INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Gymnázium, Střední odborná škola a Vyšší odborná škola
Ledeč nad Sázavou

ZÁKLADY TVORBY APLIKACÍ

(VERZE 2013)

Ing. Jan Roubíček

Vytvořeno v rámci projektu:
Implementace řízení strojů do výuky technických předmětů na SOŠ Ledeč nad Sázavou
Registrační číslo projektu: CZ.1.07/1.1.36/01.0019

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.

Obsah

Úvod

Tento učební text slouží jako podpůrný materiál pro výuku předmětů, které jsou zaměřeny na proces návrhu a na základy tvorby aplikací a webových prezentací. Nejde o klasickou učebnici, která by byla vhodná také k samostudiu, ale spíše o manuál, využitelný právě při návrhu a tvorbě nových aplikací a webů.
V jednotlivých kapitolách jsou stručně vysvětleny základní pojmy, dále prostředí používaných programů a syntaxe (způsoby zápisu) programového kódu v jednotlivých programovacích jazycích, včetně ukázkových příkladů.

Obsahová část je členěna do čtyř základních oblastí:
První oblastí (kapitoly 1 až 3) je základní rozdělení programů a rozbor postupu jejich tvorby, včetně analýzy a algoritmizace. Na tuto oblast navazuje druhá oblast (kapitoly 4 a 5), zabývající se strukturovaným programováním aplikací a dále třetí oblast (kapitola 6), která je zaměřena na objektově orientované programování. Poslední oblast (kapitoly 7 až 10) je věnována programování webových prezentací a aplikací, včetně jejich optimalizace.

Strukturované programování je aplikováno v prostředí robota Karla, konzolové aplikace jsou vytvářeny ve formě dávek příkazového řádku operačního systému Windows a dále je strukturované programování využito pro automatizaci činností v kancelářských aplikacích MS Office – tvorba maker VBA (Visual Basic for Application).
Objektově orientované programování se zaměřuje na práci ve vývojovém prostředí Visual Studio. Programování využívá programovací jazyk C#.
Webové prezentace jsou vytvářeny pomocí programovacích jazyků HTML (HyperText Markup Language) a CSS (Cascading Style Sheets).

1 Životní cyklus programu

Životním cyklem programu (softwaru) označujeme jednotlivé fáze při vývoji programů.

1.1 Analýza

Při analýze je potřeba zjistit, co by měl budoucí program dělat a jaké funkce by měl obsahovat.

- Zjištění potřeb budoucích uživatelů.
- Definice problému.
- Konkrétní zadání úlohy.

1.2 Návrh

Návrh zahrnuje přípravu před vlastním programováním příslušné aplikace.

- Slovní formulace
  - Nástin postupu řešení problému.
  - Rozfázování tvorby (rozdělení na dílčí fáze).
- Algoritmizace
  - Výběr vhodné formy pro zápis algoritmu.
  - Vlastní tvorba algoritmu.

1.3 Programování

Programování znamená vytvoření programu v určitém programovacím jazyce. Součástí programování je také odladění programu (Debugging) = ověření jeho funkčnosti, odstranění chyb.

1.3.1 Kompilace

Kompilace je převod programového kódu do spustitelné formy (například vytvoření exe souboru). Zkompilovaný program je již nezávislý na programovacím jazyce.

1.4 Testování

Testování je činnost, jejímž primárním cílem je odhalit chyby v programu. Lze zjistit pouze přítomnost chyby, nikoliv její nepřítomnost!
Snažíme se program spouštět za nejrůznějších (původně i nepředpokládaných) podmínek a při tom sledujeme, jak se s nimi program vypořádá. Testovat lze ručně nebo pomocí různých skriptů a testovacích nástrojů.

1.4.1 Druhy chyb

1.4.1.1 Syntaktické (Syntax error)

Chybný zápis příkazů. Program nelze spustit, ani zkompilovat.

1.4.1.2 Běhové

Vznikají až při běhu programu. Nazývají se výjimkami, neboť většinou indikují vznik výjimečné situace. Klasickým příkladem je dělení nulou.

1.4.1.3 Logické

Program lze sice spustit, ale nepracuje správně. Obtížně se identifikují.

1.5 Zpracování programové dokumentace.

- Specifikace minimálních (případně také maximálních) požadavků na hardwarové a softwarové vybavení (například minimální požadovaná paměť a kapacita disku).
- Vytvoření návodů na používání programu, případně souborů nápovědy.

1.6 Implementace

Implementace znamená instalaci programu včetně jeho konfigurace (nastavení vzhledu a dalších volitelných parametrů) a jeho využívání.

1.7 Údržba a aktualizace programu.

Údržba představuje činnosti sloužící k zajištění jeho spolehlivosti. Při zjištění nedostatků je třeba program aktualizovat (například naprogramováním a instalací opravných balíčků nebo nové verze).

1.8 Zastarání programu

Ukončení vývoje a aktualizací programu.

2 Druhy programů

2.1 Rozdělení programů podle platformy, pro kterou jsou určeny

- Desktopové – pro určitý operační systém
  - Windows (příslušné verze – např. od verze XP)
  - Linux
  - Apple Mac OS
- Webové
  - Určené pro používání ve webových prohlížečích.
- Konzolové
  - Pracují v textovém prostředí příkazového řádku.
- Mobilní – pro mobilní telefony, tablety, …

2.2 Rozdělení programů podle funkčního účelu

Aplikační software rozdělujeme podle oblasti uživatelského požití.

- Kancelářské aplikace
  - Pro zpracování textu
  - Pro tvorbu tabulek
  - Pro tvorbu prezentací
  - …
- Grafické
  - Pro práci s rastrovou grafikou
  - Pro práci s vektorovou grafikou
  - Pro 3D grafiku
- Výpočtové programy
  - Pro řešení určitého speciálního technického problému
  - Speciální kalkulačky
- Výukové programy
  - Pro výuku různých oborů prostřednictvím počítačů a dalších elektronických prostředků
  - E-learningové aplikace
  - Testovací programy
- Utility
  - Pomocné programy (např. pro správu operačního systému)
- Počítačové hry
- Škodlivý SW
  - Programy, snažící se nabourat do cizích počítačů a získat z nich citlivé informace
- …

3 Algoritmizace

3.1 Algoritmus

Algoritmus je přesně definovaný a záměrně vytvořený postup (návod, „kuchařka“). Pokud jej krok po kroku vykonáváme, zadaný problém bez zbytečných a neefektivních činností, po konečném počtu kroků vyřešíme.

- Algoritmus je podrobný postup, jak řešit určitý problém.
- Realizací tohoto postupu musíme dojít od zadaných vstupních dat k požadovanému výsledku.
- Algoritmus se skládá z jednoznačně určených činností (kroků algoritmu).
- Jednotlivé kroky nazýváme také příkazy.

Algoritmus je takový předem definovaný postup, který umožňuje řešení celé skupiny principiálně stejných úloh. Algoritmus je tedy postup obecný.

3.2 Vlastnosti algoritmů

3.2.1 Hromadnost a univerzálnost

Algoritmus není sestrojen pouze pro jediný problém, ale na celou řadu problémů. Řeší celou přesně vymezenou skupinu konkrétních problémů, které se liší pouze vstupními hodnotami (například postup sečtení libovolného množství jakýchkoliv čísel).

3.2.2 Jednoznačnost (determinovanost)

Návaznost jednotlivých kroků musí být jednoznačně definována. Po vykonání každého kroku algoritmu je jednoznačně určeno, jakou činností se má pokračovat a tím je zajištěno, že při realizaci daného algoritmu dostaneme pro tytéž vstupní hodnoty tentýž výsledek.

3.2.3 Konečnost

Proces (výpočet) se ukončí v „rozumném“ čase. Realizace algoritmu končí po konečném počtu kroků požadovaným výsledkem. Nesmí se stát, že by výpočet nikdy neskončil.

3.2.4 Diskrétnost

Jednotlivé kroky algoritmu jsou oddělené, nezávislé

3.2.5 Správnost

Výsledek vydaný algoritmem musí být správný.

3.2.6 Rezultativnost

Algoritmus při zadání vstupních dat vždy vrátí nějaký výsledek (může se jednat i jen o chybové hlášení).

3.3 Postup při tvorbě algoritmu

- Prvotní analýza (rozbor) úlohy – jde především o pochopení podstaty problému.
- Rozdělení na jednoduché (elementární) problémy (kroky).
- Stanovení sledu řešení jednotlivých kroků.
- Kontrola.

3.4 Části algoritmu

- Začátek (počáteční bod, od kterého začíná zpracování) = spouští algoritmus.
- Jednotlivé kroky = dílčí, přesně vymezené a oddělené operace.
- Sekvence = přesně definovaná posloupnost kroků.
- Větvení = rozhodování mezi dvěma a více možnými stavy na základě vyhodnocení určité podmínky (při splnění podmínky pokračuje algoritmus určitou sekvencí, při jejím nesplnění následuje sekvence jiná).
- Cyklus = skupina opakujících se kroků.
- Skok = přechod k jiné části algoritmu.
- Podprogramy = předem definované části algoritmu, které fungují jako jeden krok, ale pro jejichž vykonání existuje samostatný algoritmus.
- Konec (konečný bod, po jehož provedení již zpracování nepokračuje) = ukončuje algoritmus.

3.5 Zápis algoritmu

Pro vyjádření algoritmu řešené úlohy lze využít různých způsobů:

3.5.1 Slovní zápis

Nejběžnější způsob vyjádření postupů chování programu nebo běžných činností (zákony, vyhlášky, normy, návody k použití, kuchařky, …).
Výhodou tohoto způsobu je především srozumitelnost a nezávislost na znalosti odborné terminologie a syntaxe. Nevýhodou je naopak nepřehlednost a nemožnost automatického převodu do formy počítačového programu.

3.5.2 Symbolický jazyk

Vyjádření algoritmu zápisem jednotlivých operací přesně stanovenou syntaxí.
Výhodou je formalizovaný zápis a z toho plynoucí snadné převedení algoritmu do konkrétního programovacího jazyka (např. m ← hmotnost výrobku). Často se používá skutečný programovací jazyk doplněný slovním popisem.

3.5.3 Grafické vyjádření

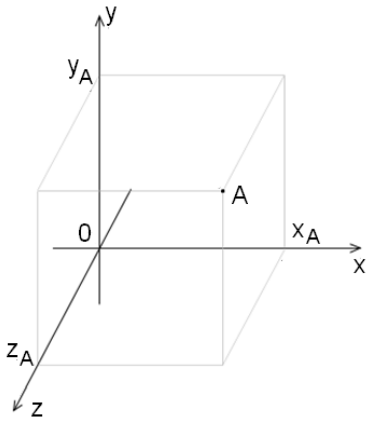
Zobrazení pomocí diagramů (například sekvenčních nebo vývojových), schémat a grafů, často doplněné symbolickým jazykem.
Výhodou je přehlednost a jednoduchost převodu do programovacího jazyka. Mnohdy slouží jako podklad pro programovou dokumentaci.

3.6 Souřadné systémy

Při programování grafických aplikací nebo určitých drah je často nutné přesné definování cest pohybu kreslícího nebo jiného nástroje, případně dalších objektů.
Souřadný systém je soustava základních údajů (referenčních bodů, přímek nebo křivek), umožňující přesné určení souřadnic polohy tělesa v rovině nebo prostoru. Poloha bodu je v daném systému souřadnic určena skupinou čísel (případně dalších symbolů), které se nazývají souřadnice bodu. Tyto souřadnice mohou představovat vzdálenosti nebo úhly vzhledem k referenčním bodům a přímkám (nebo křivkám).

- **Absolutní souřadnice** = jsou vztaženy k počátku souřadnic (zadáváme přesnou vzdálenost bodu od počátku – od bodu 0).
- **Relativní souřadnice** = jsou vztaženy k předchozímu definovanému bodu (definujeme zvýšení nebo snížení vzdálenosti vůči předchozímu bodu).

3.6.1 Kartézský souřadný systém



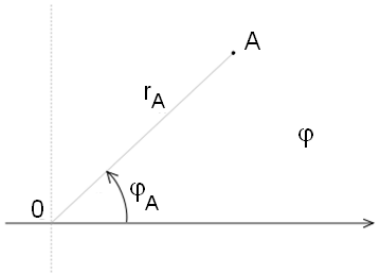
Tento souřadný systém je pojmenován podle francouzského filosofa René Descarta (latinsky Renatus Cartesius).
Pomocí kartézského systému souřadnic umísťujeme bod v trojrozměrném prostoru zadáním jeho vzdálenosti a směru od stanoveného počátku podél tří **vzájemně kolmých os** souřadného systému, označených **x**, **y**, **z**. Počátek je stanoven v bodě se souřadnicemi [0; 0; 0]. Pokud pracujeme v rovině (dvourozměrná plocha), odpadá osa **z**. Na obrázku jsou zakresleny kladné směry jednotlivých souřadných os, v obráceném směru by se souřadnice zadávaly se znaménkem mínus.

3.6.1.1 Zápis souřadnic

Souřadnice bodů zapisujeme do hranatých závorek a oddělujeme je čárkami (pokud si vystačíme s celými čísly) nebo středníky (potřebujeme-li zadávat desetinná čísla).

- **Absolutní souřadnice** = [x; y; z] nebo [x; y].
- **Relativní souřadnice** = Δx; Δy; Δz nebo Δx; Δy
  @ x; y; z nebo @ x; y

3.6.2 Polární souřadný systém



Polární souřadný systém je dvourozměrný systém souřadnic, ve kterém je každý bod určen úhlem (natočením) a vzdáleností. Každý bod je tedy definován pomocí dvou čísel: **vzdálenosti bodu od počátku a úhlem**, který svírá spojnice bodu s počátkem a přímky s úhlem 0° (v kartézském souřadném systému by tato přímka byla kladná část osy x). Úhly měříme proti směru hodinových ručiček. Pro označení vzdálenosti bodu obvykle používáme značku **r** (poloměr) a pro úhel písmeno řecké abecedy **φ** nebo **α**. Pokud potřebujeme zadávat úhly ve směru hodinových ručiček, zadáváme je se znaménkem mínus.

3.6.2.1 Zápis souřadnic

Při zápisu souřadnic bodu oddělujeme vzdálenost a úhel natočení znaménkem je menší <.

- **Absolutní souřadnice** = r < φ (zřídka).
- **Relativní souřadnice** = @ r < φ.

3.6.2.2 Přepočet polárních souřadnic na kartézské

Pro převod polárních souřadnic na kartézské používáme goniometrické funkce sinus a cosinus.

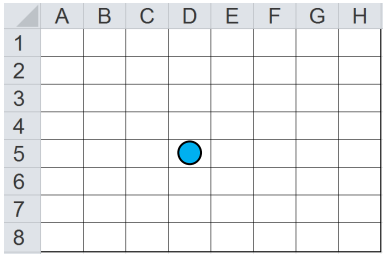
$x = r \cdot \cos \varphi$

$y = r \cdot \sin \varphi$

3.6.3 Příklady symbolického zápisu souřadnic

1) Zadání koncových bodů úsečky pomocí kartézských souřadnic.
   [10, 10]
   [60, 50] nebo @ 50, 40
2) Zadání koncového bodu úsečky pomocí polárních souřadnic.
   @ 64 < 40 (čteme … relativně 64 pod úhlem 40 stupňů)

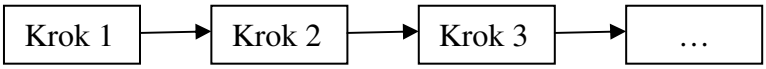
3.6.4 Tabulkový systém



K určování polohy objektu lze také využít tabulkový systém, kdy je plocha rozdělena do určitého počtu sloupců a řádků (šachovnice). Jednotlivé řádky označujeme čísly a sloupce písmeny. Vlastní poloha objektů je charakterizována průsečíkem konkrétního sloupce a řádku (například **D5**). Při relativním způsobu adresace zadáváme posun objektu vůči předchozí poloze (například **R[2]C[3]** – R značí počet řádků (row) a C počet sloupců (column)). Kladná čísla určují posun doprava a dolů, opačný směr je nutno zadat se záporným znaménkem (například **R[-2]C[-3]**).

3.7 Sekvenční diagramy

Sekvenční diagramy slouží ke grafickému ztvárnění postupu řešení jednoduchých lineárních úloh. Zobrazují postupné (sekvenční) kroky algoritmu. Daný krok je možno provést teprve po provedení všech předchozích kroků (žádný krok nelze přeskočit).



3.8 Vývojové diagramy

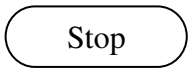
Vývojové diagramy slouží ke grafickému ztvárnění postupu řešení určitých úloh. Symboly těchto diagramů jsou normalizovány (normy ČSN 36 9011, ISO 5807 = Dokumentační symboly a konvence pro vývojové diagramy toku dat, programu a systému, síťové diagramy programu a diagramy zdrojů systému).

3.8.1 Symboly vývojových diagramů

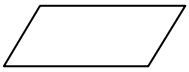
Pro znázornění jednotlivých kroků algoritmu se ve vývojových diagramech používají symboly, které jsou navzájem propojeny pomocí orientovaných šipek. Tyto symboly reprezentují jednotlivé procesy, šipky určují tok řízení.

- **Začátek algoritmu** = jednoznačně definuje začátek algoritmu.

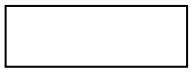
- **Konec algoritmu** = jednoznačně definuje konec algoritmu.



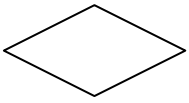
- **Vstup a výstup dat** = vstupně / výstupní operace s daty (čti / piš).



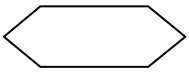
- **Zpracování** = transformace informace (změna hodnoty, výpočet, …). Vstupů může být několik, výstup smí být pouze jeden.



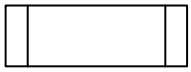
- **Větvení, rozhodování** = daný výstup je aktivován po vyhodnocení podmínek uvnitř symbolu. Podmínky jsou vyhodnoceny jako pravda (true) nebo nepravda (false). Je jeden vstup a alternativní výstupy.



- **Cyklus** = opakování určité operace nebo více operací. Má dva vstupy (jeden sekvenční a jeden pro návrat po provedení příslušného bloku operací) a dva výstupy (jeden vstupující do opakovaného bloku operací a druhů, který pokračuje do další části programu).

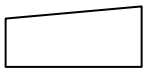


- **Podprogram** = ucelená, relativně samostatná část programu, která se může v algoritmu vyskytovat i vícekrát.



- **Spojka** = přechod z jedné části vývojového diagramu na jinou. Používá se k přerušení spojnice a k jejímu pokračování na jiném místě.

- **Ruční vstup** = zařízení pro ruční zadávání dat (např. klávesnice). Symbol má pouze jeden výstup.



- **Dokument** = zařízení pro tištěný výstup (např. tiskárna). Symbol má pouze jeden vstup.

3.8.2 Proměnné

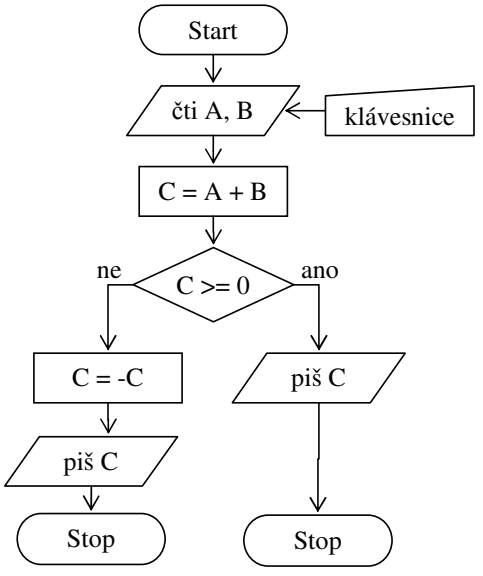
Proměnné jsou symbolicky pojmenované údaje, které se při zpracování nahradí konkrétními hodnotami. Mohou v nich být uloženy textové řetězce, čísla, obrázky nebo počítačové soubory. V průběhu vykonávání algoritmu mohou měnit svou hodnotu. Proměnná, která nemění v průběhu algoritmu svou hodnotu, se nazývá **konstanta**.

3.8.2.1 Typy proměnných:

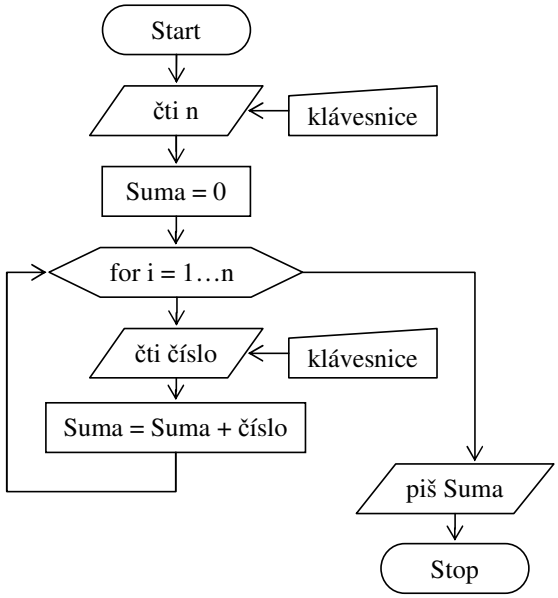
- Vstupní = jejich hodnota je dána na začátku algoritmu.
- Pracovní = slouží jako dočasné uložiště hodnoty v průběhu algoritmu.
- Výstupní = obsahují hodnoty výsledků algoritmu.

3.8.3 Příklady vývojových diagramů

1) Výpočet a výpis absolutní hodnoty součtu dvou čísel



2) Součet n čísel



4 Úvod do programování

Programování představuje převod algoritmu do konkrétního programu pomocí programovacího jazyka. Pro uživatelské používání, které je již nezávislé na zvoleném programovacím jazyce, je třeba program zkompilovat do spustitelné podoby.

4.1 Programovací jazyk

Každý program je vytvářen pomocí určitého programovacího jazyka v příslušném vývojovém prostředí a následně je zkompilován (převeden) do spustitelné podoby pro uživatelské použití.
Programovací jazyk je prostředek pro zápis programů, jež mohou být provedeny na počítači. Jde o komunikační nástroj mezi:

- uživatelem počítače, který jeho jazykovými prostředky specifikuje algoritmus řešení daného problému, a
- počítačem, jenž svými technickými prostředky algoritmus interpretuje a realizuje tak transformaci vstupních údajů na výstupní.

Programovací jazyky existují v řadě verzí a implementací; existují také standardy pro programovací jazyky, pro jednotlivé implementace se potom uvádí, kterému standardu vyhovují.

4.1.1 Syntaxe programu

Syntaxe programu = Gramatika jazyka, neboli pravidla pro vytváření příslušných řetězců (příkazů) daného programovacího jazyka. Přesně specifikuje, jak mají být zapsány jednotlivé příkazy, aby je počítač správně interpretoval.

4.1.2 Druhy programovacích jazyků

- **Strojový kód** = nejnižší forma programovacích jazyků.
- **Jazyk symbolických adres** (JSA, Assembly Language) = programový kód je tvořen symbolickou reprezentací jednotlivých strojových instrukcí a konstant potřebných pro vytvoření strojového kódu programu pro daný procesor. Je závislý na konkrétním procesoru a zapsaný program je obtížně přenositelný na jinou platformu.
- **Vyšší programovací jazyky** = bývají přenositelné mezi různými platformami.
  - Basic (Beginner's All-Purpose Symbolic Instruction Code)
  - Fortran
  - Pascal
  - Visual Basic
  - C
  - C+, C++, C#
  - Java
  - …

4.2 Postupy programování

4.2.1 Programování shora

Při programování začínáme „kostrou" programu, píšeme jenom to podstatné, nerozptylujeme se zbytečnými detaily. Postupně voláme dílčí procedury a funkce, které vykonávají požadovanou činnost (ještě je nemáme, víme jen, co by měly dělat, ale nevíme, jak to budou dělat).
Program postupně zjemňujeme, doplňujeme do něj dříve vynechané procedury a funkce.

4.2.2 Programování zdola

Nejprve napíšeme procedury a funkce realizující dílčí akce, které budou v budoucím programu zapotřebí, a teprve následně s jejich pomocí programujeme větší celky.

4.3 Stavy programu

4.3.1 Design-time

Vytváření programu (návrhový režim). Probíhá zápis kódu, tvorba objektů a nastavování jejich vlastností a funkcí.

4.3.2 Run-time

Spuštěný program. Postupné vykonávání jednotlivých příkazů, interakce s uživatelem.

4.3.3 Break-mode

Přerušení programu z důvodu chyby nebo vyvolané uživatelem, využívá se například při ladění programu.

4.4 Testování

Testováním rozumíme činnost, jejímž primárním cílem je odhalit chyby v programu.

4.4.1 Chyby programu

4.4.1.1 Syntaktické (Syntax error)

Chybný zápis příkazů. Nejsnáze se odhalují, neboť program nelze spustit, ani zkompilovat.

4.4.1.2 Běhové (Run-time error)

Vznikají až při běhu programu. Nazývají se výjimkami, jelikož většinou indikují vznik výjimečné situace. Klasickým příkladem je dělení nulou nebo špatný datový typ (například text místo čísel).

4.4.1.3 Logické

Program lze sice spustit, ale nepracuje správně (zobrazuje chybné výsledky). Program zdánlivě pracuje, proto se obtížně identifikují.

4.4.2 Odhalování chyb

Testováním lze zjistit pouze přítomnost chyby, nikoliv její nepřítomnost!
Snažíme se program spouštět za nejrůznějších (původně i nepředpokládaných) podmínek a sledujeme, jak se s nimi program vypořádá. Odhalovat chyby se můžeme pokoušet ručně nebo lze využít různé skripty a testovací nástroje.

5 Strukturované programování

Strukturované programování označuje programovací techniku, kdy se algoritmus rozděluje na dílčí úlohy, které se spojují v jeden celek. Tento způsob programování využívala většina starších programovacích jazyků (Basic, Fortran, Pascal, C, …).
Při strukturovaném programování využíváme základní tři programovací principy:

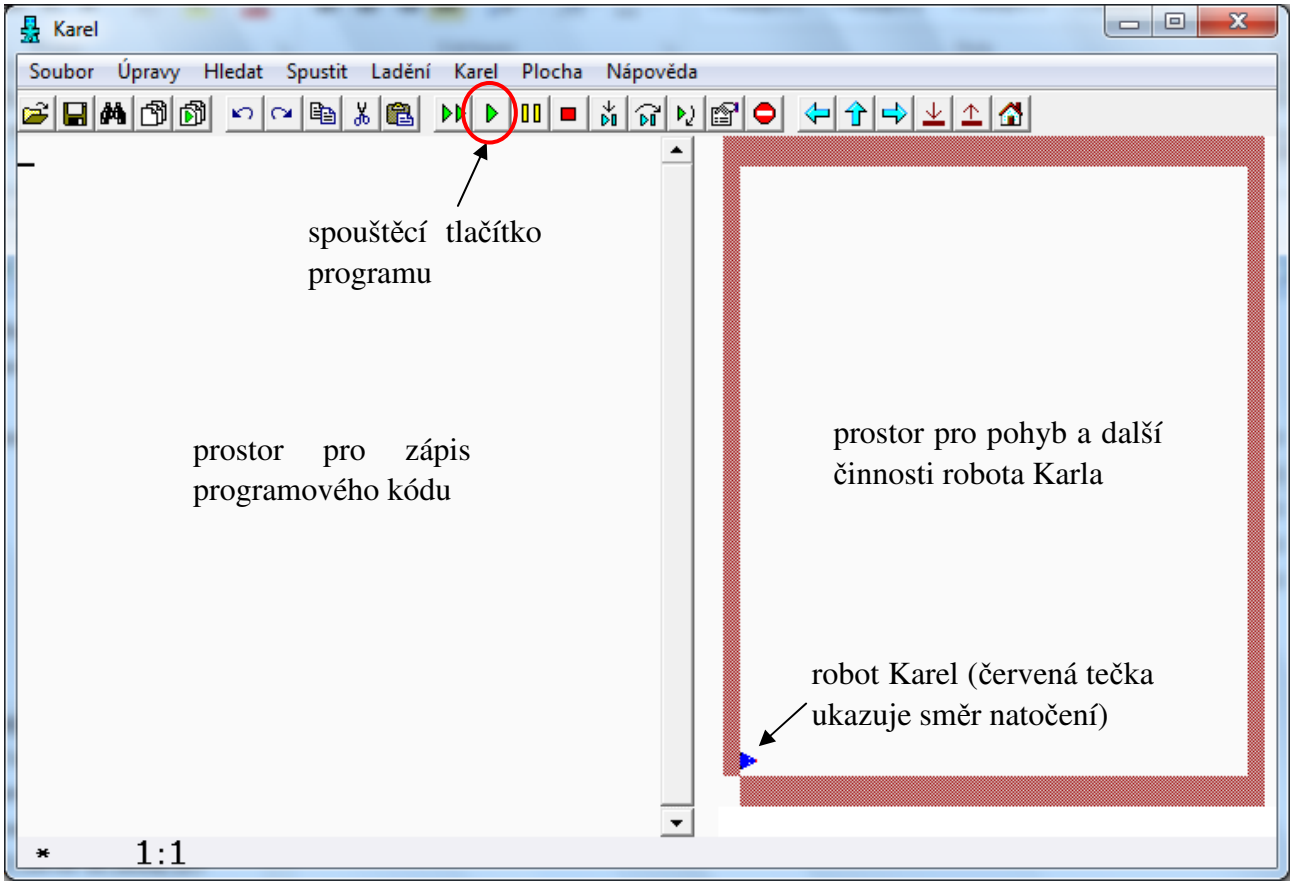
- **Sekvence** = rozdělení programu na dílčí podprogramy (procedury a funkce). Z hlavního programu potom dochází k volání těchto dílčích podprogramů. Slouží ke zpřehlednění kódu a zamezení duplicit.
- **Selekce** = rozložení úlohy na dvě (nebo několik) podúloh. Úloha se vyřeší tím, že se provede právě jedna podúloha (v závislosti na nějaké podmínce).
- **Iterace** = řešení úlohy sestává z opakovaného provádění určité podúlohy (cyklus).

5.1 Robot Karel

Program Karel je určen na výuku strukturovaného programování. Je vhodný hlavně k procvičení používání procedur a jednoduchých algoritmů.
Karel je jméno robota, který umí vykonávat příkazy podle programů (procedur), které mu napíšete.

5.1.1 Prostředí programu

Robot Karel se pohybuje v rámci šachovnice. Rozměry šachovnice se mohou lišit, přičemž pohyb robota může být navíc limitován zdmi. Na šachovnici mohou být rozmístěny značky (na jednom poli šachovnice může být 0 až 8 značek).



5.1.2 Struktura programu

Program se skládá z procedur (podprogramů). Prvním řádkem procedury je slovo PŘÍKAZ, za kterým následuje název procedury. Uvnitř procedury je posloupnost příkazů:

- základní povely pro ovládání robota (například krok nebo otočení),
- speciální příkazy pro řízení běhu programu (příkazy podmínek a cyklů),
- názvy jiných procedur (podprogramů).

Z důvodu přehlednosti programového kódu bývá zvykem psát každý příkaz na nový řádek. Je povoleno napsat na jeden řádek i více příkazů, ale v takovém případě musí být jednotlivé příkazy odděleny středníkem.
Posledním řádkem programu je slovo KONEC.

5.1.2.1 Komentáře

Do programu lze také psát komentáře pro vysvětlení zdrojového kódu.

- První typ komentáře začíná **//**, všechno až do konce řádku se považuje za komentář.
- Druhý typ komentáře je text, který začíná **/*** a končí ***/**. Takový komentář může být dlouhý i několik řádků.

5.1.3 Základní příkazy

Příkazy v námi používané verzi programovacího prostředí lze Karlovi dávat anglicky nebo česky. Základními příkazy, kterým robot Karel rozumí, jsou:

- PŘÍKAZ = PROCEDURE … začátek každé procedury (programu nebo podprogramu),
- KROK = STEP … posun robota Karla o jedno políčko v aktuálním směru,
- VLEVO = LEFT … otočení robota Karla o 90° vlevo (vlevo v bok),
- POLOŽ = PUT … položení jedné značky na aktuální pozici,
- ZVEDNI = PICK … odebrání jedné značky z aktuální pozice,
- DOMŮ = HOME … návrat do levého dolního rohu šachovnice,
- KONEC = END … ukončení každé procedury (programu nebo podprogramu).

Příklad

```
PŘÍKAZ Prvni
  KROK
  KROK
  POLOŽ
KONEC
```

5.1.4 Podmínky a cykly

Programový kód můžeme větvit zabudováním určité podmínky. Provedení akce je poté podmíněno splněním této podmínky. Každá podmínka začíná slovem KDYŽ, a končí slovem KONEC. Pro zadání podmínky a alternativní činnosti používáme příkazy:

- KDYŽ = IF … zadání podmínky, dále následují činnosti vyplývající z kladného vyhodnocení dané podmínky,
- JINAK = ELSE … alternativní činnost Karla při negativním vyhodnocení předchozí podmínky.

Pro definování podmínky používáme situaci nárazu do překážky (zdi) nebo výskytu značky na aktuální pozici:

- ZEĎ = WALL … před Karlem je překážka (zeď),
- ZNAČKA = MARK … na Karlově pozici se vyskytuje značka (1 až 8),

- NENÍ … NOT (negace) … vytvoření opaku (NENÍ ZEĎ, NENÍ ZNAČKA).

Příklad

```
KDYŽ NENÍ ZNAČKA
  POLOŽ
JINAK
  ZVEDNI
KONEC
```

Pokud má robot Karel provést určitou činnost vícekrát, není třeba jí zadávat opakovaně, ale zabalíme ji do cyklu.

- OPAKUJ = REPEAT … definice FOR cyklu,
- DOKUD = WHILE … cyklus s podmínkou na počátku,
- AŽ = UNTIL … cyklus s podmínkou na konci.

Příklady

```
OPAKUJ 4
  POLOŽ
KONEC
```

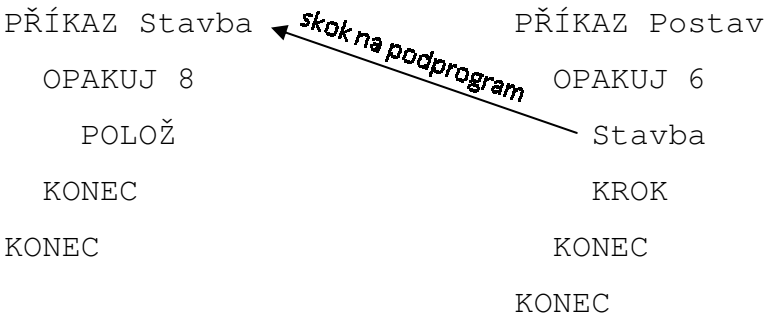
```
DOKUD NENÍ ZEĎ
  KROK
KONEC
```

```
OPAKUJ
  KROK
AŽ ZEĎ
```

5.1.5 Podprogramy

V případě, že určitou sekvenci příkazů provádíme v programu vícekrát, je vhodné tuto sekvenci pojmenovat a dále využívat jako podprogram. Další vhodné použití podprogramů bude v případě rozsáhlejších programů, kdy rozdělení na podprogramy přispěje k větší přehlednosti zdrojového kódu.

Příklad

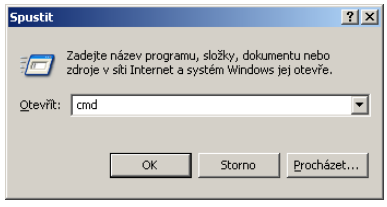


5.2 Konzolové aplikace

Konzolová aplikace je počítačový program, který nevyužívá grafické rozhraní, ale pracuje pouze v textovém rozhraní. K ovládání se používá počítačová klávesnice. Často se také využívají pro automatizované úlohy, které nevyžadují interakci s uživatelem.

5.2.1 Programování dávkových souborů

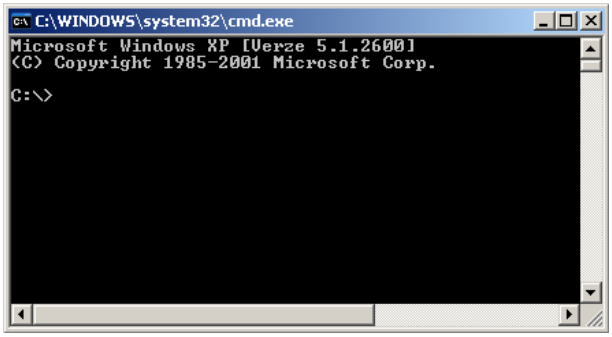
Dávkový soubor je v operačních systémech MS DOS a Microsoft Windows textový soubor obsahující sérii příkazů, které jsou zpracovány interpretem příkazového řádku. Po spuštění dávkového souboru čte interpret (obvykle command.com nebo **cmd.exe**) tento dávkový soubor a postupně spouští jednotlivé příkazy, které jsou umístěny na samostatných řádcích. Často se využívají pro administraci počítačových systémů.



Chceme-li v MS Windows pracovat přímo v příkazovém řádku,

spustíme jej nejsnáze z dialogového okna **Spustit** příkazem `cmd`. Toto dialogové okno lze vyvolat klávesovou zkratkou ***Win + R***. Příkazový řádek ukončíme příkazem `exit`, čímž dojde k návratu do grafického režimu operačního systému.
Potřebujeme-li spouštět častěji určitou skupinu příkazů, je vhodnější vytvořit samostatný soubor s příponou .bat, a to nejlépe v poznámkovém bloku (ten lze spustit například z dialogového okna **Spustit** příkazem `notepad`).

5.2.2 Prostředí příkazového řádku



Spustíme-li příkazový řádek, otevře se v samostatném okně, v jehož titulkovém pruhu je uvedena cesta k jeho programovému souboru **cmd.exe**, a na prvních řádcích se zobrazí název a verze operačního systému, z něhož je příkazový řádek spuštěn. Následuje **Prompt**, což je výzva, uváděná na každém řádku a zobrazující nejčastěji absolutní cestu do aktuálního adresáře a bývá ukončena znakem `>`. Za tuto výzvu zadáváme příslušné příkazy.

5.2.3 Základní příkazy

5.2.3.1 Vyčištění obrazovky

Příkazem `cls` dojde ke smazání všech dříve provedených příkazů a zobrazí se pouze samostatná výzva prompt. Tento příkaz nemá žádné parametry.

5.2.3.2 Nápověda k jednotlivým příkazům

Nevíme-li, jaká je syntaxe (tvar zápisu) určitých příkazů, nebo jaké lze použít parametry, lze zobrazit nápovědu k těmto příkazům. Existují dva způsoby vyvolání této nápovědy:

```
příkaz /?
help příkaz
```

5.2.3.3 Zobrazení a změna systémového datumu a času

Aktuální datum zobrazíme příkazem `date` a aktuální čas příkazem `time`. Po zadání příkazu zadáme jejich novou hodnotu ve formátu den-měsíc-rok nebo hodiny:minuty:sekundy. Nechceme-li datum nebo čas měnit, pouze potvrdíme zobrazené údaje klávesou ***Enter*** nebo použijeme příkaz s parametrem /t (`date /t` nebo `time /t`).

5.2.3.4 Změna aktuálního disku

Potřebujeme-li změnit aktuální diskovou jednotku, stačí zadat pouze název této jednotky (např. `D:`).

5.2.3.5 Práce s adresáři (složkami)

Pro práci s adresáři (složkami) používáme následující 3 základní příkazy, za nimiž následuje cesta (absolutní nebo relativní) k požadovanému adresáři, který chceme vytvořit, nastavit jako aktuální nebo jej odstranit.

- Vytvoření adresáře (make directory) = `md`
- Změna aktuálního adresáře (change directory) = `cd`
  - přechod do kořenového adresáře aktuálního disku = `cd\`
  - přechod do nadřazeného adresáře = `cd ..`

- Odstranění prázdného adresáře (remove directory) = `rd`
  - odstranění adresáře, včetně jeho obsahu (podadresářů a souborů) = parametr `/s`
  - automatické odstranění bez zobrazení potvrzovací výzvy = parametr `/q`

Příklady: `md smlouvy` = vytvoření adresáře v aktuálním adresáři.
`md ..\dokumenty\smlouvy` = vytvoření adresáře se zadanou relativní cestou.
`md C:\dokumenty\smlouvy` = vytvoření adresáře se zadanou absolutní cestou.
`rd smlouvy` = odstranění prázdného adresáře z aktuálního adresáře.
`rd C:\dokumenty\smlouvy /s` = odstranění adresáře se zadanou absolutní cestou, včetně všech jeho podadresářů.

5.2.3.6 Výpis obsahu adresáře na obrazovku

K výpisu obsahu adresáře slouží příkaz `dir`, za nímž následuje cesta (absolutní nebo relativní) k požadovanému adresáři a případné parametry:

- zobrazení výpisu po stránkách (pro pokračování výpisu po zaplnění obrazovky je vyžadováno stisknutí klávesy) = `/p`
- zobrazení obsahu určitého adresáře, včetně všech jeho podadresářů = `/s`
- výpis ve zjednodušeném formátu (pouze názvy adresářů a souborů) v několika sloupcích = `/w`
- seřazený výpis = /o
  g = nejprve adresáře, potom teprve soubory,
  n = podle názvů,
  e = podle typu (přípony),
  s = podle velikosti,
  - = obrácené (sestupné) pořadí.

Příklady: `dir ..\dokumenty\smlouvy` = výpis obsahu relativně zadaného adresáře.
`dir c:\dokumenty\smlouvy /s/p` = výpis obsahu absolutně zadaného adresáře, včetně všech jeho podadresářů po stránkách.
`dir /os` = výpis obsahu aktuálního adresáře, vzestupné řazení podle velikosti.
`dir /o-s` = výpis obsahu aktuálního adresáře, sestupné řazení podle velikosti.

5.2.3.7 Práce se soubory

Mezi základní operace se soubory patří jejich kopírování, přesunutí, přejmenování a odstranění.

- Kopie souboru = `copy zdroj` (cesta + název souboru) `cíl` (cesta)
- Přesun souboru = `move zdroj` (cesta + název souboru) `cíl` (cesta)
- Přejmenování souboru = `ren zdroj` (cesta + název souboru) `cíl` (název souboru)
- Odstranění souboru = `del zdroj` (cesta + název souboru) `[parametry]`
  - vyžadování potvrzení při odstraňování každého souboru = `/p`
  - potlačení potvrzení při odstraňování souborů při využití souborových masek = `/q`
  - odstranění příslušných souborů i ze všech podadresářů = `/s`

Příklady: `copy C:\dokumenty\smlouvy\kupni.txt D:\urgence` = absolutní zadání.
`copy smlouvy\kupni.txt D:\urgence` = relativně-absolutní zadání.
`copy smlouvy\kupni.txt urgence` = relativní zadání.
`move kupni.txt ..\urgence` = přesun z aktuálního do relativně zadaného adresáře.
`ren smlouvy\kupni.txt prodej.txt` = přejmenování s relativní cestou.

5.2.4 Souborové masky (hvězdičková konvence)

Hromadné vybrání více souborů nahrazením určitého počtu znaků v názvu souboru (v jeho jméně nebo v příponě) umožňují zástupné symboly, tzv. souborové masky.

- ***** = nahrazuje libovolný počet znaků (žádný znak až maximálně 255 znaků).
- **?** = nahrazuje přesný počet znaků (? – jeden znak, ?? – dva znaky, …).

Souborové masky lze využít zejména při vyhledávání, kopírování a mazání souborů.

Příklady: *.* … vybrání všech souborů s libovolným názvem.
***.doc** … vybrání všech souborů s příponou .doc.
d*.avi … vybrání všech souborů s příponou .avi, jejichž název začíná písmenem d.
***.do?** … vybrání všech souborů s tříznakovou příponou končící libovolným znakem.
????.txt … vybrání všech souborů s příponou .txt, jejichž jméno obsahuje 4 znaky.

5.2.5 Další příkazy

5.2.5.1 Výpis textového souboru

Obsah textového souboru vypíšeme na obrazovku příkazem `type`, s uvedenou cestou k tomuto souboru.

Příklad: `type C:\dokumenty\smlouvy\kupni.txt`

5.2.5.2 Kopírování adresářů (složek)

Pokud potřebujeme kopírovat kromě souborů také celé adresáře, použijeme příkaz `xcopy`, za nímž následuje zdrojový a cílový adresář, případně další parametry.

- zkopírování adresářů a podadresářů s výjimkou prázdných = `/e`
- zkopírování adresářů a podadresářů včetně prázdných = `/s`
- zkopírování stromové struktury adresářů bez souborů (s výjimkou prázdných, lze řešit kombinací `/t /e`) = `/t`
- zkopírování atributů souborů (samotný příkaz xcopy odstraňuje atributy read only) = `/k`
- potlačení výzvy k přepsání existujících cílových souborů `/y`
- zobrazení výzvy k přepsání existujících cílových souborů = `/-y`

Příklad: `xcopy C:\Dokumenty\Dopisy D:\Dokumenty /e`

5.2.6 Směrování výstupu do souboru

Potřebujeme-li určitý výstup zachovat pro pozdější využití, nestačí jej zobrazit na obrazovce, ale je nutno směrovat tento výstup do textového souboru. Změnu výstupu definujeme znaménkem > (je větší).

Příklad: `dir C:\dokumenty > vypis.txt` … výpis obsahu adresáře do .txt souboru.

5.2.7 Vytvoření virtuální logické jednotky (disku)

Chceme-li, aby se určitá složka tvářila jako samostatný disk, lze ji substituovat příkazem `subst`.

Příklad: `subst K: D:\dokumenty`

Odstranění substituce provedeme stejným příkazem s parametrem /d (`subst K: /d`).

5.2.8 Úprava příkazového řádku

Nevyhovuje-li nám tvar (formát) výzvy Prompt, lze ji upravit pomocí příkazu `prompt`, doplněného o vhodné parametry (běžný tvar = `prompt $P$G`).

Běžné alfanumerické znaky

`$D` = aktuální datum
`$G` = symbol > (je větší)
`$N` = aktuální disková jednotka
`$P` = absolutní cesta do aktuálního adresáře
`$T` = aktuální čas
`$S` = mezera
`$V` = verze operačního systému
`$_` = zalomení řádku výzvy

5.2.9 Dávkové soubory

V dávkových souborech lze použít všechny běžné příkazy příkazového řádku, ale také mnoho dalších příkazů. Pokud požadujeme dávku v určitém místě přerušit, zařazujeme příkaz `Pause`. Pro pokračování dávky musí uživatel stisknout ***Enter*** nebo ***Space*** (případně i jinou klávesu). Potlačení výpisu zprávy při použití příkazu Pause zajistíme směrováním výstupu … `Pause > nul`.
Přerušení probíhajících příkazů lze provést klávesovou zkratkou ***Ctrl + C***.
Pro okomentování části kódu, nebo k přechodnému potlačení vykonávání určitých příkazů slouží příkaz `rem`, umístěný na začátek řádku s komentářem.
Příklad: `rem kopírování stromové struktury`

5.2.9.1 Příkaz `echo`

Příkaz `echo` slouží k vypsání určitého textu na obrazovku. Využíváme jej k zobrazení informací při komunikaci prováděné dávky s uživatelem (např. požadavky na stisknutí určité klávesy, zadání hodnoty parametru nebo informace o právě prováděných příkazech).
Příklad: `echo Ahoj`
Speciální funkci mají následující dvě varianty tohoto příkazu:
`@echo off` = vypnutí zobrazení jednotlivých vykonávaných příkazů.
`@echo on` = zapnutí zobrazení jednotlivých vykonávaných příkazů.

5.2.9.2 Využití parametrů

Dávku můžeme sestavit více univerzální tím, že některé konkrétní hodnoty nahradíme parametry, které se budou získávat teprve v průběhu vykonávané dávky (například bude vyžadováno jejich zadání uživatelem).

- Definice (nastavení) parametru = `set parametr = hodnota`

Příkaz `set` lze doplnit o následující dva parametry:

- zadání příkazu uživatelem = `/p`
- vypočtení výrazu uvedeného za znakem rovná se = `/a`

Pro získání hodnoty parametru zadáváme jeho označení mezi znaky **%** (procento).

Příklady

`set /p A=Zadej hodnotu A:` … definování proměnné A, hodnotu zadá uživatel.
`set /p B=Zadej hodnotu B:` … definování proměnné B, hodnotu zadá uživatel.
`set /a S=%A%+%B%` … definování proměnné S jako součet hodnot proměnných A a B.

- Výpis hodnoty parametru provedeme pomocí příkazu `echo`.

Příklad: `echo Soucet = %S%` … výpis konkrétního textu, doplněného o konkrétní hodnotu parametru.

5.2.9.3 Práce s adresáři a soubory s využitím parametrů

Při tvorbě dávky, která provádí určité operace s adresáři a se soubory, lze určité konkrétní adresáře nebo soubory (případě části souborových masek) nahradit parametry, jejichž hodnota bude požadována po uživateli v průběhu prováděné dávky.

Příklady: `set /p sloz=Zadej nazev slozky:` … uživatel zadá hodnotu parametru.
`md %sloz%` … vytvoření nové složky, pojmenované hodnotou parametru.
`set /p poc=Zadej pocatecni znak:` … uživatel zadá hodnotu parametru.
`del %poc%*.*` … dojde ke smazání všech souborů v aktuálním adresáři, jejíž název začíná příslušným znakem.

5.2.9.4 Podmínky v dávkových souborech

Činnosti, prováděné dávkou, lze větvit podle splnění nebo nesplnění zvolených podmínek. K vyhodnocení podmínky využíváme příkaz `if`.

Ověření existence určitého adresáře nebo souboru

Pokud požadujeme, aby se určitá akce provedla pouze za předpokladu, že objekt, kterého se akce týká, existuje, můžeme tuto existenci ověřit pomocí příkazu `if exist`, za nímž následuje název příslušného objektu (případně i s cestou) a dále podmíněný příkaz. Pro provedení činnosti při neexistenci objektu používáme příkaz `if not exist`.

Příklady: `if exist soubor.doc del soubor.doc` … smazání souboru za předpokladu jeho existence.
`if not exist test md test` … pokud neexistuje složka test, tak se vytvoří.

Volba z několika možností

Pokud chceme po uživateli, aby zvolil z několika možností, co má dávka provést, lze vytvořit rozcestník, kdy uživatel zvolí jednu z možností a podle toho se přiřadí hodnota parametru, která určí další prováděnou činnost. Vyhodnocení zvoleného parametru je prováděno příkazem `if`, následovaného příkazem skoku `goto` (cílové místo skoku je označeno zvoleným názvem začínajícím dvojtečkou).

Příklad

```
@echo off
echo 1 ... Prvni volba
echo 2 ... Druha volba
echo 3 ... Treti volba
```

} zobrazení menu.

`set /p volba=Zvol variantu:` = nastavení hodnoty parametru uživatelem.
`echo Zvoleno: %volba%` = zobrazení zvolené hodnoty parametru.

```
if %volba%==1 goto prvni
if %volba%==2 goto druha
if %volba%==3 goto treti
```

} rozhodování podle zadané hodnoty parametru
!!! pozor na == … rovná se.

`echo Chybna volba` = činnost, prováděné při volbě jiné hodnoty parametru.
`goto konec` = odskok na přesně definované místo v dávce.

```
:prvni
```

příkazy prováděné při první volbě.

```
goto konec
:druha
```

příkazy prováděné při druhé volbě.

```
goto konec

:treti
```
příkazy prováděné při třetí volbě.
```
goto konec

:konec
echo Konec programu!
```

`pause` = po stisknutí klávesy dojde k ukončení dávky.

V podmínkách lze používat porovnání dvou hodnota, ale nelze použít znaménka > a <. Nahrazujeme je následujícími zkratkami:

EQU = rovná se.	LSS = menší než.	GTR = větší než.
NEQ = nerovná se.	LEQ = menší nebo rovno.	GEQ = větší nebo rovno.

5.2.9.5 Příklad dávkového souboru (sečtení dvou čísel)

```
@echo off

rem Nastavení výchozích hodnot proměnným
set s=0
set cislo=0
echo Pro ukonceni stiskni k.

rem Průběžné přičítání další hodnoty
:soucet
set /p cislo=Zadej cislo:
if %cislo% ==k goto konec
set /a s=%s%+%cislo%
goto soucet

rem Zobrazení celkového součtu a ukončení dávky
:konec
echo Soucet = %s%
echo Konec!

pause >nul
```

5.2.9.6 Volání externí dávky

Potřebujeme-li během provádění dávky spustit jiný dávkový soubor, voláme jej příkazem `call`.
Příklad: `call vycisti.bat`

5.3 VBA (Visual Basic for Application)

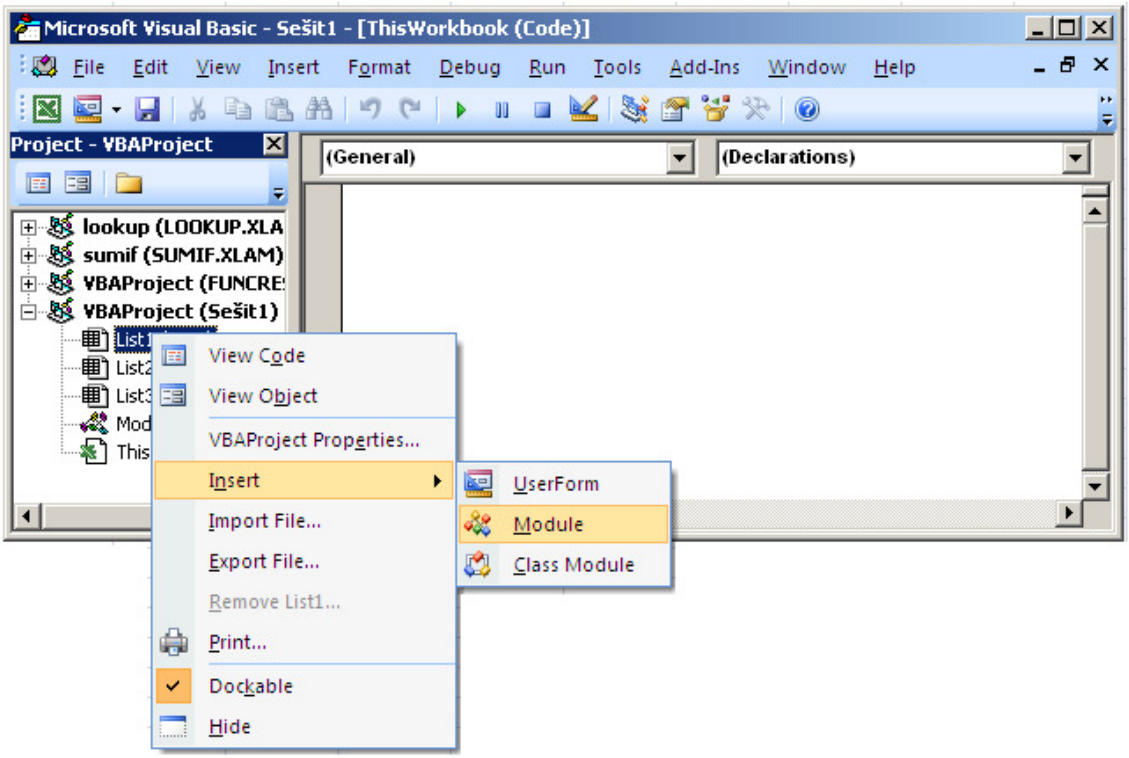
VBA (Visual Basic for Application) je odnoží programovacího jazyka Visual Basic. Využití nachází zejména v kancelářských aplikacích od firmy Microsoft (Word, Excel, Access, …). Vývojové prostředí je v angličtině.

- **Projekt VB** = pomocí něj realizujeme program.
- **Moduly VB** = obsahují obvykle obecné programy (procedury) používané v celé aplikaci.

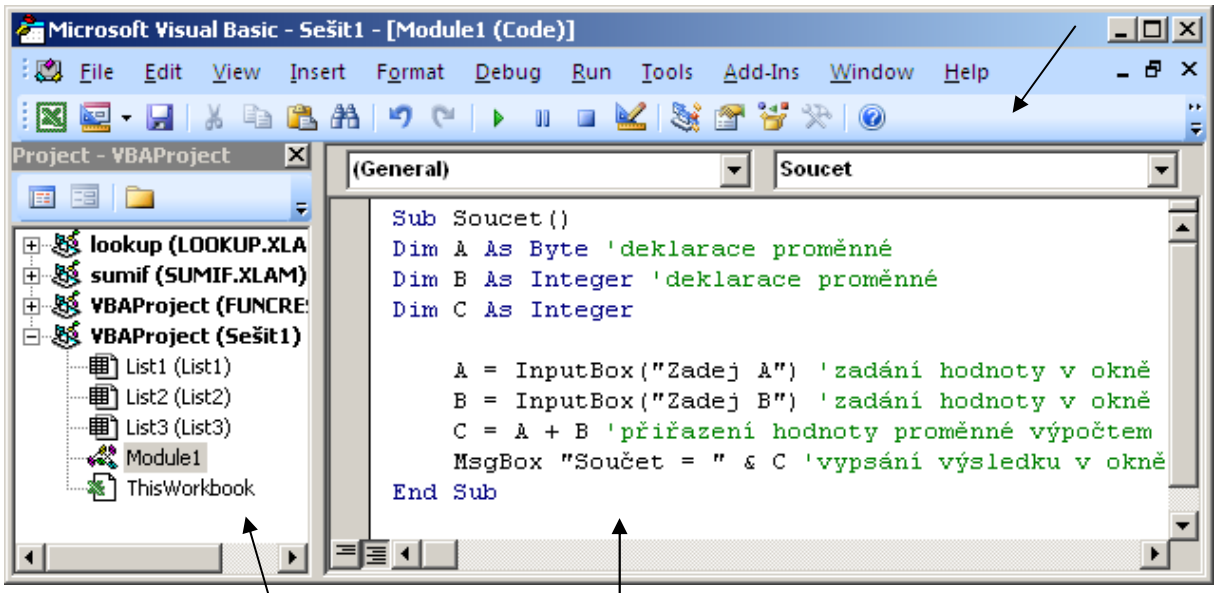
5.3.1 Vývojové prostředí

Pro programování aplikací ve VBA využíváme vývojové prostředí zabudované v aplikacích MS Office (Word, Excel, Access, …). Spuštění vývojového prostředí z běžící aplikace provádíme nejsnáze klávesovou zkratkou Alt+F11.

Nový modul, ve kterém budeme programovat jednotlivé procedury (makra) a funkce vytvoříme pomocí nabídek vyvolaných kliknutím pravého tlačítka myši na název listu (v Excelu) nebo na ThisDocument (ve Wordu).



řádek nabídek (menu)



struktura projektu

programový kód (jednotlivé procedury a funkce)

5.3.2 Struktura programového kódu

Procedura (Subroutine) začíná vždy příkazem **Sub**, následovaným názvem této procedury a prázdnými závorkami. Každá procedura je ukončena příkazem **End Sub**. Pro předčasné ukončení procedury slouží příkaz **Exit Sub**.

Jednotlivá klíčová slova (příkazy) oddělujeme od ostatních objektů kódu mezerou. Pokud se příkaz nevejde na 1 řádek, k jeho rozdělení používáme znak podtržítka _.

Příklad

```
MsgBox "Chyba!" & Err.Description, _
vbCritical, "Kritická chyba"
```

Pro zpřehlednění kódu, abychom se v něm zorientovali i po delší době, slouží komentáře, které můžeme zapisovat dvěma způsoby:

- **rem** komentář … na začátku řádku,
- **'** komentář … na začátku řádku nebo i na konci za příkazem.

5.3.3 Proměnné (Variables)

Proměnné umožňují programátorovi pojmenovat místo v paměti a ukládat si tam potřebná data. Označují se jménem a mají přiřazen určitý datový typ (jaké hodnoty mohou obsahovat).

5.3.3.1 Druhy proměnných (podle dostupnosti)

- **Globální** = jsou platné pro celý modul. Musí být deklarovány mimo proceduru nebo funkci.
- **Lokální** = jsou platné pouze pro danou proceduru nebo funkci. Deklarujeme je na začátku příslušné procedury nebo funkce.
- **Soukromé** (Private) = jsou platné pouze pro aktuální modul.
- **Veřejné** (Public) = jsou dostupné také mimo aktuální modul (v celém projektu).

5.3.3.2 Datové typy proměnných

Datový typ udává, jaký typ dat bude příslušná proměnná obsahovat.

- **Logický**
  - Boolean (nabývá hodnoty True nebo False).
- **Celá čísla**
  - Byte (1B = 0 až 255),
  - Integer = krátké celé číslo (2B = -32 768 až 32 767),
  - Long = dlouhé celé číslo (4B = -2 147 483 648 až 2 147 483 647).
- **Desetinná čísla**
  - Single = jednoduchá přesnost (4B),
  - Double = dvojitá přesnost (8B),
  - Decimal = desetinné číslo (12B).
- **Měna**
  - Currency (8B).
- **Datum a čas**
  - Date (8B).
- **Text**
  - String.

5.3.3.3 Deklarace proměnných

Všechny proměnné je nutné před použitím deklarovat. Na začátku programového kódu (v deklarační části modulu) zapisujeme příkaz **Option explicit**, který slouží k vynucení povinnosti deklarovat proměnné.
Deklarace udává jméno proměnné, její datový typ a dostupnost.

Příklad

```
Private hmotnost As Integer
```

5.3.3.4 Druhy proměnných (podle počtu hodnot, které obsahují)

- **Skalární** = obsahuje jedinou hodnotu, která se může v průběhu provádění procedury nebo funkce měnit.

  Deklarace: `Dim strana As Single`

- **Konstanta** = skalární typ proměnné, jejíž hodnota se nemění.

  Deklarace: `Const PI = 3.141592654`

- **Pole** = obsahuje skupinu hodnot. Jednotlivé hodnoty rozlišujeme celočíselným indexem v závorce. Počáteční index je 0.

  Deklarace `Dim pole(0 To 5) As Integer` nebo zkráceně `Dim pole(5) As Integer` (deklaruje pole o 6 prvcích).

  Pro indexování od jedničky používáme příkaz `Option Base 1`.

5.3.4 Základní operátory

Při výpočtech a dalších operacích využíváme následující operátory:

- **Matematické operátory**

  + - * / ^ … sčítání, odčítání, násobení, dělení, mocnění,

  **** … celočíselné dělení , **mod** … zbytek po celočíselném dělení

- **Zřetězení (spojování textových řetězců)**

  &

- **Přiřazení**

  =

- **Porovnání**

  == … rovná se, **< >** …nerovná se, **>** …větší než, **<** … menší než, **>=**, **<=**

- **Logické operátory**

  and … a, **or** … nebo, **not** … negace

5.3.5 Programování s dialogovým oknem Windows

K zadávání hodnot proměnných a zobrazení informací a výsledků lze využít oken Windows (InputBoxy a MessageBoxy).

- **Zadávání hodnot proměnných**
  - `A = InputBox("Zadej A", "Zadání")`
- **Zobrazení hodnot proměnných**
  - `MsgBox "Součet = " & C`

Chceme-li zalomit text v okně Windows, zřetězujeme s **vbCrLf**.

Příklad: Sečtení dvou čísel

```
Sub Soucet()
Dim A As Byte 'deklarace proměnné - celé číslo 0 až 255 (1 byte)
Dim B As Integer 'deklarace proměnné - celé číslo (2 byty)
Dim C As Integer

    A = InputBox("Zadej A") 'zadání hodnoty v okně windows
    B = InputBox("Zadej B") 'zadání hodnoty v okně windows
    C = A + B 'přiřazení hodnoty proměnné výpočtem
    MsgBox "Součet = " & C 'vypsání výsledku v okně windows (text + hodnota)
End Sub
```

5.3.5.1 Podmínky (větvení)

K variantnímu programování s podmínkami využíváme programovou konstrukci `If – Then / Else / End If`.

Příklad: Ověření hesla

```
Sub ZadejHeslo()
Dim Heslo As String 'deklarace textového řetězce

    Heslo = InputBox("Zadej heslo: ")
    If UCase(Heslo) = "ABC" Then 'UCase převede zadané na velká písmena
       MsgBox "Heslo OK."
    Else
       MsgBox "Chybné heslo!"
    End If
End Sub
```

Příklad: Zjištění maxima ze tří čísel

```
Sub Maximum()
Dim A As Integer
Dim B As Integer
Dim C As Integer
Dim Xm As Integer
Dim Ym As Integer

    A = InputBox("Zadej číselnou hodnotu A:")
    B = InputBox("Zadej číselnou hodnotu B:")
    C = InputBox("Zadej číselnou hodnotu C:")
    If A > B Then 'začátek 1. podmínky
        Xm = A
    Else
        Xm = B
    End If 'konec 1. podmínky
    If Xm > C Then 'začátek 2. podmínky
        Ym = Xm
    Else
        Ym = C
    End If 'konec 2. podmínky
    MsgBox "Maximální hodnota:" & Ym
End Sub
```

Pokud za sebou následuje více podmínek, lze je řešit programovou konstrukci `If – Then / ElseIf / Else / End If`.

Příklad: Porovnání dvou čísel

```
Sub Porovnani()
Dim A As Integer
Dim B As Integer

    A = InputBox("Zadej A")
    B = InputBox("Zadej B")
    If A > B Then 'podmínka 1
        MsgBox "A je větší než B"
    ElseIf A < B Then 'podmínka 2
        MsgBox "A je menší než B"
    Else '3. varianta
        MsgBox "A je rovno B"
    End If 'konec podmínky
End Sub
```

5.3.5.2 Konstrukce Select Case

Pokud má program vyhodnotit podmínku, která může mít více výstupů, je vhodné použít programovou konstrukci `Select Case` (vybrat případ – situaci), která umožňuje přehledné strukturování možných variant výstupů.

Příklad: Přidělení slevy podle počtu jednotek

```
Sub VyberSlevu()
 'Deklarace proměnných
 Dim Pocet As Integer
 Dim Sleva As Single
 'Zadání hodnoty vstupní proměnné - InputBox
 Pocet = InputBox("Zadej počet jednotek", "Zadání")
 'Výběr z více výsledků vyhodnocení podmínky - konstrukce Select Case
 Select Case Pocet 'Volba podle hodnoty vstupní proměnné
    Case 0 To 1: Sleva = 0
    Case 2 To 5: Sleva = 0.05
    Case 6 To 9: Sleva = 0.08
    Case Is >= 10: Sleva = 0.1
 End Select
 'Zobrazení hodnoty výstupní proměnné (vynásobené 100 s %) - MessageBox
 MsgBox "Sleva je " & 100 * Sleva & " %.", vbOKOnly, "Vyhodnocení"

End Sub
```

5.3.5.3 Ošetření běhových chyb

Při programování je třeba ošetřit možné výskyty běhových chyb (například dělení nulou nebo zadání hodnoty neodpovídající datovému typu proměnné). V případě výskytu chyby musí program vhodně reagovat, nejlépe přechodem k jiné části programu nebo předčasným ukončením procedury. Využíváme zde programovou konstrukci `On Error GoTo …`

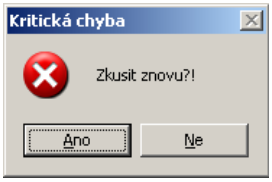
Příklad: Ověření správnosti typ zadané hodnoty

```
Sub Kontrola()
Dim Cislo As Single
On Error GoTo Chyba 'dojde-li k chybě, pokračuj na řádku Chyba:

    Cislo = InputBox("Zadej číselnou hodnotu:")
    MsgBox "Hodnota je v pořádku"
    Exit Sub 'předčasný konec procedury

Chyba:
    MsgBox "Nezadal jsi číslo!"
End Sub
```

5.3.5.4 Doplnění dalších údajů k příkazům InputBox a MsgBox.

Okna Windows lze doplnit o vhodný text v titulkovém pruhu a o různé ikony a tlačítka. Jednotlivé parametry oddělujeme čárkou, pokud chceme zobrazit více tlačítek a ikon, oddělujeme je plusem.

- `vbOKOnly` = pouze tlačítko OK,
- `vbOKCancel` = tlačítka OK a Storno,
- `vbYesNo` = tlačítka Ano a Ne,
- `vbCritical` = ikona Stop,
- `vbExclamation` = ikona Varování,
- `vbInformation` = ikona Informace.



Příklad

```
MsgBox "Zkusit znovu?!", vbYesNo + vbCritical, "Kritická chyba"
```

Příklad: Výpočet věku s ošetřením a rozlišením běhových chyb

```
Sub SpoctiVek()
Dim DatumNarozeni As Date 'deklarace proměnné - datum a čas
Dim Vek As Byte 'deklarace proměnné - celé číslo 0 až 255
Dim Okno As Integer 'deklarace proměnné - celé číslo

On Error GoTo Chyba 'dojde-li k chybě, pokračuj na řádku Chyba:

Zadani:
   DatumNarozeni = InputBox("Zadej datum narození: ")
   Vek = (Date - DatumNarozeni) \ 365.25 'celočíselné dělení
   MsgBox Vek, , "Věk"
   Exit Sub 'předčasný konec procedury
Chyba:
    If Err.Number = 13 Then 'nesoulad datových typů (type mismatch)
       Okno = MsgBox("Zkusit znovu?!", _
            vbYesNo + vbCritical, "Kritická chyba") 'ikona stop
       If Okno = vbYes Then
            Resume 'vrať se zpět (tam kde došlo k chybě) a zkus příkaz znovu
       End If
    ElseIf Err.Number = 6 Then 'přetečení (overflow)
       MsgBox "Neexistující datum!", vbCritical, "Kritická chyba"
       Resume Zadani 'vrať se zpět na řádek Zadani:
    Else 'všechny ostatní chyby
       MsgBox "Chyba!" & Err.Number & vbCr & Err.Description, _
       vbCritical, "Kritická chyba" ' _ zápis na dva řádky
    End If
End Sub
```

5.3.5.5 FOR cyklus

Cyklus se známým počtem opakování řešíme programovou konstrukcí `For` / `Next`.

Syntaxe

```
Dim i As Integer
   příkazy před cyklem
For i = 1 To 10 Step 1        (pokud není zadán krok, je automaticky 1)
   příkazy v cyklu
Next i
   příkazy za cyklem
```

Příklad: Výpočet součtu a průměru n čísel

```
Sub SoucetPrumer()
Dim i As Byte
Dim n As Byte
Dim Suma As Single
Dim Plus As Single


    n = InputBox("Zadej počet čísel:")
    Suma = InputBox("Zadej 1. číslo:", "Výpocet součtu n čísel")
    For i = 2 To n 'For cyklus ... začátek
        Plus = InputBox("Zadej " & i & ". číslo:" _
            & vbCr & vbCr & "Mezisoučet je: " & Suma, "Výpocet součtu n čísel")
            'zobrazení mezisoučtu, změna textu v titulkovém pruhu
        Suma = Suma + Plus
    Next i 'For cyklus ... konec
    MsgBox "Součet " & i - 1 & " hodnot = " & Suma
    MsgBox "Průměr " & i - 1 & " hodnot = " & Suma / n
End Sub
```

5.3.5.6 Cykly s podmínkou

Cykly, které probíhají za předpokladu splnění nebo nesplnění určité podmínky řešíme programovou konstrukcí `Do` / `Loop`.

- **Cyklus s podmínkou na začátku**
  - `Do While / Loop`
  - `Do Until / Loop`
- **Cyklus s podmínkou na konci**
  - `Do / Loop While`
  - `Do / Loop Until`

Při použití While probíhá cyklus tak dlouho, dokud je podmínka splněna.
Při použití Until končí cyklus v okamžiku splnění podmínky.

Příklad: Přihlášení s ověřením hesla

Podmínka na začátku

```
Sub Prihlaseni()
Dim Heslo As String
Dim Pokus As Byte

    Pokus = 0 'nastavení výchozí hodnoty proměnné
    Heslo = InputBox("Zadej heslo:")

    Do While Heslo <> "ABC" 'cyklus s podmínkou na začátku
        Pokus = Pokus + 1
        If Pokus = 3 Then
            MsgBox "3 chybná hesla ... Konec!", vbCritical 'ikona stop
            Exit Sub 'předčasný konec procedury
        End If
        MsgBox "Chybné heslo!", vbExclamation 'varovná ikona
        Heslo = InputBox("Zadej správné heslo:")
    Loop 'konec cyklu
    MsgBox "Přihlášení OK.", vbInformation 'informační ikona
End Sub
```

Podmínka na konci

```
Sub Prihlaseni1()
Dim Heslo As String
Dim Pokus As Byte

    Pokus = 0 'nastavení výchozí hodnoty proměnné
    Heslo = InputBox("Zadej heslo:")
    Do 'začátek cyklu
        Pokus = Pokus + 1
        If Pokus = 3 Then
            MsgBox "3 chybná hesla ... Konec!", vbCritical 'ikona stop
            Exit Sub 'předčasný konec procedury
        End If
        MsgBox "Chybné heslo!", vbExclamation 'varovná ikona
        Heslo = InputBox("Zadej správné heslo:")
    Loop Until Heslo = "ABC" 'konec cyklu s podmínkou na konci
    MsgBox "Přihlášení OK.", vbInformation 'informační ikona
End Sub
```

5.3.6 Programování v prostředí Excelu

VBA lze využít k tvorbě maker, které pracují s jednotlivými listy a buňkami excelovských sešitů. Místo InputBoxů načítáme hodnoty přímo z buněk, a stejně tak výsledné hodnoty zobrazujeme v konkrétních buňkách místo MessageBoxů.

5.3.6.1 Adresace buněk pro získání nebo vložení hodnot

Jednotlivé části adresy – list, oblast buněk, konkrétní buňky (řádek, sloupec) oddělujeme tečkami.

- Vybrání buňky, nastavení aktuální oblasti
  - `ActiveSheet.Range("B12:G15").Select`
- Vložení výsledku vzorce do buněk
  - `Worksheets("List1").Range("A1, A3, A5").Formula = 1 + 1`
  - `Worksheets(1).Range("B1:B5").Formula = 1 + 1`
  - `ActiveSheet.Range("D5:H10").Cells(2, 2).Formula = 1 + 1`
  - `Selection.Formula = 1 + 1`
  - `Selection.Cells(2, 2)Formula = 1 + 1`
  - `ActiveCell.Formula = 1 + 1`
- Posun oproti konkrétní buňce
  - `Offset`(počet řádků, počet sloupců)
- Přiřazení hodnoty proměnné
  - `X = Worksheets(2).Cells(2, 5).Value`

Slovníček

Sheet … list	Formula … vzorec
Range … oblast	Value … hodnota
Cell … buňka	Select … vybrat
Active … aktivní	

5.3.6.2 Výpočty v Excelu

Příklad: Sečtení dvou čísel

```
Sub Hodnoty()
' Delkarace proměnných (celočíselných)
Dim A As Integer
Dim B As Integer
Dim Soucet As Integer
' Přiřazení hodnoty z buňky proměnné
A = ActiveSheet.Range("A3").Value ' z konkrétní buňky
B = Worksheets("List1").Cells(3, 2).Value ' z konkrétního listu, 3. řádek, 2. sloupec
Soucet = A + B
' Vložení hodnoty proměnné do buňky
ActiveCell.Formula = Soucet ' do aktivní
ActiveSheet.Range("C3").Formula = Soucet ' do konkrétní
End Sub
```

Příklad: Výpočet druhé mocniny hodnoty aktuální buňky, její zobrazení ve vedlejší buňce

```
Sub Mocnina1()
Dim Cislo, Mocnina As Long
Cislo = ActiveCell.Value 'Načtení hodnoty z aktuální buňky
Mocnina = Cislo ^ 2
ActiveCell.Offset(0, 1).Value = Mocnina 'Zobrazení hodnoty s posunem
End Sub
```

5.3.6.3 For Each cyklus

Pokud potřebujeme provést určitou akci pro všechny buňky v určité oblasti, požíváme cyklus opakování pro všechny objekty (For Each / Next).

Syntaxe

```
Dim bunka As Range
   příkazy před cyklem
For Each bunka In Selection
   příkazy v cyklu
Next bunka
   příkazy za cyklem
```

Příklad: Výpočet druhých mocnin ve vedlejších buňkách

```
Sub Mocnina2()
Dim Cislo, Mocnina As Long
Dim Bunka As Range
'For Each cyklus - opakování pro každou buňku v označené oblasti
For Each Bunka In Selection
    Cislo = Bunka.Offset(0, 0).Value 'Načtení hodnoty z aktuální buňky
    Mocnina = Cislo ^ 2
    Bunka.Offset(0, 1).Value = Mocnina 'Zobrazení hodnoty s posunem
Next Bunka 'Další buňka
End Sub
```

5.3.7 Pole

Pole je typ proměnné, která obsahuje skupinu hodnot, které identifikujeme pomocí indexu. Počáteční index (pokud ho nedefinujeme jinak) je 0.

5.3.7.1 Deklarace pole

Pole definujeme většinou mimo vlastní proceduru na začátku modulu. Pokud nechceme číslovat hodnoty v poli od nuly, je požadovaný způsob třeba nastavit opět na začátku modulu. Při deklaraci pole zadáváme počet hodnot pole v závorce.

Příklad

```
Option Explicit 'povinná deklarace proměnných
Option Base 1 ' indexy polí začínají od 1
Dim Hodnoty(6) As Single 'deklarace globální proměnné
```

5.3.7.2 Naplnění pole hodnotami

Při práci s polem používáme pro projití celého pole nejčastěji For cyklus. V každém kroku cyklu se provádí určitá činnost s jednou hodnotou pole, určenou postupně zvyšujícím se indexem.

Naplnění pole pomocí InputBoxů

```
Sub NaplnPole()
Dim i As Integer 'deklarace lokální proměnné

   For i = 1 To 6 'For cyklus - začátek
       Hodnoty(i) = InputBox("Zadej hodnotu " & i, "Zadání hodnot do pole")
       ' zadávání hodnot pomocí okna windows,
       ' změna textu v titulku
   Next i 'konec cyklu
End Sub
```

Naplnění pole z buněk Excelu

```
Sub NaplnPole_Excel()
Dim i As Long

    For i = 1 To 6
        Hodnoty(i) = Worksheets(2).Cells(i, 3).Value
        'načtení hodnot z buněk (list 2, sloupec 3)
    Next i
End Sub
```

5.3.7.3 Výpis hodnot pole

```
Sub VypisPole()
Dim i As Integer

    For i = 1 To 6
        MsgBox "Výpis " & i & ". hodnoty ... " & Hodnoty(i)
        'vypsání výsledku v okně windows
    Next i
End Sub
```

5.3.7.4 Výpočty s polem

Příklad: Sečtení hodnot pole

```
Sub SoucetPole()
Dim i As Integer
Dim Suma As Single

    Suma = 0
    For i = 1 To 6
        Suma = Suma + Hodnoty(i)
    Next i
    MsgBox "Součet pole = " & Suma
End Sub
```

Příklad: Určení maximální hodnoty pole a jejího indexu

```
Sub NajdiMax()
Dim i As Integer
Dim Max As Single
Dim k As Integer

    Max = Hodnoty(1)
    k = 1 'nastavení výchozí hodnoty proměnné
    For i = 2 To 6
        If Hodnoty(i) > Max Then
            Max = Hodnoty(i)
            k = i
        End If
    Next i
    MsgBox "Maximální hodnota v poli = " & Max & _
        " ... odpovídá " & k & ". hodnotě" ' _ zápis na dva řádky
End Sub
```

5.3.8 Programování funkcí

Funkce je podprogram, který obsahuje určité vstupní argumenty a vrací vypočtenou hodnotu. Programový kód ohraničujeme příkazy **Function** a **End Function**. Vstupní argumenty je nutno deklarovat. Pokud je argumentů více, oddělujeme je čárkou.

Příklad: Funkce pro výpočet tržeb

```
Function Trzba(Cena As Single, Mnozstvi As Integer) As Single
    'v závorce deklarace dvou argumentů
    'za závorkou datový typ výsledku funkce

    Trzba = Cena * Mnozstvi
End Function
```

5.3.8.1 Využití funkce v proceduře

Příklad: Výpočet tržeb s použitím Excelu

```
Sub Trzba_Excel()
Dim Cena As Single
Dim Mnozstvi As Integer

        Cena = Worksheets(2).Cells(2, 5).Value
        Mnozstvi = Worksheets(2).Cells(2, 6).Value
        'načtení hodnot argumentů funkce z buněk (list 2, E2 a F2)

        Worksheets(2).Cells(2, 8).Value = Trzba(Cena, Mnozstvi)
        'výpis výsledku funkce s dosazenými argumenty v buňce (list 2, H2)
End Sub
```

5.3.8.2 Zabudované funkce Excelu při programování vlastních funkcí

Pokud chceme využít při programování vlastních funkcí některé funkce Excelu, je třeba je zapsat ve tvaru `Application.funkce(argumenty)`. Tyto funkce je nutno zadávat v původních anglických tvarech, například místo ZAOKROUHLIT zapisujeme Round.

Příklad: Výpočet obsahu kruhu, zaokrouhleno na dvě desetinná místa

```
Function ObsahKruhu(Polomer As Single) As Single
    ObsahKruhu = Application.Round(Application.Pi() * Polomer ^ 2, 2)
End Function
```

5.3.8.3 Podmínky ve funkcích

Stejně, jako v procedurách, také ve funkcích lze využít podmínky a cykly.

Příklad: Vyhodnocení velikosti čísla

```
Function VyhodnoceniCisla(Cislo As Single) As String
    If Cislo > 0 Then  '1. podmínka
        VyhodnoceniCisla = "Číslo je kladné"  'přiřazení textové hodnoty proměnné
                                              'po splnění 1. podmínky
    ElseIf Cislo < 0 Then  '2. podmínka
        VyhodnoceniCisla = "Číslo je záporné"  'přiřazení textové hodnoty proměnné
                                               'po splnění 2. (variantní) podmínky
    Else
        VyhodnoceniCisla = "Číslo je nula"  'přiřazení textové hodnoty proměnné
                                            'při nesplnění předchozích podmínek
    End If
End Function
```

6 Objektově orientované programování

Objektově orientované programování (OOP) je založeno na vzájemném propojování různých objektů. Jednotlivé objekty patří do tříd (class), které programátor nadefinuje. Třída je vlastně vzor, který definuje, jak se má objekt této třídy chovat a jaké má mít vlastnosti (properties). Tento způsob programování využívá většina moderních programovacích jazyků (Visual Basic, Java, C++, C#, …)

6.1 Základní pojmy

- **Objekt** (Object) = instance určité třídy. Na základě jedné třídy může být vytvořeno libovolné množství objektů se vzájemně nezávislými vlastnostmi.
- **Rozhraní objektu** (Interface) = vnější tvář objektu. Uživateli nemusí záležet na tom, co se děje uvnitř objektu, ale stačí mu znát členy jeho třídy, které jsou přístupné externím procesům.
- **Vlastnost** (Property) = charakteristika objektu, jako je barva, velikost, jméno, …
- **Metoda** (Method) = činnost, která může být s objektem provedena (např. zobrazení výsledku v textovém poli).
- **Událost** (Event) = změna stavu objektu, na kterou umí objekt reagovat (např. kliknutí myši).

6.1.1 Projekt

Projektem nazýváme celý program v návrhovém režimu, který nakonec konvertujeme (kompilujeme) do .exe souboru. Jde o soubor formulářů, modulů, tříd, událostí, …

- **Formuláře** = okna pro komunikaci mezi uživatelem a programem.
- **Moduly** = programové balíky, obsahující obecné programy (procedury) používané v celé aplikaci (v různých formulářích).
- **Třídy** (Class) = programové definice jednotlivých typů objektů. Třída sama o sobě ještě není objektem, je pouze jakousi formou (vzorem), na základě které se objekt vytvoří.
- **Události** (Events) = akce, které jsou vyvolané uživatelem, operačním systémem, nebo samotnou aplikací.

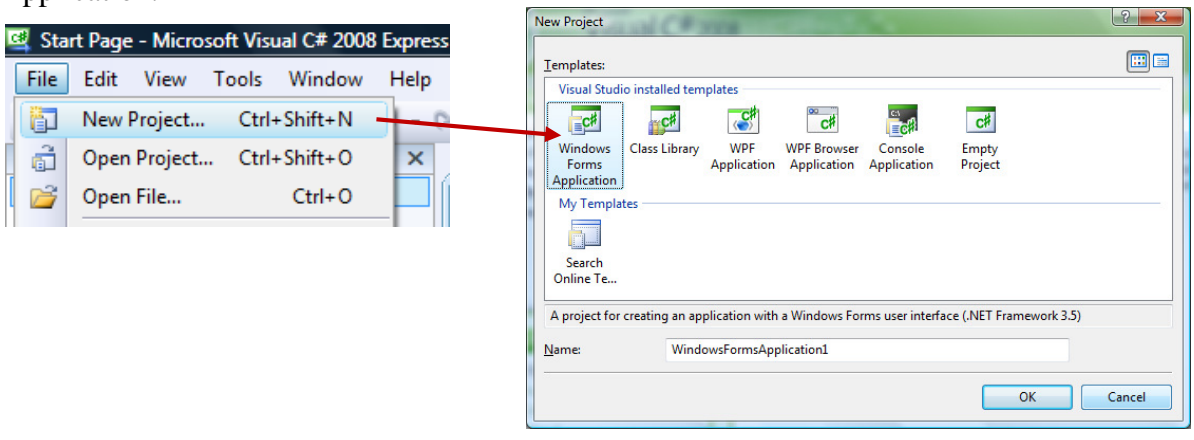
6.2 Úvod do programování v C#

Programovací jazyk C# (*sí šárp*) patří mezi současné vyšší programovací jazyky. Vyvinut byl firmou Microsoft. K programování v tomto jazyce využíváme Visual Studio, což je programovací prostředí, usnadňující tvorbu programového kódu. Vývojové prostředí je v angličtině.

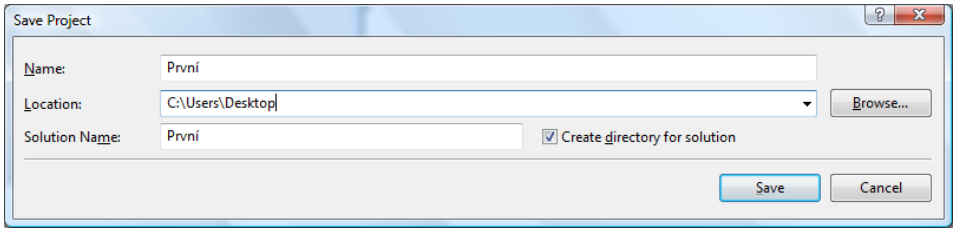
6.2.1 Postup programování ve Visual Studiu

Pro vytvoření nového programu pomocí jazyka C# aplikujeme následující postup:

- **Založení nového programu (projektu)** – File ➔ New Project ➔ Windows Forms Application.

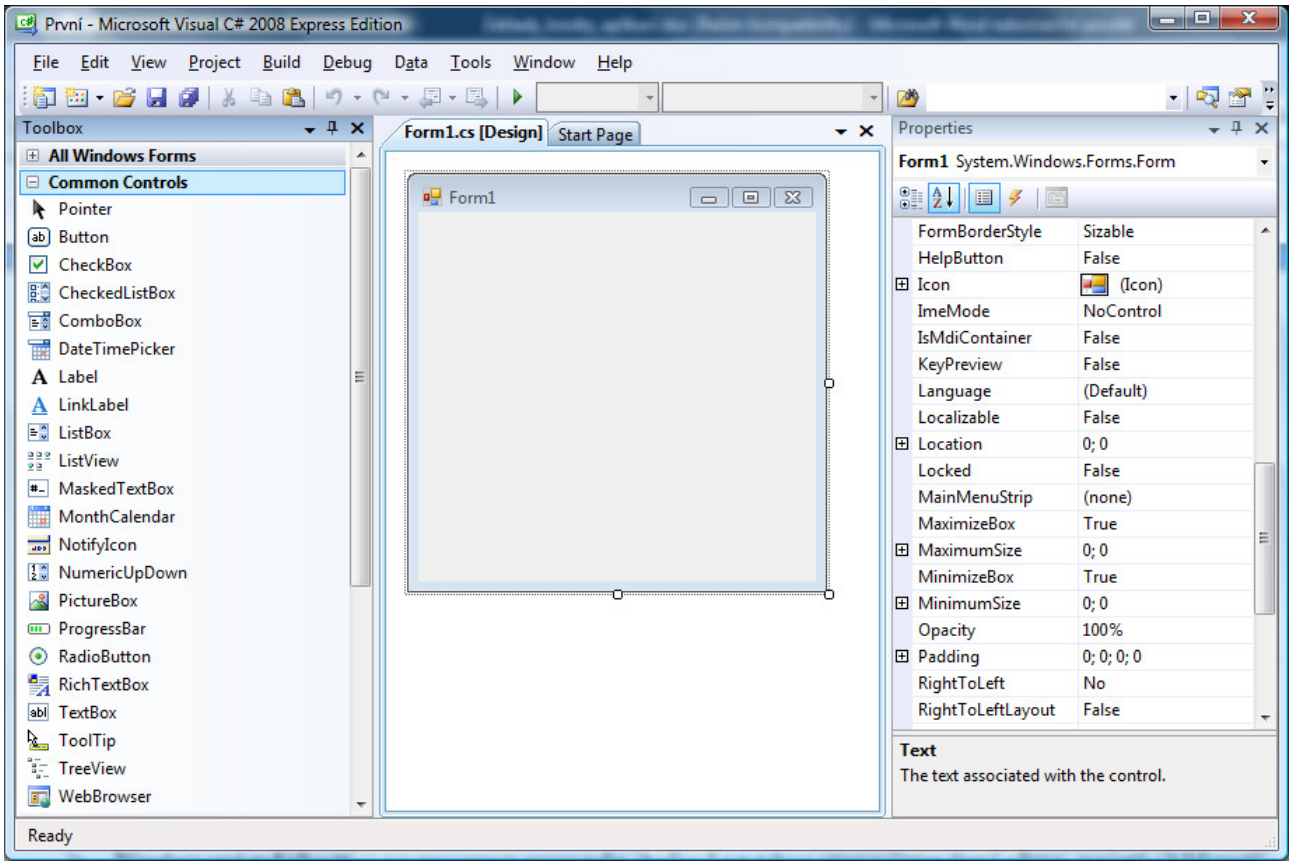


- **Pojmenování programu** – Name:

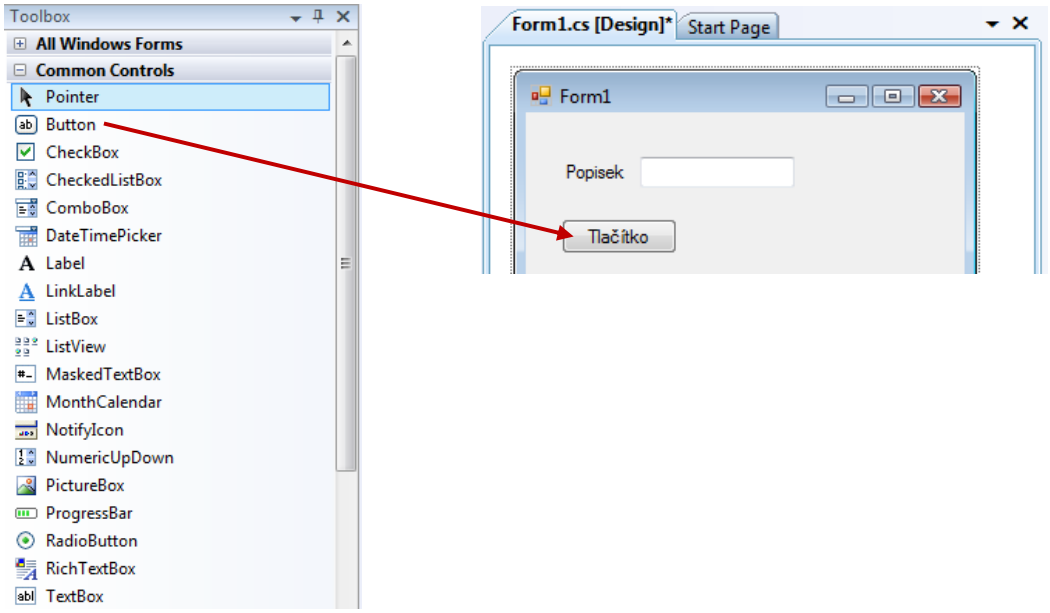


File ➔ Save All

- **Zobrazení panelu (podokna) nástrojů** – View ➔ Toolbox ➔ Common Controls (ovládací prvky).
- **Zobrazení podokna vlastností** – View ➔ Properties Window.



- **Tvorba ovládacích prvků na formuláři** – přetažení z ToolBoxu.



- **Nastavení vlastností (properties) ovládacích prvků**

- **Nastavení událostí** = co program provede, pokud se něco stane (otevření okna, najetí / kliknutí myší, změna stavu zatržítka, …).
    - V Properties Window přepnutí na události (Events) – ikona blesku.
    - Označení ovládacího prvku, kterému přiřazujeme událost.
    - Poklepání na zvolenou událost.
    - Vytvoření programového kódu, reagujícího na událost.
- **Vytvoření programu z návrhu** – Debug ➔ Start Debugging (F5).
    - Program se vytvoří a spustí.
    - Při chybě v kódu se objeví chybové hlášení Build failed (sestavování se nezdařilo); klikneme na tlačítko No a opravíme chyby (poklepáním na text chyby v okně Error List se přeneseme na výskyt příslušné chyby).
- **Uložení celého projektu** – File ➔ Save All.
    - Uloží se celá stromová struktura.
    - Spouštěcí soubory … *.csproj a *.sln
    - Soubory s kódem programu *.cs
      Form1.cs … soubor, který tvoříme (reakce na události),
      Form1.Designer.cs … soubor grafického vzhledu okna a ovládacích prvků,
      Program.cs
    - Spouštěcí .exe soubor … bin\Debug*.exe
- **Otevření existujícího projektu** – *.csproj nebo *.sln
    - Pokud se nezobrazí Designer, zobrazíme průzkumníka řešení View ➔ Solution Explorer ; dále PTM na Form1.cs ➔ Open.

6.2.2 Ovládací prvky formulářů

Ovládací prvky programů Windows Forms představují jednotlivé objekty v grafickém uživatelském rozhraní (v oknech operačního systému Windows).

- TextBox = textové pole.
- Label = popisek.
- CheckBox = zatržítko (zaškrtávací políčko).
- Button = tlačítko.
- NumericUpDown = číselník.
- RadioButton = přepínač.
- Containers ➔ GroupBox = skupinový rámeček.

6.2.3 Vlastnosti (Properties)

Každý ovládací prvek musí mít své pojmenování **(Name)** tak, aby s ním mohl program pracovat a dále má určité specifické vlastnosti podle typu objektu. Jednotlivé vlastnosti nastavujeme v okně Properties při vybraném konkrétním objektu.

6.2.3.1 Vlastnosti formulářového okna

- **MinimizeBox** = povolení / zákaz minimalizace okna … (True / False).
- **MaximizeBox** = povolení / zákaz maximalizace okna … (True / False).

- **FormBorderStyle** = nastavení chování okrajů okna (např. zákaz změny rozměrů okna = FixedSingle).
- **Size** = přesné rozměry okna.
- **Text** = text v titulkovém pruhu okna.

6.2.3.2 Vlastnosti textového pole

- **BackColor** = barva pozadí.
- **ForeColor** = barva popředí (písma).
- **Font** = parametry písma:
    - Name = název fontu (např. Times New Roman),
    - Size = velikost,
    - Unit = jednotka velikosti (Pixel, Point, Inch, Millimeter, …),
    - Bold = tučné písmo (True / False),
    - Italic = kurzíva (True / False),
    - Strikeout = přeškrtnutí (True / False),
    - Underline = podtržení (True / False).
- **BorderStyle** = styl ohraničení prvku.
- **MaxLength** = maximální počet znaků v textovém poli.
- **Text** = přednastavený text.
- **TextAlign** = zarovnání textu (Left / Right / Center).
- **Visible** = viditelnost textového pole … True / False.
- **Enabled** = aktivní nebo neaktivní prvek (zda lze do textového pole zapisovat) … True / False.
- **ReadOnly** = zákaz ručního zadávání dat do ovládacího prvku (zda lze do textového pole zapisovat) … True / False
- **Multiline** = víceřádkové textové pole.

6.2.3.3 Vlastnosti popisků

- **Text** = text popisku.
- **BackColor** = barva pozadí.
- **ForeColor** = barva popředí (písma).
- **Font** = parametry písma (stejné jako u textového pole).
- **Visible** = viditelnost popisku … True / False.

6.2.3.4 Vlastnosti tlačítek

- **Text** = text na tlačítku.
- **BackColor** = barva tlačítka (pozadí).
- **ForeColor** = barva popředí (písma).
- **Font** = parametry písma (stejné jako u textového pole).
- **Size** = přesné rozměry tlačítka.
- **Visible** = viditelnost tlačítka … True / False.
- **Enabled** = aktivní nebo neaktivní prvek (zda lze na tlačítko kliknout) … True / False.

6.2.3.5 Vlastnosti zatržítek

- **Text** = text u zatržítka.
- **Checked** = výchozí stav zatržení … True / False.

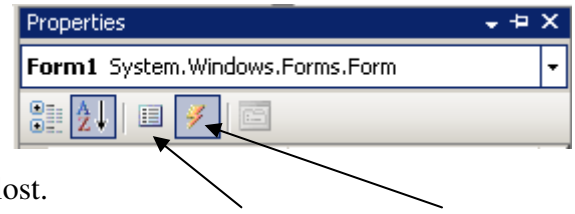
6.2.4 Uspořádání ovládacích prvků

- Format ➔ Center in Form ➔ … = centrování v okně,
- Format ➔ Align ➔ … = vzájemné zarovnání jednotlivých prvků,
- Format ➔ Horizontal spacing ➔ … = vodorovné rozmístění jednotlivých prvků (vzájemné),
- Format ➔ Vertical spacing ➔ … = svislé rozmístění jednotlivých prvků (vzájemné).

6.3 Programování obslužných programů událostí v C#

Každá činnost programu musí být určitým způsobem vyvolána (aktivována). K tomu slouží události, na které potom reagují metody.

6.3.1 Události (Events)



- V Properties Window přepnutí na události (Events) – ikona blesku.
- Označení ovládacího prvku, kterému přiřazujeme událost.
- Poklepeme na zvolenou událost.
- Pokud chceme přiřadit určitému prvku existující proceduru, nepoklepáváme na událost, ale vybereme požadovanou proceduru ze seznamu.

6.3.1.1 Vybrané události

- **Click** = stisknutí tlačítka (myší, Enterem, mezerníkem).
- **MouseEnter** = najetí myší.
- **MouseHover** = najetí myší a zastavení.
- **MouseLeave** = opuštění myší.
- **MouseDown** = stisk tlačítka myši.
- **MouseUp** = puštění tlačítka myši.
- **KeyDown / KeyUp** = stisknutí / uvolnění klávesy.
- **TextChange** = změna v textovém poli.
- **CheckedChanged** = změna zatržení zaškrtávacího políčka.
- **Load** = otevření okna programu.
- **Resize** = změna velikosti okna (i po obnovení z minimalizace).
- …

6.3.2 Úvod do programování metod

Programování metod představuje vytvoření programového kódu (procedury), reagujícího na určitou událost. Pokud se kód nezobrazuje, zvolíme v řádku nabídek View ➔ Code (F7).
Chceme-li přejmenovat existující událost, řešíme to v kódu kliknutím pravým tlačítkem myši na událost a volbou Refraktor ➔ Rename.
Pro odebrání události klikneme v Events pravým tlačítkem myši na událost ➔ Reset

6.3.2.1 Syntaxe programového kódu C#

- Podprogramy (procedury) uzavíráme do složených závorek `{ }`.
- Komentáře:
  - jednořádkové: `// komentář`
  - víceřádkové: `/* komentář */`
- Rozlišují se malá a velká písmena !!!
- Každý příkaz ukončujeme středníkem.

- Přiřazení vlastnosti objektu: `JménoObjektu.Vlastnost=Hodnota;`

  Příklady

  ```
  Pozdrav.Text="Ahoj";
  Pozdrav.ForeColor=Color.Red; nebo =Color.FromArgb(255,0,0);
  ```

- Změna vlastnosti celého okna: `Vlastnost=Hodnota;`

  Příklad

  ```
  BackColor=Color.Yellow;
  ```

- Obecná syntaxe přiřazování: kam = co
- Přiřazení vlastnosti jednoho objektu druhému:
  `CílovýObjekt.Vlastnost=ZdrojovýObjekt.Vlastnost;`

  Příklad

  ```
  TextNovy.ForeColor=TextPuvodni.ForeColor;
  ```

6.3.2.2 Příklady kódů

- Aktivování určitého prvku při spuštění okna – v kódu události Load procedura `NázevPrvku.Select();`
- Zavření okna – `Close ();`
- Zobrazení textu v novém okně Windows – `MessageBox.Show(" ... ");`

6.3.3 Definování pořadí jednotlivých ovládacích prvků

- Nastavení pořadí u vlastnosti TabIndex (v podokně Properties).
- **Hromadné nastavení indexů:**
  - View ➔ Tab Order
  - postupné klikání LTM na dílčí objekty
- Pohyb po jednotlivých ovládacích prvcích klávesou Tab.
- **Nastavení tlačítka pro klávesu Enter** = vlastnost okna AcceptButton.
- **Nastavení tlačítka pro klávesu Esc** = vlastnost okna CancelButton.

6.3.4 Datové typy

Každé proměnné je třeba před jejím použitím přiřadit datový typ (jaké hodnoty v ní budou uloženy). Při deklaraci proměnných a jejich datových typů je třeba dávat pozor na velikost písmen.

- Textový řetězec = **string**
- Logická hodnota (True / False) = **bool**
- Celé číslo = **int**, **Int32**
- Desetinné číslo = **double**, **float**
- Barva = **Color**
- Tlačítko = **Button**
- Výčet hodnot pro rozbalovací seznam ScrollBars = **ScrollBars**

6.3.4.1 Deklaraci proměnných

Deklarace proměnných je platná pouze v rámci příslušné procedury reagující na příslušnou událost.

Syntaxe

```
DatovýTyp NázevProměnné
```

Příklady

```
int Cislo;
string Jmeno;
Color BarvaPodkladu;
```

6.3.4.2 Převod (konverze) datových typů

Mnohdy potřebujeme pro výpočty pracovat s číselnými hodnotami, které jsou zadávány například pomocí textových polí, kde se vyskytují ve formě textu, nebo naopak chceme číselný výsledek zobrazit v textové formě.

- `Convert.ToInt32(PřeváděnáHodnota)`
- `Convert.ToDouble(PřeváděnáHodnota)`
- `Convert.ToString(PřeváděnáHodnota)`

Příklady

```
Cislo=Convert.ToDouble(PoleCislo.Text);
PoleCislo.Text=Convert.ToString(Cislo);
PoleCislo.Text=Cislo.ToString("N2"); N2 = oddělovač tisíců, 2 des. místa
```

6.3.4.3 Deklarace proměnné s inicializací:

V tomto případě se jedná o deklarace s přiřazením konkrétní hodnoty. Tímto způsobem často deklarujeme konstanty.

Příklad

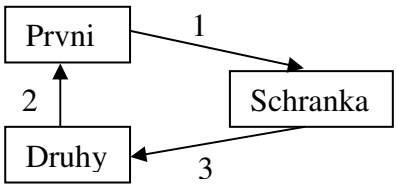
```
double pi=3.141592;
```

6.3.5 Záměna hodnot dvou proměnných

Pro vzájemnou výměnu hodnot proměnných je třeba využít pomocnou proměnnou, která bude sloužit k dočasnému uchování nahrazované hodnoty (čísla, textu, barvy).

Příklad

```
String Schranka;
Schranka=První.Text;
První.Text=Druhy.Text;
Druhy.Text=Schranka;
```



6.3.6 Matematické funkce

Syntaxe

```
Math.funkce(argumenty)
```

Příklad: Druhá odmocnina

```
Math.Sqrt(a);
```

6.3.7 Ošetření běhových chyb

Nastane-li při ladění programu chyba, je třeba toto ladění ukončit volbou Debug ➔ Stop Debuging ⊡
Dále lze programový kód ošetřit tak, aby při vzniku chyby reagoval program vhodným způsobem (například zobrazením chybového hlášení a požadavkem o opravu chybně zadané hodnoty).

- Označení bloku programu { }
- Kliknutí pravým tlačítkem myši, dále volba Surround With ➔ Try

- Do části `catch { }` zadat alternativu při vzniku chyby

```
try
  { blok programu
  }
catch
  { alternativa
  }
```

Příklad: `MessageBox.Show("Chybná hodnota")`

- Rozlišení typu chyby
  - chyba formátu – catch(FormatException, ex)
  - přetečení – catch(Overflow, ex)
  - dělení nulou – catch(DevideByZeroException, ex)

6.3.8 Podmínky

Podmínky využíváme k větvení programu, kdy je určitý příkaz proveden pouze za předpokladu splnění určité podmínky (výraz v podmínce nabývá hodnoty True).

- `if (podmínka)`
  jeden příkaz;
- `if (podmínka)`
  `{` více příkazů;
  `}`

  pokud je více příkazů, uzavíráme je do složených závorek `{ }`
- **Spojování podmínek**
  || … sjednocení (or)
  && … průnik (and)
  ! … negace

Příklady

`if (podmínka 1 || podmínka 2)` nebo
`if (podmínka 1 && podmínka 2)` nebo
`if (! podmínka)` … provede se při hodnotě podmínky False

- **Podmínění části programu** = označení bloku programu; kliknutí pravým tlačítkem myši a volba Surround Width ➔ if, dále úprava podmínky.

Příklad

```
if(!blokovaci.Checked)
{ blok programu
}
```

blok programu se provede, pokud není zaškrtnuto zatržítko

- **Variantní rozhodování** = různé příkazy při vyhodnocení podmínky jako True nebo False.

```
if (podmínka)
  příkaz;
else
  alternativní příkaz;
```

6.3.9 FOR cyklus

- Opakování akce n-krát
- **Zabalení do cyklu** = označení bloku programu; kliknutí pravým tlačítkem myši a volba Surround Width ➔ For, dále úprava kódu.

```
for (inicializace výchozí hodnoty; podmínka pro opakování; změna hodnoty počitadla)
{ opakující se část programu
}
```

Příklad: FOR cyklus pro 10 opakování

```
for(int Pocet=0; Pocet<10; Pocet ++)
{ opakující se část programu
}
```

6.4 Programování grafiky v C#

Kromě výpočtů lze programovat také grafické prvky aplikací. Objektem pro kreslení může být celé okno (Window) nebo Container ➔ Panel.

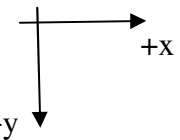
- **Událost Paint** = provede se vždy při zobrazení okna (spuštění, odkrytí, obnovení z minimalizace).
- **Událost Resize** = změna velikosti okna.

  `Refresh ()` = metoda, zajišťující překreslení obsahu okna.

  Při programování grafických objektů je vhodné zakázat změny rozměrů okna = nastavení vlastnosti **MaximizeBox** na **False** a **FormBorderStyle** na **FixedSingle**.
- **Vytvoření kreslící plochy** = deklarace proměnné s inicializací (datový typ Graphics).

  `Graphics KresliciPlocha = e.Graphics;`
- **Zjištění rozměrů okna / kreslící plochy** = slouží pro zadávání maximálních rozměrů a rohů.
  - `int SirkaOkna = ClientSize.Width;`
    `int VyskaOkna = ClientSize.Height;`
  - `int SirkaPlochy = Platno.ClientSize.Width;`
    `int VyskaPlochy = Platno.ClientSize.Height;`
- **Zadání bodu uživatelem** = určení souřadnic bodu pomocí textových polí (TextBoxů).
  - `int X = Convert.ToInt32(PoleX.Text);`
    `int Y = Convert.ToInt32(PoleY.Text);`

6.4.1 Základní objekty



- **Úsečka** = zadáváme barvu pera a souřadnice počátečního a koncového bodu.

  `KresliciPlocha.DrawLine(Pens.Blue,`x_1`,` y_1`,` x_2`,` y_2`);`
- **Obdélník** (nebo **čtverec**) = zadáváme barvu pera, souřadnice jednoho rohu a přírůstky.

  `KresliciPlocha.DrawRectangle(Pens.Blue,` x_1`,` y_1`, Δx, Δy);`
- **Elipsa** (nebo **kružnice**) = zadáváme barvu pera a opsaný obdélník.

  `KresliciPlocha.DrawEllipse(Pens.Blue,` x_1`,` y_1`, Δx, Δy);`
- **Vybarvený objekt** (obdélník nebo elipsa) = místo `Draw` zadáváme `Fill` a místo pera (`Pens`) zadáváme štětec (`Brushes`).
- **Deklarace proměnné pro barvu pera nebo štětce**
  - `Pen barva = Pens.Blue;`
  - `Brush vypln = Brushes.Green;`
- **Vytvoření nového objektu** (pera) = nová instance třídy Pen.
  - `Pen NovePero = new Pen(Color.Red, 8);`

  Příklad použití nového objektu (pera)
  - `KresliciPlocha.DrawLine(NovePero,`x_1`,` y_1`,` x_2`,` y_2`);`

6.4.2 Animace

Animace představuje rozpohybování grafických objektů. Programování animací spočívá v nastavení časovače (v jakých časových intervalech se bude obraz překreslovat) a dále ve změně referenčních souřadnic grafických objektů při každém překreslení.

- **Nastavení časovače** = ToolBox ➔ Components ➔ Timer
    - přetažení Timeru do okna formuláře; zobrazí se v dolní části
    - v Properties ➔ Enabled = True (rozběh časovače)
      Interval [ms] = 1000 (po sekundách)
- **Deklarace globálních proměnných** = aby si program pamatoval aktuální souřadnice, je třeba je deklarovat mimo vlastní proceduru.
- **Nastavení události časovače** (Timeru) = Events ➔ Tick

  V kódu nastavení změny určitých proměnných, například `X += 5;`
- Aby formulář neblikal, nastavujeme u něj vlastnost DoubleBuffered na True,

  aby nebyl objekt při animaci aktivní, nastavujeme mu vlastnost Tabstop na False.
- **Spuštění animace** (například tlačítkem) = pro výchozí nastavení časovače Enabled = False programujeme v události tlačítka Click změnu této vlastnosti `Casovac.Enabled = true;`
- **Zastavení animace** (například tlačítkem) = v události tlačítka Click programujeme změnu vlastnosti Enabled `Casovac.Enabled = false;`
- **Zabudování obrázku do programu** = Project ➔ Animace properties ➔ Resources ➔ Add Resource ➔ Add Existing File, …
    - událost Paint formuláře

```
Graphics Platno = e.Graphics;
Platno.DrawImage(Properties.Resource.Obrazek, x, y);
Refresh();
```

7 Úvod do webových aplikací

7.1 Postup při tvorbě webu

- Ujasnění si cíle a smyslu webu.
- Sběr informací, které budou tvořit počáteční obsah webu.
- Utřídění informací, vytvoření informační architektury.
- Promyšlení systému odkazů, návrh navigačního designu.
- Návrh grafického designu (nejdříve na papír, potom v grafickém programu).
- Optimalizace grafiky, aby byla co nejmenší.
- Napsání a naprogramování zdrojového kódu jednotlivých webových stránek.
- Otestování funkčnosti a intuitivnosti webu.
- Umístění webu na síť Internet a otestování funkčnosti v různých webových prohlížečích (Internet Explorer, Opera, Mozila Firefox, Google Chrome, ...).

7.2 Prvotní informace

Při prvotní analýze je potřeba zjistit, co by měl budoucí web (webová prezentace nebo webová aplikace) dělat, a jaké funkce by měl obsahovat.

- Co je cílem webu?
- Pro jakou cílovou skupinu uživatelů je web určen?
- Jaké informace by měl web obsahovat?
- Zda, a kde budou informace přibývat?
- Zda existují nějaké speciální požadavky na web?

7.2.1 Charakteristika cílových skupin uživatelů webu

Vzhled a vlastnosti webových prezentací a aplikací je nutné přizpůsobit specifikům předpokládaných cílových uživatelů. Zejména je nutno zohlednit:

- demografické znaky (věk, pohlaví, rodinný stav, povolání, …),
- životní styl (návyky, zájmy, zkušenosti, …),
- psychologické znaky (charakter člověka, jeho názory, …),
- počítačová vyspělost (počítačové znalosti, velikost monitoru, operační systém, typ prohlížeče a jeho verze).

7.3 Informační architektura

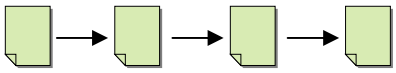
Cílem informační infrastruktury je uspořádat a zorganizovat informace na webu tak, aby mohly být využity uživatelem co nejefektivněji. Základní zásadou při jejím budování je vyjít vstříc požadavkům a přáním uživatele.

V případě rozsáhlejších webů je potřeba informace roztřídit do několika kategorií, což jsou tematicky podobné skupiny informací. Pokud je těchto kategorií více než 7, může se informační struktura stát nepřehlednou.

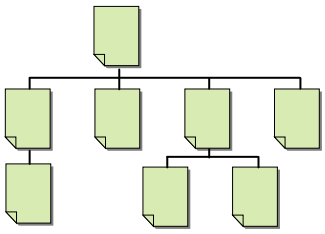
Při budování informační architektury je třeba stanovit hierarchii důležitosti jednotlivých kategorií a jednotlivých textů. Tuto hierarchii vytváříme shora od nejdůležitějších a obecných informací dolů k méně důležitým a detailním.

7.3.1 Typy informačních architektur

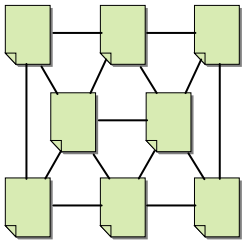
- **Posloupnost** (lineární struktura) = jednotlivé webové stránky následují postupně za sebou. Příkladem může být průvodce určitou činností.



- **Hierarchie** = webové stránky se postupně větví. Klasickým příkladem je menu, kde každá jeho položka směřuje odkaz na jinou část webu.



- **Pavučina** = neuspořádaná struktura odkazů.



7.4 Navigační design

Navigace na webu uživateli umožňuje:

- přemístit se za určitým cílem z místa na místo,
- zjistit, kde se právě nachází,
- uvědomit si všechny směry, kudy a kam může jít,
- pochopit, co všechno je na webu za informace.

7.4.1 Typy navigace

- **Základní (celková, globální) navigace** = nabídka, umožňující uživateli pohyb mezi hlavními oblastmi webu. Měla by být k dispozici na každé dílčí stránce webu.
- **Místní (lokální) navigace** = odkazy na informace, které se rozvíjejí z aktuální stránky.
- **Doplňková navigace** = odkazy na informace, které s aktuální informací v hierarchii nemusí přímo souviset.
- **Kontextová (související) navigace** = zpřístupňuje informace, které s aktuální informací přímo souvisí.
- **Zvyková navigace** = odkazy, na které jsou uživatelé zvyklí (např. kontaktní informace).
- **Externí navigace** = odkazy na jiné weby ven z aktuálního webu.
- **Mapa serveru** = vizuální vyjádření informační struktury webu.
- **Vyhledávání** = umožňuje prohledávání informací v rámci daného webu.
- **Rejstřík** = seznam klíčových slov s odkazy na jejich výskyt.

7.5 Použitelnost webu

Použitelnost je vlastnost webu, která charakterizuje schopnost uživatele tento web používat. Jde o míru jednoduchosti jeho ovládání.

7.5.1 Klíčové vlastnosti použitelnosti

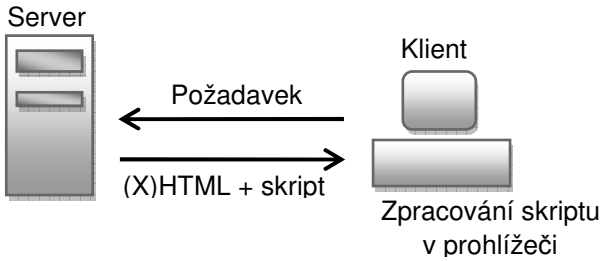
- **Pochopitelnost** = jak lehce je návštěvník schopen vykonat základní úkony při první návštěvě webu.

- **Efektivita** = jak rychle je návštěvník schopen provést požadované úkony, když se na webu již zorientoval.
- **Zapamatovatelnost** = jak lehce se návštěvník opět zorientuje při pozdější návštěvě webu.
- **Chybování** = jak moc návštěvník na webu chybuje a jak jsou tyto chyby závažné.
- **Subjektivní uspokojení** = jak příjemné je pro uživatele používání webu.

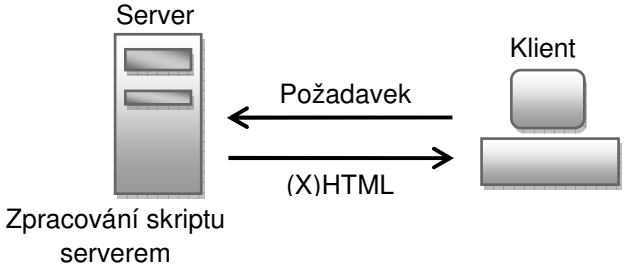
7.6 Tvorba www stránek

Webové (www) stránky jsou vytvářeny v několika programovacích jazycích:

- HTML (Hypertext Markup Language) – hypertextový značkovací jazyk pro tvorbu obsahové stránky webu,
- XHTML (eXtensible Hypertext Markup Language) – nástupce jazyka HTML, založený na jazyku XML (eXtensible Markup Language), který kromě pevně definovaných značek může vytvářet i vlastní značky (X znamená rozšiřitelný),
- CSS (Cascading Style Sheets) – kaskádové styly pro vytvoření grafického vzhledu webu,
- Skriptovací jazyky pracující na straně klienta (Java Script, VB Script, …),



- Skriptovací jazyky pracující na straně serveru (PHP, Asp.Net, Perl, Python, …).



7.7 Domovská stránka webu

Hlavní www stránka, kterou zobrazí webové prohlížeče při spuštění jako první má název **index.html** nebo **default.html** a musí být umístěna v kořenové složce webového serveru.

8 Tvorba www stránek v HTML

8.1 Základní struktura (X)HTML dokumentu

Každá webová stránka obsahuje následující části:

- Typ a verze dokumentu (Doctype).
- Hlavička www stránky (head).
- Tělo www stránky (body).

8.1.1 Typ a verze dokumentu

Tato první část každé www stránky definuje programovací jazyk, ve kterém je tato stránka vytvořena a jeho verzi, případně také odkaz na umístění zdrojového souboru tohoto jazyka.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
„http://www.w3c.org/TR/html4/loose.dtd“>
```

nebo

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
„http://www.w3c.org/TR/xhtml1/DTD/xhtml1-strict.dtd“>
```

8.1.2 Vlastní www stránka

Hlavička i tělo dokumentu jsou uzavřeny v párovém tagu (značce) html. Hlavička obsahuje titulek www stránky a metainformace určené prohlížeči a vyhledávacím robotům.

```
<html>
  <head>
     <meta http-equiv="Content-Type" content="text/html;
     charset=windows-1250" />
     další metatagy
     <title> název www stránky </title>
  </head>
  <body>
     vlastní obsah www stránky
  </body>
</html>
```

8.2 Druhy tagů (značek)

Jednotlivé tagy (značky) uzavíráme do špičatých závorek `< >`, ukončení tagu `/`.

- **Párové tagy** – ohraničují nějaký obsah (např. odstavec, text s určitým formátem, odkaz, …).
  Syntaxe: `<tag parametr1="hodnota" parametr2="hodnota" …>` text `</tag>`
  Příklad: `<p align="left">` Ahoj `</p>`
- **Nepárové tagy** – vkládají určitý objekt (např. obrázek nebo horizontální linku) na konkrétní místo.
  Syntaxe: `<tag parametr1="hodnota" parametr2="hodnota" … />`
  Příklad: `<img src="domek.jpg" alt="muj hrad" width="160" />`

8.3 Jednotky

Hodnoty jednotlivých parametrů se mohou být v různých jednotkách.

Vzdálenost: pixel (obrazový bod) [px]
% šířky nebo výšky stránky

Barvy: triviální anglické názvy (např. blue, red, yellow, …)

hexadecimální vyjádření RGB – #rrggbb (2^{24} = 16,7 milionu různých barev)

URL: relativně (od aktuálního místa), například ../images/foto.jpg
absolutně (celá cesta od místa uložení), například www.server.cz/images/foto.jpg

Velikost písma: 1 = 8 b, 2 = 10 b, 3 = 12 b, 4 = 14 b, 5 = 18 b, 6 = 24 b, 7 = 36 b
72 b = 1“ = 2,54 cm

8.4 Komentáře

Komentáře slouží jako doprovodné informace ke zdrojovému kódu, které ho vysvětlují.

Syntaxe: `<!-- komentář -->`

8.5 Tag <body>

Ohraničuje obsahovou část (tělo) každé www stránky, která se zobrazuje v okně prohlížeče.

Základní parametry: Barva podkladu = `bgcolor` [triviální název nebo #rrggbb]
Obrázek jako podklad = `background` [URL]
Barva písma = `text` [triviální název nebo #rrggbb]
Levý a pravý okraj stránky = `leftmargin` [px]
Horní a dolní okraj stránky = `topmargin` [px]

8.6 Základní tagy pro zobrazení textu

Text vždy uzavíráme do párových tagů (nadpis, odstavec), případně ještě do párových formátovacích tagů (tučné písmo, kurzíva, velikost písma, …). Formátování v HTML kódu se v současnosti již nedoporučuje, k nastavení vzhledu slouží kaskádové styly CSS.

- **Nadpisy** (heading): `<h1>` až `<h6>`, kde `<h1>` je hlavní (největší) nadpis, `<h2>` je menší nadpis, …

Vlastnosti jednotlivých nadpisů většinou nastavujeme v kaskádových stylech. V HTML nejčastěji nastavujeme parametr zarovnání = `align` (left, center, right).

- **Odstavce** (paragraph): `<p>`

Vlastnosti jednotlivých odstavců většinou nastavujeme v kaskádových stylech. V HTML nejčastěji nastavujeme parametr zarovnání = `align` (left, center, right, justify).

- **Zalomení řádku v rámci odstavce** řešíme nepárovým tagem brake `<br />`
- **Oddíly** `<div>` slouží k seskupení více nadpisů a odstavců do jednoho objektu, který lze souhrnně formátovat (např. pomocí stylů).
- **Řádkový element** `<span>` slouží k vytvoření objektu v rámci odstavce, umožňující dílčí formátování pouze této části.
- **Formátování:** tučné písmo (bold) `<b>`, kurzíva (italic) `<i>`, Velikost `<font size ="x">`, větší písmo `<big>`, menší písmo `<small>`, horní index (superscript) `<sup>`, dolní index (subscript) `<sub>`, neproporcionální písmo `<code>` nebo `<tt>` (teletyper).

Formátovací tagy se nesmí křížit (uzavíráme je v opačném pořadí).

Příklad: `<b><i>`Tučný text psaný kurzívou `</b></i>` … chybně !!!
`<b><i>`Tučný text psaný kurzívou `</i></b>` … správně.

8.6.1 Zápis speciálních znaků

Pokud potřebujeme zobrazit určitý znak, který není možno zadat z klávesnice, používáme zápis začínající znakem & a končící středníkem.

Příklady: © copyright … `&copy;`
& (ampersand) … `&`

< … <
> … >
“ (uvozovky) … "
€ (euro) … €
± … ±
libovolný znak … &#nnn; (nnn = ASCII kód)

8.6.2 Nedělitelná mezera ` `

Pokud chceme oddělit dvě slova tak, aby se na konci řádku nerozdělila, vytvoříme nedělitelnou mezeru ` ` (nonbrake space).
Pokud chceme v textu za sebou více mezer (více než jednu) používáme opět nedělitelné mezery, neboť prohlížeč interpretuje libovolné množství sousedících mezer vždy pouze jako jednu.
Pokud požadujeme, aby se určitý text vůbec nezalamoval, zadáváme ho do párového tagu `<nobr>` (no break).

8.7 Vodorovná linka (horizontal rule)

Vodorovnou (horizontální) oddělovací čáru vytvoříme nepárovým tagem `<hr />`, který můžeme ještě doplnit vhodnými parametry.
Vlastnosti (parametry) linky většinou nastavujeme v kaskádových stylech. V HTML nejčastěji nastavujeme parametry zarovnání – `align` (left, center, right), tloušťka čáry [px] – `size` a délka čáry [px nebo % šířky stránky] – `width`.
Příklad: `<hr width="80%" align="center" />`

8.8 Seznamy a výčty

Zobrazení určitého seznamu lze na www stránkách řešit pomocí odrážek `<ul>` (unordered list) nebo číslování `<ol>` (orderer list). Jednotlivé položky uzavíráme do tagu `<li>` (list item). U jednotlivých seznamů lze navíc pomocí parametru `type` změnit typ (tvar) odrážek nebo čísel. U číslovaných seznamů lze navíc změnit počáteční číslici pomocí parametru `start`.

Typy: disc … ●
square … ■
circle … ○

1 … 1. 2. 3.
A … A. B. C.
a … a. b. c.
I … I. II. III. IV.
i … i. ii. iii. iv.

Příklad:

```
<ul type="square">
      <li>První položka</li>
      <li>Další položka</li>
</ul>
```

Definiční výčty řešíme pomocí párových tagů `<dl>` (definition list) `<dt>` (definition term) a `<dd>` (description).
Příklad:

```
<dl>
      <dt>První pojem</dt>
            <dd>Definice pojmu</dd>
      <dt>Další pojem</dt>
            <dd>Definice pojmu</dd>
</dl>
```

8.9 Odkazy (hyperlinks)

K vytvoření odkazu používáme párový tag `<a>` (anchor) doplněný o další parametry.
Syntaxe: `<a parametry="hodnoty">` text odkazu `</a>`

8.9.1 Druhy odkazů

- Na jinou www stránku.
- Na záložku na aktuální nebo jiné www stránce.
- Na externí dokument (.doc, .pdf) nebo na obrázek (.png, .jpg., .gif).
- Na e-mail.

8.9.2 Parametry odkazů

- **Cíl odkazu:** `href="URL #záložka"`
- **Definice záložky:** `name="#záložka"`

Obousměrný odkaz: `<a name="#záložka1" href="#záložka2">` … `</a>`
`<a href="#záložka1" name="#záložka2">` … `</a>`

- **Odkaz na email**: `href="mailto:email1;...?cc=emailx&subject=text"`
  `mailto:` … adresát, `cc` … kopie, `subject` … předmět,
  `?` … první oddělovač, `&` … další oddělovače.
- **Komentář k odkazu:** `title="text"`
- **Způsob otevření:** `target="_blank"` … v novém okně
  `target="_self"` … v aktuálním okně
  `target="název okna"` … v konkrétním okně

8.10 Obrázky a videa

Obrázky a videa vkládáme do www stránky pomocí nepárového tagu `<img>`, doplněného vhodnými parametry.

8.10.1 Parametry obrázků a videí

- Zdroj obrázku (source) = `src="URL"`
- Zdroj videa = `dynsrc="URL"`
- Komentářový popisek = `alt="text"`
- Zarovnání obrázku vůči textu (obtékání) = `align` (left, right)
- Šířka obrázku / videa [px] = `width`
- Výška obrázku / videa [px] = `height`
- Ohraničení obrázku / videa [px] = `border`
- Volný prostor kolem rámečku (horizontální, vertikální) = `hspace`, `vspace`
- Počet přehrání videa = `loop` (-1 … nepřetržité přehrávání)
- Způsob spouštění videa = `start` (`mouseover` … při najetí myši, `fileopen` … při načtení)

Příklad: `<img src="images/alik.jpg" alt="pes Alík" width="120" />`

8.10.2 Obrázek jako odkaz

```
<a href="URL cílové stránky"> <img src="URL obrázku" /> </a>
```

8.11 Tabulky

Běžným způsobem strukturování textu je použití tabulek `<table>`, skládajících se z řádků `<tr>` a buněk `<td>`. Buňky v prvním řádku tabulky, obsahující názvy jednotlivých sloupců, definujeme pomocí tagu hlavičky tabulky `<th>`.

8.11.1 Parametry tabulky `<table>`

- Zarovnání na stránce = `align` (left, center, right)
- Barva podkladu = `bgcolor`
- Ohraničení [px] = `border`
- Vzdálenost okraje obsahu od hrany [px] = `cellpadding`
- Mezera mezi buňkami [px] = `cellspacing`
- Šířka tabulky [px] nebo [% šířky stránky] = `width`
- Vnější ohraničení = `frame` (void … žádné, above … horní, below … dolní, lhs … levé, rhs … pravé, hsides … vodorovné, vsides … svislé)
- Vnitřní mřížka = `rules` (none … žádná , rows … mezi řádky, cols … mezi sloupci, all … celá)

8.11.2 Parametry řádků `<tr>`

- Vodorovné zarovnání všech buněk = `align` (left, center, right)
- Svislé zarovnání všech buněk = `valign` (top, middle, bottom)
- Barva podkladu = `bgcolor`

8.11.3 Parametry buněk `<td>`, `<th>`

- Vodorovné zarovnání = `align` (left, center, right)
- Svislé zarovnání = `valign` (top, middle, bottom)
- Šířka buňky (celého sloupce) [px] nebo [% šířky tabulky] = `width`
- Vodorovné sloučení n buněk = `colspan`
- Svislé sloučení n buněk = `rowspan`
- Nezalamování řádků textu = `nowrap` (bez hodnoty)

8.12 Rámy (frames)

Okno prohlížeče lze rozdělit na několik samostatných částí (rámů) a v každém z těchto rámů můžeme zobrazit jinou www stránku (html soubor). Rámy definujeme za hlavičkou dokumentu místo těla pomocí párového tagu `<frameset>`. Jednotlivé rámy vkládáme pomocí nepárového tagu `<frame>`.

8.12.1 Struktura souboru s rámy

Soubor s rámy by se měl většinou zobrazit jako první, proto je většinou pojmenován index.html. Na začátku kódu definujeme, stejně jako u běžných Html dokumentů, typ a verzi. Dále následuje hlavička s titulkem a případně metainformacemi a místo těla (body) jsou definovány rámy.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Frameset//EN"
„http://www.w3c.org/TR/xhtlm1/DTD/xhtml1-frameset.dtd“>
<html>
      <head>
            <title> Titulek stránky </title>
      </head>
```

```
    <frameset>
        vložení (odkazy na obsah) jednotlivých rámů pomocí <frame>
    </frameset>
</html>
```

8.12.2 Parametry tagu <frameset>

- výšky jednotlivých rámů [px] nebo [%] = `rows`
- šířky jednotlivých rámů [px] nebo [%] = `cols`
- ohraničení rámů = `frameborder` (1 nebo 0)
- mezera mezi rámy [px] = `framespacing`

Příklad

```
<frameset rows="60%,40%"> nebo <frameset rows="60%,*">
```

`*` značí zbytek

Pokud chceme rozdělit webovou stránku na více rámců vodorovně i svisle, lze vytvořit strukturu framesetů.

Příklad

```
<frameset rows="100, *">
    vložení (odkaz na obsah) prvního vodorovného rámce pomocí <frame>
    <frameset cols="15%, *">
        vložení (odkaz na obsah) prvního svislého rámce pomocí <frame>
        vložení (odkaz na obsah) druhého svislého rámce pomocí <frame>
    </frameset>
</frameset>
```

8.12.3 Parametry tagu <frame>

- název rámu = `name`
- odkaz na soubor s obsahem rámu (source) = `src="url"`
- zobrazení posuvníků = `scrolling` (auto, no, yes)
- zákaz změny velikosti rámu = `noresize` (bez hodnoty)
- ohraničení rámů = `frameborder` (1 nebo 0)

Příklad

```
<frame name="menu" src="menu.html" scrolling="no" frameborder="0"
noresize />
```

8.12.4 Ošetření pro zobrazení v prohlížeči, který nepodporuje rámy

Pro tento případ vkládáme za tagy `<frame>` před ukončení framesetu párový tag `<noframes>`.

Příklad

```
<noframes>
    <p>Tento prohlížeč nepodporuje rámy</p>
</noframes>
```

8.13 Metainformace

Metainformace obsahují obecné informace o HTML dokumentu a slouží k optimalizaci www stránek pro internetové prohlížeče a vyhledávače. Zapisujeme je do hlavičky (`<head>`) dokumentu pomocí nepárového tagu `<meta>`, doplněného vhodnými parametry.

Syntaxe: `<meta name="jméno" content="hodnota" />`, nebo
`<meta http-equiv="ekvivalent http" content="hodnota" />`

8.13.1 Typ a kódování dokumentu

Nejdůležitější metainformací je zadání typu dokumentu (html) a jeho kódování (jazyk). V Evropě je nejčastějším způsobem kódování Windows-1250 (Central European), které umožňuje zobrazovat české znaky s diakritikou.

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250" />
```

8.13.2 Klíčová slova pro vyhledávače

- **Autor stránek:** `<meta name="author" content="jméno a příjmení" />`
- **Popis obsahu stránky:** `<meta name="description" content="popis stránky" />`
- **Klíčová slova:** `<meta name="keywords" content="klíčová slova" />`

8.13.3 Pravidelná aktualizace stránky

Pro aktualizaci (znovunačtení) www stránky používáme:

```
<meta http-equiv="refresh" content="interval v sekundách" />
```

8.13.4 Přesměrování na jinou www stránku

`<meta http-equiv="refresh" content="x; url=y" />`,
x … časová prodleva [s], y … url adresa, na kterou se přesměrovává.
Při nastavení časové prodlevy na nulu, dojde k přesměrování okamžitě.

9 Kaskádové styly

Pomocí kaskádových stylů CSS (Cascading Style Sheets) definujeme vzhled www stránek tím, že hromadně měníme parametry jednotlivých HTML tagů.

9.1 Způsob zápisu

Syntaxe: `selektor {deklarace}`
Selektor = co chceme ovlivnit (upravovaný tag).
Deklarace = nastavované parametry (`vlastnost:hodnota;`).
Příklad: `H1{color:red;}`
Definování více selektorů najednou: `selektor1, selektor2, …`
Více deklarací najednou: `{deklarace1; deklrace2; …;}`
Příklad: `H1, H2, H3{color:red; background:black;}`

9.2 Jednotky

Hodnoty jednotlivých parametrů se mohou být v různých jednotkách.
Vzdálenost: Absolutní: palec [in], centimetr [cm], milimetr [mm], bod [pt]
Relativní: pixel – obrazový bod [px]
Barvy: triviální anglické názvy (white, blue, silver, …)
hexadecimální vyjádření RGB … #rrggbb
procentní vyjádření RGB … rgb(r%,g%,b%) … Příklad: rgb(50%,100%,0%)
URL: url(URL) … Příklad: `url(http:\\www.seznam.cz)`
Procenta: %

9.3 Komentáře

Komentáře slouží jako doprovodné informace ke zdrojovému kódu, které ho vysvětlují.
Syntaxe: `/* komentář */`

9.4 Umístění (definice) stylů

- **U jednotlivých tagů** (např. `<p>`):
  `<tag style="` deklarace selektorů `">`
- **V hlavičce dokumentu** (v `<head>`):
  ```
  <style type="text/css">
     <!-- jednotlivé deklarace selektorů //-->
  </style>
  ```
- **V externím souboru (.css)**: jednotlivé deklarace selektorů v samostatném textovém souboru.
  V .html souboru v `<head>` je umístěn pouze odkaz na externí soubor:
  `<link rel="stylesheet" type="text/css" href="URL" />`

9.5 Nastavení vlastností stránky

Vlastnosti stránky nastavujeme v deklaraci selektoru body [px] nebo [cm].

9.5.1 Nastavení okrajů stránky

- Horní okraj = `margin-top`
- Levý okraj = `margin-left`
- Dolní okraj = `margin-bottom`

- Pravý okraj = `margin-right`

Zkrácený zápis: `body{margin:top right bottom left;}` nebo
`body{margin:top left;}`

9.5.2 Barvy a obrázky na pozadí

- Barva podkladu = `background-color` Př.) `{background-color:white;}`
- Barva písma = `color` Př.) `{color:#0000ff;}`
- Obrázek na pozadí = `background-image` Př.) `{background-image:url(...);}`
- Opakování obrázku = `background-repeat` Př.) `{background-repeat:no-repeat;}`
  bez opakování … no-repeat, s opakováním … repeat, opakování ve směru … repeat-x, repeat-y
- Pohyb obrázku při rolování = `background-attachment`
  bez pohybu = fixed, pohybující se = scroll
- Umístění obrázku = `background-position` Př.) `{background-position:left;}`

9.6 Nastavení vlastností textu

Vlastnosti písma a odstavce nastavujeme především v deklaracích selektorů zahrnujících text (p, h1 až h6, a, div, li, ul, ol, …). Tyto deklarace začínají často slovem font nebo text.

9.6.1 Vlastnosti písma

- Font písma = `font-family`
  patkové … serif, bezpatkové … sans-serif, neproporcionální … monospace, přesně definované … např. Tahoma
- Velikost písma = `font-size`
- Řez písma = `font-style`
  normální … normal, kurziva …italic nebo skloněné písmo … oblique
- Tloušťka písma = `font-weight`
  normální … normal, tučné … bold, číselně po stovkách … 100 až 900
- Kapitálky = `font-variant`
  malé kapitálky … small-caps

Příklad

```
h2{font-family:sans-serif;
   font-style:normal;
   font-weight:bold;
   }
```

9.6.2 Vlastnosti textu

- Mezery mezi slovy = `word-spacing`
- Proložení znaků = `letter-spacing`
- Řádkování = `line-height`
  normální … normal, násobky … x%, přesně … x
- Podtržení a blikání = `text-decoration`
  žádné … none, podtržení … underline, nadtržení … overline, přeškrtnutí … line-through, blikající … blink
- Velikost písmen = `text-transform`
  všechna velká … uppercase, všechna malá … lowercase, první velké … capitalize

- Zarovnání = `text-align`
  vlevo … left, vpravo … right, na střed … center, do bloku … justify
- Svislé zarovnání = `vertical-align`
  nahoru … top, ke středu buňky … middle, dolů … bottom, první řádky vedle sebe … baseline
- Odsazení prvního řádku = `text-indent`
- Nezalamování řádků = `white-space`
  nowrap

9.6.3 Vlastnosti seznamů

- Styl odrážek a číslování =`list-style-type`
  disc … ●, square … ■, circle …○,
  decimal, lower-alpha, upper-alpha, lower-roman, upper-roman
- Obrázkové odrážky = `list-style-image:url(odrazka.gif)`

9.7 Nastavení vlastností rámečků, oddílů a tabulek

9.7.1 Vnitřní okraje a ohraničení

U oddílu, odstavce, tabulky, … lze nastavit tzv. vnitřní okraj, což je odsazení obsahu prvku od okraje. Slouží k tomu parametry `padding` pro všechny okraje nebo `padding-top`, `padding-bottom`, `padding-left` a `padding-right` pro dílčí okraje.
Pro ohraničení nějakého objektu používáme parametr `border-style` s následujícími hodnotami: none … bez orámování, solid … plné ohraničení, double … dvojité ohraničení, dotted … tečkované ohraničení, dashed … čárkované ohraničení, outset … vyvýšení, inset … zanoření, …

9.7.1.1 Další vlastnosti rámečků

- Barva ohraničení = `border-color`
- Tloušťka ohraničení = `border-width`

Stejně, jako u okrajů, lze nastavit ohraničení pouze u některé strany (`border-bottom-style`).

9.7.2 Plovoucí elementy

9.7.2.1 Rozměry jednotlivých objektů

Pro nastavení přesných rozměrů rámečků nebo také jednotlivých objektů (oddíl, odstavec, …) používáme deklarace vlastností šířky … `width` a výšky … `height`, případně `max-width` a `max-height`.

9.7.2.2 Plovoucí objekty

- **Nastavení obtékání** provedeme pomocí vlastnosti `float`:
  bez obtékání … none,
  zarovnání vlevo nebo vpravo … left / right
- **Zrušení obtékání** za plovoucím objektem pomocí vlastnosti `clear`:
  bez obtékání … both,
  zrušení obtékání vlevo nebo vpravo … left / right

9.7.2.3 Umístění plovoucích objektů na stránce

- **Konkrétní umístění objektu** na stránce provádíme pomocí vlastnosti `position`:

podle html kódu … static,
podle absolutních souřadnic … absolute, dále zadáváme vlastnosti `left`, `right` a `top`,
stálé umístění v okně prohlížeče (není ovlivněno posuvníkem) … fixed, dále zadáváme vlastnosti polohy jako u absolute,
relativní umístění vůči sousednímu prvku … relative, dále zadáváme vlastnosti polohy vůči sousednímu prvku.

- Při **překrývání objektů** nastavujeme pořadí pomocí vlastnosti `z-index`, kde nejvyšší hodnota bude v popředí.

9.8 Pseudotřídy

Pseudotřídy slouží k přiřazení stylu na základě pseudoelementu (splnění určité podmínky).
Syntaxe: `Selektor:podmínka{deklarace}`

9.8.1 Pseudoelementy pro odstavce

- První řádek = `first-line`
- Iniciála = `first-letter`
  Př.) `p:first-letter{font-size: 30px; font-weight:bold;}`

9.8.2 Pseudotřídy pro odkazy

- Nenavštívený odkaz = `a:link`
- Navštívený odkaz = `a:visited`
- Při najetí myši = `a:hover`

↓ pozor na pořadí !!!

9.9 Třídy (class)

Třídy slouží k vytvoření nového stylu, který bude použit pouze u části určitých tagů.
Syntaxe: `selektor.třída{deklarace}` Př.) `p.p1{text-align:justify;}`
Univerzální třída, použitelná pro různé tagy: `.třída{deklarace}` Př.) `.cerveny{color:red;}`
Použití v html: `<tag class="třída"> … </tag>` Př.) `<p class="p1"> … </p>`

9.9.1 Příklad využití tříd a pseudotříd pro dynamické odkazy

```
a.menu{background:red; color:white;}
a.menu:hover{background:white; color:red; text-decoration:none;}
```

9.10 ID prvky

ID prvky slouží, stejně jako třídy, k vytvoření nového stylu, ale na rozdíl od tříd je lze v html dokumentu použít pouze jednou. Používají se nejčastěji u oddílů `<div>`.
Syntaxe: `#název prvku{deklarace}`
Použití v html: `<tag id="název prvku"> … </tag>`
Deklarace jednotlivých tagů v ID prvku: `#název_prvku tag{deklarace}`
Př.) `#nov p{text-align:left;}`

9.11 Posuvník (Scrollbar)

Ke změně designu posuvníků lze využít CSS deklarace selektorů html a body.

- Barva podkladu posuvníků = `scrollbar-track-color`
- Barva posuvníků a barva kolem šipek = `scrollbar-face-color`
- Stín uvnitř posuvníků a nahoře = `scrollbar-highlight-color`

- Barva levých a horních okrajů posuvníků = `scrollbar-3dlight-color`
- Barva pravých a spodních okrajů posuvníků = `scrollbar-darkshadow-color`
- Stín uvnitř posuvníků napravo a dole = `scrollbar-shadow-color`
- Barva šipek = `scrollbar-arrow-color`

10 Optimalizace webových stránek – SEO

Optimalizace pro vyhledávače SEO (Search Engine Optimalization) představuje optimalizaci webových stránek tak, aby byly tyto stránky snadno vyhledatelné webovými roboty (vyhledavači) a zobrazovali se pokud možno na začátcích výsledků hledání. Hlavním cílem je zajistit co největší návštěvnost www stránek.

10.1 Činnosti SEO

- Volba správné a logické struktury www stránek (viz kapitola Úvod do webových aplikací).
- Dodržování zásad sémanticky správného a k vyhledávačům „přátelského“ (X)HTML kódu.
- Zajištění uživatelsky přitažlivého, smysluplného a zajímavého textového i obrazového obsahu www stránek.
- Volba vhodné domény pro umístění webu.
- Budování zpětných odkazů vedoucích na jednotlivé www stránky z externích webů.
- Průběžná analýza pozice webu ve vyhledavačích a snaha o zlepšování pozice ve výsledcích vyhledávání.

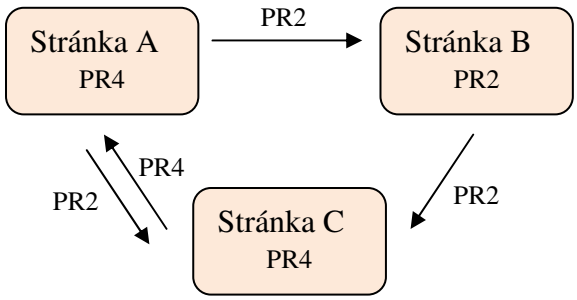
10.2 Funkce webových vyhledavačů

Úkolem internetového vyhledávače (robota, pavouka) je plížit se Internetem a prohledávat maximální počet webových stránek, jejichž obsah ukládá do své databáze, tzv. indexu (indexuje). Index je databáze uložených www stránek, kterou doplňují roboti (vyhledavače). Tento index se stále doplňuje a aktualizuje podle toho, jak narůstá počet nově nalezených a znovu navštěvovaných stránek.
Uživatelé pracují s vyhledávači tak, že zadají do pole pro hledání určitá klíčová slova a vyhledávací stroj poté zobrazí seznam výsledků hledání, seřazený podle určitého hodnocení.

10.2.1 Hodnocení www stránek

Hodnotící systém se liší podle konkrétního vyhledávače (Google, Seznam.cz, …). Výsledkem hodnocení je číslo, které určuje, jak dobře si na známkovací stupnici konkrétní www stránka (nikoliv celý web) stojí. Čím vyšší číslo bude, tím častěji bude příslušnou stránku vyhledávací robot navštěvovat a bude ji umisťovat výše ve výsledcích vyhledávání.

- **PageRank** (PR) = tento systém využívá Google. Počítá hodnotu stránky podle toho, kolik a jak důležitých stránek se na hodnocenou stránku odkazuje. Výchozí hodnotou je PageRank výchozí stránky. Část své hodnoty stránka propůjčuje stránkám, na které se odkazuje. Čím více odkazů stránka obsahuje, tím menší hodnotu PageRanku těmto stránkám propůjčuje.



- **S-Rank** = tento systém využívá Seznam.cz. Princip hodnocení je podobný jako u PageRanku, ale navíc zohledňuje také atraktivní odkazy vedoucí ze stránky.

Hodnoty PageRanku nebo S-Ranku pro určitou www stránku lze zjistit on-line pomocí aplikace na ***ranky.cz***.

10.2.2 Vyhledávání klíčových slov¨

Pozici ve výsledcích hledání neurčuje pouze vlastní hodnocení konkrétní stránky, ale také umístění hledaných klíčových slov v (X)HTML kódu. Výše bude umístěna například stránka, která bude obsahovat klíčová slova v titulku a metainformacích.

10.3 Volba domény

Doména je internetová adresa (URL = Uniform Resource Locator), na které je web (struktura www stránek) umístěn.

10.3.1 Důležitost názvu domény

- Dobrá uživatelská zapamatovatelnost (aby se na určitou stránku lidé často vraceli).
- Prioritní hledání klíčových slov v doménových názvech vyhledávači.
- Snadno zapamatovatelný název pro marketingovou podporu mimo Internet (např. reklama v televizi, billboardy, …).

10.3.2 Druhy domén

- **Generické domény** = domény prvního řádu (com, net, info, org, …), vhodné pro celosvětovou působnost.
- **Národní domény** = domény prvního řádu (cz, sk, de, eu, …), vhodné pro lokální působnost.
- **Webové domény** = domény druhého (případně třetího) řádu, obsahující vlastní název webu (firmy). Pokud je delší (víceslovný) název a je to realizovatelné, je vhodné registrovat více těchto domén s podobnými názvy (např. NabytekNovak a Nabytek-Novak). V případě regionálně působících firem je vhodné tento region zabudovat do názvu domény (např. Nabytek-Ledec).

10.4 Volba klíčových slov

Klíčová slova jsou slova, která uživatelé zadávají do vyhledávačů a která očekávají, že bude příslušný web obsahovat. Současné vyhledávače většinou hledají bez ohledu na zadávání klíčových slov s diakritikou nebo bez ní (např. Ledec = Ledeč).

10.4.1 Příklady klíčových slov

- Obor činnosti firmy nebo podniku.
- Název nabízeného produktu nebo služby.
- Charakteristické znaky produktů nebo služeb (např. levné, nadměrné, apod.)
- Název regionu nebo města.
- …

10.4.2 Klíčové fráze

Nejčastěji vyhledáváme nejprve podle obecného klíčového slova a následně vyhledávání upřesňujeme přidáváním dalších klíčových slov. Tím vzniká tzv. dlouhý ocas (long tail), což je dlouhá přesná fráze. Na začátku hledání je vyhledáno velké množství obecných informací a na konci je jen málo vyhledaných konkrétních informací.
Správným cílem SEO je oslovení těch návštěvníků, ze kterých se mohou stát zákazníci (konec dlouhého ocasu). Vysoká návštěvnost není hlavním cílem. Je třeba se snažit především o cílenou návštěvnost, tedy nabídnout návštěvníkům přesně ty informace, které hledají.

Příklad fráze: Prodej levného nábytku Ledeč nad Sázavou

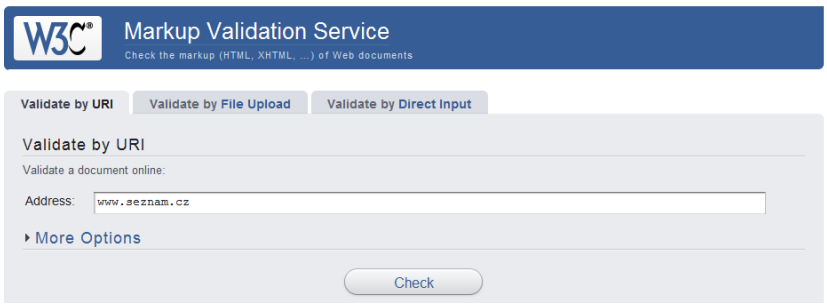
10.4.3 Zjištění počtu vyhledávání klíčových slov

Počet vyhledávání určitých klíčových slov nebo frází lze zjistit prostřednictvím on-line aplikací například pro Google lze využít ***adwords.google.com/select/KeywordToolExternal*** nebo pro Seznam.cz lze ***www.sklik.cz*** (pouze při přihlášení).

10.5 Validita (X)HTML kódu

Kód webových stránek, který má být přívětivý pro vyhledávače, by měl být validní, měl by tedy odpovídat používané specifikaci jazyka (X)HTML. K ověření kódu lze využít on-line validátory, které ověří správnost syntaxe příslušného kódu. Příkladem je ***validator.w3.org***.

- Zadání URL adresy existující stránky (Validate by URI) nebo cesty k lokálnímu umístění www stránky (Validate by File Upload). Dále kliknutí na tlačítko ***Check***.



- Při nalezených chybách se zobrazí jejich počet a popis. Navíc se zobrazí typ a kódování (X)HTML dokumentu.
- Pokud je dokument validní (bez chyb), zobrazí se text *This document was successfully checked*.

10.5.1 Sémantický a prezentační kód

Aby byl kód www stránek validní, je třeba striktně oddělovat obsahovou a formátovací část.

- **Sémantický kód** = jeho úkolem je pouze popisovat význam obsahu (nadpis, odstavec, …).
- **Prezentační kód** = definuje způsob formátování, nejčastěji prostřednictvím kaskádových stylů.

10.6 Oblasti vyhledávání klíčových slov

Při tvorbě webu je třeba zajistit, aby vyhledávače nalezli co nejvíce výskytů klíčových slov. Nesmí se to ale přehnat, neboť při extrémních výskytech určitého slova naopak vyhledávače danou stránku při vyhledávání penalizují.

10.6.1 Popisné informace

- **Titulek** (title) = většina vyhledávačů jej zobrazují jako hlavní část výsledků vyhledávání. Jde o nejdůležitější místo pro umístění klíčových slov, ale neměl by být příliš dlouhý.
- **Metadata** = základní informace umístěné v hlavičce (X)HTML dokumentu. Mezi nejdůležitější pro vyhledávače patří popis www stránky (description) a seznam klíčových slov (keywords). Zde je vhodné uvést co nejvíce relevantních klíčových slov a frází.
- **Definice znakové sady** (charset) = kódování, definující použitou znakovou sadu, což je důležité pro správné zobrazení znaků (zejména s diakritikou a speciálních znaků). Nejběžnějšími znakovými sadami, podporujícími české znaky jsou **iso-8859**, **windows-1250**, **utf-8** a **utf-16**.

10.6.2 Nadpisy

Nadpisy patří mezi základní prvky strukturování webového obsahu. Zejména nadpisy úrovní h1 a h2 patří mezi vyhledávači prohledávané prvky, které by opět měly obsahovat vybraná klíčová slova.

10.6.3 Odkazy

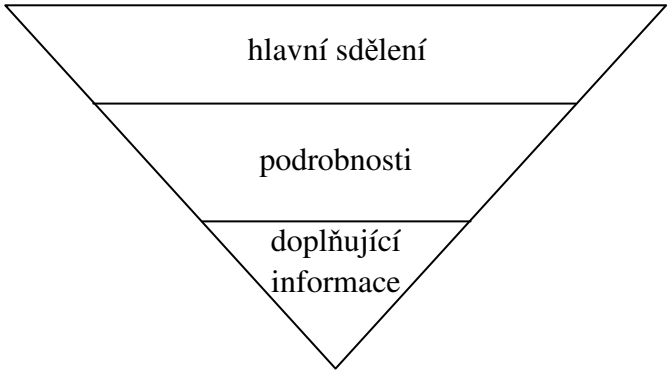
Odkazy slouží nejen uživatelům k přechodu na jinou stránku, ale také vyhledávačům, které tak mohou procházet i hlouběji ve struktuře uložené informace a nikoliv pouze domovské stránky. Text odkazu je opět objektem, ve kterém vyhledávače hledají případný výskyt klíčových slov, neměl by tedy obsahovat nic neříkající text (špatně je například *„Klikněte zde“*).

10.6.4 Popisné texty objektů (Alt texty)

Některé objekty (například obrázky) nejde fulltextově prohledávat a je tedy vhodné je navíc označit textovým popisem, který budou následně vyhledávače schopni identifikovat.

10.7 Obrácená pyramida

Informace by na webu měly být strukturovány tak, aby uživatele vedly.



- Název (slogan) produktu nebo služby a základní popis.
- Všechny pro uživatele důležité informace (technické, cenové, … údaje produktu nebo služby).
- Méně podstatné informace určené těm, kteří se o produkt nebo službu zajímají do hloubky.

10.8 SEO nástroje

Analytické nástroje pro optimalizaci webu lze nalézt na adrese ***www.seonastroje.cz***.

- **Analýza klíčových slov** = zjištění počtu výskytů jednotlivých klíčových slov v různých částech (X)HTML dokumentu.
- **Ranky** = zjištění hodnocení stránky PageRank a S-rank.
- **SEO analýza** = analýza webové stránky, zjištění jejích nedostatků a doporučení pro optimalizaci.
- **…**

10.9 Mapa webu

Mapa webu je strukturovaný seznam odkazů v rámci webu (ve formátu xml nebo html). Jde o velmi užitečný nástroj zejména u větších webu. Mapu webu můžeme vytvořit ručně nebo využít on-line nástroje, například na ***www.xml-sitemaps.com***.

Literatura

[1] SCHWIMMER, M., BREDEN, M. *Excel 2007 VBA – Velká kniha řešení*. Brno: Computer Press, 2009. ISBN 978-80-251-2698-1

[2] KRÁL, M. *Excel VBA – Výukový kurz*. Brno: Computer Press, 2010. ISBN 978-80-251-2358-4

[3] VYSTAVĚL, R. *Moderní programování – učebnice pro začátečníky*. Praha: Moderní programování, 2009. ISBN 978-80-903951-6-9

[4] VYSTAVĚL, R. *Moderní programování – sbírka úloh k učebnici pro začátečníky*. Praha: Moderní programování, 2008. ISBN 978-80-903951-5-2

[5] VYSTAVĚL, R. *Moderní programování – učebnice pro středně pokročilé*. Praha: Moderní programování, 2008. ISBN 978-80-903951-2-1

[6] VYSTAVĚL, R. *Moderní programování – sbírka úloh k učebnici pro středně pokročilé*. Praha: Moderní programování, 2009. ISBN 978-80-903951-3-8

[7] HANZLÍKOVÁ, J. *Webdesign pro úplné začátečníky*. Brno: Computer Press, 2004. ISBN 80-251-0159-2

[8] HAUSER, M., HAUSER, T., WENZ, CH. *HTML a CSS, Velká kniha řešení*. Brno: Computer Press, 2006. ISBN 80-251-1117-2

[9] DOMES, M. *SEO jednoduše*. Brno: Computer Press, 2011. ISBN 978-80-251-3456-6