kód	Popis
btmActionMask	$0007	Maska pro příznaky výběru akce
btmNotify	$0000	Při stisku se generuje oznámení nmClick (bez parametrů)
btmKeyDown	$0001	Při stisku se emuluje stisk klávesy. Kód klávesy kb_ je uložen v položce Param komponenty **TCustomButton**.
btmEndModal	$0002	Při stisku se ukončí modální stav. Důvod ukončení modálního stavu je uložen v položce Param komponenty **TCustomButton**.
btmGoto	$0003	Při stisku se přepne stránka komponenty **TPageControl**, metodou **GotoPage**. Odkaz na komponentu **TPageControl** musí být uložen v položce Pages a identifikátor stránky v položce Param komponenty **TCustomButton**.
btmCall	$0004	Při stisku se přepne stránka komponenty **TPageControl**, metodou **CallPage**. Odkaz na komponentu **TPageControl** musí být uložen v položce Pages a identifikátor stránky v položce Param komponenty **TCustomButton**.
btmReturn	$0005	Při stisku se přepne stránka komponenty **TPageControl**, metodou **ReturnPage**. Odkaz na komponentu **TPageControl** musí být uložen v položce Pages.
btmNext	$0006	Při stisku se přepne stránka komponenty **TPageControl**, metodou **GotoNextPage**. Odkaz na komponentu **TPageControl** musí být uložen v položce Pages.
btmPrev	$0007	Při stisku se přepne stránka komponenty **TPageControl**, metodou **GotoPrevPage**. Odkaz na komponentu **TPageControl** musí být uložen v položce Pages.
btmPush	$0008	Pokud je příznak nastaven, pak tlačítko reaguje už při stisku. V opačném případě reaguje až při uvolnění.
btmSwitch	$0010	Tlačítko pracuje jako dvoustavový přepínač. Stav tlačítka je uložen v položce Pressed.

Konstanty btmNotify, btmKeyDown, btmEndModal, btmGoto, btmCall, btmReturn, btmPrev a btmNext nelze mezi sebou kombinovat. Vždy může být uvedena pouze jedna varianta.

5.1.12. Konstanty mf_

Konstanty mf_ upřesňují chování komponenty **TCustomMenu** a jejich potomků. Kombinace těchto konstant je uložena v položce Flags komponenty **TCustomMenu**.

Identifikátor	Kód	Popis
mfMenuBar	$0001	Komponenta menu je lista (TMenuBar).
mfBorder	$0002	Kolem komponenty menu je vykreslen okraj.

5.1.13. Konstanty mif_

Konstanty s prefixem _mif upřesňují chování položky menu v komponentách TCustomMenu a TCustomMenuOverlay a jejich potomcích. Kombinace konstant je uložena v položce Flags struktury **TMenuItem** (viz. kapitola 5.2.11).

Identifikátor	Kód	Popis
mifActionMask	$0007	Maska pro bity specifikující akci prováděnou při výběru položky.
mifNotify	$0000	Generování oznánemí **nmClick**. V položce oznámení **ItemId** bude uložnen identifikátor položky Id ze struktury **TMenuItem**.
mifGotoPage	$0001	Skok na stránku specifikovanou v položce **WParam** struktury **TMenuItem**. Tato stránka se musí nacházet v komponentě **TPageControl** specifikovanou položkou Pages komponenty menu.
mifCallPage	$0002	Volání stránku specifikované položkou **WParam** struktury **TMenuItem**. Tato stránka se musí nacházet v komponentě **TPageControl** specifikovanou položkou Pages komponenty menu.
mifReturn	$0003	Provedení návratu do výšší urovně menu, nebo návrat do stránky, ze které byl provedeno volání aktuální stránky.
mifEndModal	$0004	Ukončení modálního stavu. Výsledek modálního stavu musí být uveden v položce **WParam** struktury **TMenuItem**.
mifSubMenu	$0005	Vyvolání podmenu určeného položkou **SubMenu** struktury **TMenuItem**.
mifDisabled	$0008	Položka menu je zakázána a nelze ji vybrat. Přesné chování je závislé na typu komponenty menu.

5.1.14. Konstanty sbf_

Konstanty s prefixem sbf_ upřesňují vlastnosti komponenty posuvníku - TCustomScrollBar (viz. kapitola 5.3.17). Jejich kombinace je uložena v položce Flags této komponenty.

Identifikátor	Kód	Popis
sbfDragging	$0001	Zatím neimplementováno.
sbfFixedPage	$0002	Fixní velikost indikátoru posuvníku. Pokud tento příznak není nastaven, pak je velikost indikátor posuvníku nastaven proporcinálně podle velikosti položky PageStep.
sbfBroadcast	$0004	Pokud je nastaven tento příznak, generuje pusuvník při každé změně aktuální hodnoty oběžník

		cmScrollbarChanged (viz. kapitola 5.3.17.2)
sbfNotify	$0008	Pokud je nastaven tento příznak, generuje posuvník při každé změně aktuální hodnoty oznámení nmChange (viz. kapitola 5.3.17.2)
sbfButtons	$0010	Pokud je nastaven tento příznak, pak je posuvník vykreslen se dvěma tlačítky na levé a pravé (resp. horní a dolní) straně. Tyto tlačítka lze použít ke změně aktuální hodnotu posuvníku o hodnotu danou položkou Step.

5.2. Typy

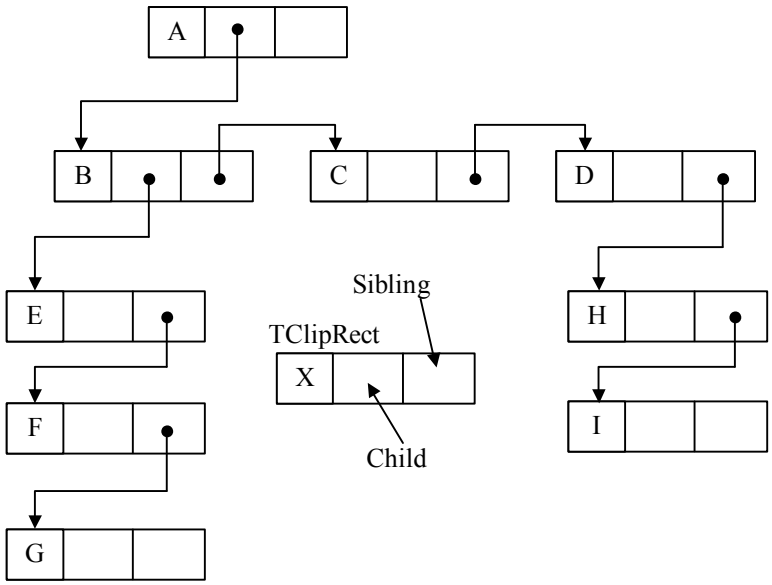
5.2.1. Typ TClipRect

Struktura **TClipRect** popisuje jeden obdélník ořezávací oblasti (viz. třída **TClipRegion** v kapitole 5.3.1).

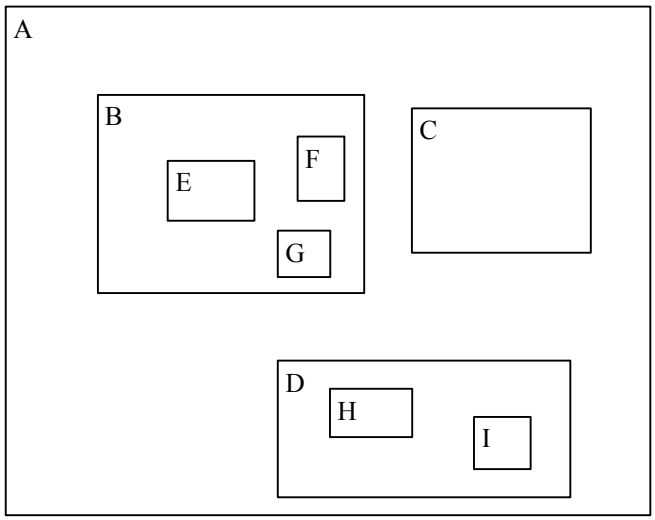
```
PClipRect = ^TClipRect;
TClipRect = record
  Bounds  : TRect;
  Sibling : PClipRect;
  Child   : PClipRect;
end;
```

Položka	Popis
Bounds	Hranice obdélníku
Sibling	Ukazatel na další obdélník v seznamu.
Child	Ukazatel na první obdélník v seznamu.

Struktury TClipRect jsou uloženy ve stromové struktuře, viz. následující obrázek.



Tato stromová struktura odpovídá např. našledujícímu rozvržení obdelníků:



V oblasti jsou obsaženy pouze obdélníky E,F,G a H, I. Ostatní obdélníky jsou pomocné a slouží k rychlejšímu zpracování struktury.

5.2.2. Typ TAClipRect

Typ **TAClipRect** je pole struktur **TClipRect**, jedná se o interní strukturu pro uložení seznamu ořezávacích obdélníků.

```
PAClipRect = ^TAClipRect;
TAClipRect = array[0..$FFF8 div SizeOf( TClipRect ) - 1] of
               TClipRect;
```

5.2.3. Typ TNotification

Typ **TNotification** definuje strukturu tzv. oznámení. Oznámení jsou krátké zprávy komponent určené uživatelské aplikaci. Oznámení slouží k informování aplikace obvykle např. o změně stavu komponenty. Oznámení lze vyvolat metodami třídy **TControl.Notify** a **TControl.NotifyEx** (viz. kapitoly 5.3.4.36 a 5.3.4.35).

```
  PNotification = ^TNotification;
  TNotification = record
    Control : PControl;
    Code    : Word;
    case Integer of
      0: ( WParam  : Word;
           LParam  : Longint;
           LParam2 : Longint );

      1: ( __Res1  : Word;
           WParam2 : Word;
           WParam3 : Word;
           WParam4 : Word;
           WParam5 : Word;
         );

      2: ( Result  : Word );

      3: ( ItemId  : Word );

      4: ( Index   : Integer );
```

```
      5: ( Row     : Integer;
           Col     : Integer );

      6: ( __Res2  : Word;
           __Res3  : Word;
           Text    : PString );

      7: ( __Res4  : array[0..7] of Byte;
           Accept  : Boolean;
         );
  end;
```

Popis jednotlivých položek struktury je uveden v následující tabulce:

Položka	Popis
Control	Odkaz na komponentu, která ozámení vyvolala.
Code	Indefikátor oznámení, tj. jedna z konstant s prefixem nm_ (viz. kapitola 5.1.3).

Ostatní položky struktury tvoří parametry oznámení a jejich význam je uveden vždy v popisu k danému oznámení.

5.2.4. Typ TEventHandler

Procedurální typ **TEventHandler** definuje prototyp procedury určené k obsluze vstupních událostí aplikačním programem (viz. položky BeforeHandle a AfterHandle třídy **TControl** v kapitolách 5.3.4.18 a 5.3.4.17).

```
TEventHandler = procedure( AControl: PControl; var AEvent: TEvent );
```

Parametr AControl obsahuje vždy ukazatel na komponentu, která tuto proceduru vyvolala. Parametr AEvent obsahuje strukturu vstupní události.

5.2.5. Typ TNotificationHandler

Procedurální typ **TNotificationHandler** definuje prototyp procedury určené k obsluze oznámení komponent aplikačním programem (viz. položky BeforeNotify a AfterNotify třídy **TControl** v kapitolách 5.3.4.20 a 5.3.4.19).

```
TNotificationHandler = procedure( var ANotification: TNotification );
```

Parametr ANotification obsahuje strukturu oznámení.

5.2.6. Typ TPaintHandler

Procedurální typ **TPaintHandler** definuje prototyp funkce určené pro dodatečné vykreslování do oblasti komponenty aplikačním programem (viz. položky BeforePaint a AfterPaint třídy **TControl** v kapitolách 5.3.4.21 a 5.3.4.22).

```
TPaintHandler = procedure( AControl: PControl; ACanvas: PCanvas );
```

Parametr AControl obsahuje ukazatel na komponentu, která tuto proceduru zavolala.

Parametr ACanvas obsahuje ukazatel na inicializovanou instanci třídy ACanvas určenou pro kreslení do komponenty odkazované parametrem AControl.

5.2.7. Typ TKeyMapper

Procedurální typ **TKeyMapper** definuje prototyp funkce určené pro mapování kódů kláves kb_ na kódy virtuálních kláves vk_ (viz. položka KeyMapper třídy **TControl** v kapitole 5.3.4.15)

```
TKeyMapper = function( AKeyCode: Word ): Word;
```

Parametr funkce AKeyCode obsahuje kód klávesy s prefixed kb_ (viz. dokumentace ke knihovně IoDrv) nebo případně ASCII hodnotu klávesy. Funkce vrací jeden z kódu virtuálních kláves vk_ (viz. kapitola 5.1.4) nebo hodnotu nula v případě, že funkce mapování kódu klávesy neprovedla.

5.2.8. Type TCustomizeProc

Procedurální typ **TCustomizeProc** definuje prototyp funkce určené pro konfiguraci vlastností komponenty podle zadaná třídy nastavení (viz. kapitola 4.2.1)

```
TCustomizeProc = procedure( AControl: PControl; AClass: Word );
```

Parametr procedury AControl obsahuje odkaz na komponentu, jejíž vlastnosti se mají modifikovat. Parametr AClass obsahuje identifikátor třídy nastavení komponenty, tj. konstantu s prefixem cc_ (viz. kapitola 5.1.6).

Tyto funkce je potřeba zaregistrova v globální tabulce pomocí procedury **RegisterCustomizeProc** (viz. kapitola 5.4.1).

5.2.9. Typ TBitmapGetter

Procedurální typ TBitmapGetter definuje prototyp funkce určené pro získání ukazatele na bitmapu ze zadaného identifikátoru bitmapy.

```
TBitmapGetter = function( AId : Integer ): PBitmap;
```

Parametr AId obsahuje identifikátor bitmapy. Funkce vrací ukazatel na strukturu bitmapy.

Funkce tohoto typu lze vytvořit pomocí speciální aplikace pro zpracování bitmap.

5.2.10. Typ THandlePhase

Jednotlivé varianty výčtového typ **THandlePhase** slouží jako identifikátory fází zpracování vstupních událostí metodou **HandleEvent** (viz. kapitola 5.3.4.26).

```
THandlePhase = (
  phPreProcess,    { Pred komponentou v ohnisku   }
  phFocused,       { Komponenta v ohnisku         }
```

```
  phPostProcess   { Po komponente v ohnisku      }
);
```

Komponenta **TGroup** a její potomci obsahují položku Phase definovanou následovně:

Phase : THandlePhase;

Tato položka je automaticky nastavována při obsluze vstupních událostí a lze její obsah je platný pouze v rámci metody **HandleEvent** nebo procedur **AfterHandle** a **BeforeHandle**. Více v kapitole 5.3.5.15.

5.2.10.1. Typ TMessageBoxFunc

Procedurální typ **TMessageBoxFunc** definuje prototyp funkce pro vytvářející speciální dialogové okno pro zobrazení jednoduchého hlášení uživateli. Více v kapitolách 5.3.11.6 a 5.3.11.12.

```
TMessageBoxFunc = function( const ABounds: TRect;
  const ATitle, AText: string; AFlags: Word ): PCustomDialog;
```

Funkce vytvoří a inicializuje dialogové okno se zadanými parametry. Tj. umístění a rozměry, titulkem, textem a případnými příznaky. Funkce vrátí ukazatel na instanci dialogového okna, tj. potomka třídy **TCustomDialog**.

5.2.11. Typ TMenuItem

Struktura **TMenuItem** definuje jednu položku menu (viz. kapitola 5.4.10).

```
PMenuItem = ^TMenuItem;
TMenuItem = record
  Next     : PMenuItem;   { Dalsi polozka v seznamu    }
  Id       : Word;        { Identifikator polozky      }
  Flags    : Word;        { Priznaky mifXXX            }
  Name     : PString;     { Nazev polozky              }
  Text     : PString;     { Text zarovnany doprava     }
  Width    : Integer;     { Sirka polozky              }

  case Integer of
    0: ( WParam  : Word );
    1: ( LParam  : Longint );
    1: ( SubMenu : PMenu );
end;
```

Položka	Popis
Next	Odkaz na následující položku menu. Položky menu tvoří spojový seznam. Poslední položka menu se odkazuje na hodnotu **nil**.
Id	Pomocný identifikátor položky. Zatím nevyužito.
Flags	Příznaky definující vlastnosti položky, tj. zda se jedná o podmenu, zda je položka povolená, příp. jaká akce se má provést při výběru položky (viz. příznaky s prefixem mif_ v kapitole 5.1.13).
Name	Text položky zarovnaný k levému okraji.
Text	Text položky zarovnaný k pravému okraji.

Width	Šířka položky, využito pouze v případě TMenuBox a TMenuPopup (viz. dokumentace ke knihovně GrCtrls).
WParam	Pomocný parametr pro akci prováděnou při výběru položky. Např. číslo stránky, výsledek modálního stavu apod.
LParam	Pomocný parametr pro akci prováděnou při výběru položky.
SubMenu	Odkaz na další úroveň menu, tj. na strukturu TMenu. Tato položka je platná pouze tehdy, pokud položka Flags obsahuje příznak mifSubMenu.

5.2.12. Typ TMenu

Struktura **TMenu** definuje jednu úroveň menu (viz. kapitola 5.4.10).

```
PMenu = ^TMenu;
TMenu = record
  Owner   : PMenu;
  Items   : PMenuItem;
  TopItem : PMenuItem;
  Default : PMenuItem;
end;
```

Položka	Popis
Owner	Odkaz na vyšší úroveň menu. Využito pouze u komponenty TCustomMenuOverlay a jejich potomků.
Items	Odkaz na spojový seznam položek menu (viz. kapitola 5.2.11)
TopItem	Odkaz na první zobrazenou v případě rolování menu. Využíva pouze komponenta TCustomMenuOverlay a její potomci.
Default	Odkaz na vybranou položku v dané úrovni menu.

5.3. Třídy

5.3.1. Třída TClipRegion

Třída **TClipRegion** definuje oblast určenou k ořezávání vykreslovaných objektů v překrývajících se komponentách. Oblast je definována seznamem nepřekrývajících se obdélníku (viz. struktur **TClipRect** v kapitole 5.2.1).

```
PClipRegion = ^TClipRegion;
TClipRegion = object( TObject )
public
  constructor Init( AMaxCount: Integer );
  destructor  Done; virtual;

  function  Assign( const ABounds: TRect ): Boolean;
  function  Empty: Boolean;
  procedure Clear;
  function  Exclude( const ABounds: TRect ): Boolean;
  procedure ForEach( AEnumFunc: Pointer );
```

```
    procedure ForEachIntersect( AEnumFunc: Pointer;
                const ABounds: TRect );
    function  GetClipRectCount: Integer;
  end;
```

5.3.1.1. Konstruktor TClipRegion.Init

Konstruktor **Init** provádí inicializaci instance třídy.

```
constructor Init( AMaxCount: Integer );
```

Parametry:

AMaxCount	Parametr určuje maximální počet obdélníků, ze kterých se oblast může skládat.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor alokuje na hromadě pole struktur TClipRect (viz. kapitola 5.2.1) o délce AMaxCount, tj. prostor o velikosti AMaxCount * 16B.

5.3.1.2. Destruktor TClipRegion.Done

Destruktor **Done** provádí uvolnění prostředků alokovaných konstruktorem třídy.

```
destructor Done; virtual;
```

Parametry:

Destruktor nemá žádné parametry.

Návratové hodnoty:

Destruktor nevrací žádnou hodnotu.

5.3.1.3. Metoda TClipRegion.Assign

Metoda **Assign** provádí inicializaci seznamu obdélníku oblasti jednoduchou obdélníkovou oblastí.

```
function  Assign( const ABounds: TRect ): Boolean;
```

Parametry:

ABounds	Rozměry a umístění obdélníku.

Návratové hodnoty:

V případě úspěchu metoda vrací hodnotu True. V opačném případě hodnotu False.

Poznámky:

Metoda odstraní z oblasti všechny obdélníky a vloží do ní obdélník daný parametrem ABounds.

5.3.1.4. Metoda TClipRegion.Empty

Metoda **Empty** zjišťuje zda je oblast prázdná, tj. neobsahuje žádný neprázdný obdélník.

```
function  Empty: Boolean;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

V případě, že je oblast prázdná, tj. počet obdélníku v oblasti je roven 0, pak metoda vrací hodnotu True.

5.3.1.5. Metoda TClipRegion.Clear

Metoda **Clear** odstraní všechny obdélníky z oblasti.

```
procedure  Clear;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

5.3.1.6. Metoda TClipRegion.Exclude

Metoda **Exclude** provádí vyjmutí zadaného obdélníku z oblasti.

```
function  Exclude( const ABounds: TRect ): Boolean;
```

Parametry:

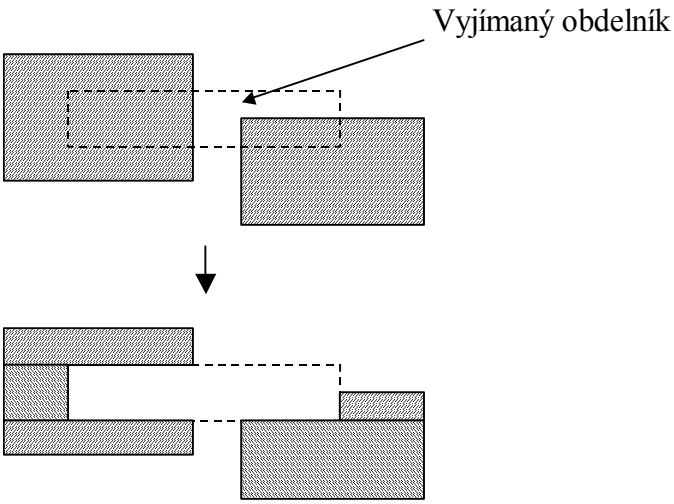
ABounds	Rozměry a umístění vyjímaného obdélníku.

Návratové hodnoty:

V případě úspěchu metoda vrací hodnotu True. V opačném případě hodnotu False,

Poznámky:

Při vyjímání obdélníku z oblasti může dojít ke zvětšení celkového počtu obdélníků, ze kterých je oblast složena. Viz. následující obrázek:



Původní oblast byla složena ze dvou obdélníků. Po vyjmutí označeného obdélníku, byly původní obdélníky zrušeny a bylo vytvořeno další pět menších. V případě, že při této operaci je překročen maximální počet obdélníků v oblasti (definovaný parametrem konstruktoru) vrací metoda hodnotu False.

5.3.1.7. Metoda TClipRegion.ForEach

Metoda **ForEach** prochází všechny obdélníky v oblasti a pro každý volá callback funkci.

```
procedure ForEach( AEnumFunc: Pointer );
```

Parametry:

AEnumFunc	Callback funkce volaná pro každý obdélník v oblasti. Viz. poznámky níže.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **ForEach** pro každý obdelník v oblasti zavolá funkci danou parametrem AEnumFunc. Tato funkce musí být **lokální** a definovaná podle následujícího prototypu:

```
function EnumFunc( R : TRect ): Boolean; far;
```

V případě, že tato funkce vrátí hodnotu True, je procházení obdélníků oblasti ukončeno.

Následující příklad vypíše všechny obdélníky, z nichž se oblast skládá:

```
procedure PrintRegionRects( AClipRegion: PClipRegion );

  function EnumFunc( R: TRect ): Boolean; far;
  begin
    WriteLn( R.A.X, 'x', R.A.Y, ' - ', R.B.X, 'x', R.A.X );
    EnumFunc := False;
  end;

begin
  AClipRegion^.ForEach( @EnumFunc );
end;
```

5.3.1.8. Metoda TClipRegion.ForEachIntersect

Metoda **ForEachIntersect** prochází všechny obdélníky v oblasti, které mají neprázdný průnik se zadaným obdélníkem a pro každý volá callback funkci.

```
procedure ForEachIntersect( AEnumFunc: Pointer;
            const ABounds: TRect );
```

Parametry:

AEnumFunc	Callback funkce volaná pro každý obdélník v oblasti. Viz. poznámky níže.
ABounds	Obdélník určující prohledávanou část oblasti.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **ForEachIntersect** pro každý obdélník v oblasti, který má neprázdný průnik se zadaným obdélníkem ABounds, zavolá funkci danou parametrem AEnumFunc. Tato funkce musí být **lokální** a definovaná podle následujícího prototypu:

```
function EnumFunc( R : TRect ): Boolean; far;
```

V případě, že tato funkce vrátí hodnotu True, je procházení obdelníků oblasti ukončeno.

5.3.1.9. Metoda TClipRegion.GetClipRectCount

Metoda **GetClipRectCount** zjišťuje počet obdélníků, z nichž se oblast skládá.

```
function  GetClipRectCount: Integer;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací počet obdélníků.

Poznámky:

Metoda **GetClipRectCount** interně pro výpočet počtu obdélníků využívá metodu **ForEach**.

5.3.2. Třída TCanvas

Třída **TCanvas** je definuje rozhraní pro kreslení do obdélníkové oblasti komponenty. Nabízí řadu funkcí pro vykreslování různých grafických objektů jako je úsečka, kružnice, obdélník, text apod.

```
PCanvas = ^TCanvas;
TCanvas = object( TObject )
public
{--- Interni polozky --- }
  Surface   : PDrawSurface;
  LastError : TDrawStatus;
  Origin    : TPoint;
{--- Polozky pristupne uzivateli ---}
  Clip      : PClipRegion;
  Pen       : TPenRef;
  Brush     : TBrushRef;
  Font      : TFontRef;
  ROP       : Byte;

  constructor Init( ASurface: PDrawSurface; AMaxClips: Integer );
  destructor Done; virtual;
  procedure  Reuse;

  procedure DrawPoint( X, Y: Integer );
  procedure DrawLine( X1, Y1, X2, Y2: Integer );
  procedure DrawCircle( X, Y, Radius: Integer );
  procedure DrawEllipse( X, Y, A, B: Integer );
  procedure DrawRect( X1, Y1, X2, Y2: Integer );
  procedure DrawArc( X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer );
  procedure FillRect( X1, Y1, X2, Y2: Integer );
  procedure FillCircle( X, Y, Radius: Integer );
  procedure FillEllipse( X, Y, A, B: Integer );
  procedure FillArc( X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer );
  procedure DrawText( X, Y: Integer; const AText: string );
  procedure DrawTextRect( const Rect: TRect; const AText: string;
              Flags: Byte );
  procedure DrawBitmap( X, Y: Integer; Bitmap: PBitmap );
  procedure DrawBitmapRect( const R: TRect;
              OffsetX, OffsetY: Integer; Bitmap: PBitmap );
  procedure DrawFrame( X1, Y1, X2, Y2: Integer;
              LTColor, BRColor: TColorRef );
  procedure Scroll( R: TRect; DeltaX, DeltaY: Integer );
  procedure Copy( R: TRect; X, Y: Integer );

  function  GetTextOffset( const Text: string;
              RefPos, Pos: Byte ): Integer;
  function  GetTextWidth( const Text: string ): Integer;
  function  GetTextHeight: Integer;
  function  GetTextLength( const Text: string; RefPos: Byte;
              Width: Integer ): Byte;
end;
```

5.3.2.1. Položka TCanvas.Surface

Položka **Surface** obsahuje odkaz na kreslící povrch svázaný s komponentou **TCanvas**. Položka je určena pouze pro čtení.

```
Surface   : PDrawSurface;
```

5.3.2.2. Položka TCanvas.LastError

Položka **LastError** obsahuje chybu naposledy volané vykreslovací funkce. Viz. konstanty s prefixem ds_ definované v knihovně IoDrv. Položka je určena pro ladící účely.

```
LastError : TDrawStatus;
```

5.3.2.3. Položka TCanvas.Origin

Položka **Origin** obsahuje souřadnice, které jsou přičteny při každém volání některé z vykreslovacích funkcí k zadaným souřadnicím. Ve většině případů jsou zde uloženy globální souřadnice levého horního rohu komponenty. Položka je určena pouze pro čtení.

```
Origin    : TPoint;
```

5.3.2.4. Položka TCanvas.Clip

Položka **Clip** obsahuje odkaz na instanci třídy **TClipRegion** a definuje ořezávací oblast. Položka je určena pouze pro čtení.

```
Clip      : PClipRegion;
```

5.3.2.5. Položka TCanvas.Pen

Položka **Pen** definuje tzv. pero, tj. jeho styl, barvu a šířku. Pero se používá při vykreslování čárových objektů, jako je bod, úsečka, kružnice, elipsa apod. Viz. struktura **TPenRef** v knihovně IoDrv.

```
Pen       : TPenRef;
```

5.3.2.6. Položka TCanvas.Brush

Položka **Brush** definuje tzv. štětec, tj. jeho styl, barvu, vzor a případně jeho počátek. Štětec se používá při vykreslování vyplněných objektů, jako je vyplněný obdélník, kruh apod. Viz. struktura **TBrushRef** v knihovně IoDrv.

```
Brush     : TBrushRef;
```

5.3.2.7. Položka TCanvas.Font

Položka **Font** definuje písmo. Viz. struktura **TFontRef** v knihovně IoDrv.

```
Font      : TFontRef;
```

5.3.2.8. Položka TCanvas.ROP

Položka ROP upřesňuje chování vykreslovacích funkcí. Definuje logickou operaci barvy zapisovaných bodů s jejich původní barvou (viz. Konstanty rop_ v knihovně IoDrv).

```
ROP       : Byte;
```

5.3.2.9. Konstruktor TCanvas.Init

Konstruktor **Init** inicializuje strukturu instance třídy.

```
constructor Init( ASurface: PDrawSurface; AMaxClips: Integer );
```

Parametry:

ASurface	Odkaz na kreslící povrch, tedy instanci TDrawSurface.
AMaxClips	Maximální počet obdelníku ořezávací oblasti TClipRegion.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor alokuje na hromadě instanci třídy **TClipRegion** a volá metodu **Reuse** (viz. kapitola 5.3.2.11).

5.3.2.10. Destruktor TCanvas.Done

Destruktor **Done** uvolní prostředky alokované konstruktorem třídy.

```
destructor Done; virtual;
```

Parametry:

Destruktor nemá žádné parametry.

Návratové hodnoty:

Destruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor uvolní ořezávací oblasti TClipRegion alokovanou při volání konstruktoru.

5.3.2.11. Metoda TCanvas.Reuse

Metoda Reuse nastaví parametry vykreslovacích funkcí tj. pero, štetec, font apod. na implicitní hodnoty.

```
procedure  Reuse;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda nastaví struktury Pen, Brush, Font a ROP na implicitní hodnoty uvedené v následující tabulce:

Struktura	Položka		Hodnota
Pen.	Style	=	psSolid
	Color	=	clWhite
	Width	=	1
Brush.	Style	=	bsSolid
	Color	=	clWhite
Font.	Style	=	fsTransparent
	Color	=	clWhite
	BkColor	=	clBlack
	Leading	=	0
	Id	=	fidDefault
ROP		=	ropCopy

5.3.2.12. Metoda TCanvas.DrawPoint

Metoda **DrawPoint** vykreslí bod na zadaných souřadnicích aktuálně nastaveným perem.

```
procedure DrawPoint( X, Y: Integer );
```

Parametry:

X	X-ová souřadnice vykreslovaného bodu.
Y	Y-ová souřadnice vykreslovaného bodu.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Chování vykreslovací rutiny ovlivňují položky ROP a Pen.

5.3.2.13. Metoda TCanvas.DrawLine

Metoda **DrawLine** vykreslí úsečku aktuálně nastaveným perem.

```
procedure DrawLine( X1, Y1, X2, Y2: Integer );
```

Parametry:

X1	X-ová souřadnice počátečního bodu úsečky.
Y1	Y-ová souřadnice počátečního bodu úsečky.
X2	X-ová souřadnice koncového bodu úsečky.
Y2	Y-ová souřadnice koncového bodu úsečky.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Úsečka je vykreslena z bodu [X1, Y1] do bodu [X2, Y2] včetně těchto bodů. Chování vykreslovací rutiny ovlivňují položky ROP a Pen.

5.3.2.14. Metoda TCanvas.DrawCircle

Metoda **DrawCircle** vykreslí kružnici aktuálně nastaveným perem.

```
procedure DrawCircle( X, Y, Radius: Integer );
```

Parametry:

X	X-ová souřadnice středu kružnice.
Y	Y-ová souřadnice středu kružnice.
Radius	Poloměr kružnice

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Chování vykreslovací rutiny ovlivňují položky ROP a Pen.

5.3.2.15. Metoda TCanvas.DrawEllipse

Metoda **DrawEllipse** vykreslí elipsu aktuálně nastaveným perem.

```
procedure DrawEllipse( X, Y, A, B: Integer );
```

Parametry:

X	X-ová souřadnice středu elipsy.
Y	Y-ová souřadnice středu elipsy.
A	Délka *vodorovné* osy elipsy.
B	Délka *svislé* osy elipsy.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Chování vykreslovací rutiny ovlivňují položky ROP a Pen.

5.3.2.16. Metoda TCanvas.DrawRect

Metoda **DrawRect** vykreslí obdélník aktuálně nastaveným perem.

```
procedure DrawRect( X1, Y1, X2, Y2: Integer );
```

Parametry:

X1	X-ová souřadnice levého horního rohu obdélníku.
Y1	Y-ová souřadnice levého horního rohu obdélníku.
X2	X-ová souřadnice pravého dolního rohu obdélníku.
Y2	Y-ová souřadnice pravého dolního rohu obdélníku.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Chování vykreslovací rutiny ovlivňují položky ROP a Pen.

5.3.2.17. Metoda TCanvas.DrawArc

Metoda **DrawArc** vykreslí eliptický oblouk.

```
procedure DrawArc( X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer );
```

Zatím neimplementováno.

5.3.2.18. Metoda TCanvas.FillRect

Metoda **Fillect** vyplní opdélník aktuálně nastaveným štětcem.

```
procedure FillRect( X1, Y1, X2, Y2: Integer );
```

Parametry:

X1	X-ová souřadnice levého horního rohu obdélníku.
Y1	Y-ová souřadnice levého horního rohu obdélníku.
X2	X-ová souřadnice pravého dolního rohu obdélníku.
Y2	Y-ová souřadnice pravého dolního rohu obdélníku.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Chování vykreslovací rutiny ovlivňují položky ROP a Brush.

5.3.2.19. Metoda TCanvas.FillCircle

Metoda **FillCircle** vyplní kruh aktuálně nastaveným štětcem.

```
procedure FillCircle( X, Y, Radius: Integer );
```

Parametry:

X	X-ová souřadnice středu kružnice.
Y	Y-ová souřadnice středu kružnice.
Radius	Poloměr kružnice

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Chování vykreslovací rutiny ovlivňují položky ROP a Brush.

5.3.2.20. Metoda TCanvas.FillEllipse

Metoda **FillEllipse** vyplní elipsu aktuálně nastaveným štětcem.

```
procedure FillEllipse( X, Y, A, B: Integer );
```

Parametry:

X	X-ová souřadnice středu elipsy.
Y	Y-ová souřadnice středu elipsy.
A	Délka *vodorovné* osy elipsy.
B	Délka *svislé* osy elipsy.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Chování vykreslovací rutiny ovlivňují položky ROP a Brush.

5.3.2.21. Metoda TCanvas.FillArc

Metoda **FillArc** vyplní výseč elipsy aktuálně nastaveným štětcem.

```
procedure FillArc( X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer );
```

Zatím neimplementováno.

5.3.2.22. Metoda TCanvas.DrawText

Metoda **DrawText** vykreslí text na zadanou pozici.

```
procedure DrawText( X, Y: Integer; const Text: string );
```

Parametry:

X	X-ová souřadnice levého horního rohu textu.
Y	Y-ová souřadnice levého horního rohu textu.
Text	Vykreslovaný text

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Vykreslovaný není ovlivněno řídícími znaky jako je CR, LF, Tab apod. Pokud text tyto znaky obsahuje, jsou vykresleny jako implicitní znak daného fontu.

5.3.2.23. Metoda TCanvas.DrawTextRect

Metoda **DrawTextRect** vykreslí text do zadaného obd0lníku.

```
procedure DrawTextRect( const Rect: TRect; const Text: string;
            Flags: Byte );
```

Parametry:

Rect	Obdélník, do něhož bude text vykreslen
Text	Vykreslovaný text
Flags	Příznaky určující, jak bude text vykreslen. Může se jednat o kombinaci následujících příznaků (viz. také kapitola 5.1.1):

tfLeft	Zarovnání textu doleva k levé straně obdélníku
tfRight	Zarovnání textu doprava k pravé straně obdélníku.
tfCenterX	Zarovnání textu na střed obdélníku v ose X
tfTop	Zarovnání textu nahoru k horní straně obdélníku
tfBottom	Zarovnání textu dolů k spodní straně obdélníku
tfCenterY	Zarovnání textu na střed obdélníku v ose Y
tfMultiline	Výpis textu na více řádků. Znaky CR (#13) jsou reprezentovány jako konec řádku.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je v parametru Flags nastaven příznak tfMultiline, pak je znak CR (#13) reprezentovány jako konec řádku. Pokud příznak nastaven není, pak je tento znak zobrazen jako každý jiný znak.

5.3.2.24. Metoda TCanvas.DrawBitmap

Metoda **DrawBitmap** vykreslí bitmapu na zadané souřadnice.

```
procedure DrawBitmap( X, Y: Integer; Bitmap: PBitmap );
```

Parametry:

X	X-ová souřadnice levého horního rohu bitmapy.
Y	Y-ová souřadnice levého horního rohu bitmapy.
Bitmap	Vykreslovaná bitmapa

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Vykreslovaná bitmapa musí být v dekomprimovaném tvaru. Musí mít menší nebo stejnou bitovou hloubku než je bitová hloubka kreslícího povrchu.

5.3.2.25. Metoda TCanvas.DrawBitmapRect

Metoda **DrawBitmapRect** vykreslí bitmapu nebo její část na zadané souřadnice.

```
procedure DrawBitmapRect( const R: TRect;
            OffsetX, OffsetY: Integer; Bitmap: PBitmap );
```

Parametry:

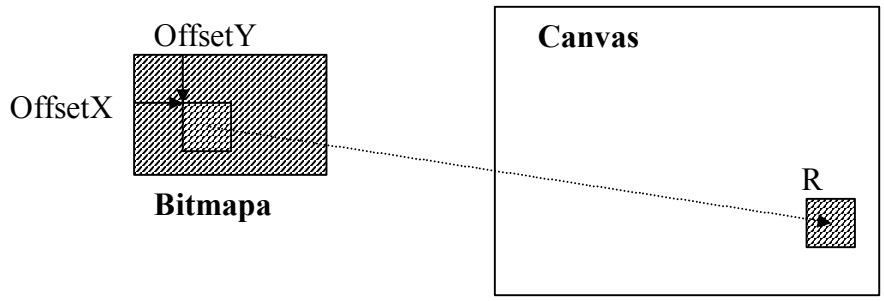
R	Obdelník, do něhož bude bitmapa vykreslena.
OffsetX	Posunutí bitmapy v horizontálním směru.
OffsetY	Posunutí bitmapy ve vertikálním směru.
Bitmap	Vykreslovaná bitmapa

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Vykreslovaná bitmapa musí být v dekomprimovaném tvaru. Musí mít menší nebo stejnou bitovou hloubku než je bitová hloubka kreslícího povrchu.



5.3.2.26. Metoda TCanvas.DrawFrame

Metoda **DrawFrame** vykreslí 3D stínovaný obdelník.

```
procedure DrawFrame( X1, Y1, X2, Y2: Integer;
            LTColor, BRColor: TColorRef );
```

Parametry:

X1	X-ová souřadnice levého horního rohu obdélníku.
Y1	Y-ová souřadnice levého horního rohu obdélníku.
X2	X-ová souřadnice pravého dolního rohu obdélníku.
Y2	Y-ová souřadnice pravého dolního rohu obdélníku.
LTColor	Barva levé a horní čáry rámečku.
BRColor	Barvá dolní a pravé čáry rámečku.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda vykreslí obdelník u něhož lze nezávisle určit barvu levé a horní čáry a dolní a pravé čáry.

5.3.2.27. Metoda TCanvas.Scroll

Tato metoda není v současné době implementována.

Metoda **Scroll** provádí rolování zadaného výřezu o zadaný počet pixelů.

```
procedure Scroll( R: TRect; DeltaX, DeltaY: Integer );
```

5.3.2.28. Metoda TCanvas.Copy

Tato metoda není v současné době implementována.

Metoda **Copy** provádí kopírování zadaného výřezu na zadané místo.

```
procedure Copy( R: TRect; X, Y: Integer );
```

5.3.2.29. Metoda TCanvas.GetTextOffset

Metoda **GetTextOffset** vrací šířku části zadaného textu.

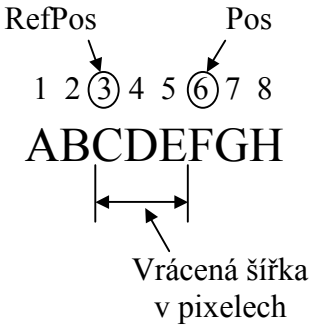
```
function GetTextOffset( const Text: string;
          RefPos, Pos: Byte ): Integer;
```

Parametry:

Text	Libovolný text bez řídících znaků (CR, LF apod.)
RefPos	Referenční pozice v textu
Pos	Pozice znaku, jehož relativní souřadnici funkce vrátí.

Návratové hodnoty:

Metoda vrací relativní souřadnici X prvního pixelu znaku na pozici Pos vůči začátku znaku na pozici RefPos. Jinými slovy metoda vrací šířku části textu vymezenou pozicemi RefPos a Pos. Viz. následující obrázek:



Šířky jednotlivých znaků jsou dány specifikovaným fontem.

Pokud je aktuální font neplatný (font s daným identifikátorem nebyl zaregistrován), pak metoda vrací hodnotu nula.

Poznámky:

5.3.2.30. Metoda TCanvas.GetTextWidth

Metoda **GetTextWidth** vrací šířku zadaného textu v pixelech.

```
function GetTextWidth(const Text: string ): Integer;
```

Parametry:

Text	Libovolný text bez řídících znaků (CR, LF apod.)

Návratové hodnoty:

Metoda vrací šířku zadaného textu v pixelech. Pokud je aktuální font neplatný (font

s daným identifikátorem nebyl zaregistrován), pak metoda vrací hodnotu nula.

Poznámky:

5.3.2.31. Metoda TCanvas.GetTextHeight

Metoda **GetTextHeight** vrací výšku znaků aktuálního fontu.

```
function GetTextHeight: Integer;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací výšku znaků aktuálního fontu v pixelech. Pokud je zadaný identifikátor fontu neplatný (font s daným identifikátorem nebyl zaregistrován), pak metoda vrací hodnotu nula.

Poznámky:

Výška znaků fontu nemusí vždy odpovídat výšce vykresleného znaku. Obecně se jedná o výšku mřížky znaku.

5.3.2.32. Metoda TCanvas.GetTextLength

Metoda **GetTextLength** vrací délku textu, jenž lze zobrazit v oblasti o zadané šířce.

```
function GetTextLength(const Text: string;
          RefPos: Byte; Width: Integer ): Byte;
```

Parametry:

AFontId	Identifikátor fontu (viz. kapitola **Chyba! Nenalezen zdroj odkazů.**).
Text	Libovolný text bez řídících znaků (CR, LF apod.)
RefPos	Referenční pozice v textu.
Width	Šířka oblasti v pixelech.

Návratové hodnoty:

Metoda vrací počet znaků textu začínajícího na referenční pozici (parametr RefPos), který lze zobrazit v oblasti o zadané šířce (Width). Pokud je aktuální fontu neplatný (font s daným identifikátorem nebyl zaregistrován), pak metoda vrací hodnotu nula.

Poznámky:

5.3.3. Třída TBitmapCanvas

Třída **TBitmapCanvas** je speciálním potomkem třídy TCanvas určeným pro přímé kreslení do bitmapy.

```
PBitmapCanvas = ^TBitmapCanvas;
TBitmapCanvas = object( TCanvas )
public
  constructor Init( ABitmap: PBitmap );
  destructor Done; virtual;
end;
```

5.3.3.1. Konstruktor TBitmapCanvas.Init

Konstruktor **Init** inicializuje strukturu instance třídy.

```
constructor Init( ABitmap: PBitmap );
```

Parametry:

ABitmap Bitmapa alokovaná v paměti RAM.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Nevolejte konstruktor **Init** přímo, ale použijte vždy globální funkci **GetBitmapCanvas** (viz. kapitola 5.4.8).

5.3.3.2. Destruktor TBitmapCanvas.Done

Destruktor **Done** uvolní prostředky alokované konstruktorem třídy.

```
destructor Done; virtual;
```

Parametry:

Destruktor nemá žádné parametry.

Návratové hodnoty:

Destruktor nevrací žádnou hodnotu.

Poznámky:

Nevolejte destruktor **Done** přímo, ale použijte vždy globální funkci **ReleaseBitmapCanvas** (viz. kapitola 5.4.9).

5.3.4. Třída TControl

Třída **TControl** je základní abstraktní třída, ze které jsou odvozeny veškeré komponenty vizualizačního systému. Třída TControl definuje standardní chování komponent a poskytuje rozhraní pro snadný návrh komponent a snadnou manipulaci s komponentami z uživatelské aplikace.

```
  PControl = ^TControl;
  TControl = object( TObject )
  public
    Id           : Word;
    Owner        : PGroup;
    Next         : PControl;
    Bounds       : TRect;
    EventMask    : Word;
    Caret        : TPoint;
    CaretSize    : TPoint;
    State        : Word;
    Options      : Word;
    Palette      : PString;
    Font         : Word;
    ModalResult  : Word;
    GrowMode     : Word;
    HelpCtx      : Word;
    Validator    : PValidator;
    KeyMapper    : TKeyMapper;
    AfterHandle  : TEventHandler;
    BeforeHandle : TEventHandler;
    AfterNotify  : TNotificationHandler;
    BeforeNotify : TNotificationHandler;
    BeforePaint  : TPaintHandler;
    AfterPaint   : TPaintHandler;

  public

    constructor Init( const ABounds: TRect );
    destructor Done; virtual;
    procedure Customize( AClass: Word );

{protected}

    procedure HandleEvent( var AEvent: TEvent ); virtual;
    procedure HandleNotification( var ANotification: TNotification );
                virtual;
    procedure Paint( ACanvas: PCanvas ); virtual;

    function  GetCanvas: PCanvas; virtual;
    function  GetCanvasRect( R: TRect ): PCanvas; virtual;
    procedure ReleaseCanvas( ACanvas: PCanvas );

    procedure SetState( AState: Word; AEnable: Boolean ); virtual;

    procedure ProcessEvent( var AEvent: TEvent );
    procedure NotifyEx( var ANotification: TNotification );
    procedure Notify( ACode: Word );

    procedure ClearEvent( var AEvent: TEvent );

    procedure GetEvent( var AEvent: TEvent; ANoTimer: Boolean );
                virtual;

    function  MouseEvent( var AEvent: TEvent; AMask: Word ): Boolean;
    procedure PutEvent( const AEvent: TEvent );
```

```
    procedure SetBounds( const ABounds: TRect );
    procedure CalcBounds( var ABounds: TRect; ADelta: TPoint );
                virtual;
    procedure ChangeBounds( const ABounds: TRect ); virtual;
    procedure SizeLimits( var AMin, AMax: TPoint ); virtual;

  {public}

  {-- Pomocne metody --}

    procedure Repaint; virtual;
    procedure RepaintRect( const R: TRect ); virtual;

    function  Prev: PControl;
    function  NextControl: PControl;
    function  PrevControl: PControl;
    function  TopControl: PControl;

    function  Contains( APos: TPoint ): Boolean; virtual;
    procedure MakeLocal( ASource: TPoint; var ADest: TPoint );
    procedure MakeGlobal( ASource: TPoint; var ADest: TPoint);
    procedure GetExtent( var AExtent: TRect );

    procedure SetPalette( const APalette: string; AShared: Boolean );
    procedure SetColor( AIndex: Byte; AColor: TColorRef );
    function  GetColor( AIndex: Byte ): TColorRef;

    procedure SetFont( AFont: Word );
    function  GetFont: Word;

    function  GetHelpCtx: Word; virtual;

    function  Execute: Word; virtual;
    procedure EndModal( AModalResult: Word );
    procedure SetValidator( AValidator: PValidator }; virtual;
    function  Transfer( AMode: Integer ): Integer; virtual;
    function  Validate: Boolean; virtual;
    function  Error( AErrCode: Integer ): Boolean; virtual;

  {-- Pomocne metody modifikujici stav komponenty --}

    function  Focus: Boolean;
    procedure Select;
    procedure PutInFrontOf( ATarget: PControl );
    procedure MakeFirst;

    procedure Show;
    procedure Hide;
    procedure Disable;
    procedure Enable;

    procedure Locate( var ABounds: TRect );
    procedure MoveTo( AX, AY: Integer);
    procedure GrowTo( AX, AY: Integer);

  {-- Metody pro praci s kurzorem }

    procedure ShowCaret;
    procedure HideCaret;
    procedure SetCaretPos( APos: TPoint );
    procedure SetCaretSize( ASize: TPoint );

  {-- Metody pro praci s casovacem }
```

```
    function  ScheduleTimer( AId: Integer; ATime: Longint;
                AOptions: Word ): Boolean;
    procedure CancelTimer( AId: Integer );

  end;
```

5.3.4.1. Položka TControl.Id

Položka **Id** obsahuje identifikátor komponenty přiřazený manuálně aplikačním programátorem. Položka je určena pro zápis i pro čtení. Položka může nabývat hodnoty v rozsahu 1 až 65535 ($FFFF). Hodnoty 65280 ($FF00) a vyšší jsou rezervovány.

```
Id            : Word;
```

5.3.4.2. Položka TControl.Owner

Položka **Owner** obsahuje odkaz na vlastníka komponenty. Obsah položky se nastavuje automaticky při vkládání příp. vyjímání komponenty do příp. z seznamu vlastníka. Položka je určena pouze pro čtení.

```
Owner         : PGroup;
```

5.3.4.3. Položka TControl.Next

Položka **Next** obsahuje odkaz na následující komponentu v seznamu vlastníka. Obsah položky se nastavuje automaticky. Položka je určena pouze pro čtení.

```
Next          : PControl;
```

5.3.4.4. Položka TControl.Bounds

Položka **Bounds** obsahuje rozměry a pozici komponenty v rámci svého vlastníka. Položka je určena pouze pro čtení. Změnu velikosti nebo pozice komponenty lze provést pouze pomocí metod **Locate**, **MoveTo** příp. **GrowTo**.

```
Bounds        : TRect;
```

5.3.4.5. Položka TControl.EventMask

Položka **EventMask** obsahuje bitovou masku událostí, které komponenta může obdržet. Položka EventMask obsahuje kombinaci příznaku typu událostí ev_. Položka je obvykle nastavována v rámci konstruktoru komponenty a je určena pouze pro čtení.

```
EventMask     : Word;
```

5.3.4.6. Položka TControl.Caret

Položka **Caret** obsahuje aktuální pozici textového kurzoru (tedy jeho levého horního rohu) relativně k levému hornímu rohu komponenty. Položka je určena pouze ke čtení. Pozici textového kurzoru lze nastavit pomocí metody **SetCaretPos** (viz. kapitola 5.3.4.80).

```
Caret         : TPoint;
```

5.3.4.7. Položka TControl.CaretSize

Položka **CaretSize** udává rozměry textového kurzoru v pixelech. Položka je určena pouze ke čtení. Velikost textového kurzoru lze nastavit pomocí metody **SetCaretSize** (viz. kapitola 5.3.4.81).

```
CaretSize    : TPoint;
```

5.3.4.8. Položka TControl.State

Položka **State** obsahuje množinu příznaků s prefixem sf_ (viz. kapitola 5.1.8), které určují aktuální stav komponenty. Položka je určena pouze ke čtení. Změnu stavu komponenty lze provést pomocí metody **SetState** (viz. kapitola 5.3.4.32), nebo příp. pomocí dalších speciálních metod.

```
State        : Word;
```

5.3.4.9. Položka TControl.Options

Položka **Options** obsahuje kombinaci příznaků nastavení komponenty of_ (viz. kapitola kapitola 5.1.9). Položka lze číst i zapisovat. Obvykle je hodnota položky nastavena konstruktorem komponenty a není ji potřeba měnit.

```
Options      : Word;
```

5.3.4.10. Položka TControl.Palette

Položka **Palette** obsahuje ukazatel na paletu komponenty. Jedná se o řetězec znaků, kde každý znak řetězce reprezentuje jednu barvu palety. Pokud komponenta nemá nastaven příznak ofSharedPalette v položce Options, pak je tento řětezec uvolněn při volání destruktoru komponenty. Položka je určena pouze pro čtení. Pro nastavení lze použít metodu **SetPalette** (viz. kapitola 5.3.4.55). Více o práci s paletou komponenty je uvedeno v kapitole 4.5.1.

```
Palette      : PString;
```

5.3.4.11. Položka TControl.Font

Položka **Font** obsahuje identifikátor implicitního fontu komponenty. Jedná se spíše o interní položku. Pro čtení implicitního fontu komponenty použijte metodu **GetFont** (viz kapitola 5.3.4.59) a pro jeho nastavení metodu **SetFont** (viz. kapitola 5.3.4.58).

```
Font         : Word;
```

5.3.4.12. Položka TControl.ModalResult

Položka **ModalResult** je spíše interní položkou a má význam pouze u komponent v modálním stavu. Může obsahovat jednu z konstant mr_ (viz. kapitola 5.1.7). Položka je nastavována pomocí metody **EndModal**, která ukončuje modální stav. Hodnotu této položky pak vrátí metoda **Execute**.

```
ModalResult  : Word;
```

5.3.4.13. Položka TControl.GrowMode

Položka **GrowMode** určuje způsob, jakým se bude měnit pozice a velikost komponenty při změně velikosti vlastníka komponenty. Položka obsahuje kombinaci příznaků gm_ (viz. kapitola 5.1.10).

```
GrowMode      : Word;
```

5.3.4.14. Položka TControl.HelpCtx

Položka **HelpCtx** (Help Context) obsahuje identifikátor stránky nápovědy přiřazené komponentě. Položka je určena pouze pro zápis. Pro čtení identifikátoru nápovědy použijte metodu **GetHelpCtx** (viz. kapitola 5.3.4.60).

```
HelpCtx       : Word;
```

5.3.4.15. Položka TControl.KeyMapper

Položka **KeyMapper** obsahuje ukazatel na uživatelskou funkci provádějící speciální mapování kláves. Více v kapitole 4.6.4. Položka je určena pro čtení i pro zápis. Pokud obsahuje hodnotu **nil**, pak k mapování kláves nedochází. Položka je určena pro čtení i pro zápis.

```
KeyMapper     : TKeyMapper;
```

5.3.4.16. Položka TControl.Validator

Položka **Validator** obsahuje ukazatel na instanci validátoru (tj. potomka třídy **TValidator** – viz. manuál ke knihovně Valids), který řeší validaci a přesun hodnoty komponenty z/do zadaného místa v paměti. Položka **Validator** je určena pouze pro čtení. Pro její nastavení slouží metoda **SetValidator** (viz. kapitola 5.3.4.63). Instance validátoru je odstraněna z paměti při volání destruktoru komponenty.

Některé komponenty např. **TUpDown**, **TScrollBar**, **TTrackBar** nepodporují všechny typy validátorů, ale pouze validátory odvozené od třídy **TOrdinalValidator**. Na tento fakt bude upozorněno v dokumentaci ke komponentě.

```
Validator : PValidator;
```

5.3.4.17. Položka TControl.AfterHandle

Položka **AfterHandle** obsahuje ukazatel na uživatelskou funkci provádějící zpracování vstupních událostí po vlastní obsluze komponenty, tj. po zpracování události metodou **HandleEvent**. Více v kapitole 4.6.2. Položka je určena pro čtení i pro zápis.

```
AfterHandle   : TEventHandler;
```

5.3.4.18. Položka TControl.BeforeHandle

Položka **BeforeHandle** obsahuje ukazatel na uživatelskou funkci provádějící zpracování vstupních událostí před vlastní obsluhou komponenty, tj. před voláním

metody **HandleEvent**. Více v kapitole 4.6.2. Položka je určena pro čtení i pro zápis.

```
BeforeHandle : TEventHandler;
```

5.3.4.19. Položka TControl.AfterNotify

Položka **AfterNotify** obsahuje ukazatel na uživatelskou funkci zpracovávající oznámení komponenty. Tato funkce je volána poté, co je oznámení zpracováno metody **HandleNotification**. Položka je určena pro čtení i pro zápis.

```
AfterNotify  : TNotificationHandler;
```

5.3.4.20. Položka TControl.BeforeNotify

Položka **BeforeNotify** obsahuje ukazatel na uživatelskou funkci zpracovávající oznámení komponenty. Tato funkce je volána těsně před tím, než je oznámení předáno metodě **HandleNotification**. Položka je určena pro čtení i pro zápis.

```
BeforeNotify : TNotificationHandler;
```

5.3.4.21. Položka TControl.BeforePaint

Položka **BeforePaint** obsahuje ukazatel na uživatelskou funkci, která je volána těsně předtím než je komponenta vykreslena vlastní metodou **Paint**. Položku lze číst i zapisovat.

```
BeforePaint  : TPaintHandler;
```

5.3.4.22. Položka TControl.AfterPaint

Položka **AfterPaint** obsahuje ukazatel na uživatelskou funkci, která je volána poté, co je komponenta vykreslená vlastní metodou **Paint**. Položku lze číst i zapisovat.

```
AfterPaint   : TPaintHandler;
```

5.3.4.23. Konstruktor TControl.Init

Konstruktor **Init** inicializuje instanci komponenty.

```
constructor Init( const ABounds: TRect );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSelectable **or** ofSharedPalette
Font	fidDefault
Palette	#$00 (sdílená paleta)

5.3.4.24. Destruktor TControl.Done

Destruktor **Done** uvolní prostředky alokované konstruktorem třídy.

```
destructor Done; virtual;
```

Parametry:

Destruktor nemá žádné parametry.

Návratové hodnoty:

Destruktor nevrací žádnou hodnotu.

Poznámky:

Pokud není v položce Options nastaven příznak ofSharedPalette, pak destruktor uvolní z hromady paletu komponenty, tj. řetězec na který se odkazuje položka Palette.

5.3.4.25. Metoda TControl.Customize

Metoda **Customize** slouží ke konfiguraci komponenty podle zadané třídy nastavení komponenty.

```
procedure Customize( AClass: Word );
```

Parametry:

AClass	Třída nastavení komponenty, tj. konstanta cc_ (viz. kapitola 5.1.6).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda provede nastavení položek komponenty podle zadané třídy nastavení. Pokud třída nastavení neexistuje (tj. nebyla zaregistrována metodou RegisterCustomizeProc), pak metoda neprovede nastavení komponenty nemodifikuje.

5.3.4.26. Metoda TControl.HandleEvent

Metoda **HandleEvent** slouží k obsluze vstupních událostí přicházejících do komponenty.

```
procedure HandleEvent( var AEvent: TEvent ); virtual;
```

Parametry:

AEvent Struktura události (viz. kapitola 4.6).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **HandleEvent** je zavolána automaticky vždy, když se na vstupu objeví nová událost, která přísluší dané komponentě.

Metoda **TControl.HandleEvent** provádí základní obsluhu vstupních událostí, tj. výběr komponenty při prvním kliknutím do její oblasti. Potomci třídy **TControl** mohou metodu **HandleEvent** předefinovat. V každém případě by však vždy měli zavolat metodu **HandleEvent** předka (tj. **inherited** HandleEvent(AEvent);).

Pokud metoda **HandleEvent** vstupní událost zpracuje, pak musí událost označit za zpracovanou pomocí metody **ClearEvent** proto, aby nebyla předána ke zpracování případným dalším komponentám (více o zpracování vstupních událostí v kapitole 4.6).

Aplikace by neměla volat metodu **HandleEvent** přímo. Pokud je potřeba předhodit konkrétní komponentě vstupní událost, lze zavolat k tomu určenou metodu **ProcessEvent** (viz. kapitola 5.3.4.34).

5.3.4.27. Metoda TControl.HandleNotification

Metoda **HandleNotification** slouží k obsluze oznámení komponenty, nebo komponent do ní vložených.

```
procedure HandleNotification( var ANotification: TNotification );
          virtual;
```

Parametry:

ANotification Struktura oznámení (viz. kapitola 4.7).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **TControl.HandleNotification** je definována jako prázdná. Potomci třídy **TControl** mohou tuto metodu předefinovat.

Pokud metoda **HandleNotification** oznámení zpracuje, pak musí toto oznámení označit za zpracované pomocí metody **ClearNotification** proto, aby nebyla předána ke zpracování případným dalším komponentám (více o zpracování oznámení v kapitole 4.7).

Běžné komponenty metodu **HandleNotification** nemusí modifikovat. Metoda je určená spíše pro speciální komponenty, tj. jakési prefabrikáty složené z více komponent, které plní určitou úlohu jako celek.

Aplikace by neměla volat metodu **HandleNotifiaction** přímo. Pokud je potřeba předhodit konkrétní komponentě oznámení, lze zavolat k tomu určené metody **Notify** nebo **NotifyEx** (viz. kapitoly 5.3.4.36 a 5.3.4.35).

5.3.4.28. Metoda TControl.Paint

Metoda **Paint** slouží k vykreslení aktuálního stavu komponenty.

```
procedure Paint( ACanvas: PCanvas ); virtual;
```

Parametry:

ACanvas

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **Paint** je volána automaticky (nikdy ji nevolejte přímo), vždy když je potřeba překreslit oblast displeje, kterou komponenta překrývá. Potomci třídy **TControl** obvykle musí metodu **Paint** předefinovat, tak aby správně vykreslovala stav komponenty. Přímo ve třídě **TControl** je metoda **Paint** definována jako prázdná.

Při vykreslování by měla metoda volit mezi dvěma režimy na základě příznaku položky **Options - ofFlickSafe**. Pokud je tento příznak nastaven, měla by metoda **Paint** vykreslit komponentu tak, aby při překreslování nedocházelo k efektu problikávání, tj. aby oblast komponenty byla vykreslena tak, aby žádný bod komponenty nebyl vykreslen dvakrát různou barvou. Tento způsob vykreslování je obvykle pomalejší. Pokud příznak **ofFlickSafe** nastaven není, metoda **Paint** může komponentu vykreslit libovolným způsobem.

5.3.4.29. Metoda TControl.GetCanvas

Metoda **GetCanvas** vrací odkaz na inicializovanou instanci třídy **TCanvas**, pomocí které je možné kreslit do oblasti komponenty.

```
function  GetCanvas: PCanvas; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací odkaz na instanci třídy **TCanvas** (viz. kapitola 5.3.2). Pokud komponenta není viditelná nebo je zcela překryta jinou komponentou, pak metoda vrací hodnotu **nil**.

Poznámky:

Po spuštění aplikace je automaticky vytvořena jedna instance třídy **TCanvas**, která je sdílena všemi komponentami. Metoda **GetCanvas** provede nezbytná nastavení této instance (inicializuje seznam ořezávacích obdélníku a nastaví implicitní vlastnosti pera, štětce, fontu apod.) a předá ji aplikaci. Metodu **GetCanvas** není tedy možné volat rekurzivně. Pokud aplikace zavolá metodu **GetCanvas** rekurzivně, pak je aplikace ukončena runtimeovou chybou s číslem 240.

Po zavolání metody **GetCanvas** musí aplikace neprodleně provést nezbytné vykreslení a zavolat metodu **ReleaseCanvas**, která uvolní sdílenou instanci **TCanvas** pro jiné komponenty. Pokud metoda **GetCanvas** vrátí hodnotu **nil**, pak aplikace nesmí metodu **ReleaseCanvas** zavolat. Obvyklý kód využívající metody GetCanvas může vypadat následovně:

```
var
  Canvas : PCanvas;

Canvas := Control^.GetCanvas;
if Canvas <> nil then
begin
  with Canvas^ do
  begin
    { Vykreslování do oblasti komponenty }
    Pen.Color := clRed;
    DrawLine( 0, 0, 10, 10 );
  end;
  ReleaseCanvas( Canvas );
end;
```

Metoda **GetCanvas** se používá především k získání instance **TCanvas** pro překreslení části komponenty, při změně jejího stavu (např. při příchodu vstupní události apod)., kdy obvykle není vhodné volat metodu **Repaint** pro překreslení celé oblasti komponenty z důvodu její časové náročnosti.

5.3.4.30. Metoda TControl.GetCanvasRect

Metoda **GetCanvasRect** vrací odkaz na inicializovanou instanci třídy **TCanvas**, pomocí které je možné kreslit do částí oblasti komponenty.

```
function  GetCanvasRect( R: TRect ): PCanvas; virtual;
```

Parametry:

R	Umístění a rozměry obdélníku relativního k levému hornímu rohu komponenty, do něhož se bude kreslit.

Návratové hodnoty:

Metoda vrací odkaz na instanci třídy **TCanvas** (viz. kapitola 5.3.2). Pokud komponenta není viditelná nebo je zcela překryta jinou komponentou, pak metoda vrací hodnotu **nil**.

Poznámky:

Metoda **GetCanvasRect** se chová podobně jako metoda **GetCanvas** s tím rozdílem, že seznam ořezávacích obdélníku vrácené instance **TCanvas** je omezen na oblast danou parametrem R metody. Po zavolání metody **GetCanvas** musí aplikace neprodleně provést nezbytné vykreslení a zavolat metodu **ReleaseCanvas**, která uvolní sdílenou instanci **TCanvas** pro jiné komponenty.

5.3.4.31. Metoda TControl.ReleaseCanvas

Metoda **ReleaseCanvas** uvolní instanci **TCanvas** získanou metodou **GetCanvas** nebo **GetCanvasRect** pro pozdější použití.

```
procedure ReleaseCanvas( ACanvas: PCanvas );
```

Parametry:

ACanvas	Odkaz na instanci třídy **TCanvas** získanou metodou **GetCanvas** nebo **GetCanvasRect**.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

5.3.4.32. Metoda TControl.SetState

Metoda **SetState** slouží ke změně stavu komponenty, tj. položky State.

```
procedure SetState( AState: Word; AEnable: Boolean ); virtual;
```

Parametry:

AState	Modifikovaný příznak stavu komponenty. Tj. jedna z konstant sf_ (viz. kapitola 5.1.8).
AEnable	Parametr udávající zda se má příznak AState nastavit (True) nebo vynulovat (False).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metodu **SetState** je jen zřídkakdy nutné volat přímo. Metoda je volána automaticky při změně stavu komponenty, tj. při volání metody **Show**, **Hide**, **Disable**, **Enable** apod. nebo při výběru komponenty, změně ohniska atd.

Potomci třídy **SetState** mohou metodu předefinovat, tak aby komponenta správně reagovala na změnu stavu, viz. následující příklad, který způsobí překreslení komponenty v případě změny příznaku sfFocused. Potomci vždy musí zavolat metodu SetState předka (tj. **inherited** SetState(AState, AEnable);).

```
procedure TMyControl.SetState( AState: Word; AEnable: Boolean );
begin
  inherited SetState( AState, AEnable );
  if AState = sfFocused then
  begin
     Repaint;
  end;
end;
```

5.3.4.33. Metoda TControl.SetOptions

Metoda **SetOptions** slouží ke změně položky Options komponenty.

```
procedure SetOptions( ASet, AReset: Word ); virtual;
```

Parametry:

ASet	Maska příznaků, které se mají nastavit. Tj. kombinace příznaků s prefixem of_ (viz. kapitola 5.1.9).
AReset	Parametr příznaků, které se mají vynulovat.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda provádí nastavení položky Options tak, že nejprve vynuluje příznaky dané parametrem AReset a poté nastaví příznaky dané parametrem ASet:

Options := (Options and not AReset) or ASet.

5.3.4.34. Metoda TControl.ProcessEvent

Metoda **ProcessEvent** vyvolá zpracování vstupní události komponentou.

```
procedure ProcessEvent( var AEvent: TEvent );
```

Parametry:

AEvent	Vyplněná struktura vstupní události určené ke zpracování.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **ProcessEvent** postupně volá proceduru **BeforeHandle**, metodu **HandleEvent** a proceduru **AfterHandle** (viz. kapitola 4.6.3). Použijte tuto metodu v případě, že je potřeba konkrétní komponentě předat uměle vytvořenou vstupní událost.

5.3.4.35. Metoda TControl.NotifyEx

Metoda **NotifyEx** vyvolá zpracování oznámení komponentou.

```
procedure NotifyEx( var ANotification: TNotification );
```

Parametry:

ANotification	Vyplněná struktura oznámení určeného ke zpracování.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Před voláním metody musí být nastaveny položky Control a Code struktury ANotification, příp. další parametry upřesňující oznámení. Metoda **NotifyEx** zavolá postupně proceduru **BeforeNotify**, metody **HandleNotification** a proceduru **AfterNotify** komponenty. Pokud ani jedna s těchto procedur neoznačí oznámení za obsloužené pomocí procedury **ClearNotification** (viz. kapitola 5.4.4). Pak je oznámení předáno vlastníkovi komponenty.

Příklad:

```
var
  N : TNotification;

  { Vyplneni struktury oznameni }
  N.Control := @Self;
```

```
N.Code := nmXXX;
{ Parametr oznameni }
N.WParam := ...;
{ Vyvolani obsluhy oznameni }
NotifyEx( N );
{ Zpracovani navratove hodnoty }
... := N.LParam
```

5.3.4.36. Metoda TControl.Notify

Metoda **Notify** vyvolá zpracování jednoduchého oznámení bez parametrů komponentou.

```
procedure Notify( ACode: Word );
```

Parametry:

ACode	Identifikátor oznámení nm_ (viz. kapitola 5.1.3).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **Notify** je zjednodušenou variantou metody **NotifyEx**. Umožňuje odeslat jednoduché oznámení bez parametrů.

5.3.4.37. Metoda TControl.ClearEvent

Metoda **ClearEvent** označí událost za obslouženou.

```
procedure ClearEvent( var AEvent: TEvent );
```

Parametry:

AEvent	Struktura obsluhované události.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **ClearEvent** označí událost za obslouženou. Takto označená událost se nebude dále posílat dalším komponentám, které by ji případně mohly obsloužit. Metoda **ClearEvent** je určena k volání pouze v rámci metody **HandleEvent**. Viz. následující příklad:

```
procedure TMyControl.HandleEvent( AEvent: TEvent );
begin
  case AEvent.Code of
    evMouseDown:
    begin
```

```
      { Zpracovani udalosti }
    end;

    evKeyDown:
    begin
      case VirtKey of
        vkLeft:
        begin
          { Zpracovani udalosti }
        end;

        vkRight:
        begin
          { Zpracovani udalosti }
        end;

      else
        Exit;
      end;
    end;

  else
    Exit;
  end;

  ClearEvent( AEvent );
end;
```

5.3.4.38. Metoda TControl.GetEvent

Metoda **GetEvent** provádí vyzvednutí události ze vstupní fronty vstupních ovladačů.

```
procedure GetEvent( var AEvent: TEvent; ANoTimer: Boolean );
            virtual;
```

Parametry:

AEvent	Struktura, která bude po návratu vyplněna vstupní událostí.
ANoTimer	Příznak určující, zda se ze vstupní fronty může vyjmout událost časovače (tj. událost typu evTimer). Událost časovače může být vyjmuta, pokud je tento parametr nastaven na hodnotu False.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud ve vstupní frontě není připravena žádná událost, pak se metoda vrátí a ve struktuře AEvent bude nastavena položka Code na evNothing.

Metodu **GetEvent** obvykle není potřeba volat přímo, využívají jí další metody komponent jako je **Execute**, **MouseEvent** apod.

Třída **TControl** implementuje metodu **GetEvent** tak, že zavolá metodu **GetEvent** svého vlastníka, takže nakonec je zavolána metoda **GetEvent** kořenu stromu

komponent, tj. komponenty **TApplication** (viz. kapitola 5.3.11).

5.3.4.39. Metoda TControl.MouseEvent

Metoda **MouseEvent** je určena pro speciální zpracování vstupních událostí od myši v rámci metody **HandleEvent**. Metoda čeká na specifikovanou vstupní událost.

```
function  MouseEvent( var AEvent: TEvent; AMask: Word ): Boolean;
```

Parametry:

AEvent	Struktura, která bude po návratu metody naplněna vstupní událostí.
AMask	Maska událostí (tj. kombinace příznaků ev_), které způsobí návrat metody.

Návratové hodnoty:

Metoda **MouseEvent** vrací hodnotu True, pokud nastala událost uvolnění tlačítka myši. V ostatních případech vrací hodnotu False.

Poznámky:

Metoda **MouseEvent** čeká ve smyčce na vstupní událost uvolnění tlačítka myši nebo na jednu z událostí specifikovanou maskou AMask. Pokud se taková událost na vstupu objeví metoda se vrátí s vyplněnou strukturou AEvent. Pokud je vstupní událost právě uvolnění tlačítka myši, metoda vrací hodnotu True. Použití metody ukazuje následující příklad.

```
{ Obsluha vstupnich udalosti tlacitka, které reaguje az na
  uvolneni tlacitka mysi }

procedure TMyButton.HandleEvent( var AEvent: TEvent );
var
  Down  : Boolean;
begin
  inherited HandleEvent( AEvent );

  case AEvent.Code of

    evMouseDown:
    begin
      { Priznak stisknuteho tlacitka }
      Down := False;

      { V nasledujici smycce setrvame do uvolneni tlacitka mysi }

      repeat
        { Pokud je ukazatel mysi nad tlacitkem, je vykresleno
          v poloze dole, v opacnem pripade v poloze nahore }

        if Down <> Contains( AEvent.Pos ) then
        begin
          Down := not Down;
          PaintButton( Down );
        end;
```

```
      { Metoda MouseEvent se vrati vždy, když uzivatel posune
        ukazatel mysi (udalost evMouseMove) nebo v pripade, ze uvolni
        tlacitko mysi (udalost evMouseUp) }

      until not MouseEvent( AEvent, evMouseMove );

      { Zde jsme se dostali, protože uzivatel uvolnil tlacitko mysi.
        Pokud je priznak Down nastaven na True, pak bylo tlacitko
        mysi uvolneno nad tlacitkem }

      if Down then
      begin
        { Akce při stisku tlacitka }
        Click;
        { Vratime tlacitko do polohy nahore }
        PaintButton( False );
      end;

      ClearEvent( AEvent );
    end;

  end;
end;
```

5.3.4.40. Metoda TControl.PutEvent

Metoda **PutEvent** vloží událost do uživatelské fronty.

```
procedure PutEvent( const AEvent: TEvent );
```

Parametry:

AEvent	Vyplněná struktura vstupní události.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **PutEvent** vloží vstupní událost do uživatelské fronty. Tato událost je vyjmuta při dalším zpracování vstupní události, tj. voláním metody **GetEvent**.

Příklad: emulace stisku klávesy Enter

```
var
  E : TEvent;

AssignKeyCode( E, kbEnter )
PutEvent( E );
```

5.3.4.41. Metoda TControl.SetBounds

Metoda **SetBounds** nastavuje položku Bounds komponenty.

```
procedure SetBounds( const ABounds: TRect );
```

Parametry:

ABounds	Požadované rozměry a umístění komponenty relativně k levému hornímu okraji vlastníka.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **SetBounds** je interní metoda. Pro změnu umístění nebo rozměrů komponenty použijte metody **Locate**, **MoveTo** příp. **GrowTo**.

Kromě nastavení samotné položky Bounds, metoda zjišťuje zda se komponenta překrývá s jinými komponentami a případně modifikuje příznak sfOverlapped všech komponent ve skupině, kterých se změna dotýká.

5.3.4.42. Metoda TControl.CalcBounds

Metoda **CalcBounds** zjišťuje umístění a velikost komponenty při změně velikosti vlastníka.

```
procedure CalcBounds( var ABounds: TRect; ADelta: TPoint );
            virtual;
```

Parametry:

ABounds	Nové rozměry a umístění komponenty relativně k levému hornímu okraji vlastníka. Pouze výstupní parametr.
ADelta	Změna velikosti rozměrů vlastníka v ose X a Y.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **CalcBounds** je interní metoda. Volá se při změně velikosti vlastníka komponenty.

Při výpočtu nového umístění a rozměrů komponenty metoda **CalcBounds** bere v úvahu položku GrowMode (viz. kapitola 5.3.4.13). Potomci třídy **TControl** mohou tuto metodu v případě potřeby předefinovat.

5.3.4.43. Metoda TControl.ChangeBounds

Metoda **ChangeBounds** slouží k nastavení nového umístění a rozměrů komponenty a

zajišťuje její překreslení.

```
procedure ChangeBounds( const ABounds: TRect ); virtual;
```

Parametry:

ABounds	Požadované rozměry a umístění komponenty relativně k levému hornímu okraji vlastníka.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **ChangeBounds** je interní metoda a proto ji nikdy nevolejte přímo. Jejím úkolem je nastavení nových souřadnic komponenty a překreslení komponenty na požadovaném místě s požadovanými rozměry.

Implicitně metoda **ChangeBounds** volá metody **SetBounds** (viz. kapitola 5.3.4.41) a **Repaint** (viz. kapitola 5.3.4.45). Potomci třídy **TControl** mohou tuto metodu v případě potřeby předefinovat.

5.3.4.44. Metoda TControl.SizeLimits

Metoda **SizeLimits** definuje maximální a minimální rozměry komponenty.

```
procedure SizeLimits( var AMin, AMax: TPoint ); virtual;
```

Parametry:

AMin	Minimální rozměry komponenty.
AMax	Maximální rozměry komponenty.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **SizeLimits** nastavuje parametry AMin a AMax na hodnotu minimálních a maximálních rozměrů komponenty. Implicitně vrací AMin = (0, 0) a AMax = (32767, 32767). Potomci třídy **TControl** mohou tuto metodu předefinovat.

5.3.4.45. Metoda TControl.Repaint

Metoda **Repaint** překreslí oblast displeje, kterou komponenta pokrývá.

```
procedure Repaint; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **Repaint** připraví instanci třídy **TCanvas** a zavolá metodu **Paint** (viz. kapitola 5.3.4.28).

5.3.4.46. Metoda TControl.RepaintRect

Metoda **RepaintRect** překreslí zadanou část oblasti, kterou komponenta pokrývá.

```
procedure RepaintRect( const R: TRect ); virtual;
```

Parametry:

R	Obdélník v relativní k levému hornímu rohu komponenty, který se má překreslit.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **RepaintRect** připraví instanci třídy **TCanvas** a zavolá metodu **Paint** (viz. kapitola 5.3.4.28).

5.3.4.47. Metoda TControl.Prev

Metoda **Prev** vrací předchozí komponentu ve skupině.

```
function  Prev: PControl;
```

Parametry:

Metoda vrací odkaz na předchozí komponentu ve skupině. Pokud je komponenta první vrací metoda odkaz na poslední komponentu.

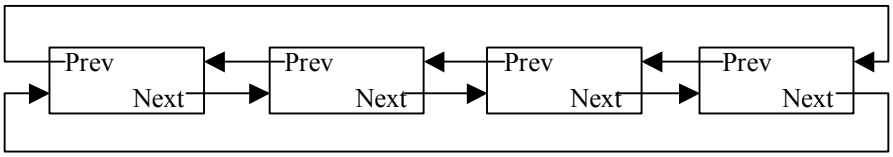
Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **Prev**, je doplňkem položky **Next** komponenty, která obsahuje odkaz na následující komponentu ve skupině. Odkazy získané pomocí položky **Next** a metody

Prev tvoří kruhový seznam. Viz. následující obrázek:



5.3.4.48. Metoda TControl.NextControl

Metoda **NextControl** vrací odkaz na následující komponentu ve skupině.

```
function  NextControl: PControl;
```

Parametry:

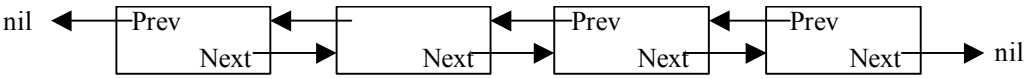
Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací odkaz na následující komponentu ve skupině. Pokud je komponenta poslední vrací hodnotu **nil**.

Poznámky:

Odkazy získané pomocí položky **NextControl** a metody **PrevControl**, na rozdíl od položky **Next** a metody **Prev** netvoří kruhový seznam. Viz. následující obrázek:



5.3.4.49. Metoda TControl.PrevControl

Metoda **PrevControl** vrací odkaz na předchozí komponentu ve skupině.

```
function  NextControl: PControl;
```

Parametry:

Metoda nemá žádné parametry.

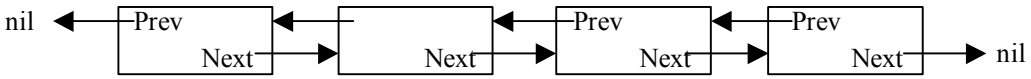
Návratové hodnoty:

Metoda vrací odkaz na předchozí komponentu ve skupině. Pokud je komponenta první vrací hodnotu **nil**.

Poznámky:

Odkazy získané pomocí položky **NextControl** a metody **PrevControl**, na rozdíl od

položky **Next** a metody **Prev** netvoří kruhový seznam. Viz. následující obrázek:



5.3.4.50. Metoda TControl.TopControl

Metoda **TopControl** vrací odkaz na nejbližší modální komponentu směrem ke kořeni stromu komponent.

```
function  TopControl: PControl;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací odkaz na modální komponentu. Pokud žádná komponenta, směrem ke kořeni stromu komponent není modální, metoda vrací hodnotu **nil**.

Poznámky:

Metoda může vrátit hodnotu **nil** jedině v tom případě, že komponenta nebo její vlastník není vložen do stromu komponent aplikace. Jinak vždy vrátí odkaz na konrétní komponentu, obvykle instanci třídy **TApplication** nebo **TDialog**.

5.3.4.51. Metoda TControl.Contains

Metoda **Contains** zjišťuje zda se zadaný bod nachází uvnitř komponenty.

```
function  Contains( APos: TPoint ): Boolean; virtual;
```

Parametry:

APos Globální souřadnice bodu

Návratové hodnoty:

Metoda vrací hodnotu True, pokud se zadaný bod nachází v oblasti komponenty. V opačném případě vrací hodnotu False.

Poznámky:

Metoda nebere v úvahu fakt, že se komponenty mohou překrývat, příp. že mohou být jinak skryty.

5.3.4.52. Metoda TControl.MakeLocal

Metoda **MakeLocal** převádí globální souřadnice do lokálních v rámci komponenty.

```
procedure MakeLocal( ASource: TPoint; var ADest: TPoint );
```

Parametry:

APos	Vstupní globální souřadnice
ADest	Výstupní lokální souřadnice

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

5.3.4.53. Metoda TControl.MakeGlobal

Metoda **MakeGlobal** převádí globální souřadnice do lokálních v rámci komponenty.

```
procedure MakeGlobal( ASource: TPoint; var ADest: TPoint );
```

Parametry:

APos	Vstupní lokální souřadnice bodu.
ADest	Výstupní globální souřadnice bodu.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

5.3.4.54. Metoda TControl.GetExtent

Metoda **GetExtent** vrací rozměry komponenty v podobě obdélníku.

```
procedure GetExtent( var AExtent: TRect );
```

Parametry:

AExtent	Struktura obdélníku, která bude po návratu funkce naplněna rozměry komponenty.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda vrací obdélník, kde levý horní roh je nastaven vždy na hodnotu (0,0) – a pravý horní roh je nastaven na hodnotu (Bounds.B.X – Bounds.A.X, Bounds.B.Y – Bounds.A.Y).

5.3.4.55. Metoda TControl.SetPalette

Metoda **SetPalette** slouží k nastavení celé palety komponenty.

```
procedure SetPalette( const APalette: string; AShared: Boolean );
```

Parametry:

APalette	Ukazatel na řetězec znaků reprezentujících paletu komponenty.
AShared	Příznak určující zda se jedná o sdílenou paletu (True) nebo o paletu, kterou komponenta vlastní (False).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Řetězec předaný metodě APalette musí mít správnou délku, tj. délku rovnající se počtu indexu v paletě komponenty. Struktura a délka palety je vždy uvedena v popisu konkrétní komponenty.

Parametr AShared určuje zda se jedná o sdílenou paletu. Pokud je paleta není sdílená, předpokládá se, že je alokována na hromadě a destruktor komponenty tuto paletu před uvolněním instance komponenty uvolní.

5.3.4.56. Metoda TControl.SetColor

Metoda **SetColor** slouží k nastavení jednoho indexu palety komponenty.

```
procedure SetColor( AIndex: Byte; AColor: TColorRef );
```

Parametry:

AIndex	Pozice barvy v palety komponenty. Pozice jsou číslovány od jedničky.
AColor	Barva, která nahradí stavající barvu palety se na zadané pozici.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je paleta komponenty sdílená, je při volání metody **SetColor** vytvořena její kopie na hromadě a v té je teprve provedena požadovaná změna.

Pokud je parametr AIndex mimo rozsah palety, není provedena žádná změna.

5.3.4.57. Metoda TControl.GetColor

Metoda **GetColor** vrací barvu na zadané pozici palety

```
function  GetColor( AIndex: Byte ): TColorRef;
```

Parametry:

AIndex	Pozice barvy v paletě komponenty. Pozice jsou číslovány od jedničky.

Návratové hodnoty:

Metoda vrací barvu z palety komponenty na zadané pozici. Pokud je parametr AIndex mimo rozsah, pak metoda vrací hodnotu 0.

Poznámky:

5.3.4.58. Metoda TControl.SetFont

Metoda **SetFont** slouží k nastavení implicitního fontu komponenty.

```
procedure SetFont( AFont: Word );
```

Parametry:

AFont	Index fontu, tj. konstanta fid_ (viz. dokumentace ke knihovně Fonts).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **SetFont** nastaví implicitní font komponenty. K získání implicitního fontu komponenty použijte metodu **GetFont** (viz. kapitola 5.3.4.59). Pokud je implicitní font komponenty nastaven na hodnotu fidDefault, pak metoda **GetFont** vrátí implicitní font vlastníka.

5.3.4.59. Metoda TControl.GetFont

Metoda **GetFont** vrací implicitní font komponenty.

```
function  GetFont: Word;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací implicitní font komponenty, tj. jednu z konstant fid_ (viz. dokumentace ke knihovně Fonts).

Poznámky:

5.3.4.60. Metoda TControl.GetHelpCtx

Metoda **GetHelpCtx** vrací identifikátor stránky nápovědy přiřazené komponentě.

```
function  GetHelpCtx: Word; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací identifikátor stránky přiřazené komponentě. Pokud komponenta nemá přiřazen žádnou stránku nápovědy, metoda vrací hodnotu nula.

Poznámky:

Přímo ve třídě **TControl** je metoda **GetHelpCtx** implementována tak, že vrací hodnotu položky HelpCtx. Metoda je definována jako virtuální a potomci třídy **TControl** ji mohou předefinovat (např. **TGroup.GetHelpCtx** v kapitole 5.3.5.16).

5.3.4.61. Metoda TControl.Execute

Metoda **Execute** uvede komponentu do modálního stavu.

```
function  Execute: Word; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací hodnotu reprezentující důvod ukončení modálního stavu, tj. jednu z konstant mr_ (viz. kapitola 5.1.7).

Poznámky:

Přímo ve třídě **TControl** je metoda **Execute** implementována tak, že neprovádí žádnou činnost a vrací konstantu mrCancel. Potomci třídy **TControl** musí metodu **Execute** předefinovat, pokud umožňují přepnutí komponenty do modálního stavu.

5.3.4.62. Metoda TControl.EndModal

Metoda **EndModal** způsobí ukončení modálního stavu komponenty, příp. ukončení modálního stavu některého z vlastníků komponenty směrem ke koření stromu komponent.

```
procedure EndModal( AModalResult: Word );
```

Parametry:

AModalResult	Důvod ukončení modálního stavu, tj. jedna z konstant mr_ (viz. kapitola 5.1.7).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Při ukončování modálního stavu, je vyvoláno oznámení nmEndModal, pomocí kterého aplikace může ukončení modálního stavu zrušit nebo odložit.

5.3.4.63. Metoda TControl.SetValidator

Metoda **SetValidator** slouží k přiřazení validátoru komponentě, tj. nastavení položky Validator (viz. kapitola 5.3.4.16).

```
procedure SetValidator( AValidator: PValidator }; virtual;
```

Parametry:

AValidator	Odkaz na inicializovanou instanci validátoru, tj. potomka třídy **TValidator** (viz. dokumentace ke knihovně Valids). Parametr může mít hodnotu **nil**.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud má třída již přiřazen validátor, pak je tento uvolněn z paměti a je přiřazen validátor daný parametrem AValidator.

Metoda je definována jako virtuální. Potomci třídy **TControl** mohou tuto metodu předefinovat, tak aby mohla provádět případné další nastavení nutné s připojením validátoru.

5.3.4.64. Metoda TControl.Transfer

Metoda **Transfer** slouží k přenosu hodnoty komponenty z/do validátoru a k samotné validaci hodnoty.

```
function  Transfer( AMode: Integer ): Integer; virtual;
```

Parametry:

AMode	Režim přenosu dat. Tento parametr může nabývat třech hodnot	
	vmLoad	Přesun hodnoty z validátoru do komponenty.
	vmStore	Přesun hodnoty z komponenty do validátoru.
	vmValidate	Kontrola hodnoty komponenty validátorem (bez přesunu).

Návratové hodnoty:

Metoda vrací chybový kód validátoru, tj. konstantu s prefixem ve_. V případě úspěšného přesunu nebo validace vrací konstantu veOk. Jinak vrací jinou chybovou konstantu např. veRange, veSyntax apod. Tyto konstanty jsou definovány v knihovně validátoru Valids.

Poznámky:

Metoda **Transfer** je přímo ve třídě **TControl** implementována tak, že vrací konstantu veOk a neprovádí žádný přesun nebo validaci. Potomci třídy **TControl** musí metodu **Transfer** předefinovat tak, aby plnila svůj účel. Obvykle jde pouze o volání metody **TransferText** příp. **TransferOrdinal** třídy **TValidator**, resp. **TOrdinalValidator**.

Metoda **Transfer** musí zkontrolovat, zda má komponenta přiřazenou položku Validator. Pokud položka validator obsahuje hodnotu nil, pak metoda neprovádí žádný přesun a pouze vrátí hodnotu veOk.

5.3.4.65. Metoda TControl.Validate

Metoda **Validate** provádí validaci hodnoty komponenty pomocí přidruženého validátoru.

```
function  Validate: Boolean; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací hodnotu True pokud validace proběhla v pořádku. V opačném případě vrací hodnotu False.

Poznámky:

Metoda Validate využívá interně metodu **Transfer** (viz. kapitola 5.3.4.64) volanou s parametrem vmValidate. Pokud validace neproběhne úspěšně, pak je zavolána metoda **Error** (viz. kapitola 5.3.4.66), která může provést opravu hodnoty komponenty nebo informovat uživatele o chybě.

5.3.4.66. Metoda TControl.Error

Metoda **Error** je volána při validaci hodnoty komponenty, v případě, že hodnota je nesprávně zadaná.

```
function  Error( AErrCode: Integer ): Boolean; virtual;
```

Parametry:

AErrCode	Parametr a AErrCode je při volání metody nastaven na chybový kód, který vrátil validátor přidružený komponentě, tedy konstanty s prefixem ve_.

Návratové hodnoty:

Metoda **Error** vrací implicitně hodnotu False. V případě, že provede opravu hodnotu, např. na implicitní hodnotu, pak může vrátit hodnotu True.

Poznámky:

Metoda **Error** je přímo ve třídě **TControl** implementována tak, že generuje oznámení **nmError**. Toto oznámení obsahuje v položce ErrCode kopii parametr AErrCode a položce Accept hodnotu False. Obsluha oznámení může např. vyvolat dialogové okno s hlašením o chybě, nebo opravit validovanou hodnotu. V případě, že opraví hodnotu, měla by nastavit položku oznámení Accept na hodnotu True (tuto hodnotu pak vrátí i metoda Error).

Metoda **Error** není volána, pokud komponenta nemá přiřazenou položku Validator.

5.3.4.67. Metoda TControl.Focus

Metoda **Focus** změní stav komponenty tak, aby se nacházela v ohnisku (viz. kapitola 4.4.3).

```
function  Focus: Boolean;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací hodnotu True, pokud se podařilo komponentu dostat do ohniska. V opačném případě vrací hodnotu False.

Poznámky:

Pokud se komponenta nachází v ohnisku, jsou v položce State komponenty nastaveny příznaky sfSelected a sfFocused a položka Current vlastníka (tj. potomka třídy **TGroup**) se odkazuje na tuto komponentu.

Pokud má komponenta nastaven příznak ofTopSelect v položce Options, může při změně ohniska dojít k prohození pořadí komponent ve skupině.

Komponenta, která se dostane do ohniska vyvolá oznámení nmEnter a naopak komponenta, která opouští ohnisko vyvolává oznámení nmExit. Pokud má komponenta nastaven příznak ofValidate v položce Options, pak těsně před tím, než opustí ohnisko vyvolá oznámení nmCanExit. Aplikace tak může změnu ohniska blokovat, např. z důvodu validace obsahu komponenty apod.

5.3.4.68. Metoda TControl.Select

Metoda **Select** označí komponentu v rámci skupiny vlastníka za vybranou (viz. kapitola 4.4.3).

```
procedure  Select;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je komponenta vybraná, pak je v položce State komponenty nastaven příznak sfSelected a položka Current vlastníka (tj. potomka třídy **TGroup**) se odkazuje na tuto komponentu.

5.3.4.69. Metoda TControl.PutInFrontOf

Metoda **PutInFrontOf** provede přesun komponenty před zadanou komponentu v seznamu vlastníka.

```
procedure PutInFrontOf( ATarget: PControl );
```

Parametry:

ATarget	Cílová komponenta, před kterou se má tato komponenta přesunout. Pokud má tento parametru hodnotu **nil**, pak je komponenta zařazena jako poslední v seznamu.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Cílová komponenta daná parametrem ATarget , pokud nemá hodnotu nil, musí mít stejného vlastníka jako tato komponenta.

Pokud je potřeba, metoda **PutInFrontOf** provede překreslení všech komponent, který se změna dotkla.

5.3.4.70. Metoda TControl.MakeFirst

Metoda **MakeFirst** přesune komponentu do popředí, tj. přesune ji na první místo v seznamu vlastníka.

```
procedure MakeFirst;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Volání metody **MakeFirst** je identické s voláním metody **PutInFrontOf**(Owner^.First);

5.3.4.71. Metoda TControl.Show

Metoda **Show** změní stav skryté komponenty na viditelnou.

```
procedure Show;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je komponenta viditelná, tj. má nastaven příznak sfVisible v položce State, metoda neprovede žádnou akci. Pokud je komponenta skrytá, pak bude po provedení metody viditelná a zároveň vyvolá oznámení nmShow.

5.3.4.72. Metoda TControl.Hide

Metoda **Hide** změní stav viditelné komponenty na skrytou.

```
procedure Hide;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je komponenta skrytá, tj. nemá nastaven příznak sfVisible v položce State, metoda neprovede žádnou akci. Pokud je komponenta viditelná, pak bude po provedení metody skrytá a zároveň vyvolá oznámení nmHide.

5.3.4.73. Metoda TControl.Disable

Metoda **Disable** změní stav povolené komponenty na zakázanou.

```
procedure Disable;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je komponenta zakázaná, tj. má nastaven příznak sfDisabled v položce State, metoda neprovede žádnou akci. Pokud je komponenta povolená, pak bude po provedení metody zakázaná a zároveň vyvolá oznámení nmDisable.

Zakázané komponenty nedostávají žádná vstupní události a nemohou tak reagovat na vstup uživatele. Zakázané komponenty mohou být viditelně odlišeny od povolených komponent.

5.3.4.74. Metoda TControl.Enable

Metoda **Enable** změní stav zakázané komponenty na povolenou.

```
procedure Disable;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je komponenta povolená, tj. nemá nastaven příznak sfDisabled v položce State, metoda neprovede žádnou akci. Pokud je komponenta zakázaná, pak bude po provedení metody povolená a zároveň vyvolá oznámení nmEnable.

5.3.4.75. Metoda TControl.Locate

Metoda **Locate** slouží ke změně umístění a rozměrů komponenty.

```
procedure Locate( var ABounds: TRect );
```

Parametry:

ABounds	Požadované umístění a rozměry komponenty v souřadnících vlastníka komponenty.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Po provedení metody je komponenta a všechny komponenty, jichž se změna dotkla překresleny. Maximální rozměry komponenty jsou limitovány metodou **SizeLimits** (viz. kapitola 5.3.4.44).

5.3.4.76. Metoda TControl.MoveTo

Metoda **MoveTo** slouží ke změně umístění komponenty.

```
procedure MoveTo( AX, AY: Integer);
```

Parametry:

AX	Požadovaná souřadnice X levého horního rohu komponenty v rámci jejího vlastníka.
AY	Požadovaná souřadnice Y levého horního rohu komponenty v rámci jejího vlastníka.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Po provedení metody je komponenta a všechny komponenty, jichž se změna dotkla překresleny.

5.3.4.77. Metoda TControl.GrowTo

Metoda **GrowTo** slouží ke změně velikosti komponenty.

```
procedure GrowTo( AX, AY: Integer);
```

Parametry:

AX	Požadovaná šířka komponenty.
AY	Požadovaná výška komponenty.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Po provedení metody je komponenta a všechny komponenty, jichž se změna dotkla překresleny. Maximální rozměry komponenty jsou limitovány metodou **SizeLimits** (viz. kapitola 5.3.4.44).

5.3.4.78. Metoda TControl.ShowCaret

Metoda **ShowCaret** povolí zobrazení textového kurzoru komponentou.

```
procedure ShowCaret;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Textový kurzor je viditelný, pokud je povolený (je nastaven příznak sfCursorVis v položce State) a zároveň, pokud je komponenta v ohnisku.

5.3.4.79. Metoda TControl.HideCaret

Metoda **HideCaret** zakáže zobrazení textového kurzoru komponentou.

```
procedure HideCaret;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

5.3.4.80. Metoda TControl.SetCaretPos

Metoda **SetCaretPos** nastaví pozici textového kurzoru.

```
procedure SetCaretPos( APos: TPoint );
```

Parametry:

APos	Požadovaná pozice textového kurzoru relativní k levému hornímu rohu komponenty.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je kurzor v komponentě viditelný, dojde okamžitě při volání této metody k jeho překreslení a požadované místo.

5.3.4.81. Metoda TControl.SetCaretSize

Metoda **SetCaretSize** nastaví velikost obdélníku reprezentujícího textový kurzor.

```
procedure SetCaretSize( ASize: TPoint );
```

Parametry:

ASize	Požadovaná velikost textového kurzoru.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je kurzor v komponentě viditelný, dojde okamžitě při volání této metody k jeho překreslení s novými rozměry..

5.3.4.82. Metoda TControl.ScheduleTimer

Metoda **ScheduleTimer** slouží k naplánování časové události.

```
function  ScheduleTimer( AId: Integer; ATime: Longint;
            AOptions: Word ): Boolean;
```

Parametry:

AId	Identifikátor časovače zvolený aplikací, tj. číslo v rozsahu 0 až 32767. Tento identifikátor musí být jednoznačný pro časovače naplánované v rámci jedné komponenty.
ATime	Čas v milisekundách.
AOptions	Parametr udává typ časovače. Může nabývat jedné ze dvou hodnot, a to toTimer (jednorázový časovač), toPeriodic (periodický časovač). Konstanty jsou definovány v knihovně IoDrv.

Návratové hodnoty:

Metoda vrací hodnotu True, pokud se podařilo časovač úspěšně naplánovat. Pokud je překročen maximální počet souběžně naplánovaných časovačů metoda vrací hodnotu False, a časovač naplánován nebude.

Poznámky:

Počet souběžně běžících časovačů je omezen na 16 a proto je potřeba s tímto prostředkem nakládat uvážlivě.

5.3.4.83. Metoda TControl.CancelTimer

Metoda **CancelTimer** slouží k předčasnému zrušení dříve naplánovaného časovače pomocí metody **ScheduleTimer**.

```
procedure CancelTimer( AId: Integer );
```

Parametry:

AId	Identifikátor časovače zvolený aplikací, při naplánování časovače. V případě, že je tento parametr rovný hodnotě –1, pak jsou zrušeny všechny naplánované časovače.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

5.3.5. Třída TGroup

Třída **TGroup** je abstraktní třída, přímý potomek třídy **TControl**, definující skupinu komponent. Ze třídy **TGroup** jsou odvozeny všechny komponenty, které mohou vlastnit další komponenty, např. **TPage**, **TPageControl**, **TWindow**, **TApplication** apod.

```
TGroup = object( TControl )
public
  Phase     : THandlePhase;
  Current   : PControl;
  Last      : PControl;

  constructor Init( const ABounds: TRect );
  destructor Done; virtual;

  procedure HandleEvent( var AEvent: TEvent ); virtual;
  procedure Insert( AControl: PControl );
  procedure InsertBefore( AControl, ATarget: PControl );
  procedure Delete( AControl: PControl );
  function  First: PControl;
  procedure ForEach( AProc: Pointer );
  function  FirstThat( AProc: Pointer ): PControl;
  function  FocusNext( AForwards: Boolean ): Boolean;
  procedure SelectNext( AForwards: Boolean );

  function  Execute: Word; virtual;
  procedure Repaint; virtual;
  procedure RepaintRect( const R: TRect ); virtual;

  function  GetCanvas: PCanvas; virtual;
  function  GetCanvasRect( R: TRect ): PCanvas; virtual;
  procedure ChangeBounds( const ABounds: TRect ); virtual;

  procedure SetState( AState: Word; AEnable: Boolean ); virtual;
  function  FindById( AId: Word ): PControl;
  function  GetHelpCtx: Word; virtual;

  function  ExecControl( AControl: PControl ): Word;

end;
```

Třída **TGroup** definuje řadu nových metod a položek, které jsou popsány v následujících kapitolách. Dále mění chování několika virtuálních metod předka a to: **Done**, **HandleEvent**, **Execute**, **Repaint**, **RepaintRect**, **GetCanvas**, **GetCanvasRect**, **ChangeBounds** a **SetState**. Tyto metody jsou popsány v kapitole 5.3.4 a v této kapitole nebudou dále rozebírány.

5.3.5.1. Položka TGroup.Phase

Položka **Phase** obsahuje aktuální fázi zpracování vstupní události. Položka je určena pouze ke čtení a to pouze v rámci metody **HandleEvent** (viz. kapitola 5.3.5.15)

```
  Phase     : THandlePhase;
```

Položka obsahuje jednu z konstant s prefixem ph_ (viz. kapitola 5.2.10)

5.3.5.2. Položka TGroup.Current

Položka **Current** obsahuje odkaz na aktuálně vybranou komponentu, tj. komponentu označenou příznakem sfSelected v položce State. Položka Current je určena pouze ke čtení a může obsahovat hodnotu **nil** v případě, že žádná komponenta není vybraná.

```
Current    : PControl;
```

5.3.5.3. Položka TGroup.Last

Položka **Last** obsahuje odkaz na poslední komponentu ve skupině, tj. komponentu, která je na pozadí. Položka je určena pouze ke čtení.

```
Last       : PControl;
```

5.3.5.4. Konstruktor TGroup.Init

Konstruktor **Init** inicializuje instanci komponenty.

```
constructor Init( const ABounds: TRect );
```

Parametry:

ABounds Umístění a rozměry komponenty v rámci jejího vlastníka.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSelectable **or** ofSharedPalette
Font	fidDefault
Palette	#$00 (sdílená paleta)

5.3.5.5. Metoda TGroup.Insert

Metoda **Insert** slouží k vložení komponenty do skupiny.

```
procedure Insert( AControl: PControl );
```

Parametry:

AControl	Odkaz na vkládanou komponentu.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Vkládaná komponenta se nesmí nacházet v žádné jiné skupině. Komponenta je vložena na začátek seznamu (tj. na popředí).

5.3.5.6. Metoda TGroup.InsertBefore

Metoda **InsertBefore** vloží komponentu do skupiny před zadanou komponentu.

```
procedure InsertBefore( AControl, ATarget: PControl );
```

Parametry:

AControl	Odkaz na vkládanou komponentu.
ATarget	Odkaz na komponentu před kterou má být komponenta daná parametrem AControl vložena, nebo hodnota nil, pokud má být komponenta vložena na konec seznamu (tj. na pozadí).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Vkládaná komponenta se nesmí nacházet v žádné jiné skupině.

5.3.5.7. Metoda TGroup.Delete

Metoda **Delete** vyjme komponentu ze skupiny.

```
procedure Delete( AControl: PControl );
```

Parametry:

AControl	Odkaz na vyjímanou komponentu.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Po vyjmutí komponenty není uvolněna instance komponenty uvolněna z paměti.

5.3.5.8. Metoda TGroup.First

Metoda **First** vrací ukazatel na první komponentu ve skupině.

```
function  First: PControl;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda **First** vrací ukazatel na první komponentu ve skupině, tj. na komponentu umístěnou nejvíce v popředí. Pokud skupina neobsahuje žádnou komponentu, metoda vrací hodnotu **nil**.

Poznámky:

5.3.5.9. Metoda TGroup.ForEach

Metoda **ForEach** prochází všechny komponenty v seznamu a pro každou volá callback funkci.

```
procedure ForEach( AProc: Pointer );
```

Parametry:

AProc	Callback funkce volaná pro každý komponentu v oblasti. Viz. poznámky níže.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **ForEach** pro každou komponentu v seznamu zavolá funkci danou parametrem AProc. Metoda postupuje od první komponenty (na popředí) k poslední komponentě (na pozadí). Tato funkce musí být **lokální** a definovaná podle následujícího prototypu:

```
procedure EnumFunc( AControl: PControl ); far;
```

Následující příklad zakáže všechny komponenty v seznamu.

```
procedure DisableControls( AGroup: PGroup );

  procedure DoDisable( AControl: PControl ); far;
  begin
    AControl^.Disable;
  end;
```

```
begin
  AGroup^.ForEach( @DoDisable );
end;
```

5.3.5.10. Metoda TGroup.FirstThat

Metoda **FirstThat** prochází všechny komponenty v seznamu a pro každou volá callback funkci, pokud tato funkce vrátí hodnotu, je procházení ukončeno.

```
function  FirstThat( AProc: Pointer ): PControl;
```

Parametry:

AProc	Callback funkce volaná pro každý komponentu v oblasti. Viz. poznámky níže.

Návratové hodnoty:

Metoda vrací ukazatel na komponentu, pro kterou volaná funkce vrátila hodnotu True. Pokud žádná z volaných callback funkcí nevrátí hodnotu True, metoda **FirstThat** vrací hodnotu **nil**.

Poznámky:

Metoda **FirstThat** postupuje od první komponenty (na popředí) k poslední komponentě (na pozadí). Callback funkce musí být **lokální** a definovaná podle následujícího prototypu:

```
function EnumFunc( AControl: PControl ): Boolean; far;
```

Následující příklad najde první komponentu která obsahuje zadaný bod.

```
function FindControl( APos: TPoint ): TControl;

  function Match( AControl: PControl ): Boolean; far;
  begin
    FintIt := AControl^.Contains( APos );
  end;

begin
  FindControl := AGroup^.ForEach( @Match );
end;
```

5.3.5.11. Metoda TGroup.FocusNext

Metoda **FocusNext** přesune do ohniska následující resp. předchozí komponentu ve skupině.

```
function  FocusNext( AForwards: Boolean ): Boolean;
```

Parametry:

AForwards	Udává směr prohledávání seznamu komponent. Pokud je tento

parametr nastaven na True, prohledává v dopředném směru, tj. od komponent na popředí ke komponentám na pozadí.

Návratové hodnoty:

Metoda vrací hodnotu True, pokud se ohnisko podařilo přesunout. V opačném případě vrací hodnotu False.

Poznámky:

Metoda **FocusNext** přesune do ohnisko další komponentu v zadaném směru, která je viditelná (tj. příznak sfVisible je nastaven), není zakázaná (tj. příznak sfDisabled není nastaven), kterou lze obecně vybrat (tj. příznak sfSelectable je nastaven).

5.3.5.12. Metoda TGroup.SelectNext

Metoda **SelectNext** vybere následující resp. předchozí komponentu ve skupině.

```
procedure SelectNext( AForwards: Boolean );
```

Parametry:

AForwards	Udává směr prohledávání seznamu komponent. Pokud je tento parametr nastaven na True, prohledává v dopředném směru, tj. od komponent na popředí ke komponentám na pozadí.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **SelectNext** vybere další komponentu v zadaném směru, která je viditelná (tj. příznak sfVisible je nastaven), není zakázaná (tj. příznak sfDisabled není nastaven), kterou lze obecně vybrat (tj. příznak sfSelectable je nastaven).

5.3.5.13. Metoda TGroup.FindById

Metoda vyhledá komponentu se zadaným identifikátorem Id.

```
function  FindById( AId: Word ): PControl;
```

Parametry:

AId	Identifikátor hledané komponenty.

Návratové hodnoty:

Pokud je v seznamu nalezena komponenta se zadaným Id, pak metoda vrací ukazatel na tuto komponentu. Pokud žádná komponenta v seznamu nemá zadané Id, pak metoda vrací hodnotu True.

Poznámky:

5.3.5.14. Metoda TGroup.ExecControl

Metoda **ExecControl** provede spuštění komponenty v modálním režimu.

```
function  ExecControl( AControl: PControl ): Word;
```

Parametry:

AControl	Odkaz na komponentu, která se má spustit v modálním stavu. Tato komponenta může být buď již vložena do této skupiny, nebo nesmí být vložena do žádné skupiny.

Návratové hodnoty:

Metoda **ExecControl** vrací hodnotu reprezentující důvod ukončení modálního stavu, tj. jednu z konstant mr_ (viz. kapitola 5.1.7).

Poznámky:

Metoda ExecControl provádí postupně následující činnosti:

- Uložení aktuálního stavu skupiny
- Vložení komponenty AControl do seznamu (pokud v seznamu není)
- Výběr komponenty AControl a přepnutí do modálního stavu
- Volání metody Transfer s parametrem vmLoad (tj. načtení hodnoty komponenty z validátoru)
- Volání metody **Execute** komponenty AControl
- Volání metody Transfer s parametr vmStore (tj. uložení hodnoty komponenty pomocí validátoru). Tento krok se provádí pouze tehdy, pokud metoda **Execute** vrátila hodnotu mrOk.
- Zrušení modálního stavu komponenty AControl
- Vyjmutí komponenty AControl ze seznamu (pokud v seznamu původně nebyla)
- Obnovení stavu skupiny

Obvykle se pomocí metody **ExecControl** spouští modální dialogová okna, nicméně ji lze aplikovat na libovolnou komponentu implementující metodu **Execute**.

5.3.5.15. Metoda TGroup.HandleEvent

Metoda **HandleEvent** slouží k obsluze vstupních událostí přicházejících do komponenty.

```
procedure HandleEvent( var AEvent: TEvent ); virtual;
```

Parametry:

AEvent Struktura události (viz. kapitola 4.6).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **TGroup.HandleEvent** nejprve zavolá metodu **HandleEvent** svého předka, tj. třídy **TControl**. Poté je událost rozeslána přislušným komponentám ve skupině:

Vlastní obsluha se liší podle typu události, tj. zdali se jedná o ohniskovou, poziční nebo jinou událost (viz. kapitola 4.6.3).

Ohniskové události Ohniskové události (viz. kapitola 4.6.3.1) jsou obsluhovány ve třech fázích. V první fázi je nastavena položka Phase (viz. kapitola 5.3.5.1) na hodnotu phPreProcess a událost je odeslána všem komponentám ve skupině, které mají nastaven příznak ofPreProcess v položce Options. Ve druhé fází je položka Phase nastavena na hodnotu phFocused a událost je odeslána pouze komponentě v ohnisku. V poslední třetí fázi, je položka Phase nastavena na hodnotu phPostProcess a událose je předána všem komponentám s nastaveným příznakem ofPostProcess.
Toto relativně složité zpracování ohniskové události umožnuje komponentám s nastaveným příznakem ofPreProcess nebo ofPostProcess zpracovávat ohniskové události, přestože samy v ohnisku nejsou. Při zpracování události metodou HandleEvent tyto komponenty musí testovat fázi zpracování události pomocí položky Phase svého vlastníka (tj. Owner^.Phase).

Poziční události Poziční události (viz. kapitola 4.6.3.2) jsou předávány pouze té komponentě, která obsahuje souřadnice této události. Položka Phase je nastavena vždy na hodnotu phFocused.

Ostatní události Ostatní události (viz. kapitola 4.6.3.3) jsou předávány všem komponentám ve skupině bez rozdílu. Položka Phase je nastavena vždy na hodnotu phFocused.

5.3.5.16. Metoda TGroup.GetHelpCtx

Metoda **GetHelpCtx** vrací identifikátor stránky nápovědy přiřazené komponentě.

```
function  GetHelpCtx: Word; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací identifikátor stránky přiřazené komponentě. Pokud komponenta nemá přiřazen žádnou stránku nápovědy, metoda vrací hodnotu nula.

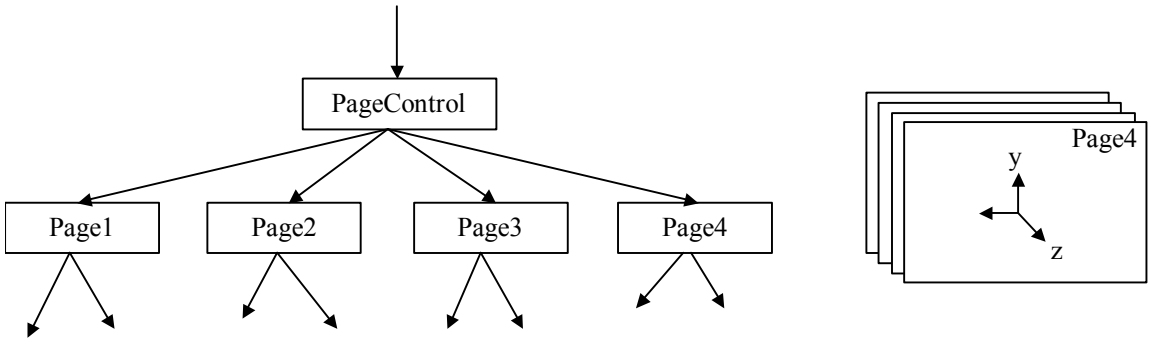
Poznámky:

Metoda **TGroup.GetHelpCtx** předefinovává metodu **GetHelpCtx** svého předka, tj. komponenty **TControl**. Metoda volá metodu **GetHelpCtx** vybrané komponenty, pokud ta vrátí hodnotu nula vrátí metoda hodnotu vlastní položky HelpCtx.

Např. Je možné zavolat metodu **Application^.GetHelpCtx**, která vrátí identifikátor stránky nápovědy komponenty, která se aktuálně nachází v ohnisku. Pokud tato komponenta nemá přiřazen identifikátor stránky nápovědy, metoda vrátí identifikátor nápovědy nejbližší komponenty směrem ke kořenu stromu komponent.

5.3.6. Třída TPageControl

Třída **TPageControl** definuje jednu se základních komponent vizualizačního systému. Jedná se o skupinu překrývajících se přepínatelných stránek (komponent **TPage**). Na následujícím obrázku je ukázka výřezu stromu komponent obsahujícího komponenty **TPageControl** a **TPage**:



Třída **TPageControl** je definována následovně:

```
PPageControl = ^TPageControl;
TPageControl = object( TGroup )
public
  constructor Init( const ABounds: TRect );
  procedure HandleEvent( var AEvent: TEvent ); virtual;
  procedure Repaint; virtual;
  procedure RepaintRect( const R: TRect ); virtual;
  procedure GotoNextPage;
  procedure GotoPrevPage;
  procedure GotoPage( APage : Word );
  procedure CallPage( APage : Word );
  procedure ReturnPage;
  function  NewPage( AId: Word ): PPage;
end;
```

Každá stránka vložená do komponenty **TPageControl** musí mít jednoznačný identifikátor Id (viz. položka Id třídy **TControl** v kapitole 5.3.4.1). Do komponenty

TPageControl nelze vkládat jiné komponenty než **TPage** a z této třídy odvozené komponenty. Třída **TPageControl** nabízí několik metod pro přepínání mezi stránkami a to

GotoPage	Výběr stránky se zadaným Id.
CallPage	Výběr stránky se zadaným Id a uložení Id aktuální stránky pro případ návratu pomocí metodu ReturnPage.
GotoNextPage	Výběr následující stránky. Pořadí stránek je definováno položkami PrevPage a NextPage komponenty TPage.
GotoPrevPage	Výběr předchozí stránky. Pořadí stránek je definováno položkami PrevPage a NextPage komponenty TPage.
ReturnPage	Návrat do stránky, která byla vybraná při posledním volání metody CallPage.

Komponenta **TPageControl** mění chování virtuální metody **HandleEvent**, tak že komponenta reaguje na stisk virtuálních kláves vkPrevWindow, vkNextWindow, vkEsc, tím že volá metody **GotoPrevPage**, **GotoNextPage** a **ReturnPage**. Dále metoda **HandleEvent** reaguje na oběžník cmPageCall viz. kapitola 5.3.6.2.

5.3.6.1. Systém přepínání stránek

Každá stránka, tj. komponenta **TPage** (viz. kapitola 5.3.7) obsahuje čtyři položky:

Položka	Význam
Id	Obsahuje identifikátor vlastní stránky, tj. číslo v rozsahu 0 až 65535.
NextPage	Obsahuje identifikátor implicitní předchozí stránky, tj. identifikátor stránky, která se zobrazí při volání metody **GotoNextPage**.
PrevPage	Obsahuje identifikátor implicitní následující stránky, tj. identifikátor stránky, která se zobrazí při volání metody **GotoPrevPage**.
ReturnPage	Obsahuje identifikátor stránky, která se zobrazí při volání metody **ReturnPage**.

Každá stránka v rámci jedné komponenty **TPageControl** musí mít jednoznačný identifikátor. Tento identifikátor musí aplikace povinně zadat při vytvoření stránky, tj. při volání metody **NewPage**.

Položky **NextPage** a **PrevPage** stránky jsou implicitně nastaveny na hodnotu 0. Pokud bude aplikace využívat možnosti implicitního přepínání stránek, pak musí vyplnit i tyto položky. K implicitnímu přepínání stránek dochází např. při stisku klávesy vkNextWindow a vkPrevWindow. Existuje množství komponenty, např. **TButton**, **TNavigator**, které umožnují při stisku přepnout na následující nebo předchozí stránku a k tomu využívají právě tyto položky.

Položka **ReturnPage** je vyplňována automaticky při volání metody pro přepínání stránek. Při volání metody **CallPage** je do položky ReturnPage nové stránky uložena hodnota aktuální stránky, tj. stránky, ze které došlo k přepnutí. Při volání metody **GotoPage**, **GotoNextPage** a **GotoPrevPage** je položka ReturnPage zkopírována z aktuální stránky. Při volání metody **ReturnPage** je položka ReturnPage využita k návratu do stránky naposledy volané metodou **CallPage**, zároveň je nastavena na

hodnotu 0.

5.3.6.2. Výběr stránky pomocí oběžníku cmPageCall

Stránky lze přepínat jednak pomocí k tomu určených metod (**GotoPage**, **CallPage** apod.) a nebo pomocí oběžníku cmPageCall, na který reaguje metoda **HandleEvent**.

Pokud se přepnutí stránky provádí pomocí oběžníku, k přepnutí dojde až v okamžiku, kdy komponenta **TPageControl** bude zpracovávat vstupní události. Tento způsob přepnutí lze využít ve dvou případech. První případ, který může zřídka nastat, je takový, že není možné zavolat metodu pro přepnutí stránky přímo, např. z důvodu potřeby dokončení nějaké činnosti v aktuální stránce. V druhém častějším případě, lze tohoto způsobu použít pro asynchronní vyvolání stránky, např. pro oznámení nějaké nenadálé události uživateli, i z jiného procesu než je proces vizualizace.

Postup asynchronního volání stránky.

```
Application^.AsyncPageCall( cidMyPageControl, cidMyPage );
```

Metoda **AsyncPageCall** třídy **TApplication** vytvoří strukturu oběžníku a vloží ji do uživatelské fronty vstupních zařízení. Konstanty cidMyPageControl a cidMyPage musí odpovídat reálným Id komponent **TPageControl** a požadované stránky **TPage**.

5.3.6.3. Konstruktor TPageControl.Init

Konstruktor **Init** inicializuje instanci komponenty.

```
constructor Init( const ABounds: TRect );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSelectable **or** ofSharedPalette **or** ofFirstClick
Font	fidDefault

Palette	#$00 (sdílená paleta)

5.3.6.4. Metoda TPageControl.NewPage

Metoda **NewPage** vytvoří a inicializuje instanci komponenty **TPage**.

```
function  NewPage( AId: Word ): PPage;
```

Parametry:

AId	Identifikátor vytvářené stránky.

Návratové hodnoty:

Metoda vrací ukazatel na vytvořenou instanci stránky, tj. komponentu TPage. V případě nedostatku paměti vrací hodnotu **nil**.

Poznámky:

Identifikátor stránky bude uložen do položky Id komponenty stránky. Vytvořenou komponentu stránky je potřeba manuálně vložit nadřízené komponenty **TPageControl** např. pomocí metody **Insert**.

5.3.6.5. Metoda TPageControl.GotoNextPage

Metoda **GotoNextPage** slouží k přepnutí na následující stránku.

```
procedure GotoNextPage;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **GotoNextPage** provede přepnutí na stránku určenou položkou NextPage komponenty **TPage** (viz. kapitola 5.3.7.2). Pokud tato položka obsahuje hodnotu 0, pak metoda neprovede žádnou akci.

5.3.6.6. Metoda TPageControl.GotoPrevPage

Metoda **GotoPrevPage** slouží k přepnutí na předchozí stránku-

```
procedure GotoPrevPage;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **GotoPrevPage** provede přepnutí na stránku určenou položkou PrevPage komponenty **TPage** (viz. kapitola 5.3.7.2). Pokud tato položka obsahuje hodnotu 0, pak metoda neprovede žádnou akci.

5.3.6.7. Metoda TPageControl.GotoPage

Metoda **GotoPage** slouží k přepnutí na stránku se zadaným Id.

```
procedure GotoPage( APage : Word );
```

Parametry:

APage	Identifikátor stránky, tj. obsah položky Id komponenty stránky .

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud parametr APage obsahuje identifikátor, který není shodný s položkou Id ani jedné komponenty **TPage** vložené do této komponenty, pak metoda neprovede žádnou akci.

5.3.6.8. Metoda TPageControl.CallPage

Metoda **CallPage** slouží k přepnutí na stránku se zadaným Id. Id aktuální stránky je uloženo do položky ReturnPage komponenty stránky se zadaným Id.

```
procedure CallPage( APage : Word );
```

Parametry:

APage	Identifikátor stránky, tj. obsah položky Id komponenty stránky .

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud parametr APage obsahuje identifikátor, který není shodný s položkou Id ani jedné komponenty **TPage** vložené do této komponenty, pak metoda neprovede žádnou

akci.

Návrat ze stránky volané metodou **CallPage** je možný pomocí metody **ReturnPage** (viz. kapitola 5.3.6.9).

5.3.6.9. Metoda TPageControl.ReturnPage

Metoda **ReturnPage** slouží k návratu ze stránky volané pomocí metody **CallPage**.

```
procedure ReturnPage;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda provede výběr stránky, jejíž Id je uloženo v položce **ReturnPage** komponenty aktuální stránky.

5.3.7. Třída TPage

Třída **TPage** definuje komponentu stránky vkládané do komponenty **TPageControl** (viz. kapitola 0). Komponenta **TPage** může být vložena pouze do třídy **TPageControl**. Třída **TPage** je definována následovně:

```
PPage = ^TPage;
TPage = object( TGroup )
public
  PrevPage   : Word;
  NextPage   : Word;
  ReturnPage : Word;

  constructor Init( const ABounds: TRect );
  procedure HandleEvent( var AEvent: TEvent ); virtual;
end;
```

5.3.7.1. Položka TPage.PrevPage

Položka **PrevPage** obsahuje identifikátor předchozí stránky, tj. stránky která bude vyvolána při stisku virtuální klávesy vkPrevWindow, nebo při volání metody **GetoPrevPage** komopnenty **TPageControl**. Hodnota identifikátoru je shodná s položkou Id požadované stránky. Položku lze číst i zapisovat v libovolném okamžiku.

5.3.7.2. Položka TPage.NextPage

Položka **NextPage** obsahuje identifikátor předchozí stránky, tj. stránky která bude vyvolána při stisku virtuální klávesy vkNextWindow, nebo při volání metody

GotoNextPage nadřízené komponenty **TPageControl**. Hodnota identifikátoru je shodná s položkou Id požadované stránky. Položku lze číst i zapisovat v libovolném okamžiku.

5.3.7.3. Položka TPage.ReturnPage

Položka **ReturnPage** obsahuje identifikátor stránky, které bude vybrána při volání metody **ReturnPage** nadřízené komponenty **TPageControl**. Položka je nastavována automaticky a je určena pouze ke čtení.

5.3.7.4. Konstruktor TPage.Init

Konstruktor **Init** inicializuje instanci komponenty.

```
constructor Init( const ABounds: TRect );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka. Komponenta TPage by měla mít levý horní roh shodný s levým horním rohem vlastníka, tedy (0, 0). Podobně pravý dolní roh by měl korespondovat s pravým dolním rohem vlastníka.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

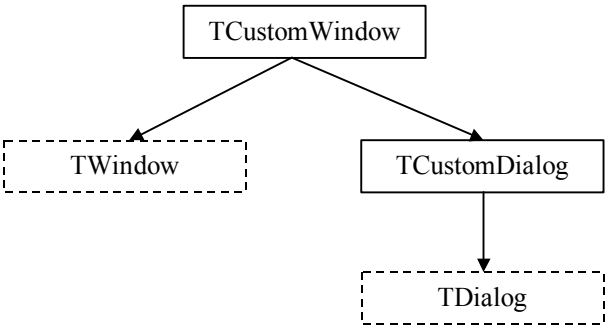
Obvykle není nutné (nedoporučuje se) volat konstruktor komponenty **TPage** přímo. K vytvoření instance stránky použijte metodu **NewPage** nadřízené komponenty **TPageControl** (viz. kapitola 5.3.6.4)

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSelectable **or** ofSharedPalette **or** ofFirstClick **or** ofBackground **or** ofTopSelect **or** ofPaintControl
Font	FidDefault
Palette	#$0F (sdílená paleta)

5.3.8. Třída TCustomWindow

Abstraktní třída **TCustomWindow** definuje komponentu a rozhraní okna. Ze třídy

TCustomWindow jsou odvozeny třídy **TCustomDialog**, **TWindow** a **TDialog**, viz následující obrázek.



Třídy **TWindow** a **TDialog** jsou definovány v knihovně GrCtrls.

Aplikace nesmí vytvářet instance třídy **TCustomWindow** a **TCustomDialog**, jelikož jsou abstraktní. Lze vytvářet instance tříd **TWindow** a **TDialog** a jejich potomků.

```
PCustomWindow = ^TCustomWindow;
TCustomWindow = object( TGroup )
public
  Frame : PCustomFrame;
  Title : PString;

  constructor Init( const ABounds: TRect; const ATitle: string );
  destructor Done; virtual;
  procedure SetTitle( const ATitle: string );
  procedure InitFrame; virtual;
  procedure HandleEvent( var AEvent: TEvent ); virtual;
  procedure SetState( AState: Word; AEnable: Boolean ); virtual;
end;
```

Třída **TCustomWindow** mění chování metody **HandleEvent**, tak že reaguje na stisk virtuální klávesy vkNext a vkPrev, které posouvají fokus v okně na následující resp. předchozí komponentu.

5.3.8.1. Položka TCustomWindow.Frame

Položka **Frame** se odkazuje na instanci rámečku okna (viz. kapitola 5.3.9). Tuto položku inicializuje metoda **InitFrame**, jinak je tato položka určena pouze ke čtení.

5.3.8.2. Položka TCustomWindow.Title

Položka Title obsahuje ukazatel na textový řetězec vypsaný v horní liště (titulek) okna. Tento text není zobrazován samotnou komponentou okna, ale rámečkem vloženým do okna (viz. položka **Frame**, a metoda **InitFrame** v kapitole 5.3.8.5).

5.3.8.3. Konstruktor TCustomWindow.Init

Konstruktor **Init** inicializuje instanci komponenty.

```
constructor Init( const ABounds: TRect; const ATitle: string );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka.
ATitle	Text titulku okna.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSelectable **or** ofTopSelect **or** ofSharedPalette
Font	fidDefault
Palette	#$00 (sdílená paleta)
Title	ukazatel na kopii ATitle alokovanou na hromadě

Konstruktor dále volá metodu **InitFrame** (viz. kapitola 5.3.8.5), jejímž úkolem je nainicializovat položku Frame.

5.3.8.4. Metoda TCustomWindow.SetTitle

Metoda **SetTitle** slouží k nastavení titulku okna.

```
procedure SetTitle( const ATitle: string );
```

Parametry:

ATitle	Požadovaný text titulku okna.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Metoda nastaví textu titulku okna, tj. položku Title a vyvolá překreslení rámečku okna.

5.3.8.5. Metoda TCustomWindow.InitFrame

Metoda **InitFrame** složí k inicializaci instance komponenty rámečku okna.

```
procedure InitFrame; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Metoda **InitFrame** je ve třídě **TCustomWindow** definována jako abstraktní a její vyvolání způsobí runtimeovou chybu. Potomci třídy **TCustomWindow**, ze kterých uživatel bude vytvářet instance musí tuto metodu předefinovat.

Úkolem metody **InitFrame** je vytvořit instanci komponenty rámečku (tedy instanci potomka třídy **TCustomFrame**, viz. kapitola 5.3.9) a odkaz na tuto instanci uložit do položky Frame.

V následujícím příkladu je ukázána funkce **InitFrame** třídy **TWindow** používající rámeček **TFrame**. Tyto třídy jsou definovány v knihovně GrCtrls.

```
procedure TWindow.InitFrame;
var
  R : TRect;
begin
  GetExtent( R );
  Frame := New( PFrame, Init( R ) );
  Frame^.Customize( ccFrame );
  Insert( Frame );
end;
```

5.3.9. Třída TCustomFrame

Třída **TCustomFrame** definuje komponentu rámečku okna. Rámeček okna je speciální komponenta odvozená, která pokrývá celou oblast okna a je v umístěna v seznamu komponent okna jako poslední komponenta (tedy na pozadí).

```
  PCustomFrame = ^TCustomFrame;
  TCustomFrame = object( TControl )
  public
    constructor Init( const ABounds: TRect );
    procedure SetState( AState: Word; AEnable: Boolean ); virtual;
  end;
```

Třída **TCustomFrame** je abstraktní. Její potomci musí předefinovat metodu Paint, tak aby vykreslovala rámeček kolem okrajů okna a zbytek okna vyplnila definovanou barvou.

Potomci **TCustomFrame** obvykle k vykreslování sebe sama využívají paletu potomka komponenty **TCustomWindow**.

5.3.10. Třída TCustomDialog

Abstraktní třída **TCustomDialog** definuje komponentu a rozhraní dialogového okna. Aplikace nesmí vytvářet instance třídy TCustomDialog. Lze ale vytvářet instance jejího potomka, tj. třídy **TDialog** definované v jednotce GrCtrls.

Třída **TCustomDialog** je definována níže:

```
PCustomDialog = ^TCustomDialog;
TCustomDialog = object( TCustomWindow )
public
  constructor Init( const ABounds: TRect; const ATitle: string );
  procedure HandleEvent( var AEvent: TEvent ); virtual;
end;
```

Dialogové okno rozšiřuje obsluhu vstupních událostí o zpracování virtuálních kláves vkEnter a vkEsc, které způsobí ukončení modálního stavu okna s výsledkem mrOk resp. mrCancel.

5.3.10.1. Metoda TDialog.Init

Konstruktor **Init** inicializuje instanci komponenty.

```
constructor Init( const ABounds: TRect; const ATitle: string );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka.
ATitle	Text titulku okna.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSelectable **or** ofTopSelect **or** ofSharedPalette
Font	fidDefault
Palette	#$00 (sdílená paleta)
Title	ukazatel na kopii ATitle alokovanou na hromadě

Konstruktor dále volá metodu **InitFrame** (viz. kapitola 5.3.8.5), jejímž úkolem je nainicializovat položku Frame.

5.3.11. Třída TApplication

Třída **TApplication** definuje kořenovou komponentu každé aplikace. Komponenta **TApplication** řeší odebírání vstupních událostí ze vstupních zařízení, provádění automatů vstupních a výstupních zařízení apod.

```
PApplication = ^TApplication;
TApplication = object( TGroup )
public
  InputDriver    : PInputDriver;
  DisplayDriver  : PDisplayDriver;
  SharedCanvas   : PCanvas;
  OnIdle         : TIdleProc;
  Settings       : PApplicationSettings;
  MessageBoxFunc : TMessageBoxFunc;

  constructor Init( AInputDriver: PInputDriver;
                    ADisplayDriver: PDisplayDriver;
                    ASettings: PApplicationSettings );

  destructor Done; virtual;

  procedure GetEvent( var AEvent: TEvent;
                      ANoTimer: Boolean ); virtual;

  procedure HandleEvent( var AEvent: TEvent ); virtual;

  procedure Run;
  procedure AsyncPageCall( APageControlId: Word; APageId: Word );
  procedure DoIdle; virtual;
  function  MessageBox( ABounds: TRect; const ATitle,
              AText: string; AFlags: Word ): Word;
  function  LoadSettings: Boolean;
  procedure StoreSettings;
end;
```

Komponenta **TApplication** slouží pouze jako kontejner pro další komponenty. Protože samotná komponenta **TApplication** neprovádí žádné vykreslování, musí být zcela pokryta vloženými komponentami (např. jedna komponenta TPageControl, nebo TDesktop apod.)

5.3.11.1. Položka TApplication.InputDriver

Položka **InputDriver** obsahuje odkaz na ovladač vstupních zařízení **TInputDriver** (viz. dokumentace ke knihovně IoDrv). Položka je inicializována parametrem konstruktoru třídy a je určena pouze pro čtení.

```
InputDriver    : PInputDriver;
```

5.3.11.2. Položka TApplication.DisplayDriver

Položka **DisplayDriver** obsahuje odkaz na ovladač zobrazovacího zařízení, tj. potomka třídy **TDisplayDriver**. Položka je inicializována parametrem konstruktoru třídy a je určena pouze pro čtení.

```
DisplayDriver : PDisplayDriver;
```

5.3.11.3. Položka TApplication.SharedCanvas

Položka **SharedCanvas** obsahuje odkaz na instanci třídy **TCanvas** používanou při vykreslování komponent. Položka je víceméně interní a není doporučeno ji číst ani zapisovat. Pro manipulaci se sdílenou instancí TCanvas používejte k tomu určené metody **GetCanvas** resp. **GetCanvasRect** a **ReleaseCanvas**.

```
SharedCanvas  : PCanvas;
```

5.3.11.4. Položka TApplication.Settings

Položka **Setttings** obsahuje odkaz na datovou strukturu **TApplicationSetttings** s nastavením vstupních a výstupních zařízení terminálu (např. jas, kontrast, nastavení kalibrace dotykového panelu apod.). v zálohované paměti RAM. Tato položka je inicializována parametrem konstruktoru třídy. K manipulaci s položkou Settings použijte k tomu určené metody **LoadSetttings** a **StoreSettings** (viz. kapitoly 5.3.11.13 a 5.3.11.14).

```
Settings      : PApplicationSettings;
```

5.3.11.5. Položka OnIdle

Položka **OnIdle** obsahuje odkaz na vzdálenou proceduru bez parametrů, která je volána ze smyčky obsluhy vstupních událostí vždy, když je fronta vstupních událostí prázdná (tedy všechny události, která do daného okamžiku nastaly, byly obslouženy).

```
OnIdle        : TIdleProc;
```

Položku (resp. proceduru OnIdle) lze využít např. k tomu, aby uspala proces terminálu a dala procesorový čas k dispozici procesům s nižší prioritou, viz. následující příklad:

```
procedure AppIdle; far;
begin
  Retos.Wait( 1 );
end;

Application^.OnIdle := AppIdle;
```

5.3.11.6. Položka MessageBoxFunc

Položka **MessageBoxFunc** obsahuje ukazatel funkce vytvářející instanci jednoduchého dialogového okna pro zobrazení zpravý uživateli. Tato funkce je vždy poplatná danému typu terminálu a je definována v konfigurační knihovně jeho ovladače. Např. ovladač terminálu Term10 obsahuje funkci vytvářející jednoduché dialogové okno bez tlačítek, ovladač terminálu Touch11 obsahuje funkci vytvářející dialogové okno s různými tlačítky apod. Tato položka je určena jak pro čtení, tak pro zápis a je automaticky nastavena při volání metody Customize objektu TApplication.

Položku MessageBoxFunc využívá metoda **MessageBox** (viz. kapitola 5.3.11.12).

```
MessageBoxFunc : TMessageBoxFunc;
```

Procedurální TMessageBoxFunc (viz. kapitola 5.2.10.1) je definován následovně:

```
TMessageBoxFunc = function( const ABounds: TRect;
  const ATitle, AText: string; AFlags: Word ): PCustomDialog;
```

Jedná se o funkci, jejímž úkolem je vytvořit instanci potomka třídy **TCustomDialog**, např. **TDialog**. Naplnit ji potřebnýma komponentama, jako je **TStaticText**, **TButton** apod.

Parametr ABounds udává umístění rozměry dialogového okna v rámci komponenty TAplication. Položka ATitle udává text titulku okna. Položka AText obsahuje text zobrazený uvnitř okna. Položka Flags obsahuje kombinaci příznaků upřesňující parametry dialogového okna, např. typy tlačítek apod.

5.3.11.7. Konstruktor TApplication.Init

Konstruktor **Init** provádí inicializaci instance třídy.

```
constructor Init( AInputDriver: PInputDriver;
                  ADisplayDriver: PDisplayDriver;
                  ASettings: PApplicationSettings );
```

Parametry:

AInputDriver	Odkaz na instanci ovladače vstupů.
ADisplayDriver	Odkaz na instanci ovladače displeje.
ASettings	Odkaz na strukturu s nastavení ovladačů vstupně výstupních zařízení v zálohované paměti RAM. Pokud má tento parametr hodnotu **nil**, pak jsou všechny zařízení nastaveny s implicitními parametry.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor třídy **TApplication** provede inicializaci samotné třídy, inicializaci vstupních ovladačů a inicializaci ovladače displeje. Nakonec provádí načtení parametrů ovladačů ze zálohované struktury ASettings.

Pokud je zálohovaná struktura ASettings poškozena (nebo neinicializována), pak je do ní při volání konstruktoru **Init** uloženo implicitní nastavení pro konkrétní vstupní nebo výstupní zařízení.

Příklad volání konstruktoru instance **TApplication** v případě použití terminálu Term10.

```
  Tree.App :=
    New( PApplication, Init (
      New( PInputDriver, Init(
        New( PT10KeybDriver, Init( ioTerm10 )),  { Klavesnice }
```

```
      nil                                        { Mys       }
    )),
    New( PT10DispDriver, Init( ioTerm10 ) ),     { Displej   }
    @g_AppSettings
  ));
```

5.3.11.8. Destruktor TApplication.Done

Destruktor **Done** provádí uvolnění paměťových a hw prostředků alokovaných instancí komponenty **TApplication**.

```
destructor Done; virtual;
```

Parametry:

Destruktor nemá žádné parametry.

Návratové hodnoty:

Destruktor nevrací žádnou hodnotu.

Poznámky:

Destruktor uvolní celý strom komponent vložený do instance komponenty aplikace. Nakonec uvolní z paměti instance ovladače vstupů a displeje.

5.3.11.9. Metoda TApplication.Run

Metoda **Run** spustí hlavní smyčku obsluhy událostí, tedy spustí aplikaci.

```
procedure Run;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda Run spustí hlavní smyčku obsluhy událostí. Tuto smyčku lze opustit (tedy ukončit aplikaci) buď voláním metody **EndModal** (viz. kapitola 5.3.4.62) a nebo stiskem klávesy vkAppExit.

5.3.11.10. Metoda TApplication.AsyncPageCall

Metoda **AsyncPageCall** slouží k přepnutí stránky komponenty **TPageControl**.

```
procedure AsyncPageCall( APageControlId: Word; APageId: Word );
```

Parametry:

APageControlId	Identifikátor komponenty PageControl.
APageId	Identifikátor stránky.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metodu je možné volat i z jiného procesu, naprosto nezávisle na stavu procesu terminálu.

Přepnutí je provedeno pomocí speciálního oběžníku cmAsyncPageCall vloženého do uživatelské vstupní fronty. Tento oběžník obsahuje zadané identifikátor komponent APageControlId a APageId. Oběžník zpracuje komponenta **TPageControl** se shodným Id, tak že zavolá vlastní metodu **CallPage**.

5.3.11.11. Metoda TApplication.DoIdle

Metoda **DoIdle** je volána ze smyčky obsluhy vstupních událostí vždy, když je fronta vstupních událostí prázdná (tedy všechny události, která do daného okamžiku nastaly, byly obslouženy).

```
procedure DoIdle; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **DoIdle** implicitně volá proceduru odkazovanou položkou OnIdle (viz. kapitola 5.3.11.5). Potomci mohou tuto metodu předefinovat.

5.3.11.12. Metoda TApplication.MessageBox

Metoda **MessageBox** slouží k zobrazení speciálního dialogového okna se zprávou pro uživatele.

```
function  MessageBox( ABounds: TRect; const ATitle,
  AText: string; AFlags: Word ): Word;
```

Parametry:

ABounds	Umístění a rozměry dialogového okna
ATitle	Titulek okna
AText	Text zobrazovaný uvnitř okna
AFlags	Kombinace příznaků upřesňujících další vlastnosti okna. Tyto příznaky jsou poplatné implementaci konkrétního terminálu (viz. manuál ke knihovně konfigurace používaného terminálu, manuály s xxxCus, např. T10Cus, T11MCus apod.).

Návratové hodnoty:

Metoda vrací jednu z konstant mr_ (viz. kapitola 5.1.7), např. mrOk nebo mrCancel specifikující důvod ukončení modálního stavu dialogového okna.

Poznámky:

Metoda **MessageBox** pro vytvoření dialogu volá funkci uvedenou v položce MessageBoxFunc (viz. kapitola 5.3.11.6). Tato funkce je implementována různě pro různé typy terminálů.

5.3.11.13. Metoda TApplication.LoadSettings

Metoda **LoadSettings** provádí načtení nastavení ovladačů vstupních zařízení a displeje ze struktury AppSettings.

```
function  LoadSettings: Boolean;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda **LoadSettings** vrací hodnotu True, pokud se podařilo načíst všechny parametry. V případě, že je struktura poškozena, metoda vrací hodnotu False. (Struktura AppSettings je po částech zajištěna kontrolním součtem).

Poznámky:

Metoda **LoadSettings** provádí načtení parametrů ovladače klávesnice, myši a displeje ze struktury, na kterou se odkazuje položka AppSettings (nastavuje se v konstruktoru třídy).

Pokud je položka AppSettings nastavena na hodnotu **nil**, pak metoda neprovádí žádnou akci a vrací hodnotu True, jako kdyby byly parametry úspěšně načteny.

5.3.11.14. Metoda TApplication.StoreSettings

Metoda **StoreSettings** provádí uložení aktuálního nastavení ovladačů vstupních

zařízení a displeje ze struktury AppSetttings.

```
procedure StoreSettings;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **StoreSettings** uloží aktuální nastavení ovladače klávesnice, myši a displeje do struktury, na kterou se odkazuje položka **AppSettings**. Pokud je tato položka nastavena na hodnotu **nil**, pak metoda neprovádí žádnou akci.

5.3.12. Třída TDesktop

Komponenta **TDesktop** je jednoduchá komponenta odvozená ze třídy **TGroup**, která vyplní celou svou plochu první barvou své palety. Tato komponenta se používá jako pozadí komponenty **TApplication**, pokud není zcela překryta jinými komponentami (jako je např. **TPageControl** apod.).

```
PDesktop = ^TDesktop;
TDesktop = object( TGroup )
public
  constructor Init( const ABounds: TRect );
end;
```

5.3.12.1. Konstruktor TDesktop.Init

Konstruktor **Init** inicializuje instanci komponenty.

```
constructor Init( const ABounds: TRect );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds

EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSharedPalette **or** ofBackground **or** ofPaintControl **or** ofFirstClick
Font	fidDefault
Palette	#$00 (sdílená paleta)

5.3.13. Třída TCustomButton

Třída **TCustomButton** je bázovou třídou pro implementaci komponenty tlačítka.

```
PCustomButton = ^TCustomButton;
TCustomButton = object( TControl )
public
  Mode    : Word;
  Pressed : Boolean;
  Down    : Boolean;
  Param   : Word;
  Pages   : PPageControl;

  constructor Init( const ABounds: TRect; AMode: Word;
                    AParam: Word );
  procedure HandleEvent( var AEvent: TEvent ); virtual;
  function  Transfer( AMode: Integer ): Integer; virtual;
  procedure Click; virtual;
end;
```

5.3.13.1. Události obsluhované komponentou TCustomButton

Komponenta **TCustomButton** reaguje na události myši (příp. dotykového panelu). Pokud je v položce Mode (viz. kapitola 5.3.13.4) nastaven příznak btmPush, je v okamžiku kliknutí do oblasti komponenty zavolána metoda **Click** (viz. kapitola 5.3.13.10). Pokud tento příznak nastaven není, pak je metoda **Click** zavolána až v okamžiku uvolnění tlačítka myši.

Komponenta dále reaguje na stisk klávesy vkEnter, která způsobí okamžité volání metody **Click**.

5.3.13.2. Oznámení generované komponentou TCustomButton

Pokud je v položce Mode (viz. kapitola 5.3.13.4) nastaven příznak btmNotify, pak komponenta při stisku tlačítka generuje oznámení **nmClick**, které je bez parametrů.

5.3.13.3. Validace hodnoty komponenty TCustomButton

Pokud má komponenta **TCustomButton** v položce Mode nastaven příznak btmSwitch, pak se komponenta chová jako dvoustavový přepínač. V takovém případě je možné ke komponentě použít validátor ordinálního typu, tj. validátor vycházející ze třídy **TOrdinalValidator**, tedy např. **TByteValidator**, **TIntegerValidator** apod. V případě, že je tlačítko stisknuté vrací hodnotu 1, v opačném případě hodnotu 0.

5.3.13.4. Položka TCustomButton.Mode

Položka **Mode** obsahuje příznaky upřesňující chování komponenty tlačítka. Může obsahovat kombinaci příznaků btm_ uvedených v kapitole 5.1.11. Položka **Mode** je inicializována parametrem konstruktoru a je určena pouze pro čtení.

```
Mode : Word;
```

5.3.13.5. Položka TCustomButton.Pressed

Položka **Pressed** udává stav tlačítka, pokud je v položce Mode nastaven příznak btmSwitch. Pokud je tlačítko stisknuto, obsahuje hodnotu True. V opačném případě obsahuje hodnotu False. Pokud v položce Mode není nastaven příznak btmSwitch, pak obsahuje vždy hodnotu False.

Položka **Pressed** je nastavována automaticky při stisku tlačítka (příp. validátorem) a je určena pouze pro čtení.

```
Pressed : Boolean;
```

5.3.13.6. Položka TCustomButton.Down

Položka **Down** udává stav tlačítka. Pokud je nastavena na hodnotu True, pak je tlačítko stisknuto. Položku **Down** se nastavuje v metodě **HandleEvent** v případě, že uživatel kliknul do oblasti tlačítka a drží tlačítko myši stisknuté. Po uvolnění tlačítka myši je položka nastavena zpět na hodnotu False. Při stisku tlačítka tímto způsobem je navíc zavolána metoda **Repaint**. Potomci třídy TCustomButton mohou tuto položku využít společně s položkou Pressed k vykreslení tlačítka ve správné poloze. Pokud platí podmínka *Down xor Pressed = True*, je potřeba tlačítko vykreslit ve stisknuté poloze. V opačném případě v poloze uvolněné.

Položka **Down** je nastavována automaticky a je určena pouze pro čtení.

```
Down : Boolean;
```

5.3.13.7. Položka TCustomButton.Param

Položka Param obsahuje pomocný parametr akce při stisku tlačítka. Může obsahovat např. kód emulované klávesy, identifikátor stránky apod.

```
Param : Word;
```

5.3.13.8. Položka TCustomButton.Pages

Položka **Pages** obsahuje odkaz na komponentu **TPageControl** (viz. kapitola 5.3.6). Pokud je v položce Mode nastavena akce tlačítka na btmGoto, btmCall, btmReturn, btmNext nebo btmPrev, pak je při stisku tlačítka zavolána příslušná metoda této komponenty s parametrem daným položkou Param (viz. kapitola 5.3.13.7)

Položku Pages je možné zapisovat i číst v libovolném okamžiku.

```
Pages : PPageControl;
```

5.3.13.9. Konstruktor TCustomButton.Init

Konstruktor **Init** provádí inicializaci instance třídy.

```
constructor Init( const ABounds: TRect; AMode: Word;
                  AParam: Word );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka.
AMode	Příznaky upřesňující chování tlačítka, viz. konstanty btm_ v kapitole 5.1.11.
AParam	Parametr akce při stisku tlačítka závislý na nastavení parametru AMode.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSelectable **or** ofSharedPalette **or** ofFirstClick
Font	fidDefault
Palette	#$00 (sdílená paleta)
Mode	parametr konstruktoru AMode
Param	parametr konstruktoru AParam
Pressed	False
Down	False
Pages	**nil**

5.3.13.10. Metoda TCustomButton.Click

Metoda **Click** provádí akci při stisku tlačítka.

```
procedure Click; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda provádí akci podle nastavení položky Mode a Param, inicializovaných konstruktorem Init (viz. kapitola 5.3.13.9)

5.3.14. Třída TPaintBox

Komponenta **TPainBox** je jednoduchá komponenta, která slouží k vytvoření obdelníkové kreslící oblasti.

Sama komponenta **TPaintBox** vykresluje pouze jednobarevné pozadí v barvě z palety s indexem 1. Aplikace musí vytvořit funkci pro vykreslování a přiřadit jí položce **AfterPaint**.

Pokud aplikace chce vyplnit celou oblast komponenty TPaintBox vlastním obsahem, pak není nutné, aby komponenta sama vykreslovala jednobarevné pozadí. Toho lze dosáhnou tím, že se v položce **Options** vynuluje příznak **ofBackground**.

```
PPaintBox = ^TPaintBox;
TPaintBox = object( TControl )
public
  constructor Init( const ABounds: TRect );
end;
```

5.3.14.1. Konstruktor TPaintBox.Init

Konstruktor **Init** inicializuje instanci komponenty.

```
constructor Init( const ABounds: TRect );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$0000
CaretSize	(1, 8)
State	sfVisible

Options	ofSharedPalette **or** ofBackground
Font	fidDefault
Palette	#$00 (sdílená paleta)

5.3.15. Třída TCustomMenu

Abstraktní třída **TCustomMenu** je bázovou třídou pro implementaci menu.

```
PCustomMenu = ^TCustomMenu;
TCustomMenu = object( TControl )
public
  ParentMenu  : PMenuControl;
  Menu        : PMenu;
  Current     : PMenuItem;
  Flags       : Word;
  ItemHeight  : Integer;
  Selected    : PMenuItem;
  Pages       : PPageControl;

  constructor Init( const ABounds: TRect );
  function  Execute: Word; virtual;
  function  FindItem(Ch: Char): PMenuItem;
  procedure GetItemRect( AItem: PMenuItem;
    var AR: TRect ); virtual;
  procedure PaintItems( AItem1, AItem2: PMenuItem ); virtual;
  procedure HandleEvent( var AEvent: TEvent ); virtual;
  function  NewSubControl( const AItemBounds: TRect; AMenu: PMenu;
    AParentMenu: PMenuControl ): PMenuControl; virtual;
  procedure ClickItem( AItem: PMenuItem ); virtual;
  procedure SetFlags( ASet, AReset: Word ); virtual;
  procedure SetItemHeight( AItemHeight: Integer );
end;
```

5.3.15.1. Události obsluhované komponentou

Komponenta **TCustomOverlay** obsluhuje události myši (příp. dotykového panelu). Provedení akce svázané s položkou menu se provádí pomocí kliknutí příp. uvolnění tlačítka myši v obdélníku položky.

Komponenta dále obsluhuje nasledující klávesy (pouze v modálním režimu):

Klávesa	**Akce**
vkUp	Označení předchozí položky
vkDown	Označení následující položky
vkEsc	Návrat do předchozí úrovně menu
vkEnter	Výběr označené položky, tj. provedení akce svázané s položkou
vkLeft	Návrat do předchozí úrovně menu, nebo označení předchozí položky.
vkRight	Výběr označené položky, tj. provedení akce svázané s položkou, nebo označení následující položky.
vkHome	Označení první položky menu
vkEnd	Označení poslední položky menu

5.3.15.2. Oznámení generované komponentou

Komponenta může generovat oznámení **nmClick**, při výběru položky, která má nastaven příznak mifNotify. Položka oznámení ItemId v takovém případě obsahuje identifikátor položky zadaný při volání funkce **NewMenuItem** (viz. kapitola 5.4.10.2).

5.3.15.3. Položka TCustomMenu.ParentMenu

Položka **ParentMenu** obsahuje odkaz na komponentu rodičovského menu, ze které bylo toto menu vyvoláno. Např. komponenta **TMenuBox** zde obsahuje ukazatel na jinou komponentu **TMenuBox** příp. **TMenuBar**.

```
ParentMenu : PCustomMenu;
```

5.3.15.4. Položka TCustomMenu.Menu

Položka Menu obsahuje odkaz na menu zobrazované touto komponentou. Položku lze pouze číst.

```
Menu : PMenu;
```

5.3.15.5. Položka TCustomMenu.Current

Položka **Current** obsahuje odkaz na aktuálně vybranou položku menu. Položka je aktualizovaná automaticky a lze ji pouze číst.

```
Current : PMenuItem;
```

5.3.15.6. Položka TCustomMenu.Flags

Položka **Flags** upřesňuje chování menu. Může obsahovat kombinaci konstant s prefixem mf_ (viz. kapitola 5.1.12). Položka je určena pouze ke čtení. Pro její změnu použijte metodu **SetFlags** (viz. kapitola 5.3.15.15).

```
Flags : Word;
```

5.3.15.7. Položka TCustomMenu.ItemHeight

Položka **ItemHeight** obsahuje výšku oblasti (tj. počet pixelů) položky menu. Pokud je hodnota této položky menší nebo rovna nule, pak je výška této oblasti dána výškou znaků použitého fontu zvetšená o absolutní hodnotu této položky. Položka je určena pouze pro čtení. Pro její nastavení slouží metoda **SetItemHeight** (viz. kapitola 5.3.15.16).

```
ItemHeight    : Integer;
```

5.3.15.8. Položka TCustomMenu.Selected

Položka **Selected** obsahuje odkaz na naposledy vybranou položku při ukončení modálního stavu menu.

```
Selected     : PMenuItem;
```

5.3.15.9. Položka TCustomMenu.Pages

Položka **Pages** obsahuje odkaz na komponentu **TPageControl** (viz. kapitola 0). Pokud je v položce Flags položky menu nastavena akce na mifGoto pak mifCall, mifReturn, pak je při výběru položky zavolána příslušná metoda této komponenty s parametrem daným položkou Param (viz. kapitola 5.2.11), tj. proveden skok na příslušnou stránku

Položku **Pages** je možné zapisovat i číst v libovolném okamžiku.

```
Pages : PPageControl;
```

5.3.15.10. Konstruktor TCustomMenu.Init

Konstruktor **Init** inicializuje instanci komponenty.

```
constructor Init( const ABounds: TRect );
```

Parametry:

ABounds Umístění a rozměry komponenty v rámci jejího vlastníka.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSelectable **or** ofSharedPalette **or** ofPostProcess
Font	fidDefault
Palette	#$00 (sdílená paleta)

5.3.15.11. Metoda TCustomMenu.GetItemRect

Metoda **GetItemRect** vrací obdélník položky menu v souřadnicích komponenty menu.

```
procedure GetItemRect( AItem: PMenuItem; var AR: TRect );
```

Parametry:

AItem	Odkaz na položku menu.
AR	Odkaz na strukturu obdélníku, do které bude uloženy pozice a rozměry položky menu.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Přímo ve třídě **TCustomMenu** je tato metoda definována jako abstraktní. Potomci třídy musí tuto metodu předefinovat.

5.3.15.12. Metoda TCustomMenu.PaintItems

Metoda **PaintItems** slouží k vykreslení jedné nebo dvou položek menu.

```
procedure PaintItems( AItem1, AItem2: PMenuItem );
```

Parametry:

AItem1	První vykreslovaná položka menu. Parametr může obsahovat hodnotu **nil**.
AItem2	Druhá vykreslovaná položka menu. Parametr může obsahovat hodnotu **nil**.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Přímo ve třídě **TCustomMenu** je tato metoda definována jako abstraktní. Potomci třídy musí tuto metodu předefinovat.

5.3.15.13. Metoda TCustomMenu.NewSubControl

Metoda **NewSubControl** slouží k vytvoření instance komponenty podmenu při výběru položky podmenu.

```
function NewSubControl( const AItemBounds: TRect;
  AMenu: PMenu; AParentMenu: PCustomMenu ): PCustomMenu;
```

Parametry:

AItemBounds	Umístění a rozměry položky podmenu.
AMenu	Odkaz na seznam položek menu podmenu.
AParentMenu	Odkaz na rodičovskou komponentu menu pro vytvářené podmenu (obvykle hodnot @Self).

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Přímo ve třídě **TCustomMenu** je tato metoda definována jako abstraktní. Potomci třídy musí tuto metodu předefinovat.

5.3.15.14. Metoda TCustomMenu.ClickItem

Metoda **ClickItem** provádí akci spojenou s danou položkou menu.

```
procedure ClickItem( AItem: PMenuItem ); virtual;
```

Parametry:

AItem	Odkaz na položku jejíž akce se má provést.

Návratové hodnoty:

Metoda **ClickItem** nevrací žádnou hodnotu.

Poznámky:

Akce spojená s položkou menu je určena položkou Flags a Param struktury **TMenuItem** (viz. kapitola 5.2.11)

5.3.15.15. Metoda TCustomMenu.SetFlags

Metoda **SetFlags** slouží k nastavení položky Flags (viz. kapitola 5.3.15.15)

```
procedure SetFlags( ASet, AReset: Word ); virtual;
```

Parametry:

ASet	Kombinace příznaků s prefixem mf_ (viz. kapitola 5.1.12), které se mají v položce Flags nastavit.
AReset	Kombinace příznaků s prefixem sbf_, které se mají v položce Flags vynulovat.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je to nutné, metoda provede překreslení komponenty pomocí metody **Repaint**.

Metoda nejprve vynuluje příznaky dané parametrem AReset a poté nastaví příznaky dané parametrem ASet. Metodu **SetFlags** lze tedy použít i pro přímé nastavení

položky Flags. V takovém případě musí být parametr AReset nastaven na hodnotu $FFFF a parametr ASet musí obsahovat požadovanou hodnotu položky Flags.

5.3.15.16. Metoda TCustomMenu.SetItemHeight

Metoda **SetItemHeight** slouží k nastavení výšky oblasti pro zobrazení textu položky, tj. nastavení položky ItemHeight (viz. kapitola 5.3.15.7).

```
procedure SetItemHeight( AHeight: Integer );
```

Parametry:

AHeight	Parametr udává výšku oblasti (tj. počet pixelů) pro zobrazení textu položky. Pokud je hodnota tohoto parametru menší nebo rovna nule, pak je výška oblasti dána výškou znaků použitého fontu zvetšená o absolutní hodnotu této položky.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je to nutné, metoda provede překreslení komponenty pomocí metody **Repaint**.

5.3.16. Třída TCustomMenuOverlay

Abstraktní třída **TCustomMenuOverlay** je bázovou třídou pro implementaci menu. Tato třída řeší veškerou obsluhu událostí vstupních zařízení, rolování menu apod. Třída **TCustomMenuOverlay** je abstraktní a nelze tedy vytvářet její instance.Instance lze vytvářet až z potomků této třídy, (tj. např. TMenuOverlay definované v knihovně GrCtrls), kteří správně implementují abstraktní metody **Paint**, **PaintItems**, **GetRowHeight**, **GetRowCount** a **GetItemAtPos**, zajišťují vykreslování komponenty a definují pozici položek menu v rámci obdélníku komponenty.

```
  PCustomMenuOverlay = ^TCustomMenuOverlay;
  TCustomMenuOverlay = object( TControl )
  public
    Pages        : PPageControl;
    Menu         : PMenu;
    Current      : PMenu;
    TopItem      : PMenuItem;
    Selected     : PMenuItem;
    ScrollBar    : PCustomScrollBar;

    constructor Init( const ABounds: TRect;
                  AScrollBar: PCustomScrollBar; AMenu: PMenu );

    destructor Done; virtual;

  {protected}
    procedure HandleEvent( var AEvent: TEvent ); virtual;
    procedure SetState( AState: Word; AEnable: Boolean ); virtual;
    procedure ClickItem( AItem: PMenuItem ); virtual;
```

```
    procedure PaintItems( AItem1, AItem2: PMenuItem ); virtual;
    function  GetRowHeight: Integer; virtual;
    function  GetRowCount: Integer; virtual;
    function  GetItemAtPos( APos: TPoint ): PMenuItem; virtual;
  end;
```

5.3.16.1. Události obsluhované komponentou

Komponenta **TCustomMenuOverlay** obsluhuje události myši (příp. dotykového panelu). Provedení akce svázané s položkou menu se provádí pomocí dvojkliku do obdélníku položky. Jednoduché kliknutí na položku menu provede pouze její označení.

Komponenta dále obsluhuje nasledující klávesy:

Klávesa	Akce
vkUp	Označení předchozí položky
vkDown	Označení následující položky
vkEsc	Návrat do předchozí úrovně menu
vkEnter	Výběr označené položky, tj. provedení akce svázané s položkou
vkLeft	Návrat do předchozí úrovně menu.
vkRight	Výběr označené položky, tj. provedení akce svázané s položkou
vkHome	Označení první položky menu
vkEnd	Označení poslední položky menu

Komponenta reaguje také na oběžník cmScrollBarChanged od přidruženého pusuvníku, jehož ukazatel je uložen v položce ScrollBar (viz. kapitola 5.3.16.8)

5.3.16.2. Oznámení generované komponentou

Komponenta může generovat oznámení **nmClik**, při výběru položky, která má nastaven příznak mifNotify. Položka oznámení ItemId v takovém případě obsahuje identifikátor položky zadaný při volání funkce **NewMenuItem** (viz. kapitola 5.4.10.2).

5.3.16.3. Položka TCustomMenuOverlay.Pages

Položka **Pages** obsahuje odkaz na komponentu **TPageControl** (viz. kapitola 0). Pokud je v položce Flags položky menu nastavena akce na mifGoto pak mifCall, mifReturn, pak je při výběru položky zavolána příslušná metoda této komponenty s parametrem daným položkou Param (viz. kapitola 5.3.13.7), tj. proveden skok na příslušnou stránku

Položku **Pages** je možné zapisovat i číst v libovolném okamžiku.

```
Pages : PPageControl;
```

5.3.16.4. Položka TCustomMenuOverlay.Menu

Položka **Menu** obsahuje odkaz na první úroveň zobrazovaného menu, tj. strukturu

TMenu vytvořenou pomocí metody NewMenu (viz. kapitola 5.4.10.1). Obsah položky je inicializován parametrem konstruktoru třídy a je určena pouze ke čtení.

```
Menu          : PMenu;
```

5.3.16.5. Položka TCustomMenuOverlay.Current

Položka **Current** obsahuje odkaz na aktuálně vybranou úroveň menu, tj. příslušnou strukturu TMenu. Obsah položky je automaticky aktualizován při přechodu do vyšší či nižší úrovně menu. Položka je určena pouze ke čtení.

```
Current       : PMenu;
```

5.3.16.6. Položka TCustomMenuOverlay.TopItem

Položka **TopItem** obsahuje odkaz na první viditelnou položku aktuálně vybraného menu. Položka je automaticky aktualizována při rolování obsahu menu a je určena pouze ke čtení.

```
TopItem       : PMenuItem;
```

5.3.16.7. Položka TCustomMenuOverlay.Selected

Položka **Selected** obsahuje odkaz na aktuálně vybranou položku. Položka je automaticky aktualizována v metodě **HandleEvent** a lze ji pouze číst.

```
Selected      : PMenuItem;
```

5.3.16.8. Položka TCustomMenuOverlay.ScrollBar

Položka **ScrollBar** obsahuje odkaz na přidruženou komponentu **TCustomScrollBar**, která zobrazuje aktuálně zobrazenou část menu z celého menu. Položka ScrollBar je inicializována parametrem konstruktoru instance a je určena pouze ke čtení.

```
ScrollBar     : PCustomScrollBar;
```

5.3.16.9. Konstruktor TCustomMenuOverlay.Init

Konstruktor **Init** provádí inicializaci instance třídy.

```
constructor Init( const ABounds: TRect;
                  AScrollBar: PCustomScrollBar; AMenu: PMenu );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka.
AScrollBar	Přidružená komponenta **TCustomScrollBar** pro zobrazování aktuálního viditelného výřezu menu. Tento parametr může být nastaven na hodnotu **nil**.
AMenu	Odkaz na strukturu stromového menu, vytvořeného pomocí metod v kapitole 5.4.10. Tento parametr nesmí mít hodnotu **nil**.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSelectable **or** ofSharedPalette **or** ofFirstClick
Font	fidDefault
Palette	#$00 (sdílená paleta)
Menu	parametr konstruktoru AMenu
Current	parametr konstruktoru AMenu
TopItem	AMenu^.Items
Selected	AMenu^.Items
ScrollBar	parametr konstrktoru ScrollBar

U přidruženého posuvník (parametr AScrollBar) je zrušen příznak ofSelectable v položce Options a příznak sbfFixedPage v položce Flags.

5.3.16.10. Metoda TCustomMenuOverlay.Done

Destruktor **Done** provádí uvolnění prostředků alokovaných konstruktorem instance.

```
destructor Done; virtual;
```

Parametry:

Destruktor nemá žádné parametry.

Návratové hodnoty:

Destruktor nevrací žádnou hodnotu.

Poznámky:

Destruktor uvolní z paměti instanci přidruženého posuvníku, na kterou se odkazuje parametr ScrollBar. Dále uvolní celou strukturu stromového menu pomocí funkce **FreeMenu** (viz. kapitola 5.4.10.5)

5.3.16.11. Metoda TCustomMenuOverlay.ClickItem

Metoda **ClickItem** provádí akci spojenou s danou položkou menu.

```
procedure ClickItem( AItem: PMenuItem ); virtual;
```

Parametry:

AItem	Odkaz na položku jejíž akce se má provést.

Návratové hodnoty:

Metoda **ClickItem** nevrací žádnou hodnotu.

Poznámky:

Akce spojená s položkou menu je určena položkou Flags a Param struktury **TMenuItem** (viz. kapitola 5.2.11)

5.3.16.12. Metoda TCustomMenuOverlay.PaintItems

Abstraktní metoda **PaintItems** slouží k překreslení obsahu dvou položek menu.

```
procedure PaintItems( AItem1, AItem2: PMenuItem ); virtual;
```

Parametry:

AItem1	Odkaz na první překreslovanou položku, může být **nil**.
AItem2	Odkaz na druhou překreslovanou položku, může být **nil**.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Metoda **PaintItems** je volána v případě změny označení (výběru) položek, v případě, že nedojde ke změně první viditelné položky menu (TopItem).

Metoda **PaintItems** je přímo ve třídě **TCustomMenuOverlay** definována jako abstraktní. Potomci této třídy jí musí předefinovat tak, aby prováděla překreslení zadaných položek menu.

5.3.16.13. Metoda TCustomMenuOverlay.GetRowHeight

Abstraktní metoda **GetRowHeight** vrací výšku jednoho řadku menu v pixelech.

```
function  GetRowHeight: Integer; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací výšku jednoho řádku menu. Předpokládá se, že všechny řádky menu jsou stejně vysoké.

Poznámky:

Metoda **GetRowHeight** je přímo ve třídě **TCustomMenuOverlay** definována jako abstraktní. Potomci této třídy jí musí předefinovat tak, aby vracela správnou výšku jednoho řádku menu.

5.3.16.14. Metoda TCustomMenuOverlay.GetRowCount

Abstraktní metoda **GetRowCount** vrací počet řádků, které lze zobrazit ve výřezu komponenty menu.

```
function  GetRowCount: Integer; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda vrací počet řádků.

Poznámky:

Metoda **GetRowCount** je přímo ve třídě **TCustomMenuOverlay** definována jako abstraktní. Potomci této třídy jí musí předefinovat tak, aby vracela správný počet řádků.

5.3.16.15. Metoda TCustomMenuOverlay.GetItemAtPos

Abstraktní metoda **GetItemAtPos** vrací ukazatel na položku menu, jejíž obdélník obsahuje zadaný bod.

```
function  GetItemAtPos( APos: TPoint ): PMenuItem; virtual;
```

Parametry:

APos	Poloha bodu v lokálních souřadnicích komponenty menu.

Návratové hodnoty:

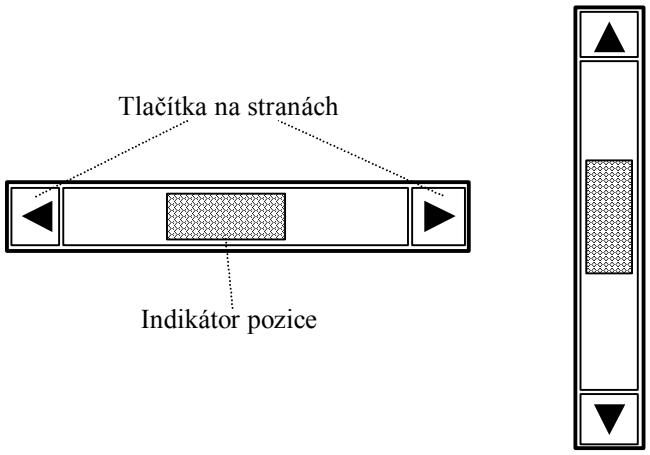
Metoda vrací ukazatel na položku menu, jejíž obdélník obsahuje zadaný bod, nebo hodnotu **nil**, pokud zadaný bod neobsahuje žádná položka menu.

Poznámky:

Metoda **GetItemAtPos** je přímo ve třídě **TCustomMenuOverlay** definována jako abstraktní. Potomci této třídy jí musí předefinovat.

5.3.17. Třída TCustomScrollBar

Abstraktní třída **TCustomScrollBar** je bázovou třídou pro implementaci posuvníku, viz. následující obrázek:



Posuvník lze použít k nejrůznějším účelům. Obvykle je však svázán s další komponentou jako je **TListBox**, **TListView**, **TMenuOverlay**, které zajišťují automatické nastavení jeho stavu při změně výběru položky.

Třída **TCustomScrollBar** je abstraktní a nelze tedy vytvářet její instance.Instance lze vytvářet až z potomků této třídy, (tj. např. TScrollBar definované v jednotce GrCtrls), kteří správně implementují abstraktní metody **Repaint**, **PaintIndicator** a **GetIndicatorCoords**.

Ke komponentě **TCustomScrollBar** lze připojit validátor pro ordinální typy, tj. potomky třídy **TOrdinalValidator**.

```
PCustomScrollBar = ^TCustomScrollBar;
TCustomScrollBar = object( TControl )
public
  Flags     : Word;
  Min       : Integer;
  Max       : Integer;
  Step      : Integer;
  PageStep  : Integer;
  Position  : Integer;

  constructor Init( const ABounds: TRect );
  procedure SetRange( AMin, AMax: Integer );
  procedure SetPosition( APosition: Integer );
  procedure SetParams( AMin, AMax, AStep, APageStep,
              APosition: Integer );
  procedure SetFlags( ASet, AReset: Word ); virtual;

{protected}
  procedure HandleEvent( var AEvent: TEvent ); virtual;
  function  Transfer( AMode: Integer ): Integer; virtual;
  procedure DoChange; virtual;
  procedure PaintIndicator; virtual;
  procedure GetIndicatorCoords( var ARect: TRect;
              var AIndPos, AIndSize: Integer ); virtual;
end;
```

5.3.17.1. Události obsluhované komponentou TCustomScrollBar

Komponenta **TCustomScrollBar** reaguje na události klávesnice a myši (resp. dotykového panelu).

Komponenta obsluhuje událost evKeyDown a reaguje pouze na klávesy uvedené v následující tabulce:

Klávesa	Akce
vkLeft	Position = Position – Step (pouze horizontalní posuvník)
vkRight	Position = Position + Step (pouze horizontální posuvník)
vkUp	Position = Position – Step (pouze vertikální posuvník)
vkDown	Position = Position + Step (pouze vertikální posuvník)
vkPageUp	Position = Position – PageStep
vkPageDown	Position = Position + PageStep
vkHome	Position = Min
vkEnd	Position = Max

Komponenta obsluhuje buď klávesy vkLeft a vkRight nebo vkUp a vkDown, podle toho zda se jedná o horizontální nebo vertikální posuvník.

Dále komponenta reaguje na události evMouseDown, evMouseDbl a evMouseRep.

5.3.17.2. Oznámení generované komponentou TCustomScrollBar

Komponenta **TCustomScrollBar** generuje oznámení **nmChange**, které je odesláno při každé změně hodnoty posuvníku. Toto oznámení je generováno pouze tehdy, jestliže v položce Flags komponenty je nastaven příznak **sbfNotify**.

Dále komponenta generuje oběžník **cmScrollBarChanged**, který je poslán vlastníkovi komponenty pomocí funkce Broadcast. Parametr oběžníku Param je nastaven na ukazatel na instanci komponenty **TCustomScrollBar**. Tento oběžník je generován pouze tehdy, jestliže v položce Flags komponenty je nastaven příznak **sbfBroadcast**.

5.3.17.3. Validace hodnoty komponenty TCustomScrollBar

Ke komponentě **TCustomScrollbar** může být použit pouze validátor ordinálního typu, tj. validátor vycházející ze třídy **TOrdinalValidator**, tedy např. **TByteValidator**, **TIntegerValidator** apod.

5.3.17.4. Položka TCustomScrollBar.Flags

Položka **Flags** upřesňuje chování posuvníku. Může obsahovat kombinaci konstant s prefixem sbf_, které jsou uvedeny v následující tabulce.

Identifikátor	Kód	Popis
sbfDragging	$0001	Zatím neimplementováno.
sbfFixedPage	$0002	Fixní velikost indikátoru posuvníku. Pokud tento příznak není nastaven, pak je velikost indikátor posuvníku nastaven

		proporcinálně podle velikosti položky PageStep.
sbfBroadcast	$0004	Pokud je nastaven tento příznak, generuje pusuvník při každé změně aktuální hodnoty oběžník cmScrollbarChanged (viz. kapitola 5.3.17.2)
sbfNotify	$0008	Pokud je nastaven tento příznak, generuje posuvník při každé změně aktuální hodnoty oznámení nmChange (viz. kapitola 5.3.17.2)
sbfButtons	$0010	Pokud je nastaven tento příznak, pak je posuvník vykreslen se dvěma tlačítky na levé a pravé (resp. horní a dolní) straně. Tyto tlačítka lze použít ke změně aktuální hodnotu posuvníku o hodnotu danou položkou Step.

5.3.17.5. Položka TCustomScrollBar.Min

Položka **Min** obsahuje dolní mez rozsahu posuvníku. Položka Min je určena pouze pro čtení. Lze ji však nastavit pomocí metod **SetRange** (viz. kapitola 5.3.17.11) příp. **SetParams** (viz. kapitola 5.3.17.13).

5.3.17.6. Položka TCustomScrollBar.Max

Položka **Max** obsahuje horní mez rozsahu posuvníku. Položka Max je určena pouze pro čtení. Lze ji však nastavit pomocí metod **SetRange** (viz. kapitola 5.3.17.11) příp. **SetParams** (viz. kapitola 5.3.17.13).

5.3.17.7. Položka TCustomScrollBar.Step

Položka **Step** obsahuje krok změny hodnoty posuvníku v případě stisku kláves vkLeft a vkRight (resp. vkUp a vkDown) nebo při stisku tlačítek umístěných v rozích posuvníku. Položka je určena pouze pro čtení a lze ji nastavit pomocí metody **SetParams** (viz. kapitola 5.3.17.13).

5.3.17.8. Položka TCustomScrollBar.PageStep

Položka **PageStep** obsahuje krok změny hodnoty posuvníku v případě stisku kláves vkPageUp a vkPageDn nebo při stisku oblasti mezi ukazatelem posuvníku a tlačítky umístěných v rozích. Položka je určena pouze pro čtení a lze ji nastavit pomocí metody **SetParams** (viz. kapitola 5.3.17.13).

5.3.17.9. Položka TCustomScrollBar.Position

Položka **Position** obsahuje aktuální hodnotu nastavenou posuvníkem. Tato hodnota se pohybuje v rozsahu daném položkami **Min** a **Max**. Položka Position je určena pouze pro čtení. Nastavit ji lze explicitně pomocí metody **SetPosition** (viz. kapitola 5.3.17.12)

Pokud je v položce Flags nastaven příznak sbfFixedPage, pak se hodnota Position mění v rozsahu položek Min a Max. Pokud tento příznak nastaven není, pak je horní mez posuvníku stanovena rozdílem položky Max a PageStep.

5.3.17.10. Konstruktor TCustomScrollBar.Init

Konstruktor **Init** provádí inicializaci instance třídy.

```
constructor Init( const ABounds: TRect );
```

Parametry:

ABounds	Umístění a rozměry komponenty v rámci jejího vlastníka.

Návratové hodnoty:

Konstruktor nevrací žádnou hodnotu.

Poznámky:

Konstruktor **Init** nastaví všechny položky instance na implicitní hodnoty, viz. následující tabulka. Ostatní, zde neuvedené položky, jsou inicializovány na hodnotu 0.

Položka	Hodnota
Bounds	parametr konstruktoru ABounds
EventMask	$FFFF
CaretSize	(1, 8)
State	sfVisible
Options	ofSharedPalette **or** ofSelectable **or** ofFirstClick
Font	fidDefault
Palette	#$00 (sdílená paleta)
Min	0
Max	100
PageStep	10
Position	0
Step	1
Flags	sbfBroadcast

5.3.17.11. Metoda TCustomScrollBar.SetRange

Metoda **SetRange** slouží k nastavení dolní a horní meze rozsahu posuvníku.

```
procedure SetRange( AMin, AMax: Integer );
```

Parametry:

AMin	Dolní mez rozsahu.
AMax	Horní mez rozsahu.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Dolní mez rozsahu musí být menší než horní mez. Pokud se voláním metody **SetRange**, změní původní dolní příp. horní mez, je posuvník automaticky překreslen s novým nastavením.

5.3.17.12. Metoda TCustomScrollBar.SetPosition

Metoda **SetPosition** slouží k nastavení aktuální hodnoty posuvníku.

```
procedure SetPosition( APosition: Integer );
```

Parametry:

APosition	Požadovaná hodnota posuvníku.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Parametr APosition musí být v mezích definovaných položkami Min a Max. Pokud je mimo tento rozsah je aktuální hodnota nastavena na krajní hodnotu rozsahu. Pokud se parametr APosition liší od aktuální hodnoty posuvníku, pak je komponenta automaticky překreslena.

5.3.17.13. Metoda TCustomScrollBar.SetParams

Metoda **SetParams** slouží k nastavení všech parametrů posuvníku.

```
procedure SetParams( AMin, AMax, AStep, APageStep,
           APosition: Integer );
```

Parametry:

AMin	Dolní mez rozsahu.
AMax	Horní mez rozsahu.
AStep	Krok při stisku tlačítek v rozích posuvníku, nebo při stisku kláves vkLeft, vkRight resp. vkUp a vkDown.
APageStep	Krok při stisku oblasti mezi indikátorem a tlačítky posuvníku, nebo při stisku kláves vkPageUp a vkPageDown.
APosition	Aktuální pozice posuvníku.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Parametr AMin musí být menší než parametr AMax. Parametr AStep a APageStep

musí být menší než rozdíl parametrů AMax a AMin. Parametr APosition musí být v rozsahu definovaném parametry AMin a AMax.

Po provedení metody je posuvník automaticky překreslen.

5.3.17.14. Metoda TCustomScrollBar.SetFlags

Metoda **SetFlags** slouží k nastavení položky Flags (viz. kapitola 5.3.17.4)

```
procedure SetFlags( ASet, AReset: Word ); virtual;
```

Parametry:

ASet	Kombinace příznaků s prefixem sbf_ (viz. kapitola 5.1.14), které se mají v položce Flags nastavit.
AReset	Kombinace příznaků s prefixem sbf_, které se mají v položce Flags vynulovat.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Pokud je to nutné, metoda provede překreslení komponenty pomocí metody **Repaint**.

Metoda nejprve vynuluje příznaky dané parametrem AReset a poté nastaví příznaky dané parametrem ASet. Metodu **SetFlags** lze tedy použít i pro přímé nastavení položky Flags. V takovém případě musí být parametr AReset nastaven na hodnotu $FFFF a parametr ASet musí obsahovat požadovanou hodnotu položky Flags.

5.3.17.15. Metoda TCustomScrollBar.PaintIndicator

Metoda PaintIndicator slouží k překreslení indikátoru posuvníku v okamžiku, kdy se změnila aktuální hodnota posuvníku. Metoda je abstraktní, potomci třídy **TCustomScrollBar** ji musí předefinovat.

```
procedure PaintIndicator; virtual;
```

Parametry:

Metoda nemá žádné parametry.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Potomci třídy **TCustomScrollBar** musí předefinovat metodu **Paint** a **PaintIndiciator**. Metoda **Paint** provádí vykreslení celého posuvníku např. včetně

tlačítek na stranách apod. Metoda **PaintIndicator** provádí pouze překreslení minimální části posuvníku, která zobrazuje jeho aktuální stav, tj. indikátor.

5.3.17.16. Metoda TCustomScrollBar.GetIndicatorCoords

Metoda **GetIndicatorCoords** vrací rozměry oblasti, ve které se pohybuje indikátor posuvníku, jeho pozici a rozměry. Metoda je abstraktní, potomci třídy **TCustomScrollBar** ji musí předefinovat.

```
procedure GetIndicatorCoords( var ARect: TRect;
            var AIndPos, AIndSize: Integer ); virtual;
```

Parametry:

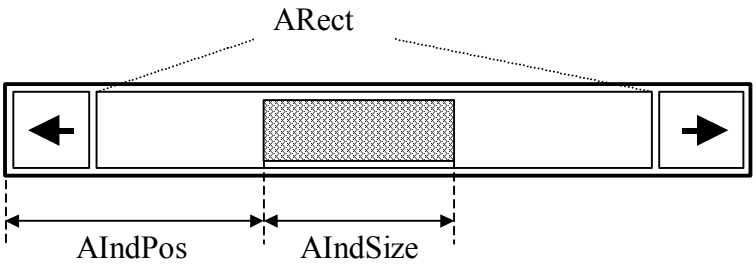
ARect	Tento parametr je po zavolání metody naplněn umístěním a rozměry obdélníku, ve kterém se pohybuje indikátor posuvníku.
AIndPos	Pozice indikátoru od levého (resp. horního) okraje posuvníku.
AIndSize	Šířka (resp. výška) indikátoru posuvníku.

Návratové hodnoty:

Metoda nevrací žádnou hodnotu.

Poznámky:

Všechny souřadnice jsou relativní vůči komponentě posuvníku.



5.4. Funkce

5.4.1. RegisterCustomizeProc

Procedura **RegisterCustomizeProc** vloží zadanou proceduru do seznamu konfiguračních procedur komponent.

```
procedure RegisterCustomizeProc( AProc: TCustomizeProc;
            ALevel: Integer );
```

Parametry:

AProc	Odkaz na konfigurační proceduru **TCustomizeProc** (viz. kapitola 5.2.8).
ALevel	Místo v seznamu, kam se má procedura vložit.

Poznámky:

Odkazy na konfigurační procedury jsou uloženy ve spojovém seznamu seřazeném podle parametru ALevel zadaného při registraci. Při konfiguraci komponent pomocí metody **Customize** (viz. kapitola 5.3.4.25) jsou postupně vyvolávány zaregistrované konfigurační procedury v pořadí daném spojovým seznamem (tedy od nejnižší hodnoty k nejvyšší hodnotě ALevel).

5.4.2. RemoveCustomizeProc

Procedura **RemoveCustomizeProc** odstraní ze seznamu konfiguračních procedur komponent zadanou proceduru.

```
procedure RemoveCustomizeProc( AProc: TCustomizeProc );
```

Parametry:

AProc	Odkaz na konfigurační proceduru **TCustomizeProc** (viz. kapitola 5.2.8).

Poznámky:

5.4.3. Procedura ClearEvent

Procedura **ClearEvent** označí strukturu události za obslouženou.

```
procedure ClearEvent( var AEvent: TEvent );
```

Parametry:

AEvent	Struktura obsloužené události.

Poznámky:

Metoda **ClearEvent** označí událost za obslouženou. Takto označená událost se nebude dále posílat dalším komponentám, které by ji případně mohly obsloužit.

Metoda **ClearEvent** je určena k volání pouze v rámci procedury **AfterHandle** příp. **BeforeHandle** (viz. kapitoly 5.3.4.17 a 5.3.4.18).

5.4.4. Procedura ClearNotification

Procedura **ClearNotification** označí strukturu oznámení za obslouženou.

```
procedure ClearNotification( var ANotification: TEvent );
```

Parametry:

AEvent	Struktura obsloužené události.

Poznámky:

Metoda **ClearNotification** označí oznámení za obsloužené. Takto označená oznámení se nebude dále posílat dalším komponentám směrem ke kořeni stromu komponent.

Metoda **ClearNotification** je určena k volání pouze v rámci procedury **AfterNotify**, **BeforeNotify** a **HandleNotification** (viz. kapitoly 5.3.4.19, 5.3.4.20 a 5.3.4.27).

5.4.5. Funkce Broadcast

Funkce **Broadcast** vytvoří událost evBroadcast a předá ji ke zpracovaní zadané komponentě.

```
function Broadcast( AReceiver: PControl; ACommand: Word;
  AParam: Pointer ): Pointer;
```

Parametry:

AReceiver	Cílová komponenta pro zpracování události.
ACommand	Příkaz, resp. identifikátor události evBroadcast, který bude uložen do položky Command struktury události.
AParam	Pomocný parametr, poplatný příkazu ACommand, který bude uložen do položky Param struktury události.

Návratové hodnoty:

Pokud byla událost zpracována, tj. při obsluze události byla zavolána metoda **ClearEvent** (viz. kapitola 5.3.4.37), pak funkce vrací hodnotu položky Param struktury události. Pokud událost nebyla obsloužena, vrací hodnotu **nil**.

Poznámky:

Metoda **Broadcast** vytvoří událost a předá ji přímo cílové komponentě, tj. zavolá její metodu **ProcessEvent** (viz. kapitola 5.3.4.34). Událost tedy neprochází uživatelskou frontou událostí.

Pokud je cílová komponenta pro zpracování zprávy potomkem komponenty **TGroup**, pak je událost rozeslána všem komponentám vloženým do této komponenty (rozdíl oproti události zpráva, viz funkce **Message** v kapitole 5.4.6).

5.4.6. Funkce Message

Funkce **Message** vytvoří událost evMessage a předá ji ke zpracování zadané komponentě.

```
function Message( AReceiver: PControl; ACommand: Word;
  AParam: Pointer ): Pointer;
```

Parametry:

AReceiver	Cílová komponenta pro zpracování události.
ACommand	Příkaz, resp. identifikátor události evBroadcast, který bude uložen do položky Command struktury události.
AParam	Pomocný parametr, poplatný příkazu ACommand, který bude uložen do položky Param struktury události.

Návratové hodnoty:

Pokud byla událost zpracována, tj. při obsluze události byla zavolána metoda **ClearEvent** (viz. kapitola 5.3.4.37), pak funkce vrací hodnotu položky Param struktury události. Pokud událost nebyla obsloužena, vrací hodnotu **nil**.

Poznámky:

Metoda **Message** vytvoří událost a předá ji přímo cílové komponentě, tj. zavolá její metodu **ProcessEvent** (viz. kapitola 5.3.4.34). Událost tedy neprochází uživatelskou frontou událostí.

Událost typu zpráva je předána pouze cílové komponentě (rozdíl oproti události oběžník, viz. funkce **Broadcast** v kapitole 5.4.5).

5.4.7. Funkce CompareStringPtr

Funkce **CompareStringPtr** zjišťuje, zda jsou dva řetězce identické.

```
function CompareStringPtr( A, B: PString ): Boolean;
```

Parametry:

A	Ukazatel na první řetězec. Může mít hodnotu **nil**.
B	Ukazatel na druhý řetězec. Může mít hodnotu **nil**.

Návratové hodnoty:

Funkce vrací hodnotu True, pokud jsou dva řetězce identické. V opačné případě vrací hodnotu False.

Poznámky:

Dva řetězce jsou identické jestliže oba ukazatele se odkazují na stejné místo v pamětí a nebo pokud jsou identické obsahy obou řetězců.

5.4.8. Funkce GetBitmapCanvas

Funkce **GetBitmapCanvas** slouží k vytvoření instance třídy **TBitmapCanvas** (viz. kapitola 5.3.3) poskytující množství metod pro přímé kreslení do bitmapy.

```
function GetBitmapCanvas( ABitmap: PBitmap ): PCanvas;
```

Parametry:

ABitmap Ukazatel na strukturu nekomprimované bitmapy.

Návratové hodnoty:

Funkce vrací ukazatel na alokovanou a inicializovanou instanci třídy **TBitmapCanvas**. V případě nedostatku paměti, nebo jiné chyby vrací hodnotu **nil**.

Poznámky:

Bitmapa daná parametrem ABitmap může mít libovolný formát, ale nesmí být komprimovaná, jinak funkce vrátí hodnotu **nil**.

5.4.9. Procedura ReleaseBitmapCanvas

Procedura **ReleaseBitmapCanvas** slouží k uvolnění instance třídy **TBitmapCanvas** vytvořené pomocí funkce **GetBitmapCanvas**..

```
procedure ReleaseBitmapCanvas( ACanvas: PCanvas );
```

Parametry:

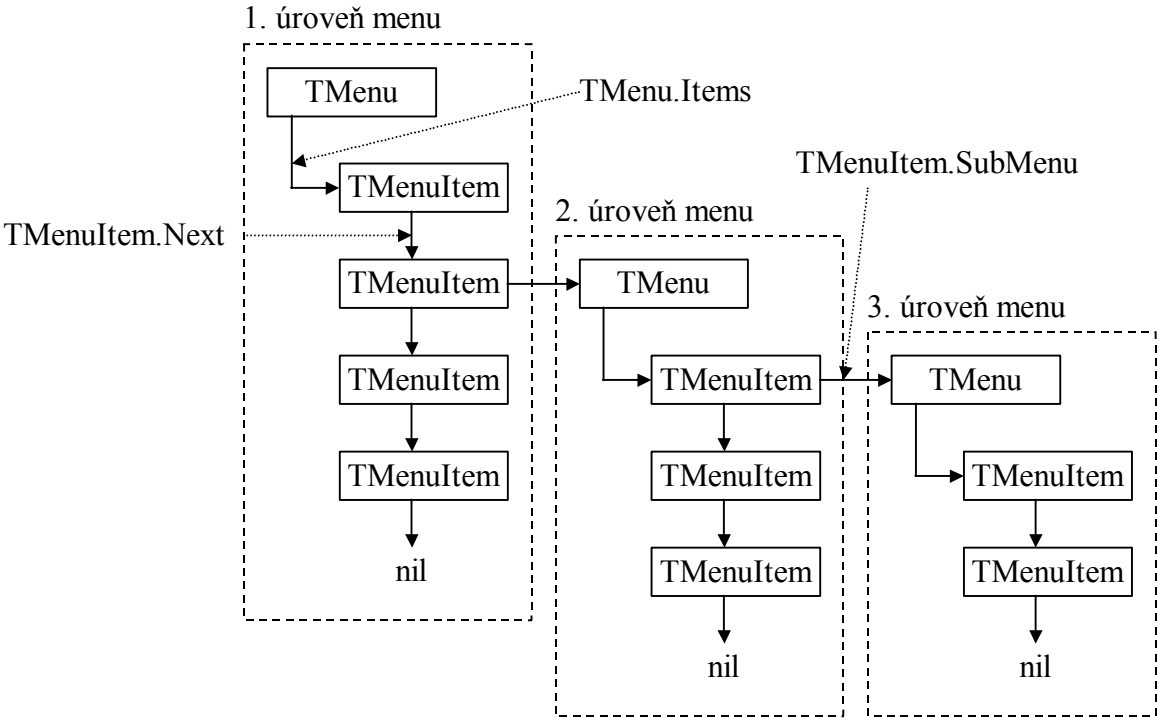
ACanvas Ukazatel na instanci třídy **TCanvas** alokované funkcí **GetBitmapCanvas**.

Poznámky:

5.4.10. Funkce pro tvorbu struktury menu

Pro komponenty **TCustomMenu** (viz. kapitola 5.3.15), **TCustomMenuOverlay** (viz. kapitola 5.3.16) a jejich potomky, tj. **TMenuBar**, **TMenuPopup**, **TMenuOverlay** definovaných v knihovně GrCtrls byly vytvořeny funkce pro vytvoření hierarchického menu. Tyto funkce jsou popsány v následujících kapitolách.

Menu je strom spojových seznamů struktur **TMenu** a **TMenuItem** (viz. kapitoly 5.2.12a 5.2.11).



Struktura **TMenu** obsahuje ukazatel na spojový seznam položek **TMenuItem** a další pomocné položky. Struktura **TMenuItem** obsahuje text položky menu a akci, která se má provést, tj. volání stránky, generování oznámení, ukončení modálního stavu nebo skok do další úrovně menu (podmenu). Celá struktura menu je uložena na hromadě.

Výše uvedenou strukturu lze vytvořit pomocí níže uvedených funkcí, viz. následující příklad:

```
var
  Menu : PMenu;

Menu :=
  NewMenu(
    NewMenuItem( 'Polozka 1-1', '', 0, 0, 0,
    NewSubMenu( 'Polozka 1-2',
      NewMenu(
        NewSubMenu( 'Polozka 2-1',
          NewMenu(
            NewMenuItem( 'Polozka 3-1', '', 0, 0, 0,
            NewMenuItem( 'Polozka 3-2', '', 0, 0, 0,
            nil))
          ),
        NewMenuItem( 'Polozka 2-2', '', 0, 0, 0,
        NewMenuItem( 'Polozka 2-3', '', 0, 0, 0,
        nil)))
      ),
    NewMenuLine(
    NewMenuItem( 'Polozka 1-4', '', 0, 0, 0,
    nil))))
  );
```

5.4.10.1. Funkce NewMenu

Funkce **NewMenu** alokuje na hromadě strukturu **TMenu** (viz. kapitola 5.2.12) a provede její inicializaci.

```
function NewMenu( AItems: PMenuItem ): PMenu;
```

Parametry:

AItem	Odkaz na první položku spojového seznamu položek menu.

Návratové hodnoty:

Funkce vrací hodnotu ukazatel na alokovanou strukturu TMenu. V případě nedostatku paměti vrací hodnotu **nil**.

Poznámky:

Funkce **NewMenu** inicializuje strukturu **TMenu** tak, že do položek Items, Default a TopItem uloží parametr AItems. Položku Owner nastaví na hodnotu **nil**.

5.4.10.2. Funkce NewMenuItem

Funkce **NewMenuItem** alokuje na hromadě strukturu **TMenuItem** (viz. kapitola 5.2.11) a inicializuje ji jako běžnou položku menu.

```
function NewMenuItem( const AName, AText: string; AId: Word; )
  AFlags: Word; AParam: Word; ANext: PMenuItem): PMenuItem;
```

Parametry:

AName	Název položky (tj. text zarovnaný doleva). Nesmí být prázdný řetězec.
AText	Text položky (tj. text zarovnaný doprava). Pokud je tento parametr prázdný řetězec, pak není zobrazován.
AId	Identifikátor položky, může být hodnota v rozsahu 0 až 65535.
AFlags	Příznaky specifikující chování položky. Tj. kombinace konstant s prefixem mif_, viz. kapitola 5.1.13.
AParam	Parametr upřesňující chování položky, viz. poznámky dále.
ANext	Odkaz na následující položku menu.

Návratové hodnoty:

Funkce vrací hodnotu ukazatel na alokovanou strukturu **TMenuItem**. V případě nedostatku paměti vrací hodnotu **nil**.

Poznámky:

Funkce **NewMenuItem** inicializuje strukturu **TMenu** tak, že do položek Items, Default a TopItem uloží parametr AItems. Položku Owner nastaví na hodnotu **nil**.

Při výběru položky menu je možné provést následující akce:

Akce	Parametr	Doplňující parametry

	AFlags	
Generování oznámení	mifNotify	Parametr AId musí obsahovat číslo položky, jenž bude uvedeno v položce ItemId vygenerovaného oznámení
Skok na stránku (Goto)	mifGotoPage	Parametr AParam musí obsahovat identifikátor stránky a zároveň položka Pages komponenty menu se musí odkazovat na instanci komponenty TPageControl obsahující tuto stránku.
Volání stránky (Call)	mifCallPage	Parametr AParam musí obsahovat identifikátor stránky a zároveň položka Pages komponenty menu se musí odkazovat na instanci komponenty TPageControl obsahující tuto stránku.
Návrat do předchozí úrovně menu, příp. do předchozí stránky	mifReturn	Položka Pages komponenty menu se musí odkazovat na instanci komponenty TPageControl.
Ukončení modálního stavu	mifEndModal	Parametr AParam musí obsahovat důvod ukončení modálního stavu, tedy jednu z konstant mr_ (viz. kapitola 5.1.7)

5.4.10.3. Funkce NewMenuLine

Funkce **NewMenuLine** alokuje na hromadě strukturu **TMenuItem** (viz. kapitola 5.2.11) a inicializuje ji jako oddělovací položku menu, tj. obvykle vodorovnou úsečku, která nahrazuje položku menu, a kterou nelze vybrat.

```
function NewMenuLine( ANext: PMenuItem ): PMenuItem;
```

Parametry:

ANext Odkaz na následující položku menu.

Návratové hodnoty:

Funkce vrací hodnotu ukazatel na alokovanou strukturu **TMenuItem**. V případě nedostatku paměti vrací hodnotu **nil**.

Poznámky:

Funkce inicializuje položku menu tak, že do položky AName struktury **TMenuItem** uloží hodnotu **nil** a zároveň položka Flags je nastavena na hodnotu 0.

5.4.10.4. Funkce NewSubMenu

Funkce **NewSubMenu** alokuje na hromadě strukturu **TMenuItem** (viz. kapitola 5.2.11) a inicializuje ji položku vytvářející vstup do další úrovně menu.

```
function NewSubMenu( const AName: string; ASubMenu: PMenu;
  ANext: PMenuItem ): PMenuItem;
```

Parametry:

AName	Název položky (tj. text zarovnaný doleva). Nesmí být prázdný řetězec.
ASubMenu	Odkaz na strukturu další úrovně menu vytvořená pomocí funkce NewMenu (viz. kapitola 5.4.10.1).
ANext	Odkaz na následující položku menu.

Návratové hodnoty:

Funkce vrací hodnotu ukazatel na alokovanou strukturu **TMenuItem**. V případě nedostatku paměti vrací hodnotu **nil**.

Poznámky:

5.4.10.5. Procedura FreeMenu

Procedura **FreeMenu** uvolní z paměti celou strukturu menu vytvořenou pomocí funkce **NewMenu**.

```
procedure FreeMenu( AMenu: PMenu );
```

Parametry:

AMenu	Odkaz na strukturu menu alokovanou na hromadě pomocí funkce **NewMenu**.

5.5. Globální proměnné

5.5.1. Proměnná Application

Globální proměnná **Application** obsahuje odkaz na komponentu **TApplication**, tedy kořen stromu komponent aplikace. Tato proměnná je automaticky inicializována při volání konstruktoru třídy **TApplication** a je určena pouze ke čtení.

```
Application : PApplication;
```