

Ostravská univerzita



Vývoj objektových aplikací 1

Ostrava, 2002

Mgr. Rostislav Fojtík

Textový podklad k on-line kurzu

Obsah:

Úvodní lekce	6
Cíl lekce.....	6
Samostatné práce.....	7
Podmínky udělení zápočtu	7
Obsah kurzu.....	7
Literatura	8
Shrnutí lekce.....	8
Základy objektově orientovaného programování.....	9
Cíl lekce.....	9
Vstupní test.....	9
Úvod	10
Základní vlastnosti	11
Zapouzdření.....	11
Dědičnost.....	11
Polymorfismus	13
Kompozice	13
Opakovací test.....	14
Shrnutí učiva	15
Rejstřík	15
Vývoj aplikací pro MS Windows.....	16
Cíl lekce.....	16
Vstupní test.....	16
Programy ve Windows	17
Vizuální vývojové nástroje.....	17
Borland Delphi	18
Grafické komponenty	21
Formulář	21
Tlačítko – Button.....	21
Label.....	22
Projekt – program.....	23
Opakovací test	24
Shrnutí	25
Shrnutí	25
Rejstřík	25
Tvorba projektů č.1	26
Cíl lekce.....	26
Vstupní test.....	26
Projekt č.1.....	27
Projekt č.2.....	30
Projekt č.3.....	32
Časté chyby	34
Opakovací test	34
Shrnutí	35
Rejstřík	35
Tvorba projektů č.2	36
Cíl lekce.....	36
Vstupní test.....	36
Edit	37
MainMenu	37

Projekt č.4	38
Projekt č.5	40
Projekt č.6	43
Časté chyby	45
Opakovací test.....	45
Shrnutí.....	46
Rejstřík	46
Tvorba projektů č.3	47
Cíl lekce	47
Vstupní test	47
Memo	48
OpendDialog	49
SaveDialog.....	49
Projekt č.7 – Jednoduchý textový editor	51
Časté chyby	58
Časté chyby	58
Opakovací test.....	58
Shrnutí.....	59
Rejstřík	59
Tvorba projektů č.4	60
Cíl lekce	60
Vstupní test	60
Projekt č.8 – jednoduchá kalkulačka.....	61
Časté chyby	70
Časté chyby	70
Opakovací test.....	70
Shrnutí.....	71
Rejstřík	71
Tvorba projektů č.5	72
Cíl lekce	72
Vstupní test	72
Projekt č.9 – kalkulačka	73
Komponenta GroupBox	75
Činnost tlačítek	75
Vlastní unita	78
Časté chyby	86
Časté chyby	86
Opakovací test.....	86
Shrnutí.....	87
Rejstřík	87
Samostatné práce.....	88
Cíl lekce	88
Úkol č.1	88
Úkol č.2.....	89
Nové grafické komponenty	90
Úkol č.3	90
Shrnutí.....	91

Orientační symboly v textu:



Cíle, ke kterým chceme dospět.



Úkoly, projekty, testy a písemné zprávy.



Otazník - průběžné otázky a úkoly.



Vykřičník - důležité pojmy a postupy.



Suma - shrnutí učební látky.

Zpracoval:

Mgr. Rostislav Fojtík

Katedra informatiky a počítačů

Přírodovědecká fakulta

Ostravská univerzita

rostislav.fojtik@osu.cz

Upozornění:

Tento soubor slouží jako textový podklad k on-line výukovému kurzu "Vývoj objektových aplikací I" pro kombinovanou formu studia rozšiřujícího studia informatiky pro učitele ZŠ a SŠ.

On-line verzi kurzu lze nalézt na adrese:

<http://tutorial.osu.cz/tutor2000/student/login.asp>, přihlašovací jméno *voa* a heslo *voa*.

Úvodní lekce



Cíl lekce

Cílem této lekce je vás seznámit s organizací výukového kurzu a požadavky na udělení zápočtu z předmětu *Vývoj objektových aplikací 1*.

Po absolvování lekce budete:

- vědět, jaké lekce kurz obsahuje a kdy by jste je měli absolvovat
- vědět, kdy odevzdat samostatné práce
- vědět, jaké jsou požadavky na udělení zápočtu z předmětu

Časová náročnost lekce: 30 minut

Pro zdárné absolvování kurzu *Vývoj objektových aplikací 1* jsou vhodnými základními předpoklady znalosti z předmětů *Algoritmy a datové struktury 1 a 2*.



Cílem kurzu *Vývoj objektových aplikací 1* je rozšířit si znalosti programování a algoritmizace. A to v oblasti objektově orientovaného programování a v oblasti využívání vizuálního vývojového nástroje Borland Delphi. Po absolvování kurzu budete umět vytvářet programy a jednoduché aplikace v jazyku *Object Pascal* a budete umět tvořit grafické rozhraní aplikace pomocí vizuálních vývojových nástrojů. Získané znalosti využijete v návazném předmět *Vývoj objektových aplikací 2*, který je zakončen zkouškou.

Název kurzu: ***Vývoj objektových aplikací 1***

Doporučený ročník: **druhý**

Semestr: **zimní**

Zakončení: **zápočet**

Tutor: **Mgr. Rostislav Fojtík**

Katedra: **KIP** (Katedra informatiky a počítačů)

Fakulta: **Přírodovědecká**

Komunikace mezi všemi účastníky kurzu je velmi důležitá. Pro její zajištění slouží následující způsoby:

- **e-mail** - elektronická pošta. Adresa vyučujícího (lektora kurzu) je rostislav.fojtik@osu.cz. Je důležité, aby studenti během semestru neměnili své e-mailové adresy!
- **elektronická konference**
 - **telefonicky** - telefonní spojení na vyučujícího Mgr. Rostislav Fojtík - 59 6160 229
 - **konzultace prezenční formou** - konzultace probíhají po předcházející domluvě. Kancelář č.24, v budově na adrese 30.dubna 22, Ostrava.

Další možnosti komunikace:

- **ICQ** - číslo vyučujícího je 66965477
- **internetová video konference** - je potřeba domluvit s vyučujícím

Samostatné práce

Pro udělení zápočtu je nutné zpracovat tři úkoly obsažené v lekci s názvem *Samostatné práce*. Vytvořené programy včetně všech zdrojových souborů musíte předložit při udělování zápočtu.

Podmínky udělení zápočtu

Aby jste získali zápočet, musíte splnit následující podmínky:

- Prohlédnout si všechny výukové lekce kurzu a vyzkoušet si naprogramovat v nich obsažené programy.
- Naprogramovat tři programy, jejichž zadání je v lekci *Samostatné práce*.
- Osobně se dostavit na katedru a před tutorem obájit a vysvětlit své programy. Termíny udělování zápočtů (obhajoby programů) budou upřesněny na konci semestru.

Obsah kurzu

Výukový kurz obsahuje níže uvedené výkladové lekce. U každé lekce je uvedena přibližná časová náročnost a datum, ke kterému by jste měli lekci absolvovat - zvládnout její učivo. Nenechávejte si studium na poslední chvíle, nestihnete absolvovat kurz!

Úvodní lekce - 30 min

Základy objektového programování – 2 hodiny, 15.říjen

Vývoj aplikací pro MS Windows – 2 hodiny, 25.říjen

Tvorba projektů č.1 – 2 hodiny, 31.říjen

Tvorba projektů č.2 – 2 hodiny, 10.listopad

Tvorba projektů č.3 – jednoduchý textový editor – 3 hodiny, 15.listopad

Tvorba projektů č.4 – jednoduchá kalkulačka – 3 hodiny, 30.listopad

Tvorba projektů č.5 – textový editor – 3 hodiny, 10.prosinec

Samostatné práce – 4 hodiny, 15.prosinec

Výukový kurz *Vývoj objektových aplikací 1* je zaměřen na vytváření objektových programů v operačním systému s grafickým rozhraním. Jako vývojový nástroj bude využíván vizuální nástroj Borland Delphi. Pro vlastní práci si můžete nainstalovat libovolnou verzi tohoto produktu. Je možné pracovat v demo či trial verzích, případně využít verze Borland Delphi 6 Personal. V této verzi jsou odladěny všechny příklady. Verze je k dispozici jako freeware. Je však nutné se zaregistrovat na stránkách www.borland.com, kde lze získat i instalační soubor. Pozor! Soubor má přes 180 MB.

Pro práci s nástrojem je bezpodmínečně nutná znalost programovacího jazyka Pascal. Z něj vychází programovací jazyk *Object Pascal*, který je součástí Delphi.

Literatura

Pro dobré zvládnutí učiva je potřeba studovat i z dalších zdrojů. Velmi doporučuji on-line kurz na stránkách Zive.cz:

KADLEC V. *Umíme to s Delphi*, <http://www.zive.cz>, on-line

Tištěné literatury o programování v Delphi je velké množství. Od jednoduchých a rozsahem malých příruček až po do detailů vyvedené a samozřejmě patřičně drahé učebnice. Uvádím alespoň několik vhodných publikací:

LISCHNER R. *Delphi v kostce*, Computer Press, Praha 2000, ISBN 8072263617

CANTU M. *Myslíme v jazyku Delphi 6 – 1.díl knihovna programátora*, Grada 2002, ISBN 8024703343

ELLER F. *Delphi 6 příručka programátora*, Grada, ISBN 8024703033

SVOBODA L., VONEŠ P., KONŠAL T., MAREŠ M. *1001 tipů a triků pro Delphi*, ComputerPress, 2001, ISBN 8072265296

PÍSEK S. *Delphi začínáme programovat*, Grada, ISBN 8024705478

TEIXERA S., PACHECO X. *Mistrovství v Delphi 6*, ComputerPress, 2002, ISBN 8072266276



Shrnutí lekce

Kurz *Vývoj objektových aplikací 1* je ukončen udělením zápočtu. Navazuje na něj v letním semestru kurz *Vývoj objektových aplikací 2*, který je ukončen zkouškou.

- Nezapomeňte do určených termínů zvládnout učivo jednotlivých lekcí.
- Pro získání zápočtu je potřeba se osobně dostavit na katedru a před tutorem obájit a vysvětlit své programy.

Základy objektově orientovaného programování



Cíl lekce

Cílem této lekce je seznámit se se základními pojmy a mechanismy objektově orientovaného programování. Objasnit pojmy jako jsou třída, objekt, metody, vlastnosti, dědičnost, zapouzdření a polymorfismus. Lekce se zabývá obecnými vlastnostmi objektově orientovaného programování bez spojením s konkrétním programovacím jazykem.

Po absolvování lekce budete:

- znát základní vlastnosti objektově orientovaného programování - zapouzdření, dědičnost a polymorfismus
- umět definovat třídu a objekt
- vědět, co jsou to vlastnosti a metody třídy
- umět rozlišit mezi dědičností a kompozicí

Časová náročnost lekce: **2 hodiny**

Vstupní test



Testové otázky a úkoly vstupního testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Kdo je autorem programovacího jazyka Pascal?
 - Blaire Pascal
 - Niclaus Wirth
 - Bill Mates
 - John von Neumann
2. Kdo je autorem programovacího jazyka Pascal?
 - integer
 - char
 - real
 - long integer
3. Kdo je autorem programovacího jazyka Pascal?
 - for (i:=0; i<N; i++) a += 20;
 - for i:=0 to N do a += 20;
 - for i:=0 to N do a =a + 20;
 - for i:=0 to N do a := a + 20;

Úvod

Doposud jsme využívali pouze *imperativního* (příkazového) stylu programování (viz. jazyk Pascal). V rámci programu jsme zavedli proměnné určitého typu, se kterými jsme prováděli určité operace. Pokud nám nevystačovala množina připravených operátorů a podprogramů, definovali jsme své vlastní. Jediné omezení ve vztahu proměnné (data) a operace bylo dáno pouze typem dat. Datový typ předepisoval, které operace lze pro proměnné daného typu provádět. Neexistovalo však implicitní další omezení v rámci povolených operací.

Příklad:

Nad proměnnou typu celé číslo (*integer*) můžeme provádět všechny operace definované pomocí příslušných operátorů, funkcí a procedur, které jsou pro typ *integer* zavedeny. Tudiž z hlediska programovacího jazyka nic nebání provádět s celočíselnou proměnnou *Den*, jejíž úkolem je uchovávat číselnou hodnotu dne v datu, výpočty jako faktoriál. Samozřejmě za předpokladu, že daná funkce je pro typ *integer* definována. Smysluplnost takového výpočtu je však přinejmenším diskutabilní.

Mezi další nevýhody programovacích jazyků tohoto typu patří horší dispozice pro rozšiřování již funkčních programů. Většinou se neobejdeme bez znalostí a změn původního kódu. Proto vývojem programovacích jazyků bylo vytvořeno *objektově orientované programování (OOP)*. Mezi programovací jazyky využívající vlastnosti OOP patří například: Simula, Eiffel, C++, Beta, Small Talk, Java, Visual Basic a Object Pascal (Delphi).

Základní vlastnosti

Základní tezí objektově orientovaného programování (OOP) je používání *objektů*, které nejsou určeny jen množinou dat, ale rovněž množinou přesně stanovených a definovaných operací, které objekt může provádět. Během analýzy programu se snažíme vytvořit a popsat objekty, které v rámci programu mají vzniknout. Objekty mezi sebou komunikují a posílají si zprávy, reagují na události.

Základní vlastnosti OOP:

1. *Zapouzdření (Encapsulation)* - základním pojmem OOP je *třída (class)*, která v sobě zahrnuje data a operace. Konkrétní výskyt třídy je *instance třídy (objekt)*.

2. *Dědičnost (Inheritance)* - Možnost odvozovat nové třídy, které dědí data a metody z jedné nebo více tříd. V odvozených třídách je možno přidávat nebo předefinovávat nová data a metody.

3. *Polymorfismus* - Česky možno vyjádřit přibližně pojmem "vícetvarost". Umožňuje jediným příkazem zpracovávat "podobné" objekty.

Základní pojmy:

Třída - je typ definován uživatelem, který se nachází v nějakém stavu, má reprezentaci a chování. Třída je podobná datovému typu.

Objekt (instance třídy) - je již proměnná určitého datového typu, který je definován třídou. Objekt je konkrétní místo v paměti.

Zapouzdření

Zapouzdření vyjadřuje schopnost spojení atributů (dat) a metod (procedur a funkcí). Objekt dané třídy kromě atributů, které určují jeho *vlastnosti* a stav, má jasně určeno, které operace může provádět. Jsou určeny jeho *schopnosti* něco konkrétního dělat.

Třída je tedy uživatelsky definovaný typ a obsahuje jak členská data, tak i metody.

K jednotlivým složkám třídy můžeme nastavit následující přístupová práva:

public - povoluje přímý přístup k prvkům třídy z vnějšku

private - zakazuje vnější přímý přístup k prvkům třídy

protected - označují se takto prvky nepřístupné vzdáleným přímým přístupem z vnějšku třídy, ale procházející děděním do odvozených tříd.

Kontrolní úkol:

Které klíčové slovo zajišťuje přímý přístup k prvkům třídy z vnějšku třídy?



Dědičnost

Dědičnost (*Inheritance*) umožňuje přidat k základní (rodičovské nebo bázové) třídě T1 další vlastnosti nebo stávající vlastnosti modifikovat a vytvořit novou odvozenou (podtřídu neboli potomka) třídu T2.

Dědičnost může být následujících typů:

Jednoduchá inheritance - třída má jen jednoho předka (rodiče). Vytváříme stromovou hierarchii tříd. Třidu v nejvyšší úrovni označujeme jako kořenovou třídu.

Vícenásobná inheritance - třída může více předků.

Opakovaná inheritance - třída může zdědit vlastnosti některého (vzdálenějšího) předka více cestami. Vztahy tříd v hierarchii jsou znázorňovány orientovaným

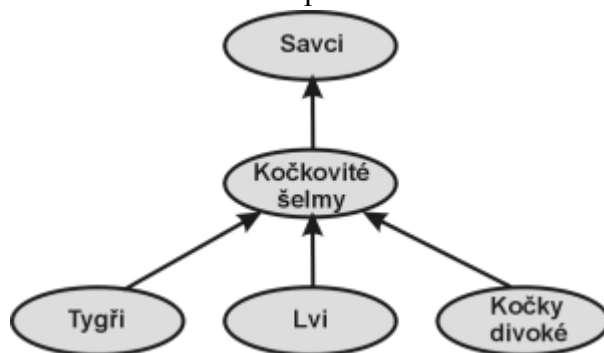
acyklickým grafem (direct acyclic graph - DAG), označovaným také jako *graf příbuznosti tříd*.

Dědičnost vyjadřuje specifikaci - **konkretizaci**. Každá nová třída (potomek) je konkrétnější než jeho předek. Podívejte se na návrh jednoduché dědičnosti *Savci* - *Kočkovité šelmy* - *Lvi*. *Savci* jsou nejobecněji navrženou třídou, *Kočkovité šelmy* jsou již konkrétnějším vyjádřením třídy *Savci*. Třída *Lvi* je ještě konkrétnějším vyjádřením vlastností a schopností.

Jednoduchá dědičnost

Jednoduchá dědičnost určuje, že každá odvozená třída (potomek) má jen jednoho předka (rodiče). Nakreslíme-li si graf jednoduché dědičnosti, získáme strom. V jeho kořenu je základní rodičovská třída. Každý následující uzel stromu je odvozená třída - potomek. Každý potomek má jen jednoho předka.

Podívejte se na obrázek příkladu jednoduché dědičnosti a všimněte si, kterým směrem míří šipky. Ty vyjadřují vzájemnou závislost tříd v rámci dědičnosti. Existence třídy potomka závisí na rodiči a ne naopak.

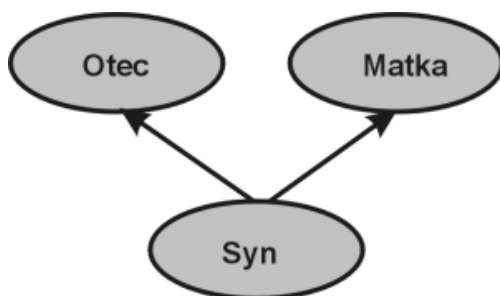


Vícenásobná dědičnost

U vícenásobné dědičnosti může mít odvozená třída (potomek) více než jednoho předka (rodiče). Realizace vícenásobné dědičnosti je obtížná a může způsobovat značné komplikace. Proto některé objektově orientované jazyky neumožňují tento typ dědičnosti vůbec zavádět. Příkladem budiž jazyk Object Pascal, který budeme v dalších lekcích používat. Ve velké většině případů lze vícenásobnou dědičnost nahradit dědičností jednoduchou.

Příklad:

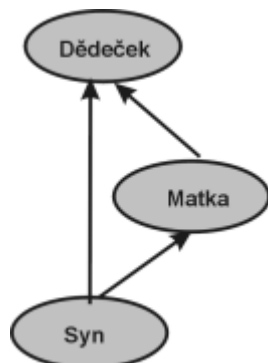
Navrhujeme třídu *Zaměstnanec* a třídu *Student*. Pak vytvoříme třídu *Doktorand*, což je student, ale částečně i zaměstnanec. To znamená, že bychom třídu *Doktorand* odvodili od obou předcházejících tříd. Chceme-li se však vícenásobné dědičnosti vyhnout, pak si stačí uvědomit, že *Doktorand* je hlavně *Student* a konkrétní změny týkající se zaměstnaneckých vlastností a schopností do třídy dodáme.



Opakovaná dědičnost

U opakované dědičnosti může odvozená třída zdědit vlastnosti potomků různými cestami. Například třída *Syn* dědí vlastnosti třídy *Dědeček* přímo nebo prostřednictvím třídy *Matka*.

Pro opakovanou dědičnost platí stejné závěry jako pro dědičnost vícenásobnou. Mnohé jazyky ji opět raději vůbec nezavádějí (Object Pascal).



Polymorfismus

Některé programovací jazyky založené na OOP podporují pouze *pozdní vazbu* (někdy nazvanou *dynamickou vazbu*). To znamená, že o povaze objektu se rozhodne až během samotného průběhu programu. Jazyky jako C++ a Object Pascal vytvářejí implicitně *vazbu brzkou* (*statickou*), která o typu objektu rozhodne již při překladu.

Umožňují rovněž pozdní vazbu a to pomocí mechanismu *virtuálních metod*. U včasné vazby se již při překladu pro jednotlivé metody přiřadí adresy podle typu třídy. Někdy však je potřeba vybrat metodu až v okamžiku, kdy je jasné instanci jaké třídy potřebuji. Představme si například, že potřebujeme vytvořit lineární spojový seznam složený z objektů tříd AA, BB, CC, kde třída CC je dědicem třídy BB a třída BB je dědicem AA. Do seznamu se umísťují objekty, o jejichž povaze se rozhoduje až při běhu programu. Dejme tomu, že každá třída má metodu `Vypis`, která se stará o výpis atributů dané třídy. Všechny tři metody pro jednotlivé třídy se sice jmenují stejně, ale dělají trochu jinou činnost (vypisují různé atributy). Při včasné vazbě by se již při překladu rozhodlo, že metoda `Vypis` bude brána ze třídy AA, i když se rozhodneme pro objekt třídy CC. Aby se tomuto předešlo definuje se příslušná metoda jako *virtuální* pomocí klíčového slova `virtual`.

```
procedure Vypis; virtual;
```

Kontrolní úkol:

Která vazba, včasná nebo pozdní, se uplatní u metody, která má v hlavičce uvedené klíčové slovo `virtual`?



Kompozice

Kompozici rozumíme schopnost vytvářet nové třídy skládáním z již existujících tříd. Na rozdíl od dědičnosti není nově vzniklá třída konkrétnějším případem své rodičovské třídy. Nová třídy již existující třídy pouze obsahuje a přidává nové vlastnosti a schopnosti.

Jednoduchou mnemotechnickou pomůckou pro rozlišení mezi vztahem dědičnosti a kompozice jsou otázky "Je?" a "Má?". Můžeme-li si kladně odpovědět na otázku "Je?", pak se jedná o dědičnost. Kladná odpověď na otázku "Má? Obsahuje?" znamená kompozici.

**Příklad:**

Mějme nalézt vztah mezi třídami *Osoba* a *Zaměstnanec*.

Je *Zaměstnanec* *Osobou*? ANO! Jedná se o dědičnost. Třída *Zaměstnanec* je konkrétnějším případem třídy *Osoba*.

Mějme nalézt vztah mezi třídami *Osoba* a *Datum*.

Je *Osoba* *Datmem*? nebo Je *Datum* *Osobou*? Ani v jednom případě nelze odpovědět kladně.

Má *Osoba* *Datum* (například datum narození)? ANO! Jedná se o kompozici. Třída *osoba* je složena z objektů jiných tříd, například z objektů *datum_narozeni*, *datum_svatby*, *datum_ukonceni_skolni_dochazky* atd.

**Kontrolní úkol:**

Nalezněte a zdůvodněte vztah mezi třídami *Rostliny* a *Květiny*. Jedná se o dědičnost nebo kompozici?

**Opakovací test**

Testové otázky a úkoly opakovacího testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Který z následujících mechanismu popisuje tvorbu nové třídy pomocí konkretizace již existující třídy?
 - kompozice
 - dědičnost
 - polymorfismus
 - zapouzdření
2. Který z následujících mechanismu popisuje tvorbu nové třídy skládáním z objektů již existujících tříd?
 - kompozice
 - dědičnost
 - polymorfismus
 - zapouzdření
3. Které z následujících klíčových slov zajišťuje přímý přístup k prvkům třídy z vnějšku třídy?
 - protecte
 - virtual
 - public
 - private

Shrnutí učiva



- Základními vlastnostmi objektově orientovaného programování jsou:
 - zapouzdření
 - dědičnost
 - polymorfismus
- Základními pojmy jsou třída a objekt (instance třídy)
- Třída obsahuje vlastnosti (členská data) a metody (schopnosti, podpogramy). Zapouzdření vyjadřuje schopnost objektu spojit v jeden celek data a metody. Pomocí přístupových práv určujeme oprávnění pro manipulaci s prvky třídy. Specifikace přístupových práv se určuje klíčovými slovy **private**, **public** a **protected**
- Dědičnost vyjadřuje schopnost vytvářet nové třídy pomocí konkretizace již existujících tříd. Typy dědičnosti jsou jednoduchá, vícenásobná a opakovaná.
- Kompozice vyjadřuje možnost složení třídy z objektů jiných tříd.
- Pomocná otázka pro zjištění zda se jedná o dědičnost zní "Je?" a pro kompozici "Má? Obsahuje?"

Rejstřík

data
dědičnost
instance třídy
kompozice
metody
objekt
polymorfismus
schopnosti
třída
vlastnosti třídy
zapouzdření

Vývoj aplikací pro MS Windows



Cíl lekce

Cílem této lekce je seznámit se se základními postupy při vývoji aplikací pro operační systém MS Windows a to pomocí vizuálních vývojových nástrojů.

Po absolvování lekce budete:

- vědět, jakou strukturu mají programy pro grafické rozhraní operačního systému MS Windows
- vědět, co jsou vizuální vývojové nástroje a jak pracují
- umět provádět základní činnosti ovládání prostředí Borland Delphi

Časová náročnost lekce: 2 hodiny



Vstupní test

Testové otázky a úkoly vstupního testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Které z následujících klíčových slov určuje, že k prvkům třídy nelze přistupovat přímo z vnějšku třídy?
 - public
 - private
 - protecte
 - published
2. Které z následujících klíčových slov určuje, že k prvkům třídy nelze přistupovat přímo z vnějšku třídy?
 - vícenásobnou dědičnost
 - jednoduchou dědičnost
 - opakovanou dědičnost
 - všechny typy dědičnosti
3. Který z následujících mechanismu popisuje tvorbu nové třídy skládáním z objektů již existujících tříd?
 - kompozice
 - dědičnost
 - polymorfismus
 - zapouzdření

Programy ve Windows

Pokud si vzpomenete na své první programy v Pascalu, které byly určeny pro operační systém MS DOS (příkazový řádek ve Windows), tak je možné na nich pozorovat jakousi linearitu provádění. Napsané příkazy následovaly jeden za druhým a takto se i prováděly. Prvním příkazem program začal, postupoval přes další až k poslednímu příkazu, u kterého program skončil. V grafickém operačním systému však průběh programů vypadá jinak. Spuštěný program jako by běžel v nekonečné smyčce a čekal na tzv. *událost*. Pod tímto pojmem si můžeme představit například zmáčknutí určité klávesy, pohyb myši, dvojklik tlačítka myši apod. Jakmile je nějaká událost vyvolána, program na ni zareaguje předem jasně definovanou činností. U kolem programu je tyto reakce přesně vymezit a definovat. Program pak se skládá z relativně samostatných a na sebe přímo nenavazujících podprogramů reakcí na události.



Kontrolní úkol:

Pokuste se vyjmenovat alespoň 5 různých událostí, které mohou nastat při běhu programu.



Námi vytvořený program pro operační systém bude vypadat například podle dále uvedeného schématu.

Hlavním úkolem programátora je sestavit množinu podprogramů, které budou provádět určitou činnost, která se vždy spustí v okamžiku určité události.

- Událost_kliknutí_na_tlačítko_Konec - procedura uzavře otevřená okna a ukončí program
- Událost_posunutí_myši_nad_tlačítko_Vypočti - procedura zajistí změnu vzhledu kurzoru myši ze šipky na ruku s ukazujícím prstem
- Událost_kliknutí_na_tlačítko_Vypočti - procedura provede výpočet příslušné matematické funkce

Jednotlivé procedury - podprogramy jsou vytvářeny relativně samostatně. V hlavním programu nejsou vypisovány sekvenčně jedna za druhou, ale jejich vyvolání provede příslušný grafický objekt v rámci okna aplikace. To však předpokládá, že daný objekt umí příslušnou událost zachytit. Aby se programátorům usnadnila práce, byly vytvořeny vizuální vývojové nástroje. Tyto aplikace mají již předem definované objekty, které splňují nároky kladené na různé grafické komponenty (tlačítka, editovací pole, combo box, list box...).

Vizuální vývojové nástroje

Grafické operační systémy kladou stále větší nároky na programátory, kteří kromě naprogramování samotného problému řešeného v programu, musí zajistit interface aplikace podle grafického prostředí. Neustálým zvyšování nároků na rychlost vývoje aplikací a zároveň zvyšování nároků na délku a složitost kódu se postupně dospělo k vizuálním programovacím nástrojům, které se také nazývají *RAD*. Mezi nejznámější

produkty pro prostředí MS Windows patří Visual Basic, Power++, Delphi, C++Builder a další. Pro operační systém Linux existuje linuxová verze Delphi s názvem Kylix.

Všechny tyto nástroje využívají předprogramované knihovny grafických komponent. Jedná se o objekty, které kromě základních vlastností (např. jméno, rozměry, barva, font textu, ...) umí rovněž reagovat na určité události vyvolané v operačním systému (např. kliknutí tlačítkem myši, zmáčknutí určité klávesy, přesun myši...) a zároveň mají předdefinované metody - schopnosti provádět určitou činnost (např. zrušit objekt, smazat svůj obsah, ...).

V našem kurzu budeme využívat vývojový nástroj firmy Borland s názvem Delphi. Vývojový nástroj Borland Delphi využívá k psaní kódu objektově orientovaný programovací jazyk Object Pascal. Uvedené programy v dalších lekcích jsou odladěny ve verzi Borland Delphi 6 verze Personal.

Komponenty se pak umísťují na formulář (okno aplikace) pomocí myši a dopisuje se pouze kód reakcí na určité události. Díky těmto možnostem lze značně urychlit celkový vývoj programů.



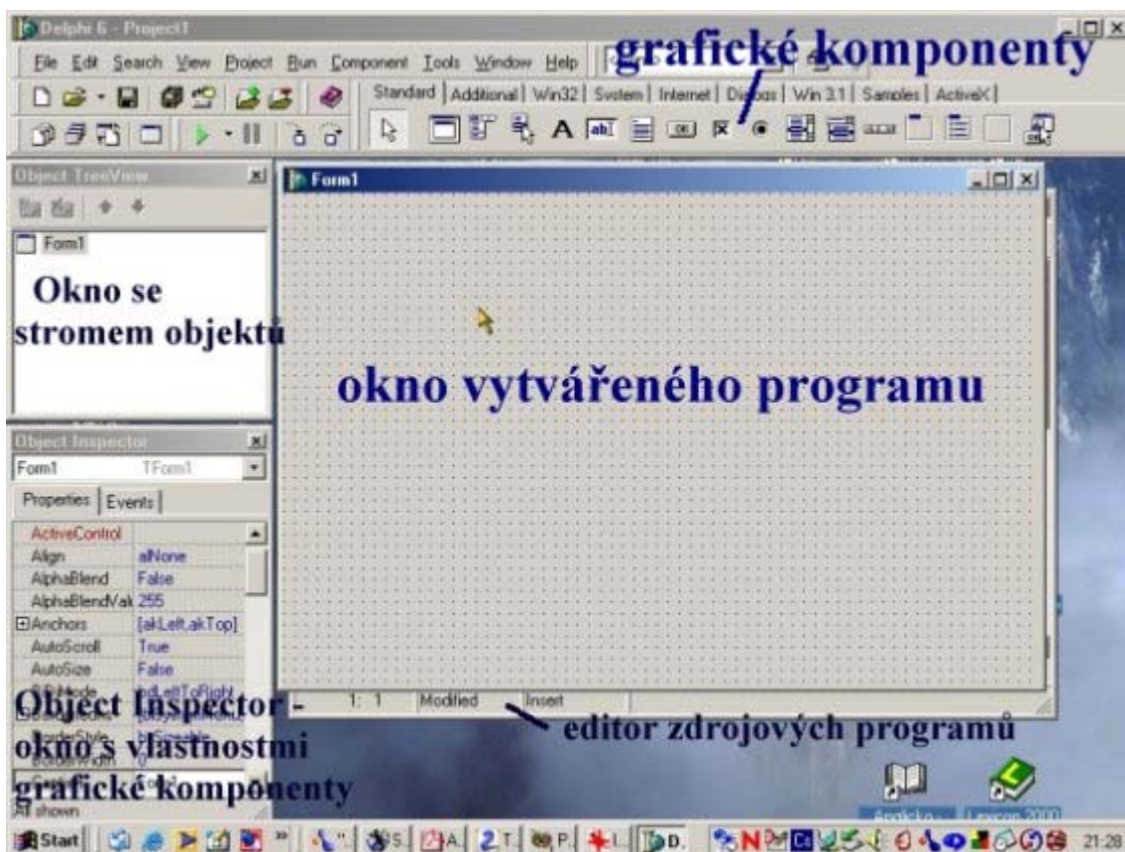
Kontrolní úkol:

Najděte si na internetu na stránkách firmy Borland informace o posledních verzích vývojového nástroje Delphi. Podívejte se na to, jak se liší jednotlivé distribuce podle ceny a možností tvorby aplikací. Odkaz na stránky: www.borland.com

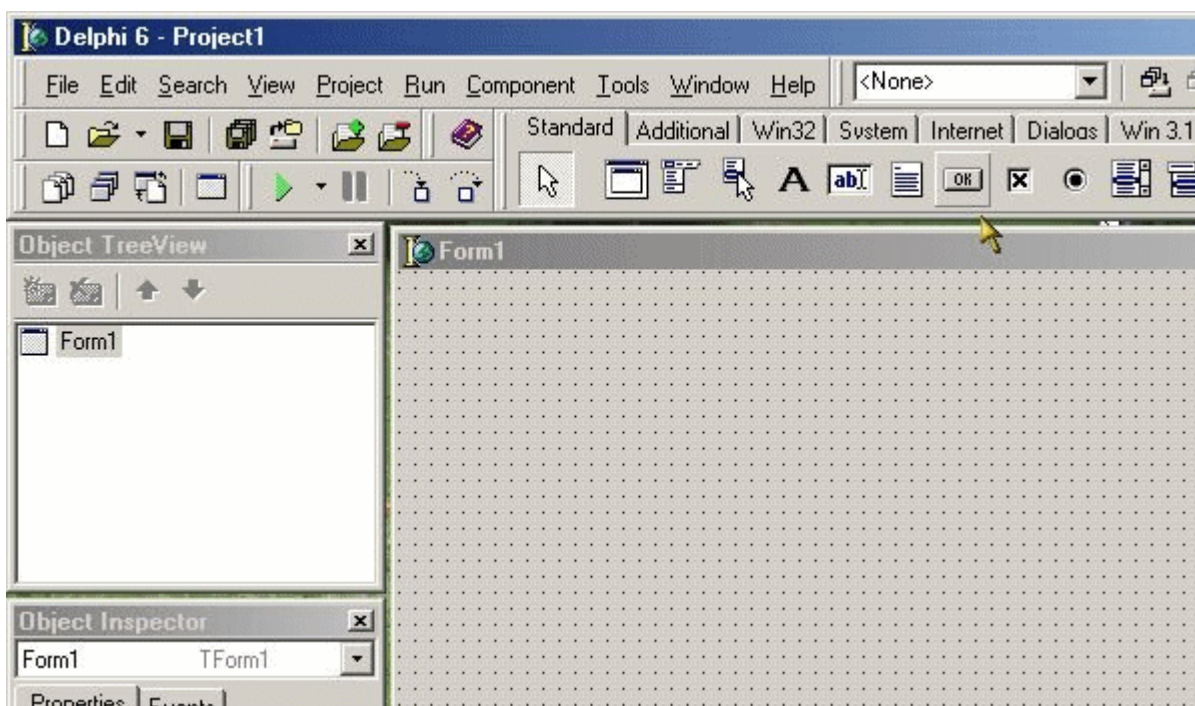
Borland Delphi

Po spuštění vývojového prostředí Borland Delphi se před vámi objeví několik oken. V horní části obrazovky je umístěno hlavní okno, ve kterém je nabídka se všemi činnostmi, které program nabízí. Navíc na záložkách jsou zde umístěny ikonky grafických komponent. Okno s názvem Form1 představuje okno vaší budoucí aplikace.

Pod formulářovým oknem je pak umístěn editor pro psaní kódu jednotlivých zdrojových souborů. V levé části jsou okna pro správu grafických komponent.



Výběr grafické komponenty se provádí levým tlačítkem myši. Označí se příslušná komponenta a kliknutím do vybraného místa na formuláři - oknu se objekt vykreslí. Objekt je možno dále upravovat a přesouvat. Každý nový objekt se objeví v levé části obrazovky v okně *Object TreeView*. Úprava vlastností jednotlivých grafických objektů se provádí v okně *Object Inspector*.



Po vložení grafické komponenty do formuláře je potřeba nastavit některé z vlastností. Ty se nastavují v s názvem *Object Inspector* záložka *Propertis*. Jako první si nastavte vždy vlastnost *Name*, tedy jméno komponenty. Z důvodu přehlednosti není vhodné zůstávat u automatických jmen jako je Button1. Při větším množství tlačítek se za chvíli mezi názvy Button1, Buton2...Button10 jen velmi obtížně budete orientovat.

Doporučená tvorba jména komponent:

První část názvu by měl signalizovat typ komponenty a pak činnost, kterou má komponenta vykonat.

Příklad:

Hlavní formulář - *FrmHlavni*

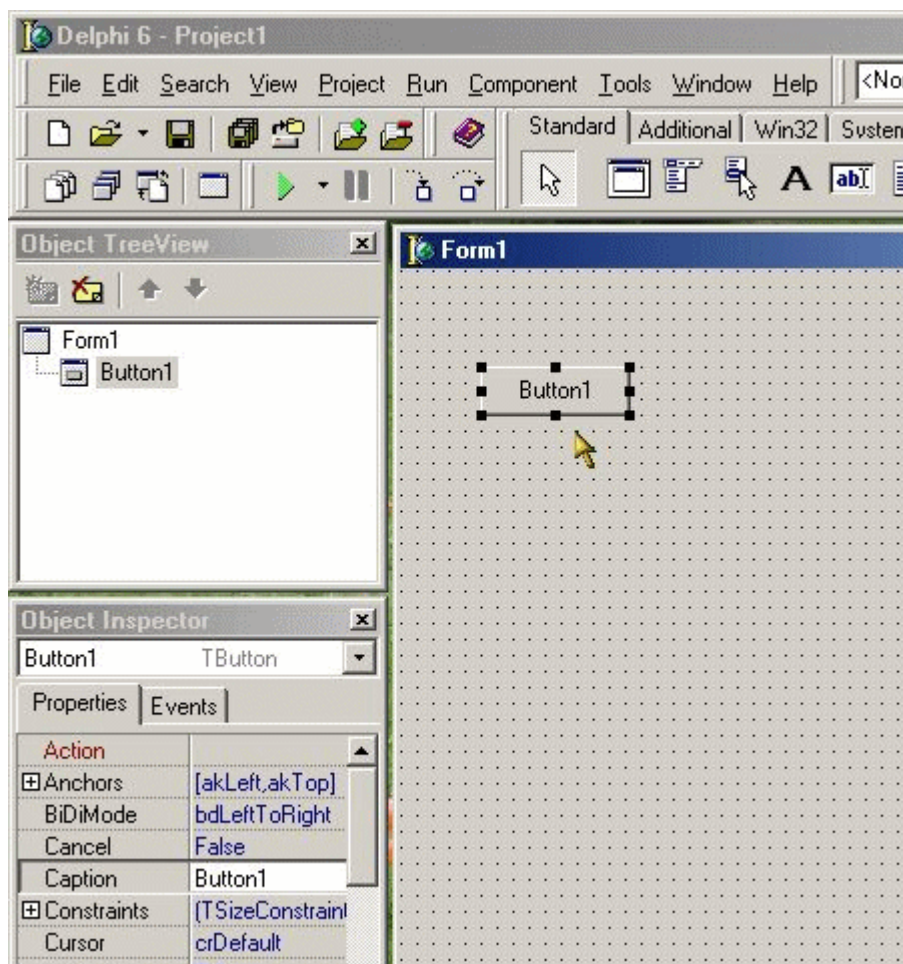
Tlačítko končící program - *BtnKonec*

Edit pro vstup dat - *EditVstup*

Záložka *Events* uvádí události, na které je komponenta schopna reagovat.

Prohlédněte si animaci, ve které u tlačítka nastavíme vlastnosti *Name* a *Caption*.

Dále ošetříme událost *OnClick*, tedy kliknutí myši na tlačítko.



Grafické komponenty

Formulář

První komponentou, na kterou se podíváme, je formulář. Jedná se vlastně okno, které pak uvidíme po spuštění programu.

Některé vlastnosti formuláře:

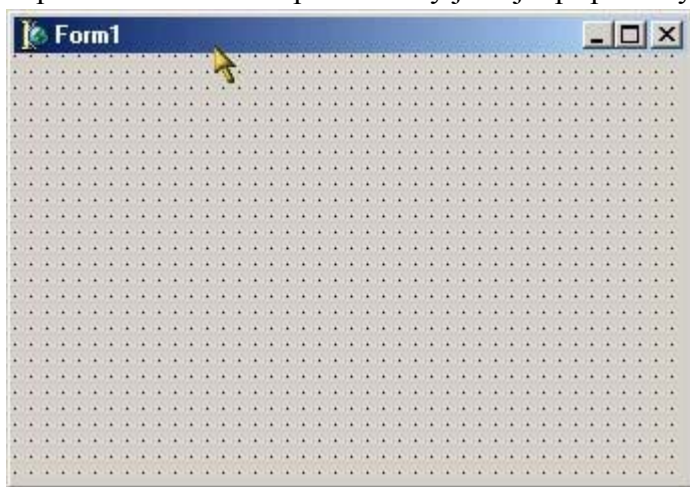
Name - jméno komponenty. Nelze používat české znaky, mezery, +, -, ? a podobně.

Caption - nápis, který uvidíme v titulku okna. Zde je naopak možno použít libovolný znak.

BorderStyle - vlastnosti ohraničení okna. Tato vlastnost má již připravené hodnoty, ze kterých si vybíráme tu, kterou potřebujeme.

FormStyle - styl okna, hodnoty jsou již připraveny.

Position - určuje zda pozice a rozměr okna, bude podle vaší volby nebo podle implicitních hodnot. Opět hodnoty jsou již připraveny.



Kontrolní úkol:

Vyzkoušejte si spustit program s různě nastavenými hodnotami některých vlastností okna a sledujte změnu jeho chování.



Tlačítko – Button

Další grafickou komponentou je tlačítko Button. Nalezneme ji v záložce s názvem *Standard*. Některé vlastnosti tlačítka (změna vlastností se mnohdy projeví až po spuštění programu):

Name - jméno komponenty. Nelze používat české znaky, mezery, +, -, ? a podobně.

Caption - nápis, který uvidíme na povrchu tlačítka. Zde je naopak možno použít libovolný znak. Znak & umístěný v textu znamená, že následující písmenko společně s klávesou ALT nahrazuje kliknutí myši. U tlačítka s textem '&Konec' stačí zmáchnout klávesy Alt + K.

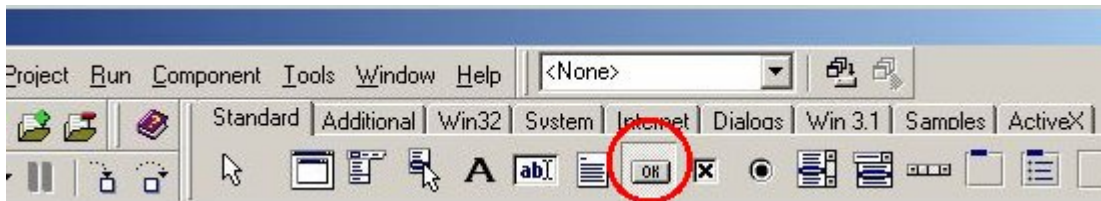
Cursor - lze nastavit, jak bude vypadat kurzor, když myši posuneme nad tlačítko

Font - nastavuje font písma na tlačítku

Enabled - má hodnoty *true* nebo *false* a určuje, zda bude tlačítko aktivní (lze zmáchnout) nebo jej nebude možné používat (nápis na tlačítku zešedne)

Visible - má hodnoty *true* nebo *false* a určuje, zda bude tlačítko na formuláři po spuštění programu vidět či nikoliv

Hint (tip, rada) - obsahuje text, který se objeví v plovoucím textovém okénku necháme-li chvíli myš bez pohybu nad komponentou. Text může sloužit jako jednoduchý tip, co komponenta dělá. Aby plovoucí text byl vidět je potřeba nastavit vlastnost *ShowHint* na hodnotu *true*.



Kontrolní úkol:

Vyzkoušejte si spustit program s různě nastavenými hodnotami některých vlastností tlačítka a sledujte změnu jeho chování.

Label

Grafickou komponentou Label - nápis nalezneme rovněž v záložce s názvem *Standard*. Některé vlastnosti komponenty:

Name - jméno komponenty. Nelze používat české znaky, mezery, +, -, ? a podobně.

Caption - nápis, který uvidíme na místě umístění komponenty

Cursor - lze nastavit, jak bude vypadat kurzor, když myši posuneme nad komponentu Label

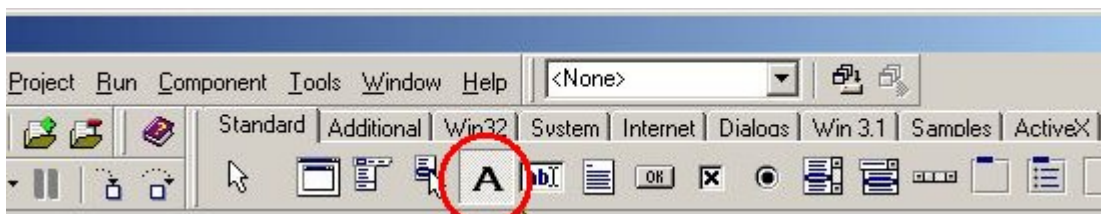
Font - nastavuje font písma

Enabled - má hodnoty *true* nebo *false* a určuje, zda bude komponenta aktivní

Visible - má hodnoty *true* nebo *false* a určuje, zda bude komponenta na formuláři po spuštění programu vidět či nikoliv

WordWrap - má hodnoty *true* nebo *false* a určuje, zda bude text v rámci prostoru komponenty zalomen či zda bude jen na jednom řádku

Hint (tip, rada) - obsahuje text, který se objeví v plovoucím textovém okénku necháme-li chvíli myš bez pohybu nad komponentou. Text může sloužit jako jednoduchý tip, co komponenta dělá. Aby plovoucí text byl vidět je potřeba nastavit vlastnost *ShowHint* na hodnotu *true*.



Kontrolní úkol:

Vyzkoušejte si spustit program s různě nastavenými hodnotami některých vlastností komponenty Label a sledujte změnu chování programu.

Projekt – program

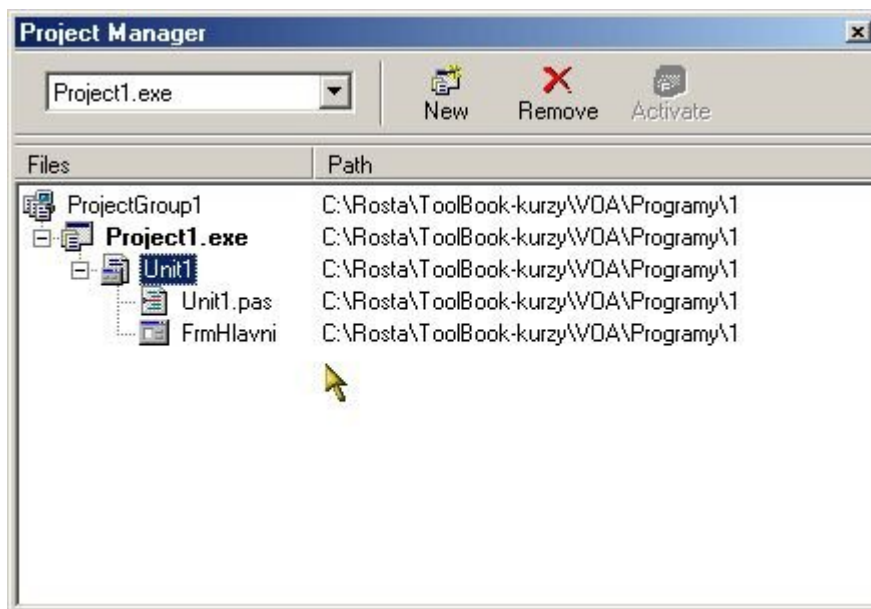
Při práci na programu je vhodné čas od čas (raději častěji) uložit všechny soubory. Na rozdíl od jednoduchého programu v Borland Pascalu v DOSu nepracujeme jen s jedním zdrojovým souborem, ale s mnoha soubory. Proto při ukládání programu (zde nazývaného *project*) je potřeba využívat volby *Save All*.

Pro každý program si vytvořte vlastní složku. Nemějte v jedné složce více než jeden program (projekt)!

Seznam souborů, které vzniknou při práci nad projektem:

- *.dpr - projektový soubor
- *.pas - zdrojové soubory jednotlivých unit
- *.dcu - již zkompileované unity
- *.dfm - soubor s informacemi o formuláři a jeho prvcích
- *.rsc - resource soubory pro ikony aplikace
- *.dpk - ukládá informace o dynamicky linkovaných knihovnách
- *.dof - soubor obsahuje nastavení kompilátoru a linkeru pro daný projekt
- *.~xxx - vlnovka před příponou znamená, že se jedná o záložní soubory

U všech souborů, které ukládáte nepoužívejte české znaky, mezery a apod. Názvy souborů tvořte co nejkratší, nejlépe do osmi znaků.



Součásti projektu si můžeme prohlédnout v okně s názvem *Project Manager*, které si spustíme přes nabídku *View*. Na obrázku vlevo je vidět, že uvedený projekt obsahuje pouze jednu unit, ke které patří formulář s názvem *FrmHlavni*. Soubor *Unit1.pas* obsahuje zdrojový kód unity s názvem *Unit1*. Název souboru a unity musí být shodný. Nelze v něm používat české znaky, mezery apod. Jeho délka by měla být do osmi znaků.

Každá unita se skládá z následujících částí:

unit - klíčové slovo, za kterým se uvádí název unity

interface - část, kde se píší deklarace a hlavičky podprogramu, datových typů, tříd, proměnných

implementation - část, ve které se definují jednotlivé podprogramy a metody tříd

end. - klíčové slovo ukončující unitu

Chceme-li vytvořený program spustit, musíme jej samozřejmě nejprve převést do strojového kódu počítače. O to se postará kompila'tor společně s linkerem. Kompilaci je možné vybrat v nabídce *Project*. Další možností je program přímo spustit v menu *Run* první položkou *Run*. Před spuštěním se program nejprve překompiluje a převede do spustitelného tvaru. Vytvoří se soubor s příponou exe (například *Project1.exe*).



Poznámka

Občas se stane, že program se nekorektně uzavře. Okno aplikace není vidět, ale program zůstal "viset" v paměti. Podíváte-li se do nabídky *Run* a v ní je aktivní volba *Program reset*, pak je to nedílnou známkou, že program ve skutečnosti neskončil. Vybráním této volby program ukončíte.



Kontrolní úkol:

Jaké výhody má rozdělení velkého programu na více modulů - unit?



Opakovací test

Testové otázky a úkoly opakovacího testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Který z následujících textů je možné použít pro vlastnost *Name* u tlačítka?
 - Btn Ulozit
 - BtnUlozit
 - BtnUložit
 - Uložit
2. Která z následujících přípon patří k souboru, který uchovává informace o projektu?
 - pas
 - dcu
 - dpr
 - dfm
3. Která z následujících přípon patří k souboru, který uchovává informace o projektu?
 - Borland Delphi
 - Kylix
 - Visual Basic
 - Borland Pascal 7.0

Shrnutí



- Spuštěný program v grafické prostředí MS Windows jako by běžel v nekonečné smyčce a čekal na tzv. *událost*. Pod tímto pojmem si můžeme představit například zmáčknutí určité klávesy, pohyb myši, dvojklik tlačítka myši apod. Jakmile je nějaká událost vyvolána, program na ni zareaguje předem jasně definovanou činností. Úkolem programu je tyto reakce přesně vymezit a definovat.
- Z důvodů urychlení a zjednodušení programování grafických částí se často používají vizuální programovací nástroje, které se také nazývají *RAD*. Mezi nejznámější produkty pro prostředí MS Windows patří Visual Basic, Power++, Delphi, C++Builder a další. Pro operační systém Linux existuje linuxová verze Delphi s názvem Kylix.
- Vizuální vývojové nástroje mají již připravené třídy pro tvorbu grafických komponent. Ty mají určité vlastnosti, metody a dovedou zachytit konkrétní události.
- Vývojový nástroj Borland Delphi využívá k psaní kódu objektově orientovaný programovací jazyk Object Pascal.
- Program – projekt vytvořený v Borland Delphi se skládá z několika modulů. Zdrojový kód se píše do souborů s názvem unita.

Rejstřík

button
form
grafická komponenta
implementation
interface
kompilátor
label
linker
projekt
unita

Tvorba projektů č.1



Cíl lekce

Cílem této lekce je seznámit se se základními postupy při tvorbě programů v Delphi na praktických příkladech.

Po absolvování lekce budete:

- umět ve svých programech používat komponenty formulář, tlačítko a label
- znát obvyklé chyby začátečníků, kterých by jste se měli vyvarovat

Časová náročnost lekce: 2 hodiny



Vstupní test

Testové otázky a úkoly vstupního testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Která z následujících grafických komponent nemá vlastnost Name?
 - Buton
 - Form
 - Label
 - Všechny komponenty tuto vlastnost mají
2. Ze kterých klíčových slov se minimálně musí skládat zdrojový text unity?
 - unit, uses, interface, end
 - uses, interface, implementation, end
 - unit, interface, implementation, end
 - program, implementation, begin, end
3. Které klíčové slovo připojuje další unitu?
 - unit
 - uses
 - init
 - include

Projekt č.1

Vytvořte program, který ve svém okně bude mít nápis "Zmáčkní tlačítko a to ukončí aplikaci" a tlačítko s nápisem "Konec", které program ukončí.

Rozbor programu:

1. Vytvořte si složku (adresář), do kterého budeme ukládat všechny soubory projektu.
 2. Spusťte Delphi.
 3. Pojmenujte formulář *FrmHlavni*. Upravte jeho velikost. Nastavte vlastnost *Caption* na text "Moje aplikace".
 4. Umístěte na plochu formuláře komponentu *label* a pojmenujte ji *LblNapis* a nastavte vlastnost *Caption* na text "Zmáčkní tlačítko a to ukončí aplikaci".
 5. Umístěte na plochu formuláře komponentu *button* a pojmenujte ji *BtnKonec*. Vlastnost *Caption* nastavte na text "&Konec".
 6. U komponenty *BtnKonec* vyberte událost *OnClick* a dopište kód ukončení aplikace.
`Application.Terminate;`
 7. Uložte všechny soubory do připravené složky a spusťte program. Vyzkoušejte vlastnosti okna a jeho částí.
 8. Ukončete program a postupně měňte níže uvedené vlastnosti a sledujte, jak se mění chování programu.
- Vyzkoušejte si změnit hodnoty následujících vlastností:

- formuláře *FrmHlavni*

- *BorderStyle*
- *FormStyle*
- *Position*

- *BtnKonec*

- *Cursor*
- *Enabled*
- *Visible*

- *LblNapis*

- *Font*
- *WordWrap*

Vždy nastavte jednu z vlastností na novou hodnotu a program spustěte. Sledujte, jak se změní vzhled a chování programu.

Soubor Project1.dpr



```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {FrmHlavni};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TFrmHlavni, FrmHlavni);
  Application.Run;
end.
```

Soubor Unit1.pas

```
// Příklady v Delphi
// Mgr. Rostislav Fojtík, 2002
// odladěno v Borland Delphi 6

{
  Vyzkoušejte si změnit hodnoty následujících vlastností:
  - formuláře FrmHlavni
    - BorderStyle
    - FormStyle
    - Position
  - BtnKonec
    - Cursor
    - Enabled
    - Visible
  - LblNapis
    - Font
    - WordWrap

  Vždy nastavte jednu z vlastností na novou hodnotu a program
  spustete. Sledujte, jak se změni vzhled a chování programu.
}

unit Unit1;

interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms,
  Dialogs, StdCtrls;

type
  TFrmHlavni = class(TForm)
    LblNapis: TLabel;
    BtnKonec: TButton;
    procedure BtnKonecClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmHlavni: TFrmHlavni;

implementation

{$R *.dfm}

procedure TFrmHlavni.BtnKonecClick(Sender: TObject);
begin
  Application.Terminate; {Ukončení programu}
end;

end.
```

Když se podíváte na soubor Project1.dpr, který se vytvořil automaticky po uložení projektu, zjistíte, že se vlastně jedná o soubor s hlavní částí programu, který zajišťuje vytvoření hlavního formuláře přes unitu Unit1.

Kromě zajištění vytvoření formuláře a spuštění aplikace se však v souboru již nenacházejí další příkazy. Vše ostatní je totiž v unitě, kterou jsme vytvořili.

Textový obsah souboru Unit1.dfm

```
object FrmHlavni: TFrmHlavni
  Left = 192
  Top = 107
  BorderStyle = bsDialog
  Caption = 'Moje aplikace'
  ClientHeight = 104
  ClientWidth = 207
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object LblNapis: TLabel
    Left = 16
    Top = 8
    Width = 175
    Height = 13
    Caption = 'Zm'#225#269'kni tla'#269#237'tko a to
ukon'#269#237' aplikaci'
  end
  object BtnKonec: TButton
    Left = 64
    Top = 56
    Width = 75
    Height = 25
    Caption = '&Konec'
    TabOrder = 0
    OnClick = BtnKonecClick
  end
end
```

Projekt č.2

Vytvořte program, který provede takovou malou "legráčku". Na formuláři bude text "Rychle klikni na tlačítko!". Když uživatel klikne na tlačítko, to zmizí a místo původního textu se objeví nápis "Pozdě!".

Rozbor programu:

1. Vytvořte si složku (adresář), do kterého budeme ukládat všechny soubory projektu.
2. V menu *File* vyberte položku *New Application* – nová aplikace. Vytvoříte nový projekt.
3. Pojmenujte formulář *FrmHlavni*. Upravte jeho velikost. Nastavte vlastnost *Caption* na text "Legráčka".
4. Umístěte na plochu formuláře komponentu *label* a pojmenujte ji *LblNapis* a nastavte vlastnost *Caption* na text "Rychle klikni na tlačítko!".
5. Umístěte na plochu formuláře komponentu *button* a pojmenujte ji *BtnRychle*. Vlastnost *Caption* nastavte na text "Rychle".
6. U komponenty *BtnRychle* vyberte událost *OnClick* a dopište kód. Nejprve změnu vlastnosti *Caption* u komponenty *LblNapis* a pak změnu vlastnosti *Visible* u komponenty *BtnRychle*.
`LblNapis.Caption:='Pozdě!';`
`BtnRychle.Visible:=false;`
7. Uložte všechny soubory do připravené složky a spusťte program.



Kontrolní úkol:

Prohlédněte si všechny soubory z projektu.



Soubor Project2.dpr

```
program Project2;

uses
  Forms,
  Unit1 in 'Unit1.pas' {FrmHlavni};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TFrmHlavni, FrmHlavni);
  Application.Run;
end.
```

Soubor Unit1.pas

```
// Příklady v Delphi
// Mgr. Rostislav Fojtík, 2002
// odladěno v Borland Delphi 6

unit Unit1;

interface

uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,  
Dialogs, StdCtrls;
```

```
type
```

```
TFrmHlavni = class(TForm)  
    LblNapis: TLabel;  
    BtnRychle: TButton;  
    procedure BtnRychleClick(Sender: TObject);  
private  
    { Private declarations }  
public  
    { Public declarations }  
end;
```

```
var
```

```
    FrmHlavni: TFrmHlavni;
```

```
implementation
```

```
{ $R *.dfm }
```

```
procedure TFrmHlavni.BtnRychleClick(Sender: TObject);
```

```
begin
```

```
    LblNapis.Caption:='Pozdě!';
```

```
    BtnRychle.Visible:=false;
```

```
end;
```

```
end.
```

Projekt č.3

Další program bude opět taková malá "legráčka". Po spuštění programu se v levé části formuláře objeví tlačítko s nápisem "Konec". Jakmile však uživatel přesune myš nad tlačítko, to zmizí a objeví se na pravé straně okna. Přesuneme-li na něj kurzor myši, opět zmizí a je znovu vlevo.

Rozbor programu:

1. Vytvořte si složku (adresář), do kterého budeme ukládat všechny soubory projektu.
2. V menu *File* vyberte položku *New Application* nová aplikace. Vytvoříte nový projekt.
3. Pojmenujte formulář *FrmHlavni*. Upravte jeho velikost. Nastavte vlastnost *Caption* na text "Legráčka".
4. Umístěte na plochu formuláře vlevo komponentu *button* a pojmenujte ji *BtnLeve*. Vlastnost *Caption* nastavte na text "Konec".
5. Umístěte na plochu formuláře vpravo komponentu *button* a pojmenujte ji *BtnPrave*. Vlastnost *Caption* nastavte na text "Konec". Vlastnost *Visible* nastavte na hodnotu *true*.
6. U komponenty *BtnLeve* vyberte událost *OnClick* a dopište kód. Nejprve zmizení komponenty *BtnLeve* a pak objevení komponenty *BtnPrave*.
`BtnLeve.Visible:=false;`
`BtnPrave.Visible:=true;`
7. U komponenty *BtnPrave* rovněž ošetřete reakci na událost *OnClick*.
`BtnPrave.Visible:=false;`
`BtnLeve.Visible:=true;`
8. Uložte všechny soubory do připravené složky a spusťte program.



Soubor Project3.dpr

```
program Project3;

uses
  Forms,
  Unit1 in 'Unit1.pas' {FrmHlavni};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TFrmHlavni, FrmHlavni);
  Application.Run;
end.
```

Soubor Unit1.pas

```
// Příklady v Delphi
// Mgr. Rostislav Fojtík, 2002
// odladěno v Borland Delphi 6

unit Unit1;

interface

uses
```



```
Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,  
Dialogs, StdCtrls;
```

```
type
```

```
TFrmHlavni = class(TForm)  
    BtnLeve: TButton;  
    BtnPrave: TButton;  
    procedure BtnLeveMouseMove(Sender: TObject; Shift:  
TShiftState;  
                                X,Y: Integer);  
    procedure BtnPraveMouseMove(Sender: TObject; Shift:  
TShiftState;  
                                X,Y: Integer);  
  
    private  
        { Private declarations }  
    public  
        { Public declarations }  
end;
```

```
var
```

```
    FrmHlavni: TFrmHlavni;
```

```
implementation
```

```
{ $R *.dfm }
```

```
procedure TFrmHlavni.BtnLeveMouseMove(Sender: TObject;  
    Shift:TShiftState;X, Y: Integer);
```

```
begin
```

```
    BtnLeve.Visible:=false;
```

```
    BtnPrave.Visible:=true;
```

```
end;
```

```
procedure TFrmHlavni.BtnPraveMouseMove(Sender: TObject;  
    Shift: TShiftState;X, Y: Integer);
```

```
begin
```

```
    BtnLeve.Visible:=true;
```

```
    BtnPrave.Visible:=false;
```

```
end;
```

```
end.
```



Časté chyby

Začátečníci se při tvorbě programů v Delphi často dopouštějí následujících chyb:

- Neukládáte každý projekt do zvláštní složky - adresáře. Velmi rychle pak ztratíte přehled o tom, které soubory patří k projektu. Často si soubory z minulých programů přepisujete soubory z programů nových.
- Při snaze změnit některou vlastnost u objektu se snažíte vložit novou hodnotu přímo do objektu. Klasickým příkladem je snaha o změnu textu u komponenty label následujícím způsobem:

```
Label1:='Pozdě!';
```

Správně:

```
Label1.Caption:='Pozdě!';
```

- Ihned nepojmenováváte komponentu, kterou jste vložili do formuláře. Odůvodnění většinou zní "Já to pojmenuji později". Později na to většinou již není čas ani chuť. Navíc pozdější přejmenovávání stojí obvykle více práce. Pamatujte, že automaticky vytvářené názvy grafických komponent jsou velmi nepřehledné.
- Při nastavování vlastností či událostí určité grafické komponenty zapomínáte komponentu označit nebo si ji vybrat mezi ostatními komponentami v okně *Object Inspector*. Pak se stává, že například chcete nastavit vlastnost *Caption* u tlačítka, ale vybraný je formulář. Svůj text zapíšete do příslušné kolonky a s údivem sledujete, že se text neobjevil na tlačítku, ale v titulku okna. Pamatujte, že počítač ani Delphi nemá telepatické schopnosti, aby tušil, kterou komponentu chcete změnit!
- Ke svému zdrojovému kódu nepřipojujete poznámky. Váš zápis se tím stává nepřehledným pro další úpravu.
- Mnohdy nad svým postupem nepřemýšlíte a jen zkoušíte, co to udělá. Kdyby jste se tak chovali v běžném životě, zřejmě by to pro vás skončilo katastroficky. Proto se nedivte, že vaše programy jsou občas zmatené a nedělají to, co by měli.



Opakovací test

Testové otázky a úkoly opakovacího testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Který z následujících zápisů zajistí, že komponenta Button1 nebude na formuláři vidět?
 - Button!.Enabled:=false;
 - Button1.Visible:=false;
 - Button1.Visible:=true;
 - Button1:=false;
2. Která z následujících přípon patří zdrojovému souboru s unitou?
 - dcu
 - prj
 - dfm
 - pas

Shrnutí



- Nezapomínejte každý projekt ukládat do zvláštní složky.
- Před zahájením práce na svém programu si udělejte rozbor problému a navrhnete nejvhodnější řešení.
- Pište do svých programů poznámky a všechny použité grafické komponenty ihned po vložení do formuláře přejmenujte.
- Neustále sledujte s jakou komponentou pracujete, ať omylem nezměníte vlastnosti jinému objektu.

Rejstřík

button
form
label
projekt

Tvorba projektů č.2



Cíl lekce

Cílem této lekce je seznámit se se základními postupy při tvorbě programů v Delphi na praktických příkladech.

Po absolvování lekce budete:

- umět ve svých programech používat komponenty Edit, MainMenu
- umět převádět řetězce na celá čísla a opačně
- vědět, jak jeden kód využít i pro jinou komponentu

Časová náročnost lekce: 2 hodiny



Vstupní test

Testové otázky a úkoly vstupního testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Která z následujících přípon patří překompilovanému souboru s unitou?
 - pas
 - dcu
 - dfm
 - dpr
2. Která z následujících přípon patří souboru, který má informace o formuláři?
 - pas
 - dcu
 - dfm
 - dpr
3. Jak se jmenuje programovací jazyk využívaný nástrojem Borland Delphi?
 - Pascal
 - Visual Pascal
 - Object Pascal
 - Basic

Edit

Grafická komponenta Edit slouží jako textové pole, do kterého může uživatel programu zapisovat data. Edit umožňuje využívat jen jeden řádek textu.

Některé vlastnosti objektu Edit:

Name - jméno objektu

Text - vlastnost, která uchovává posloupnost zapsaných znaků

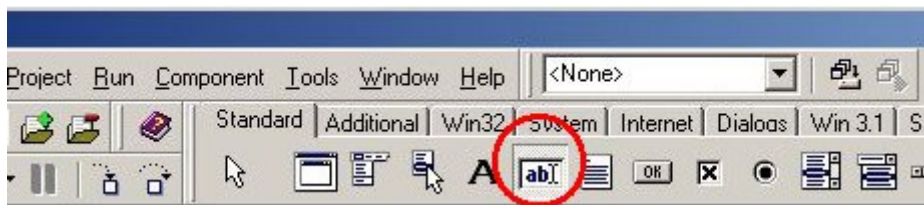
Font - nastavení písma v editovacím poli

MaxLength - číslo udává maximální počet znaků v textu, které lze zapsat

ReadOnly - má hodnoty *true* nebo *false* a určuje, zda bude moci uživatel jen text číst nebo i zapisovat

Enabled - má hodnoty *true* nebo *false* a určuje, zda bude komponenta aktivní

Visible - má hodnoty *true* nebo *false* a určuje, zda bude komponenta na formuláři po spuštění programu vidět či nikoliv



Kontrolní úkol:

Prohlédněte si další vlastnosti komponenty.



MainMenu

Grafická komponenta MainMenu slouží k vytváření hlavní nabídky.

Některé vlastnosti objektu MainMenu:

Name - jméno objektu

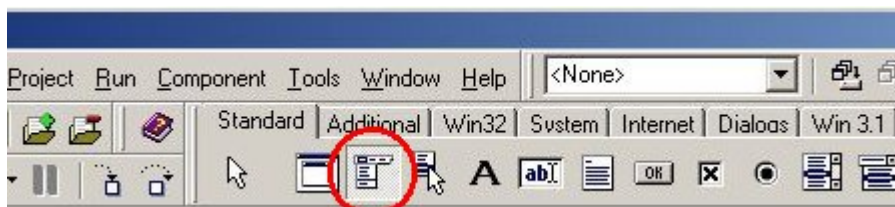
Enabled - má hodnoty *true* nebo *false* a určuje, zda bude komponenta aktivní

Visible - má hodnoty *true* nebo *false* a určuje, zda bude komponenta na formuláři po spuštění programu vidět či nikoliv

Shortcut - umožňuje nastavit klávesové zkratky pro přímé spuštění nabídky

Checked - má hodnoty *true* nebo *false* a určuje zaškrtnutí vybrané položky

Komponenta může být umístěna kdekoli ve formuláři. Sama o sobě není po spuštění programu vidět. Dvojitým kliknutím na komponentu MainMenu na formuláři se otevře editor hlavní nabídky. V něm si vytvoříme jednotlivé položky.



Kontrolní úkol:

Prohlédněte si další vlastnosti komponenty.



Projekt č.4

Vytvořte program, který bude obsahovat na formuláři objekty Edit, Button a Label. Uživatel zapíše text do Editu a stiknutím tlačítka přesune text do komponenty Label.

Rozbor programu:

1. Vytvořte si složku (adresář), do kterého budeme ukládat všechny soubory projektu.
2. Spustěte Delphi.
3. Pojmenujte formulář *FrmHlavni*. Upravte jeho velikost. Nastavte vlastnost *Caption* na text "Vstup dat".
4. Umístěte na plochu formuláře komponentu *label* a pojmenujte ji *LblVypis* a vlastnost *Caption* nechte prázdnou.
5. Umístěte na plochu formuláře komponentu *button* a pojmenujte ji *BtnPresun*. Vlastnost *Caption* nastavte na text "Přesuň text".
6. U komponenty *BtnPresunc* vyberte událost *OnClick* a dopište kód. Nejprve zkopírujeme text do Labelu a pak vymažeme Edit. Kurzor následně přesuneme zpět do Editu.
`LblVypis.Caption:=EditVstup.Text;`
`EditVstup.Clear;`
`{nebo EditVstup.Cation:='' ;}`
`EditVstup.SetFocus;`
7. Uložte všechny soubory do připravené složky a spustěte program.



Soubor Project4.dpr

```
program Project4;

uses
  Forms,
  Unit1 in 'Unit1.pas' {FrmHlavni};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TFrmHlavni, FrmHlavni);
  Application.Run;
end.
```

Soubor Unit1.pas

```
// Příklady v Delphi
// Mgr. Rostislav Fojtík, 2002
// odladěno v Borland Delphi 6

unit Unit1;

interface

uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,  
Dialogs, StdCtrls;
```

```
type
```

```
TFrmHlavni = class(TForm)  
    EditVstup: TEdit;  
    BtnPresun: TButton;  
    LblVypis: TLabel;  
    procedure BtnPresunClick(Sender: TObject);  
private  
    { Private declarations }  
public  
    { Public declarations }  
end;
```

```
var
```

```
    FrmHlavni: TFrmHlavni;
```

```
implementation
```

```
{ $R *.dfm }
```

```
procedure TFrmHlavni.BtnPresunClick(Sender: TObject);
```

```
begin
```

```
    LblVypis.Caption:=EditVstup.Text;
```

```
    EditVstup.Clear;
```

```
    {nebo EditVstup.Cation:='' ;}
```

```
    EditVstup.SetFocus; {přesune kurzor do editovacího pole  
EditVstup}
```

```
end;
```

```
end.
```

Projekt č.5

Vytvořte program, který bude umět sčítat dvě celá čísla. Z cvičných důvodů umístíme na formulář tři objekty typu Edit. Dva budou sloužit pro vstup dat a třetí, který nebude moci být editován, bude sloužit pro výstup výsledku. Stiskem tlačítkem se provede výpočet.

Rozbor programu:

1. Vytvořte si složku (adresář), do kterého budeme ukládat všechny soubory projektu.
2. V menu *File* vyberte položku *New Application* nová aplikace. Vytvoříte nový projekt.
3. Pojmenujte formulář *FrmKalkulacka*. Upravte jeho velikost. Nastavte vlastnost *Caption* na text "Kalkulačka".
4. Umístěte na plochu formuláře další komponenty a pojmenujte je.

```
FrmKalkulacka = class(TForm)
```

```
    EditVstup1: TEdit;
```

```
    EditVstup2: TEdit;
```

```
    EditVysledek: TEdit;
```

```
    BtnSecti: TButton;
```

```
    . . .
```

```
end;
```

5. U komponenty *EditVysledek* nastavte vlastnost *ReadOnly* na hodnotu *true*.

6. U komponenty *BtnSecti* vyberte událost *OnClick* a dopište kód. Je potřeba si uvědomit, že vlastnost *Text* u komponenty *Edit* pracuje s řetězcí. Ty je potřeba změnit na celá čísla. Nezapomeňte si zadefinovat pomocné proměnné

```
procedure TFrmKalkulacka.BtnSectiClick(Sender: TObject);
```

```
    var op1,op2,vysledek:integer;
```

```
begin
```

```
    op1:=StrToInt(EditVstup1.Text);
```

```
    op2:=StrToInt(EditVstup2.Text);
```

```
    vysledek:=op1 + op2;
```

```
    EditVstup1.Clear; {vymazání editovacího pole}
```

```
    EditVstup2.Clear; {vymazání editovacího pole}
```

```
    EditVysledek.Text:=IntToStr(vysledek);
```

```
    EditVstup1.SetFocus; {přesunutí kurzoru}
```

```
end;
```

7. Uložte všechny soubory do připravené složky a spusťte program.

Poznámka:

Předpokládáme, že vstupem dat budou jen celá čísla. V případě zadání jiných znaků se náš program zhroutí. Ošetření vstupů si necháme až na pozdější lekce.



Soubor Project4.dpr

```
program Project5;
```

```
uses
```

```
    Forms,
```

```
    Unit1 in 'Unit1.pas' {FrmKalkulacka};
```



```
{ $R *.res }
```

```
begin
  Application.Initialize;
  Application.CreateForm(TFrmKalkulacka, FrmKalkulacka);
  Application.Run;
end.
```

Soubor Unit1.pas

```
// Příklady v Delphi
// Mgr. Rostislav Fojtík, 2002
// odladěno v Borland Delphi 6
```

```
unit Unit1;
```

```
interface
```

```
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms,
  Dialogs, StdCtrls;
```

```
type
```

```
  TFrmKalkulacka = class(TForm)
    EditVstup1: TEdit;
    EditVstup2: TEdit;
    EditVysledek: TEdit;
    BtnSecti: TButton;
    procedure BtnSectiClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
```

```
  FrmKalkulacka: TFrmKalkulacka;
```

```
implementation
```

```
{ $R *.dfm }
```

```
procedure TFrmKalkulacka.BtnSectiClick(Sender: TObject);
```

```
var op1, op2, vysledek: integer;
```

```
begin
```

```
  op1:=StrToInt(EditVstup1.Text); {převedení textu na číslo}
  op2:=StrToInt(EditVstup2.Text); {převedení textu na číslo}
  vysledek:=op1 + op2;
  EditVstup1.Clear;   {vymazání editovacího pole}
  EditVstup2.Clear;   {vymazání editovacího pole}
  EditVysledek.Text:=IntToStr(vysledek);
  {zapsání výsledku do třetího editovacího pole}
  EditVstup1.SetFocus;   {přesunutí kurzoru}
```

```
end;
```

end.

Soubor Unit1.dfm

```
object FrmKalkulacka: TFrmKalkulacka
  Left = 192
  Top = 107
  Width = 187
  Height = 163
  Caption = 'Kalkula'#269'ka'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object EditVstup1: TEdit
    Left = 8
    Top = 16
    Width = 73
    Height = 21
    TabOrder = 0
  end
  object EditVstup2: TEdit
    Left = 96
    Top = 16
    Width = 73
    Height = 21
    TabOrder = 1
  end
  object EditVysledek: TEdit
    Left = 48
    Top = 96
    Width = 81
    Height = 21
    ReadOnly = True
    TabOrder = 3
  end
  object BtnSecti: TButton
    Left = 48
    Top = 56
    Width = 81
    Height = 25
    Caption = '&Se'#269'ti'
    TabOrder = 2
   OnClick = BtnSectiClick
  end
end
end
```

Projekt č.6

Vytvořte program, ve kterém si vyzkoušíme možnosti práce s hlavním nabídkou. Do formuláře umístíte tlačítko, které ukončí aplikaci. Ukončení umožní rovněž položka hlavního menu.

Rozbor programu:

1. Vytvořte si složku (adresář), do kterého budeme ukládat všechny soubory projektu.
2. V menu *File* vyberte položku *New Application* nová aplikace. Vytvoříte nový projekt.
3. Pojmenujte formulář *FrmHlavni*. Upravte jeho velikost. Nastavte vlastnost *Caption* na text "Pokusy s hlavní nabídkou".
4. Umístěte na plochu formuláře další komponenty a pojmenujte je.

```
TFrmHlavni = class(TForm)
  MainMenu1: TMainMenu;
  mmSoubor: TMenuItem;
  mmKonec: TMenuItem;
  BtnKonec: TButton;
  mmOkno: TMenuItem;
  mmNone: TMenuItem;
  mmNormalni: TMenuItem;
  procedure BtnKonecClick(Sender: TObject);
  procedure mmNormalniClick(Sender: TObject);
  procedure mmNoneClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```

5. U komponenty *BtnKonec* vyberte událost *OnClick* a dopište kód ukončení aplikace

```
Application.Terminate;
```

6. Stejný kód bychom mohli napsat i události *OnClick* u položky *mmKonec*. Tento postup však není vhodný. Proč psát stejný kód dvakrát? V *Object Inspectoru* vybereme u události v seznamu v našem případě jedinou možnou událost s názvem *BtnKonecClick*.

7. V položce Okno vytvořte nabídky pro změnu vzhledu okna. Jednou bude okno s běžným rámečkem a titulkovým pruhem. Druhá nabídka zruší u okna rámeček.

```
procedure TFrmHlavni.mmNormalniClick(Sender: TObject);
begin
  if (mmNormalni.Checked<>true) then
  begin
    mmNormalni.Checked:=true;
    mmNone.Checked:=false;
    FrmHlavni.BorderStyle:=bsSizeable;
    {Nastavení normálního rámečku okna}
  end;
end;
```

```
procedure TFrmHlavni.mmNoneClick(Sender: TObject);
begin
  if (mmNone.Checked<>true) then
  begin
    mmNormalni.Checked:=false;
    mmNone.Checked:=true;
    FrmHlavni.BorderStyle:=bsNone;
```

```
        {Nastavení okna bez rámečku}  
    end  
end;
```

8. Uložte všechny soubory do připravené složky a spust'te program.

Časté chyby



Začátečníci se při tvorbě programů v Delphi často dopouštějí následujících chyb:

- Znovu vás nabádám, aby jste každý projekt ukládali do zvláštní složky - adresáře. Velmi rychle v opačném případě ztratíte přehled o tom, které soubory patří k projektu.
- Nezapomeňte měnit řetězec z hodnoty *Edit.Text* na číselné hodnoty (případně naopak)!
- Stejný kód máte na více místech - ve více podprogramech. Vždy se snažte mít konkrétní algoritmus zpracován jen na jednom místě. Napíšete-li jej do více míst programu, dřív či později se začnou zdrojové texty lišit!

Opakovací test



Testové otázky a úkoly opakovacího testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Která z následujících vlastností umožňuje zakázat zápis do viditelného objektu typu TEdit?
 - visible
 - name
 - text
 - readonly
2. Kterou z následujících vlastností nemá komponenta Edit?
 - visible
 - name
 - text
 - caption
3. Jak se musí nazývat unita uložena v souboru s názvem Moje.pas?
 - Moje
 - Unit1
 - Moje.pas
 - libovolně



Shrnutí

- Grafická komponenty typu Edit slouží nejčastěji pro zápis dat do programu. Pracuje s řetězci, které se v případě manipulace s číselnými hodnotami nejprve změnit na příslušný datový typ.

- Pro práci s hlavní nabídkou se využívá komponenta MainMenu. Ikonku objektu lze umístit kdekoli do formuláře, neboť po spuštění programu není objekt vidět. Jen v horní části okna se objeví jednotlivé nabídky, které jsme v hlavním menu vytvořili.

Rejstřík

Edit

MainMenu

Tvorba projektů č.3

Cíl lekce

Cílem této lekce je seznámit se se základními postupy při tvorbě programů v Delphi na praktických příkladech.



Po absolvování lekce budete:

- umět ve svých programech používat objekty typu Memo, OpenDialog a SaveDialog
- umět vytvořit jednoduchý textový editor
- umět základní operace třídy TString

Časová náročnost lekce: 3 hodiny

Vstupní test

Testové otázky a úkoly vstupního testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.



1. Která z následujících vlastností objektu Label slouží pro nastavení zalomení textu na další řádek?
 - BreakLine
 - WordWrap
 - Caption
 - Enter
2. Který z následujících příkazů zajistí, že objekt BtnLeve nebude na formuláři po spuštění programu vidět?
 - BtnLeve.Enabled:=true;
 - BtnLeve.Visible:=true;
 - BtnLeve.Enabled:=false;
 - BtnLeve.Visible:=false;
3. Který z následujících příkazů zajistí, že objekt BtnLeve nebude na formuláři po spuštění programu aktivní?
 - BtnLeve.Enabled:=true;
 - BtnLeve.Visible:=true;
 - BtnLeve.Enabled:=false;
 - BtnLeve.Visible:=false;

Memo

Grafická komponenta Memo slouží podobně jako Edit pro zápis. Na rozdíl od Edit umožňuje využívat více řádků textu. Veškerý text v komponentě má stejný styl i velikost fontu!

Některé vlastnosti objektu Memo:

Name - jméno objektu

Lines - vlastnost, která uchovává posloupnost zapsaných znaků. Hodnota této vlastnosti je typu *TStrings*.

Font - nastavení písma v editovacím poli

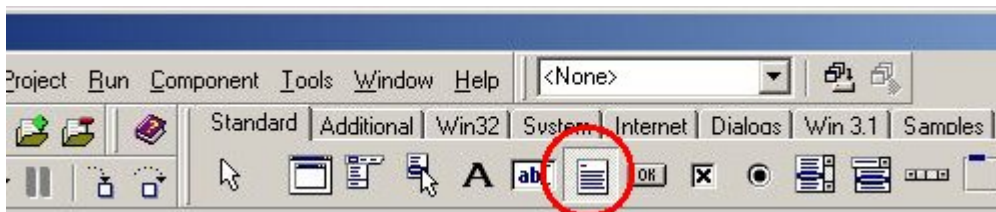
MaxLength - číslo udává maximální počet znaků v textu, které lze zapsat

ReadOnly - má hodnoty *true* nebo *false* a určuje, zda bude moci uživatel jen text číst nebo i zapisovat

Enabled - má hodnoty *true* nebo *false* a určuje, zda bude komponenta aktivní

Visible - má hodnoty *true* nebo *false* a určuje, zda bude komponenta na formuláři po spuštění programu vidět či nikoliv

WordWrap - má hodnoty *true* nebo *false* a určuje, zda bude text v rámci prostoru komponenty zalomen či zda bude na jednom řádku



Kontrolní úkol:

Prohlédněte si v Helpu metody třídy *TStrings*. Podívejte se například na metody: *Add*, *Clear*, *Delete*, *Get*, *Insert*, *LoadFromFile*, *SaveToFile*, *Move*

OpenDialog

Komponenta OpenDialog slouží k otevření dialogového okna, které slouží pro otevření určitého souboru. Přítomnost komponenty po spuštění programu není ve formuláři vidět a projeví se až po provedení určité akce. Komponenta se nachází na záložce *Dialogs*.



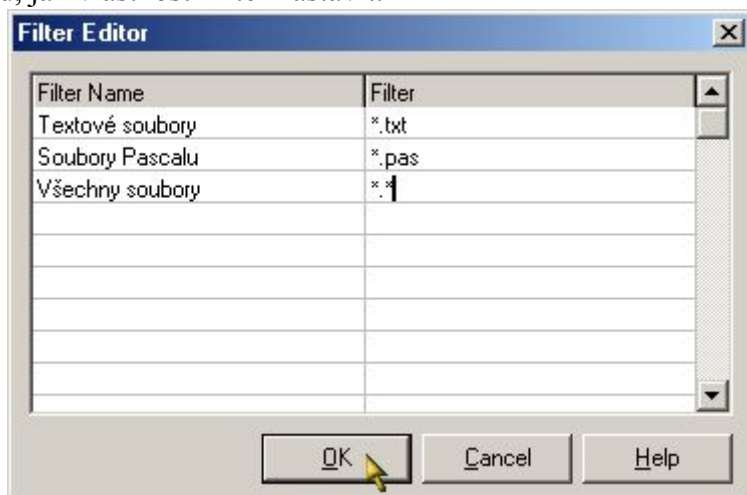
Některé vlastnosti objektu OpenDialog:

Name - jméno objektu

DefaultExt - zde napíšeme znaky implicitní přípony souboru

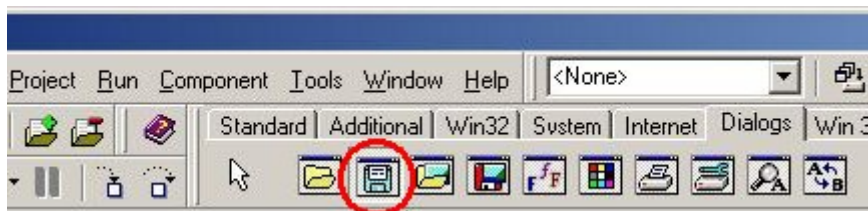
FileName - jméno souboru, který v dialogu vybereme

Filter - nastavení filtru umožní zobrazit jen soubory určitého typu. Na obrázku vidíte příklad, jak vlastnost *Filter* nastavit.



SaveDialog

Komponenta OpenDialog slouží k otevření dialogového okna, které slouží pro otevření určitého souboru. Přítomnost komponenty po spuštění programu není ve formuláři vidět a projeví se až po provedení určité akce. Komponenta se nachází na záložce *Dialogs*.



Některé vlastnosti objektu OpenDialog:

Name - jméno objektu

DefaultExt - zde napíšeme znaky implicitní přípony souboru. Uživatel při ukládání nebude muset psát název souboru včetně přípony. Ta se doplní automaticky podle hodnoty *DefaultExt*.

FileName - jméno souboru, který v dialogu vybereme. Vyplněním této vlastnosti se uživateli při otevření dialogu nabídne název, který do *FileName* zapíšeme.
Filter - nastavení filtru umožní zobrazit jen soubory určitého typu



Kontrolní úkol:

Prohlédněte si další vlastnosti komponent OpenFileDialog a SaveDialog.

Projekt č.7 – Jednoduchý textový editor

Vytvořte program, který bude pracovat jako velmi jednoduchý textový editor - trochu jako Poznámkový blok s MS Windows.

Nejprve si určíme, co chceme, aby náš editor poskytoval za funkce:

1. Umožní načíst libovolný textový soubor a zobrazit jej v objektu typu Memo.
2. Libovolný text, který vytvoříme, uloží do zvoleného souboru.
3. Bude umět měnit styl písma (tučné, kurzíva, podtržené). Samozřejmě pro všechny znaky v Memu najednou.
4. Bude nabízet volbu zalamování řádků.

Při tvorbě tohoto programu postupujte po jednotlivých krocích. Nesnažte se editor naprogramovat najednou. Rozložte si řešení na několik částí.

Kontrolní úkol:

Prohlédněte si možnosti, které nabízí aplikace "Poznámkový blok" v MS Windows a rozhodněte, které z funkcí jsme schopni s dosavadními znalostmi naprogramovat.



Rozbor programu

1. Vytvořte si složku (adresář), do kterého budeme ukládat všechny soubory projektu.
2. Spust'te Delphi.
3. Pojmenujte formulář *FrmEditori*. Upravte jeho velikost. Nastavte vlastnost *Caption* na text "Jednoduchý editor textů".
4. Umístěte na plochu všechny potřebné komponenty a pojmenujte je: `TFrmEditor = class(TForm)`

```
    MemoEditor: TMemo;  
    MainMenu1: TMainMenu;  
    mmSoubor: TMenuItem;  
    mmOtevri: TMenuItem;  
    mmUloz: TMenuItem;  
    N1: TMenuItem;  
    mmKonec: TMenuItem;  
    mmPismo: TMenuItem;  
    mmTucne: TMenuItem;  
    mmKurziva: TMenuItem;  
    mmPodtrzene: TMenuItem;  
    OpenDialog1: TOpenDialog;  
    SaveDialog1: TSaveDialog;  
    mmNovy: TMenuItem;  
    N2: TMenuItem;  
    mmZalomit: TMenuItem;
```

```
    . . .
```

```
end;
```

Poznámka: N1 a N2 jsou oddělovače mezi položkami menu. Vytváří se tak, že do vlastnosti *Caption* napíšete jeden znak mínus '-'. Vlastnost *Name* nevyplňujeme.

5. Ošetřete jednotlivé události:

```
    procedure mmTucneClick(Sender: TObject);  
    procedure mmKurzivaClick(Sender: TObject);  
    procedure mmPodtrzeneClick(Sender: TObject);  
    procedure mmKonecClick(Sender: TObject);  
    procedure mmOtevriClick(Sender: TObject);  
    procedure mmUlozClick(Sender: TObject);
```

```

procedure mmNovyClick(Sender: TObject);
procedure mmZalomitClick(Sender: TObject);
procedure FormCanResize(Sender: TObject; var NewWidth,
    NewHeight: Integer; var Resize: Boolean);

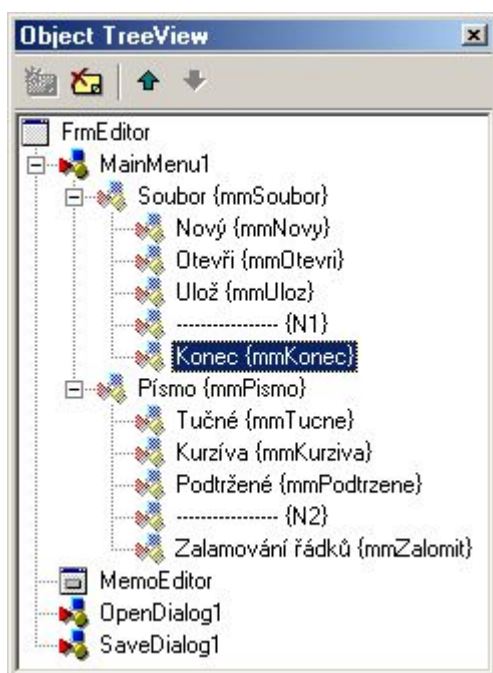
```

Poslední procedura zajišťuje změnu velikosti objektu *MemoEditor* podle změn velikosti formuláře - okna programu.

6. Uložte všechny soubory do připravené složky a spusťte program.

Objekty ve formuláři

Podívejte se na obrázek okna *Object TreeView*, ve kterém jsou vidět jednotlivé komponenty - objekty a jejich vzájemný vztah.



- MainMenu1 - hlavní nabídka:
- Položky Soubor a Písmo jsou vidět v horní části spuštěného programu.

Pod položkou Soubor budou nabídky:

- Nový - smaže obsah objektu MemoEditor
- Otevři - otevře dialogové okno pro otevření souboru
- Ulož - otevře dialogové okno pro uložení vytvořeného souboru
- -{N1} - oddělovač položek
- Konec - ukončí aplikaci. Z důvodů zjednodušení nebude program vytvářet dialog, který v případě změn v souboru se ptá na to, zda uživatel chce soubor uložit.

Pod položkou Písmo budou nabídky:

- Tučné - změni veškerý text v *MemoEditor* na tučný
- Kurzíva - změni veškerý text v *MemoEditor* na kurzivu
- Podtržené - změni veškerý text v *MemoEditor* na podtržené
- -{N2} - oddělovač položek
- Zalamování řádků - zaškrtnutí položky zalomí řádky podle velikosti okna

- MemoEditor - objekt, do kterého zapisujeme text nebo ve kterém se objeví obsah načteného souboru

- OpenDialog1 - zajistí otevření dialogového okna pro otevření souboru

- SaveDialog1 - zajistí otevření dialogového okna pro uložení souboru

Položky nabídky Soubor

Pod položkou soubor budou nabídky:

- Nový - smaže obsah objektu MemoEditor

```
procedure TFrmEditor.mmNovyClick(Sender: TObject);
begin
    {Vymaže obsah Mema}
    MemoEditor.Clear;
end;
```

- Otevři - otevře dialogové okno pro otevření souboru

```
procedure TFrmEditor.mmOtevriClick(Sender: TObject);
begin
    if (OpenDialog1.Execute) then
        MemoEditor.Lines.LoadFromFile(OpenDialog1.FileName);
end;
```

- Ulož - otevře dialogové okno pro uložení vytvořeného souboru

```
procedure TFrmEditor.mmUlozClick(Sender: TObject);
begin
    if (SaveDialog1.Execute) then
        MemoEditor.Lines.SaveToFile(SaveDialog1.FileName);
end;
```

- -{N1} - oddělovač položek

- Konec - ukončí aplikaci. Z důvodů zjednodušení nebude program vytvářet dialog, který v případě změn v souboru se ptá na to, zda uživatel chce soubor uložit.

```
procedure TFrmEditor.mmKonecClick(Sender: TObject);
begin
    Application.Terminate;
end;
```

Položky nabídky Písmo

Pod položkou Písmo budou nabídky:

- Tučný - změni veškerý text v *MemoEditor* na tučný

```
procedure TFrmEditor.mmTucneClick(Sender: TObject);
begin
    {změna písma na tučné}
    mmTucne.checked:=not (mmTucne.checked);
    if (mmTucne.checked=true) then
        MemoEditor.Font.Style := MemoEditor.Font.Style + [fsBold]
    else
        MemoEditor.Font.Style := MemoEditor.Font.Style - [fsBold];
end;
```

- Kurzíva - změni veškerý text v *MemoEditor* na kurzívu

```
procedure TFrmEditor.mmKurzivaClick(Sender: TObject);
begin
    {změna písma na kurzívu}
    mmKurziva.checked:=not (mmKurziva.checked);
    if (mmKurziva.checked=true) then
```

```

MemoEditor.Font.Style := MemoEditor.Font.Style + [fsItalic]
else
MemoEditor.Font.Style := MemoEditor.Font.Style - [fsItalic];
end;

```

- **Podtržené** - změni veškerý text v *MemoEditor* na podtržené

```

procedure TFrmEditor.mmPodtrzeneClick(Sender: TObject);
begin
  {změna písma na podtržené}
  mmPodtrzene.checked:=not(mmPodtrzene.checked);
  if (mmPodtrzene.checked=true) then
    MemoEditor.Font.Style := MemoEditor.Font.Style +
[fsUnderline]
  else
    MemoEditor.Font.Style := MemoEditor.Font.Style -
[fsUnderline]
end;

```

- **--{N2}** - oddělovač položek

- **Zalamování řádků** - zaškrnutí položky zalomí řádky podle velikosti okna

```

procedure TFrmEditor.mmZalomitClick(Sender: TObject);
begin
  mmZalomit.checked:=not(mmZalomit.checked);{zaškrnutí položky}
  MemoEditor.WordWrap:=mmZalomit.checked;{nastaví dělení řádků}
end;

```

Aby náš editor nemusel mít neustále stejnou velikost formuláře a v něm umístěného objektu typu Memo, je potřeba zajistit změnu obou objektů. Změnu velikosti můžeme například pomocí ošetření události objektu *FrmEditor* s názvem *OnCanResize*.

```

procedure TFrmEditor.FormCanResize(Sender: TObject; var NewWidth,
  NewHeight: Integer; var Resize: Boolean);
begin
  {úprava komponenty MemoEditor podle velikosti formuláře}
  MemoEditor.Height:= NewHeight - 48;
  MemoEditor.Width:= NewWidth - 10;
end;

```



Soubor editor.dpr

```

program editor;

uses
  Forms,
  hlavni in 'hlavni.pas' {FrmEditor};

{$R *.res}

begin
  Application.Initialize;
  Application.Title := 'Jednoduchý editor textů';
  Application.CreateForm(TFrmEditor, FrmEditor);
  Application.Run;
end.

```

Soubor hlavni.pas

```
// Příklady v Delphi
// Mgr. Rostislav Fojtík, 2002
// odladěno v Borland Delphi 6
// Jednoduchý textový editor

unit hlavni;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms,
  Dialogs, Menus, StdCtrls;

type
  TFrmEditor = class(TForm)
    MemoEditor: TMemo;
    MainMenu1: TMainMenu;
    mmSoubor: TMenuItem;
    mmOtevri: TMenuItem;
    mmUloz: TMenuItem;
    N1: TMenuItem;
    mmKonec: TMenuItem;
    mmPismo: TMenuItem;
    mmTucne: TMenuItem;
    mmKurziva: TMenuItem;
    mmPodtrzene: TMenuItem;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    mmNovy: TMenuItem;
    N2: TMenuItem;
    mmZalomit: TMenuItem;
    OpenDialog2: TOpenDialog;
    procedure mmTucneClick(Sender: TObject);
    procedure mmKurzivaClick(Sender: TObject);
    procedure mmPodtrzeneClick(Sender: TObject);
    procedure mmKonecClick(Sender: TObject);
    procedure mmOtevriClick(Sender: TObject);
    procedure mmUlozClick(Sender: TObject);
    procedure mmNovyClick(Sender: TObject);
    procedure mmZalomitClick(Sender: TObject);
    procedure FormCanResize(Sender: TObject; var NewWidth,
      NewHeight: Integer; var Resize: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmEditor: TFrmEditor;
```

implementation

{ \$R *.dfm }

```
procedure TFrmEditor.mmTucneClick(Sender: TObject);
begin
    {změna písma na tučné}
    mmTucne.checked:=not(mmTucne.checked);
    if (mmTucne.checked=true) then
        MemoEditor.Font.Style := MemoEditor.Font.Style + [fsBold]
    else
        MemoEditor.Font.Style := MemoEditor.Font.Style - [fsBold];
end;

procedure TFrmEditor.mmKurzivaClick(Sender: TObject);
begin
    {změna písma na kurzívu}
    mmKurziva.checked:=not(mmKurziva.checked);
    if (mmKurziva.checked=true) then
        MemoEditor.Font.Style := MemoEditor.Font.Style + [fsItalic]
    else
        MemoEditor.Font.Style := MemoEditor.Font.Style - [fsItalic];
end;

procedure TFrmEditor.mmPodtrzeneClick(Sender: TObject);
begin
    {změna písma na podtržené}
    mmPodtrzene.checked:=not(mmPodtrzene.checked);
    if (mmPodtrzene.checked=true) then
        MemoEditor.Font.Style := MemoEditor.Font.Style +
[fsUnderline]
    else
        MemoEditor.Font.Style := MemoEditor.Font.Style -
[fsUnderline]
end;

procedure TFrmEditor.mmKonecClick(Sender: TObject);
begin
    Application.Terminate;
end;

procedure TFrmEditor.mmOtevriClick(Sender: TObject);
begin
    if (OpenDialog1.Execute) then
        MemoEditor.Lines.LoadFromFile(OpenDialog1.FileName);
end;

procedure TFrmEditor.mmUlozClick(Sender: TObject);
begin
    if (SaveDialog1.Execute) then
        MemoEditor.Lines.SaveToFile(SaveDialog1.FileName);
end;

procedure TFrmEditor.mmNovyClick(Sender: TObject);
begin
    {Vymaže obsah Mema}
```



```

    MemoEditor.Clear;
end;

procedure TFrmEditor.mmZalomitClick(Sender: TObject);
begin
    mmZalomit.checked:=not (mmZalomit.checked);{zaškrnutí položky}
    MemoEditor.WordWrap:=mmZalomit.checked;{nastaví dělení řádků}
end;

procedure TFrmEditor.FormCanResize(Sender: TObject; var NewWidth,
    NewHeight: Integer; var Resize: Boolean);
begin
    {úprava komponenty MemoEditor podle velikosti formuláře}
    MemoEditor.Height:= NewHeight - 48;
    MemoEditor.Width:= NewWidth - 10;
end;

end.

```



Časté chyby

- Snaha vytvořit všechny funkce programu najednou. Rozdělte si program na jednoduché části a tvořte aplikaci po menších krocích. Neztratíte přehled ve svém kódu.
- Oddělovač mezi položkami hlavního menu se zapisuje jen pomocí jednoho znaku mínus. Nesmíte do vlastnosti *Caption* napsat více znaků mínus.



Opakovací test

Testové otázky a úkoly opakovacího testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Která z následujících vlastností umožňuje u objektu Memo změnit styl písma?
 - Font
 - FontStyle
 - Lines
 - Text
2. Kterou z následujících vlastností nemá komponenta Memo?
 - Lines
 - ReadOnly
 - Text
 - Visible
3. Která z následujících masek umožní v dialogové okně OpenFileDialog zobrazovat jen textové zdrojové soubory (unity) z Pascalu?
 - *.pas
 - *.txt
 - *.*
 - *.dcu

Shrnutí



- Grafická komponenta Memo slouží podobně jako Edit pro zápis. Na rozdíl od Edit umožňuje využívat více řádků textu. Veškerý text v komponentě má stejný styl i velikost fontu!
- Komponenta OpenFileDialog slouží k otevření dialogového okna, které slouží pro otevření určitého souboru. Přítomnost komponenty po spuštění programu není ve formuláři vidět a projeví se až po provedení určité akce. Komponenta se nachází na záložce *Dialogs*.
- Komponenta SaveDialog slouží k otevření dialogového okna, které slouží pro uložení určitého souboru. Přítomnost komponenty po spuštění programu není ve formuláři vidět a projeví se až po provedení určité akce. Komponenta se nachází na záložce *Dialogs*.
- Rozsáhlejší projekty – programy si rozdělujeme na kratší a jednodušší části a ty postupně sestavujeme.

Rejstřík

Memo

OpenDialog

SaveDialog

Tvorba projektů č.4



Cíl lekce

Cílem této lekce je seznámit se se základními postupy při tvorbě programů v Delphi na praktickém příkladu. Vytvoříme jednoduchou kalkulačku.

Po absolvování lekce budete:

- umět vytvořit jednoduchou kalkulačku
- umět převádět řetězce na celá i reálná čísla
- vytvářet pomocné proměnné a metody ve vytvořené třídě

Časová náročnost lekce: 3 hodiny



Vstupní test

Testové otázky a úkoly vstupního testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Která z vlastností objektu SaveDialog umožňuje vytvářet automaticky příponu ukládanému souboru?
 - Filter
 - DefaultExt
 - FileName
 - Tuto funkci nelze zajistit
2. Kterou z následujících zvladností nemá komponenta Memo?
 - Lines
 - ReadOnly
 - Text
 - Visible
3. Která z následujících komponent může sloužit k zápisu dat v programu?
 - Label
 - Edit
 - Buton
 - Caption

Projekt č.8 – jednoduchá kalkulačka

Vytvořte program, který bude pracovat jako velmi jednoduchý kalkulátor. Program umožní pracovat s reálnými čísly v rozsahu datového typu *real* a bude poskytovat tyto základní matematické operace: sčítání, odčítání, násobení, dělení, 1 děleno zadaným číslem, druhou mocninu a odmocninu, konstantu Pi.

Při tvorbě tohoto programu postupujte po jednotlivých krocích. Nesnažte se editor naprogramovat najednou. Rozložte si řešení na několik částí.

Tento program nebude neošetřovat vstupy. Předpokládáme, že zadávaná data jsou správně zapsaná celá nebo reálná čísla!



Kontrolní úkol:

Prohlédněte si možnosti, které nabízí aplikace "Kalkulačka" v MS Windows a rozhodněte, které z funkcí jsme schopni s dosavadními znalostmi naprogramovat.



Rozbor programu

1. Vytvořte si složku (adresář), do kterého budeme ukládat všechny soubory projektu.
2. Spusťte Delphi.
3. Pojmenujte formulář *FrmEditori*. Upravte jeho velikost. Nastavte vlastnost *Caption* na text "Jednoduchý kalkulátor".
4. Umístěte na plochu všechny potřebné komponenty a pojmenujte je:

```
TFormKalkulator = class(TForm)
    EditDisplay: TEdit;
    BtnC: TButton;
    BtnRovno: TButton;
    BtnPlus: TButton;
    BtnMinus: TButton;
    BtnKrat: TButton;
    BtnDeleno: TButton;
    BtnLomX: TButton;
    BtnSqr: TButton;
    BtnSqrt: TButton;
    BtnPi: TButton;
    . . .
end;
```

5. Ošetřete jednotlivé události:

```
procedure BtnRovnoClick(Sender: TObject);
procedure BtnPlusClick(Sender: TObject);
procedure BtnMinusClick(Sender: TObject);
procedure BtnKratClick(Sender: TObject);
```

```

procedure BtnDelenoClick(Sender: TObject);
procedure BtnLomXClick(Sender: TObject);
procedure BtnSqrClick(Sender: TObject);
procedure BtnSqrtClick(Sender: TObject);
procedure BtnCClick(Sender: TObject);
procedure BtnPiClick(Sender: TObject);

```

6. Vytvořte pomocný výčtový datový typ, který bude obsahovat symboly čísel jednotlivých matematických operací. Typ potřebujeme pro tlačítko, které vykonává výpočet.

```

TOperace=(o_nic,o_plus,o_minus,o_krat,o_deleno,o_lomx,o_sqr,o_sqrt);

```

7. V části *private* třídy *TFrmKalkulator* si vytvořte pomocné proměnné pro provádění výpočtů:

```

private
  { Private declarations }
  operand1, operand2:real;
  vysledek:real;
  operace:TOperace; {číslo matematické operace}

```

8. Uložte všechny soubory do připravené složky a spusťte program.

Zamyslíme-li se nad tím co dělá většina tlačítek vyvolávající určitou matematickou operaci, zjistíme, že část kódu se bude vždy opakovat. Již v dřívějších lekcích jsme si zdůraznili, že kód by se měl nacházet jen na jednom místě. Proto si vytvoříme pomocnou soukromou proceduru, jejíž úkolem bude přečíst text z objektu *EditDisplay*, převést jej na reálné číslo, smazat Edit a přesunout do něj kurzor.

```

TFrmKalkulator = class(TForm)
. . .
private
  { Private declarations }
  operand1, operand2:real;
  vysledek:real;
  operace:TOperace; {číslo matematické operace}
  procedure TlacitkoOperace;
public
  { Public declarations }
end;
. . .
procedure TFrmKalkulator.TlacitkoOperace;
begin
  operand1:=StrToFloat(EditDisplay.Text);
  EditDisplay.Clear;
  EditDisplay.SetFocus;
end;

```

Činnost tlačítek

Činnost jednotlivých tlačítek s matematickými operacemi se skládá z několika kroků:

1. Poznamená se číslo matematické operace pro tlačítko rovná se.
2. Zavolá se pomocná procedura, která zajistí přečtení text z objektu *EditDisplay*, jeho převedení na reálné číslo, smazání *EditDisplay* a přesunutí kurzoru.
3. U operací s jedním operandem je potřeba provést výpočet, změnit číslo na text a umístit je do objektu *EditDisplay*.

```
procedure TFrmKalkulator.BtnPlusClick(Sender: TObject);
begin
    operace:=o_plus;
    TlacidkoOperace;
end;

procedure TFrmKalkulator.BtnMinusClick(Sender: TObject);
begin
    operace:=o_minus;
    TlacidkoOperace;
end;

procedure TFrmKalkulator.BtnKratClick(Sender: TObject);
begin
    operace:=o_krat;
    TlacidkoOperace;
end;

procedure TFrmKalkulator.BtnDelenoClick(Sender: TObject);
begin
    operace:=o_deleno;
    TlacidkoOperace;
end;

procedure TFrmKalkulator.BtnLomXClick(Sender: TObject);
begin
    operace:=o_lomx;
    TlacidkoOperace;
    if (operand1 <> 0) then vysledek:= 1 / operand1
    else
        begin
            vysledek:=1;
            MessageDlg('Nulou nelze dělit!', mtWarning, [mbOK], 0);
        end;
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnSqrClick(Sender: TObject);
begin
    operace:=o_sqr;
    TlacidkoOperace;
    vysledek:=operand1 * operand1;
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
```

```

end;

procedure TFrmKalkulator.BtnSqrtClick(Sender: TObject);
begin
    operace:=o_sqrt;
    TlactkoOperace;
    vysledek:= sqrt(operand1); {druhá odmocnina}
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnCClick(Sender: TObject);
begin
    {vznuluje proměnné a zruší operaci}
    operand1:=0;
    operand2:=0;
    operace:=o_nic;
    vysledek:=0;
    EditDisplay.Text:='0';
end;

procedure TFrmKalkulator.BtnPiClick(Sender: TObject);
begin
    EditDisplay.Text:='3,1415927'; {konstant Pi}
end;

```

Tlačítko 'C' (*BtnC*) má za úkol smazat obsah komponenty *EditDisplay*, ale také vynulovat všechny pomocné proměnné.

```

procedure TFrmKalkulator.BtnCClick(Sender: TObject);
begin
    {vznuluje proměnné a zruší operaci}
    operand1:=0;
    operand2:=0;
    operace:=o_nic;
    vysledek:=0;
    EditDisplay.Text:='0';
end;

```

Tlačítko 'Pi' (*BtnPi*) má za úkol umístit do komponenty *EditDisplay* hodnotu konstanty Pi.

```

procedure TFrmKalkulator.BtnPiClick(Sender: TObject);
begin
    EditDisplay.Text:='3,1415927'; {konstant Pi}
end;

```

Činnost tlačítka '=' je založena na dokončení operace se dvěma operandy. Podle čísla operace (proměnná *operace*) provede výpočet, výsledek převede na reálné číslo a umístí jej do objektu *EditDisplay*. U operace dělení nezapomeňte ošetřit dělení nulou. Použijeme k tomu vyvolání objektu *MessageDlg*, tedy upozorňujícího dialogového okna.

```

procedure TFrmKalkulator.BtnRovnoClick(Sender: TObject);
begin
    operand2:=StrToFloat(EditDisplay.Text);
    if (operace=o_plus) then vysledek:=operand1 + operand2;
    if (operace=o_minus) then vysledek:=operand1 - operand2;

```



```

if (operace=o_krat) then vysledek:=operand1 * operand2;
if (operace=o_deleno) then
  if (operand2 <> 0) then vysledek:=operand1 / operand2
  else
    begin
      vysledek:=operand1;
      MessageDlg('Nulou nelze dělit!', mtWarning, [mbOK], 0);
    end;
EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

```

Pro převody řetězců na čísla a obráceně nabízí Object Pascal několik základních podprogramů. Podívejte se do nápovědy na jejich činnost a použité parametry.

IntToStr - převede celé číslo na řetězec

StrToInt - převede řetězec na celé číslo

FloatToStr - převede reálné číslo na řetězec ve formátu s exponentem

FloatToStrF - převede reálné číslo na řetězec ve formátu, který je určen pomocí parametrů podprogramu

StrToFloat - převede řetězec na reálné číslo



Soubor kalkul.dpr

```
program kalkul;  
  
uses  
    Forms,  
    hlavni in 'hlavni.pas' {FrmKalkulator};  
  
{$R *.res}  
  
begin  
    Application.Initialize;  
    Application.Title := 'Jednoduchý kalkulátor';  
    Application.CreateForm(TFrmKalkulator, FrmKalkulator);  
    Application.Run;  
end.
```

Soubor hlavni.pas

```
// Příklady v Delphi  
// Mgr. Rostislav Fojtík, 2002  
// odladěno v Borland Delphi 6  
  
{  
    Tento program neošetřuje vstupy. Předpokládáme, že zadávaná  
data  
    jsou správně zapsaná celá nebo reálná čísla!  
}  
  
unit hlavni;  
  
interface  
  
uses  
    Windows, Messages, SysUtils, Variants, Classes, Graphics,  
    Controls, Forms,  
    Dialogs, StdCtrls;  
  
type  
  
TOperace=(o_nic,o_plus,o_minus,o_krat,o_deleno,o_lomx,o_sqr,o_sqr  
t);  
TFrmKalkulator = class(TForm)  
    EditDisplay: TEdit;  
    BtnC: TButton;  
    BtnRovno: TButton;  
    BtnPlus: TButton;  
    BtnMinus: TButton;  
    BtnKrat: TButton;  
    BtnDeleno: TButton;  
    BtnLomX: TButton;  
    BtnSqr: TButton;  
    BtnSqrt: TButton;  
    BtnPi: TButton;  
    procedure BtnRovnoClick(Sender: TObject);  
    procedure BtnPlusClick(Sender: TObject);
```

```

    procedure BtnMinusClick(Sender: TObject);
    procedure BtnKratClick(Sender: TObject);
    procedure BtnDelenoClick(Sender: TObject);
    procedure BtnLomXClick(Sender: TObject);
    procedure BtnSqrClick(Sender: TObject);
    procedure BtnSqrtClick(Sender: TObject);
    procedure BtnCClick(Sender: TObject);
    procedure BtnPiClick(Sender: TObject);
private
    { Private declarations }
    operand1, operand2:real;
    vysledek:real;
    operace:Toperace; {číslo matematické operace}
    procedure TlacitkoOperace;

public
    { Public declarations }
end;

var
    FrmKalkulator: TFrmKalkulator;

implementation

{$R *.dfm}

procedure TFrmKalkulator.TlacitkoOperace;
begin
    operand1:=StrToFloat(EditDisplay.Text);
    EditDisplay.Clear;
    EditDisplay.SetFocus;
end;

procedure TFrmKalkulator.BtnRovnoClick(Sender: TObject);
begin
    operand2:=StrToFloat(EditDisplay.Text);
    if (operace=o_plus) then
        vysledek:=operand1 + operand2;
    if (operace=o_minus) then
        vysledek:=operand1 - operand2;
    if (operace=o_krat) then
        vysledek:=operand1 * operand2;
    if (operace=o_deleno) then
        if (operand2 <> 0) then vysledek:=operand1 / operand2
        else
            begin
                vysledek:=operand1;
                MessageDlg('Nulou nelze dělit!', mtWarning, [mbOK], 0);
            end;

    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnPlusClick(Sender: TObject);
begin
    operace:=o_plus;

```

```

    TlactitkoOperace;
end;

procedure TFrmKalkulator.BtnMinusClick(Sender: TObject);
begin
    operace:=o_minus;
    TlactitkoOperace;
end;

procedure TFrmKalkulator.BtnKratClick(Sender: TObject);
begin
    operace:=o_krat;
    TlactitkoOperace;
end;

procedure TFrmKalkulator.BtnDelenoClick(Sender: TObject);
begin
    operace:=o_deleno;
    TlactitkoOperace;
end;

procedure TFrmKalkulator.BtnLomXClick(Sender: TObject);
begin

    operace:=o_lomx;
    TlactitkoOperace;
    if (operand1 <> 0) then vysledek:=1 / operand1
    else
        begin
            vysledek:=1;
            MessageDlg('Nulou nelze dělit!', mtWarning, [mbOK], 0);
        end;
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnSqrClick(Sender: TObject);
begin
    operace:=o_sqr;
    TlactitkoOperace;
    vysledek:=operand1 * operand1;
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnSqrtClick(Sender: TObject);
begin
    operace:=o_sqrt;
    TlactitkoOperace;
    vysledek:= sqrt(operand1); {druhá odmocnina}
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnCClick(Sender: TObject);
begin

```

```

    {vznuluje proměnné a zruší operaci}
    operand1:=0;
    operand2:=0;
    operace:=o_nic;
    vysledek:=0;
    EditDisplay.Text:='0';
end;

procedure TFrmKalkulator.BtnPiClick(Sender: TObject);
begin
    EditDisplay.Text:='3,1415927';    {konstant Pi}
end;

end.

```



Časté chyby

- Pomocné proměnné definujete jako globální proměnné. K tomu není žádný důvod. Uvědomme si, že globální, případně veřejné proměnné můžeme měnit libovolným způsobem, neboť k nim není žádným způsobem omezen přístup. Mnohem vhodnější je všechny pomocné proměnné i metody, které se využívají jen uvnitř třídy, definovat jako soukromé prvky třídy!
- Vytváříte stejný kód v různých místech programu. Potřebujete-li mít stejný kód v různých místech programu, vytvořte si pomocnou (nejlépe soukromou) metodu, kterou pak můžete volat v ostatních metodách.
- Při definování pomocné procedury, která je součástí třídy, zapomínáte v hlavičce uvést jméno třídy.

```
procedure TlactitkoOperace;
begin
    operand1:=StrToFloat(EditDisplay.Text);
    EditDisplay.Clear;
    EditDisplay.SetFocus;
end;
```

Procedura pak nepatří třídě a je zcela samostatným podprogramem. V tom případě však překladač zahlásí následující chyby:

- nemá definici metody *TlactitkoOperace* ze třídy *TFrmKalkulator*
- samostatná procedura *TlactitkoOperace* nemůže přímo pracovat s proměnnou *operand1*, neboť se jedná o soukromou položku třídy *TFrmKalkulator*

Správný zápis má před jménem metody ještě název třídy:

```
procedure TFrmKalkulator.TlactitkoOperace;
begin
    operand1:=StrToFloat(EditDisplay.Text);
    EditDisplay.Clear;
    EditDisplay.SetFocus;
end;
```



Opakovací test

Testové otázky a úkoly opakovacího testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Která z následujících operací převede řetězec (text) na reálné číslo?
 - FloatToStr
 - StrToFloat
 - IntToStr
 - StrToInt
2. Která z následujících operací převede reálné číslo na řetězec (text) v určitém formátu?
 - FloatToStrF
 - StrToFloat
 - IntToStr
 - FlotToStr

Shrnutí



- Pro větší přehlednost i bezpečnost programů je vhodné všechny pomocné proměnné a podprogramy vytvářet jako soukromé prvky definované třídy.
- Pro převody řetězců na čísla a obráceně nabízí Object Pascal několik základních podprogramů. Podívejte se do nápovědy na jejich činnost a použité parametry.
IntToStr - převede celé číslo na řetězec
StrToInt - převede řetězec na celé číslo
FloatToStr - převede reálné číslo na řetězec ve formátu s exponentem
FloatToStrF - převede reálné číslo na řetězec ve formátu, který je určen pomocí parametrů podprogramu
StrToFloat - převede řetězec na reálné číslo

Rejstřík

private
protected
published
public
soukromé prvky třídy

Tvorba projektů č.5



Cíl lekce

Cílem této lekce je seznámit se se základními postupy při tvorbě programů v Delphi na praktickém příkladu. Naprogramujeme kalkulačku, která bude vycházet z minulého příkladu, ale navíc do ní nadefinujeme některé vlastní funkce.

Po absolvování lekce budete:

- umět vytvořit jednoduchou kalkulačku
- umět převádět řetězce na celá i reálná čísla
- vytvářet pomocné proměnné a metody ve vytvořené třídě
- umět měnit velikost okna za chodu programu
- umět definovat vlastní unitu

Časová náročnost lekce: 3 hodiny



Vstupní test

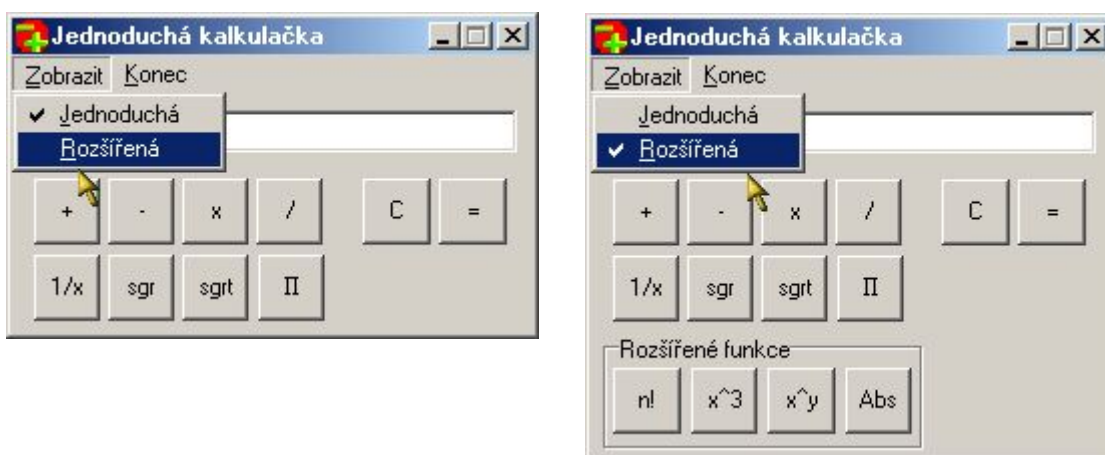
Testové otázky a úkoly vstupního testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Ve které části třídy je nejméně vhodné definovat pomocné proměnné?
 - var (globální proměnné)
 - public
 - private
 - nehraje roli
2. Která z následujících odpovědí je správně zapsaná hlavička metody Vypocet třídy TKalkul při definici metody?
 - procedure TKalkul::Vypocet;
 - procedure TKalkul.Vypocet;
 - procedure Vypocet;
 - procedure Vypocet.TKalkul;
3. Kterou z následujících vlastností nemá komponenta Edit?
 - Name
 - Font
 - ReadOnly
 - Caption

Projekt č.9 – kalkulačka

Vytvoříme program, který bude pracovat jako jednoduchý kalkulátor. Vyjdeme z minulého příkladu a rozšíříme možnosti kalkulačky o další funkce, které sami naprogramujeme.

Tento program nebude neošetřovat vstupy. Předpokládáme, že zadávaná data jsou správně zapsaná celá nebo reálná čísla!



Kontrolní úkol:

Prohlédněte si možnosti, které nabízí aplikace "Kalkulačka" v MS Windows a rozhodněte, které z funkcí jsme schopni s dosavadními znalostmi naprogramovat.



Rozbor programu

1. Vytvořte si složku (adresář), do kterého budeme ukládat všechny soubory projektu.
2. Spusťte Delphi. Otevřete dříve vytvořený projekt, ve kterém jsme vytvářeli jednoduchou kalkulačku. Uložte vše jako nový projekt do nové složky.
3. Zvětšete formulář *FrmEditori* tak, aby se do něj vešla nová tlačítka. Ty umístěte do komponenty *GroupBox*.
4. Umístěte na plochu všechny potřebné komponenty a pojmenujte je:

```
TfrmKalkulator = class(TForm)
    EditDisplay: TEdit;
    BtnC: TButton;
    BtnRovno: TButton;
    BtnPlus: TButton;
    BtnMinus: TButton;
    BtnKrat: TButton;
    BtnDeleno: TButton;
    BtnLomX: TButton;
    BtnSqr: TButton;
    BtnSqrt: TButton;
    BtnPi: TButton;
    GroupBoxRozFunkce: TGroupBox;
    BtnFakt: TButton;
    MainMenu1: TMainMenu;
    mmZobrazit: TMenuItem;
    mmJednoducha: TMenuItem;
```

```

mmRozsirena: TMenuItem;
mmKonec: TMenuItem;
BtnNaTreti: TButton;
BtnXnaY: TButton;
BtnAbs: TButton;
. . .
end;

```

5. Ošetřete jednotlivé události:

```

procedure BtnRovnoClick(Sender: TObject);
procedure BtnPlusClick(Sender: TObject);
procedure BtnMinusClick(Sender: TObject);
procedure BtnKratClick(Sender: TObject);
procedure BtnDelenoClick(Sender: TObject);
procedure BtnLomXClick(Sender: TObject);
procedure BtnSqrClick(Sender: TObject);
procedure BtnSqrtClick(Sender: TObject);
procedure BtnCClick(Sender: TObject);
procedure BtnPiClick(Sender: TObject);
procedure mmKonecClick(Sender: TObject);
procedure mmJednoduchaClick(Sender: TObject);
procedure mmRozsirenaClick(Sender: TObject);
procedure BtnFaktClick(Sender: TObject);
procedure BtnNaTretiClick(Sender: TObject);
procedure BtnXnaYClick(Sender: TObject);
procedure BtnAbsClick(Sender: TObject);

```

6. Uložte všechny soubory do připravené složky a spusťte program.

Komponenta GroupBox

Grafická komponenta *GroupBox* slouží k vymezení a sjednocení několika objektů v rámci formuláře.

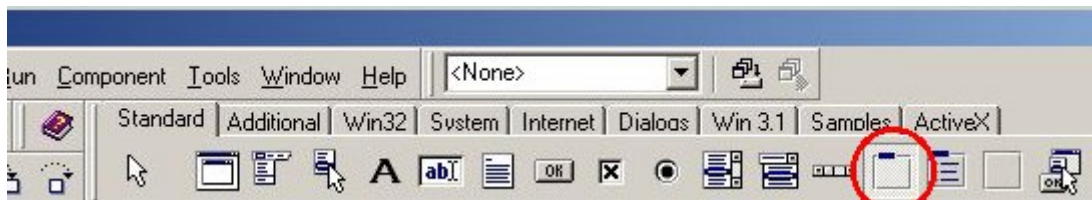
Některé vlastnosti komponenty *GroupBox*:

Name - jméno objektu

Caption - titulek

Color - barva plochy objektu

Enabled, *Visible*, *Font*, *Hint*, *ShowHint* ...



Chceme-li umístit nějaký objekt (například tlačítko) do *GroupBoxu*, je potřeba nejprve *GroupBox* označit a teprve pak tlačítko do něj vložit. Tlačítko pak již nejde vysunout z ohraničení. Přesunem *GroupBoxu* pak hýbeme všemi objekty v něm umístěnými.

Činnost tlačítek

Činnost tlačítek se základními operacemi již popisovat nebudeme. Jejich popis jsme provedli v minulé lekci. Nyní se podíváme na nové tlačítka. Tlačítko má za úkol:

1. Poznamená se číslo matematické operace pro tlačítko rovná se.
2. Zavolá se pomocná procedura, která zajistí přečtení text z objektu *EditDisplay*, jeho převedení na reálné číslo, smazání *EditDisplay* a přesunutí kurzoru.
3. U operací s jedním operandem je potřeba provést výpočet, změnit číslo na text a umístit je do objektu *EditDisplay*.

```
procedure TFrmKalkulator.BtnFaktClick(Sender: TObject);
begin
    operace:=o_fakt;
    TlactkoOperace;
    vysledek:=Faktorial(Round(operand1));
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnNaTretiClick(Sender: TObject);
begin
    operace:=o_natreti;
    TlactkoOperace;
    vysledek:=XnaTreti(operand1);
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnXnaYClick(Sender: TObject);
begin
    operace:=o_nay;
    TlactkoOperace;
end;
```

```

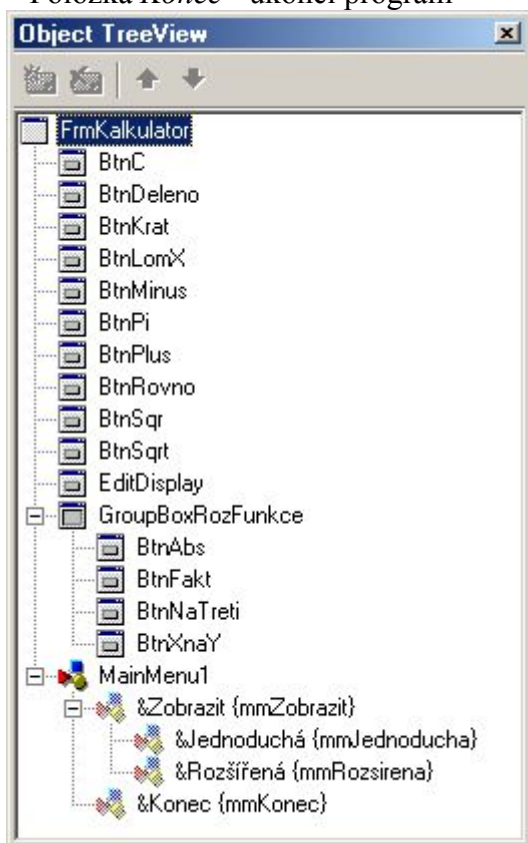
procedure TFrmKalkulator.BtnAbsClick(Sender: TObject);
begin
    operace:=o_abs;
    TlactkoOperace;
    vysledek:=Abs(operand1);
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

```

Seznam objektů

Prohlédněte si okno *Object TreeView* a doplňte podle něj všechny grafické komponenty a vytvořte nabídku hlavního menu.

- Položka zobrazit bude umožňovat výběr na jednoduchou a rozšířenou kalkulačku
- Položka *Jednoduchá* - vybírá menší okno s méně matematickými funkcemi. Procedura musí zmenšit okno, nastavit vlastnost *Checked*, schovat rozšířené funkce.
- Položka *Rozšířená* - vybírá větší okno s více matematickými funkcemi. Procedura musí zvětšit okno, nastavit vlastnost *Checked*, zobrazit rozšířené funkce.
- Položka *Konec* - ukončí program



Všimněte si, že objekty *BtnAbs*, *BtnFakt*, *BtnNaTreti*, *BtnXnaY* jsou skutečně součástí objektu *GroupBoxRozFunkce*!

- Položka *Jednoduchá* - vybírá menší okno s méně matematickými funkcemi. Procedura musí zmenšit okno, nastavit vlastnost *Checked*, schovat rozšířené funkce.

```

procedure TFrmKalkulator.mmJednoduchaClick(Sender: TObject);
begin
    if (mmJednoducha.Checked<>true) then

```

```

begin
    mmJednoducha.Checked:=true;
    mmRozsirena.Checked:=false;
    FrmKalkulator.Height:=165;
    //FrmKalkulator.Width:=268;
    GroupBoxRozFunkce.Visible:=false;
end;

```

end;

- **Položka *Rozšířená*** - vybírá větší okno s více matematickými funkcemi. Procedura musí zvětšit okno, nastavit vlastnost *Checked*, zobrazit rozšířené funkce.

```

procedure TFrmKalkulator.mmRozsirenaClick(Sender: TObject);

```

```

begin

```

```

    if (mmRozsirena.Checked<>true) then

```

```

    begin

```

```

        mmRozsirena.Checked:=true;

```

```

        mmJednoducha.Checked:=false;

```

```

        FrmKalkulator.Height:=225;

```

```

        //FrmKalkulator.Width:=268;

```

```

        GroupBoxRozFunkce.Visible:=true;

```

```

    end;

```

end;

- **Položka *Konec*** - ukončí program

```

procedure TFrmKalkulator.mmKonecClick(Sender: TObject);

```

```

begin

```

```

    Application.Terminate;

```

```

end;

```



Vlastní unita

Pokud si pozorně prohlédnete procedury z minulé stránky, zjistíte, že se zde používají funkce, které nemáme zatím definovány. Proto vytvoříme novou unitu (Hlavní nabídka *File - New - Unit*), ve které si nadefinujeme jednotlivé matematické funkce a soubor uložíme pod jménem *Funkce.pas*.

```
unit Funkce;

interface
  function Faktorial(n:integer):real;
  function Abs(n:real):real;
  function XnaTreti(n:real):real;
  function XnaY(x,y:integer):real;

implementation
  function Faktorial(n:integer):real;
  var i:integer;
      vys:real;
  begin
    vys:=1;
    for i:=1 to n do
      vys:=vys * i;
    Faktorial := vys;
  end;

  function Abs(n:real):real;
  var vys:real;
  begin
    if (n<0) then vys:=n * (-1)
    else vys:=n;
    Abs:=vys;
  end;

  function XnaTreti(n:real):real;
  var vys:real;
  begin
    vys:=n;
    vys:=vys * n * n;
    XnaTreti:=vys;
  end;

  function XnaY(x,y:integer):real;
  var vys:real;
      i:integer;
  begin
    vys:=1;
    for i:=1 to Round(Abs(y)) do
      vys:= vys * x;
    if (y<1) then vys:= 1 / vys;
    XnaY:=vys;
  end;

end.
```

Abychom mohli vytvořené matematické funkce využívat v unitě *Hlavni* je potřeba do části **uses** připojit jméno naší nové unity:

```
unit hlavni;
```

```
interface
```

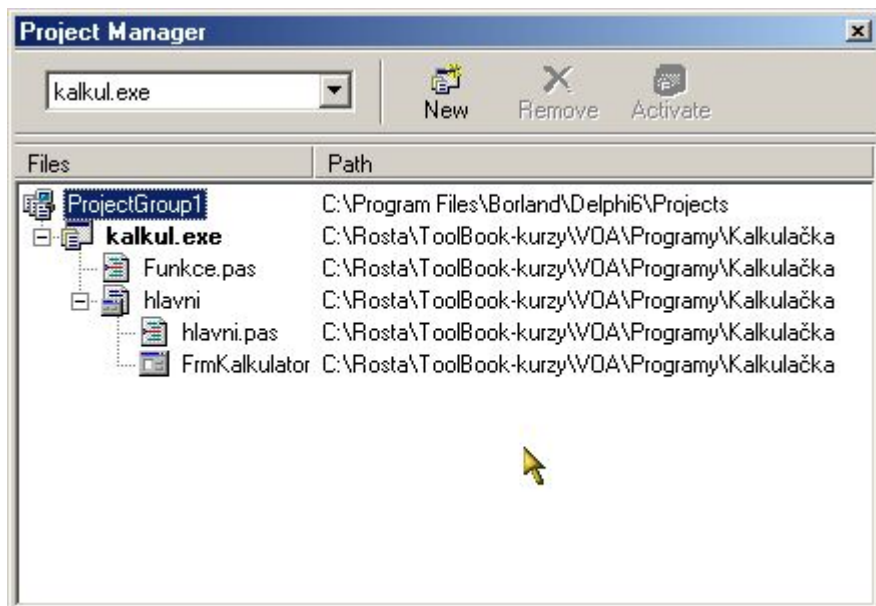
```
uses
```

```
    Windows, Messages, SysUtils, Variants, Classes, Graphics,  
    Controls, Forms, Dialogs, StdCtrls, Menus, Funkce;  
    {nezapomeňte připojit unitu s funkcemi}
```

```
...  
end.
```

Project Manager

Abychom v rozsáhlejších projektech - programech neztratili v jednotlivých souborech a formulářích přehled, nabízí Delphi nástroj *Project Manager*. Spustíme jej přes nabídku *View*, volba *Project Manager* nebo přímo stiskem kláves **Ctrl+Alt+F11**. V okně pak je vidět přehled všech unit a k nim případných přidaných formulářů. Zavřeme-li si některý z formulářů nebo ze zdrojových souborů unit, stačí dvojklikem příslušné okno vyvolat zpět.



Ikona a jméno programu

Nástroj Borland Delphi umožňuje připojit k vašemu programu vámi vybranou nebo nakreslenou ikonu. Ikonu si můžete nakreslit například pomocí nástroje *Image Editor*, který se instaluje společně s Borland Delphi. Ikonku pak připojíte v Delphi spuštěním volby *Options* v nabídce *Project*, tlačítko *Load Icon..*.

V položce *Title* napíšete jméno programu.



Soubor kalkul.dpr



```
program kalkul;  
  
uses  
    Forms,  
    hlavni in 'hlavni.pas' {FrmKalkulator},  
    Funkce in 'Funkce.pas';  
  
{$R *.res}  
  
begin  
    Application.Initialize;  
    Application.Title := 'Jednoduchý kalkulátor';  
    Application.CreateForm(TFrmKalkulator, FrmKalkulator);  
    Application.Run;  
end.
```

Pokud si pozorně prohlédnete projektový soubor *kalkul.dpr* zjistíte, že se rozrostl v části **uses**. V té přibyl řádek určující použití další unity *Funkce*, která je v souboru *Funkce.pas*.

Soubor hlavni.pas

```
// Příklady v Delphi  
// Mgr. Rostislav Fojtík, 2002  
// odladěno v Borland Delphi 6  
  
{  
    Tento program neošetřuje vstupy. Předpokládáme, že zadávaná  
data  
    jsou správně zapsaná celá nebo reálná čísla!  
}  
  
unit hlavni;  
  
interface  
  
uses  
    Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,  
    Dialogs, StdCtrls, Menus, Funkce;  
    {nezapomeňte připojit unitu s funkcemi}  
type  
  
TOperace=(o_nic,o_plus,o_minus,o_krat,o_deleno,o_lomx,o_sqr,o_sqr  
t,o_fakt,  
    o_natrei, o_nay, o_abs);  
TFrmKalkulator = class(TForm)  
    EditDisplay: TEdit;  
    BtnC: TButton;  
    BtnRovno: TButton;  
    BtnPlus: TButton;  
    BtnMinus: TButton;
```

```

    BtnKrat: TButton;
    BtnDeleno: TButton;
    BtnLomX: TButton;
    BtnSqr: TButton;
    BtnSqrt: TButton;
    BtnPi: TButton;
    GroupBoxRozFunkce: TGroupBox;
    BtnFakt: TButton;
    MainMenu1: TMainMenu;
    mmZobrazit: TMenuItem;
    mmJednoducha: TMenuItem;
    mmRozsirena: TMenuItem;
    mmKonec: TMenuItem;
    BtnNaTreti: TButton;
    BtnXnaY: TButton;
    BtnAbs: TButton;
    procedure BtnRovnoClick(Sender: TObject);
    procedure BtnPlusClick(Sender: TObject);
    procedure BtnMinusClick(Sender: TObject);
    procedure BtnKratClick(Sender: TObject);
    procedure BtnDelenoClick(Sender: TObject);
    procedure BtnLomXClick(Sender: TObject);
    procedure BtnSqrClick(Sender: TObject);
    procedure BtnSqrtClick(Sender: TObject);
    procedure BtnCClick(Sender: TObject);
    procedure BtnPiClick(Sender: TObject);
    procedure mmKonecClick(Sender: TObject);
    procedure mmJednoduchaClick(Sender: TObject);
    procedure mmRozsirenaClick(Sender: TObject);
    procedure BtnFaktClick(Sender: TObject);
    procedure BtnNaTretiClick(Sender: TObject);
    procedure BtnXnaYClick(Sender: TObject);
    procedure BtnAbsClick(Sender: TObject);
private
    { Private declarations }
    operand1, operand2:real;
    vysledek:real;
    operace:Toperace;
    procedure TlacitkoOperace;

public
    { Public declarations }
end;

var
    FrmKalkulator: TFrmKalkulator;

implementation

{$R *.dfm}

procedure TFrmKalkulator.TlacitkoOperace;
begin
    operand1:=StrToFloat(EditDisplay.Text);
    EditDisplay.Clear;
    EditDisplay.SetFocus;

```

```

end;

procedure TFrmKalkulator.BtnRovnoClick(Sender: TObject);
begin
    operand2:=StrToFloat(EditDisplay.Text);
    if (operace=o_plus) then
        vysledek:=operand1 + operand2;
    if (operace=o_minus) then
        vysledek:=operand1 - operand2;
    if (operace=o_krat) then
        vysledek:=operand1 * operand2;
    if (operace=o_deleno) then
        if (operand2 <> 0) then vysledek:=operand1 / operand2
        else
            begin
                vysledek:=operand1;
                MessageDlg('Nulou nelze dělit!', mtWarning, [mbOK], 0);
            end;
    if (operace=o_nay) then
        vysledek:=XnaY(Round(operand1), Round(operand2));

    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnPlusClick(Sender: TObject);
begin
    operace:=o_plus;
    TlactitkoOperace;
end;

procedure TFrmKalkulator.BtnMinusClick(Sender: TObject);
begin
    operace:=o_minus;
    TlactitkoOperace;
end;

procedure TFrmKalkulator.BtnKratClick(Sender: TObject);
begin
    operace:=o_krat;
    TlactitkoOperace;
end;

procedure TFrmKalkulator.BtnDelenoClick(Sender: TObject);
begin
    operace:=o_deleno;
    TlactitkoOperace;
end;

procedure TFrmKalkulator.BtnLomXClick(Sender: TObject);
begin
    operace:=o_lomx;
    TlactitkoOperace;
    if (operand1 <> 0) then vysledek:=1 / operand1
    else
        begin

```

```

        vysledek:=1;
        MessageDlg('Nulou nelze dělit!', mtWarning, [mbOK], 0);
    end;
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnSqrClick(Sender: TObject);
begin
    operace:=o_sqr;
    TlactitkoOperace;
    vysledek:=operand1 * operand1;
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnSqrtClick(Sender: TObject);
begin
    operace:=o_sqrt;
    TlactitkoOperace;
    vysledek:= sqrt(operand1); {druhá odmocnina}
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnCClick(Sender: TObject);
begin
    {vznuluje proměnné a zruší operaci}
    operand1:=0;
    operand2:=0;
    operace:=o_nic;
    vysledek:=0;
    EditDisplay.Text:='0';
end;

procedure TFrmKalkulator.BtnPiClick(Sender: TObject);
begin
    EditDisplay.Text:='3,1415927'; {konstant Pi}
end;

procedure TFrmKalkulator.mmKonecClick(Sender: TObject);
begin
    Application.Terminate;
end;

procedure TFrmKalkulator.mmJednoduchaClick(Sender: TObject);
begin
    if (mmJednoducha.Checked<>true) then
    begin
        mmJednoducha.Checked:=true;
        mmRozsirena.Checked:=false;
        FrmKalkulator.Height:=165;
        //FrmKalkulator.Width:=268;
        GroupBoxRozFunkce.Visible:=false;
    end;
end;

```

```

procedure TFrmKalkulator.mmRozsirenaClick(Sender: TObject);
begin
    if (mmRozsirena.Checked<>true) then
    begin
        mmRozsirena.Checked:=true;
        mmJednoducha.Checked:=false;
        FrmKalkulator.Height:=225;
        //FrmKalkulator.Width:=268;
        GroupBoxRozFunkce.Visible:=true;
    end;
end;

procedure TFrmKalkulator.BtnFaktClick(Sender: TObject);
begin
    operace:=o_fakt;
    TlacitkoOperace;
    vysledek:=Faktorial(Round(operand1));
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnNaTretiClick(Sender: TObject);
begin
    operace:=o_natreti;
    TlacitkoOperace;
    vysledek:=XnaTreti(operand1);
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

procedure TFrmKalkulator.BtnXnaYClick(Sender: TObject);
begin
    operace:=o_nay;
    TlacitkoOperace;
end;

procedure TFrmKalkulator.BtnAbsClick(Sender: TObject);
begin
    operace:=o_abs;
    TlacitkoOperace;
    vysledek:=Abs(operand1);
    {Výpis výsledku}
    EditDisplay.Text:=FloatToStrF(vysledek, ffGeneral, 10, 4);
end;

end.

```



Časté chyby

- Matematické funkce vytváříte ve stejné unitě, ve které se definuje formulář. Zdrojový text je málo přehledný. Navíc nelze definované matematické funkce využít v jiném projektu.
- V sekci **uses** zapomínáte uvést jméno připojované unity. Program pak nemůže nalézt vámi definované funkce.
- Program, který se objevuje ve dvou různých náhledech vytvoříte ze dvou samostatných oken - formulářů. U našeho programu použít dva formuláře pro zobrazení dvou režimu kalkulačky je zcela zbytečné. Dva formuláře znamenají dvě třídy a dvakrát napsané metody, které jsou ale jinak úplně stejné! Pouhou změnou viditelnosti některých objektů a změnou velikosti okna docílíme mnohem efektivnějšího řešení.
- Objekty se společným chováním ovládáte jednotlivě. Vhodnější je umístit všechny objekty do sjednocujícího objektu typu *GroupBox* a ovládat jen ho.



Opakovací test

Testové otázky a úkoly opakovacího testu jsou obsaženy pouze v on-line verzi kurzu. Do textového souboru nelze zakomponovat dynamicky zpracované otázky s automatickým vyhodnocováním.

1. Které klíčové slovo slouží k připojení další unity?
 - include
 - insert
 - uses
 - unit
2. Který nástroj může sloužit k vytváření vlastních ikon k programům?
 - Image Editor
 - Imagin Editor Delphi
 - Delphi Pascal
 - Image Delphi Icons
3. Jakou příponu má projektový soubor v Borland Delphi?
 - pas
 - dfm
 - dpr
 - prj

Shrnutí



- Pro větší přehlednost i bezpečnost programů je vhodné všechny pomocné proměnné a podprogramy vytvářet jako soukromé prvky definované třídy.
- Vytváříme-li obecnější funkce, které bude možné využít i v jiných projektu, pak je definujeme ve vlastních unitách. Ty se musí připojit pomocí klíčového slova **uses**.
- Komponenta typu *GroupBox* slouží k "sjednocení" vybraných objektů. Ty je možné pak ovládat pomocí *GroupBoxu*.

Rejstřík

GroupBox
Image Editor
unit
uses

Samostatné práce



Cíl lekce

Cílem této lekce je seznámit se se zadáním samostatných prací.

Zápočet z předmětu **Vývoj objektových aplikací 1** bude udělen na základě zpracovaných programů z této lekce. Vytvořené programy doneste vyučujícímu zároveň se všemi zdrojovými soubory, ve kterých budete muset umět vysvětlit uplatněné postupy.

Po absolvování lekce budete:

- umět samostatně vytvořit program
- umět pracovat s komponentami *CheckBox*, *RadioButton*, *ComboBox*
- mít základní podklady pro udělení zápočtu

Časová náročnost lekce: **4 hodiny**



Úkol č.1

V první práci budete mít za úkol zdokonalit textový editor, který jsme vytvořili v lekci s názvem *Tvorba projektů č.3 - textový editor*.

Rozšiřte funkce editoru:

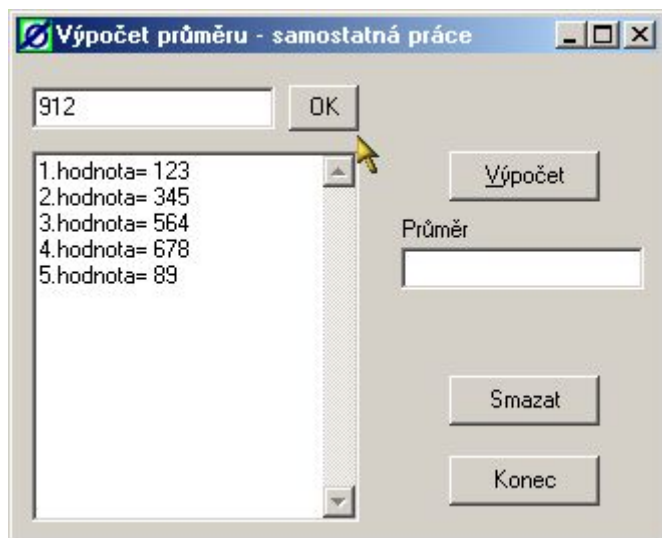
- Nabídka v hlavním menu, která umožní zakázat (povolit) zápis do objektu *Memo*
- V titulkovém pruhu okna editoru bude zobrazován kromě názvu programu rovněž název souboru, se kterým se právě pracuje
- Vytvořte si vlastní ikonku, kterou přidáte k vašemu programu (editoru)

Úkol č.2

Vytvořte program pro výpočet průměrů naměřených hodnot.

Program bude obsahovat minimálně tyto objekty:

- Edit č.1 - pro postupné zadávání hodnot
- Tlačítko *OK*- pro potvrzení zadané hodnoty a její přesunutí mezi ostatní hodnoty do komponenty Memo
- Memo - zde budou zobrazovány všechny zadané hodnoty
- Tlačítko *Výpočet* - provede výpočet průměru a запиše ho do Editu
- Edit č.2- zde budou zobrazovány vypočtené průměry
- Tlačítko *Smazat* - smaže všechny hodnoty



Prohlédněte si příklad toho, jak by mohl program pracovat. Stisknutím tlačítka si otevřete komprimovaný soubor. Formát zip. (jen v on-line verzi)



Nové grafické komponenty

Naučte se samostatně pracovat s grafickými komponentami:

- *CheckBox* - je možné označit (zaškrtnout) libovolný počet objektů tohoto typu
- *RadioButton* - označen (zaškrtnut) je právě jeden objekt tohoto typu
- *ComboBox* - rozvinovací nabídka hodnot



Seznamte se důkladně s chováním jednotlivých objektů, jeho vlastnosti, reakcemi na události a metodami. Veškeré informace lze najít v nápovědě Delphi. Jednotlivé pojmy hledejte pod názvem tříd. Tedy *TCheckBox*, *TRadioButton* a *TComboBox*.



Kontrolní úkol:

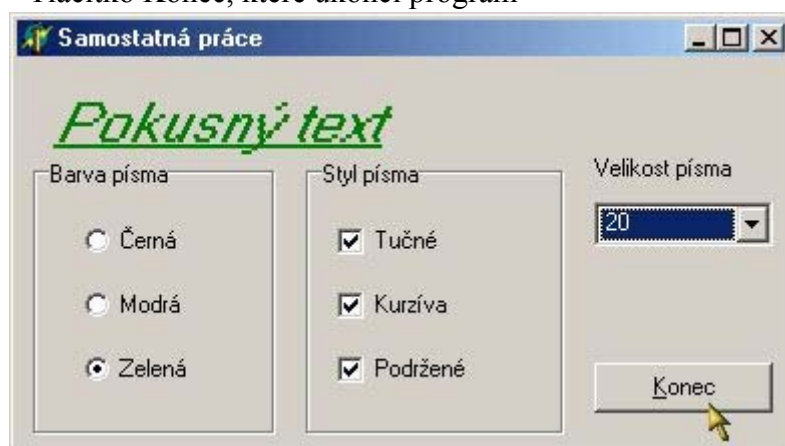
Prohlédněte si jak lze využít grafické komponenty *RadioGroup*.

Úkol č.3

Vytvořte program, u kterého bude měnit font nápisu v okně programu.

Program bude obsahovat minimálně tyto objekty:

- *Label* - v něm bude umístěn text, který se bude měnit
- *GroupBox* č.1 a v něm tři objekty typu *RadioButton*, pomocí kterých budeme měnit barvu textu (černá, modrá, zelená)
- *GroupBox* č.2 a v něm tři objekty typu *CheckBox*, pomocí kterých budeme měnit styl textu (tučné, kurzíva, podtržené)
- *ComboBox* - pomocí kterého budeme nastavovat velikost písma v hodnotách 12, 14, 16, 18, 20, 22
- Tlačítko Konec, které ukončí program



Prohlédněte si příklad toho, jak by mohl program pracovat. Stisknutím tlačítka si otevřete komprimovaný soubor. Formát zip. (jen v on-line verzi)

Shrnutí



- Zápočet z předmětu **Vývoj objektových aplikací 1** bude udělen na základě zpracovaných programů z této lekce. Vytvořené programy doneste vyučujícímu zároveň se všemi zdrojovými soubory.
- Při obhajobě svých prací je potřeba, aby jste dokázali vysvětlit jednotlivé pasáže svého zdrojového kódu.
- Nezapomeňte, že se budou hodnotit zdrojové soubory. Proto správně pojmenovávejte jednotlivé objekty a své kódy pište přehledně a s poznámkami.
- Důležitý je rovněž vzhled a jednoduchost ovládání vašich programů.