Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Matematické principy tvorby kalendáře a jejich vizualizace

Vypracoval: Pavel Sládek
Vedoucí práce: Ladislav Beránek

České Budějovice 2018

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě, v úpravě vzniklé vypuštěním vyznačených částí archivovaných Zemědělskou fakultou- elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby touto elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum: Podpis studenta:

Děkuji svému vedoucímu bakalářské práce panu doc. Ladislavu Beránkovi za ochotu, vstřícnost a veškerou pomoc, kterou mi pomohl s touto prací. Dále bych chtěl poděkovat za odporu všem, kteří mi poskytli možnost diskutovat o problémech řešených v průběhu psaní.

Abstrakt

Bakalářská práce popisuje matematické principy vybraných kalendářů. Jde o kalendář juliánský a kalendář gregoriánský. Tyto kalendáře byly vybrány, z důvodu dlouhé tradice na našem území. Jsou zde popsány rozdíly mezi kalendáři. Ukázáno je generování tabulek kalendáře a výpočet data velikonočních svátků. Praktickou část práce tvoří návrh a tvorba aplikací. Jednou z aplikací je věčný kalendář. Druhá aplikace vypočítává datum Velikonoc. Aplikace zobrazuje i dílčí hodnoty použité k výpočtu těchto aspektů. Při návrhu aplikací byla snaha o pochopitelnost kódu. Aplikace zobrazují mezivýsledky, aby poskytly možnost sledovat závislost změn na změnách vstupních dat. Obsaženy jsou i části zdrojových kódů.

Klíčová slova: kalendář, principy, aplikace

Abstract

This bachelor thesis covers mathematical principles of selected types of calendars – Julian and Gregorian These were selected because of long-term history of usage in our country and its predecessors. Comparison of calendar types is covered. There is described how to create the table of Gregorian and Julian calendar and derivations of Easters. There is also computer application for calculating these aspects and demonstration of these principles. This application algorithm is designed with the aim of uncovering parts of calculations and therefore it can be used for studying these principles. The thesis also includes described parts of source code.

Keywords: Calendars, princples, application

Obsah

1 Matematické principy kalendářů

Cílem bakalářské práce je nastínit matematické principy kalendářů. Například princip fungování věčného kalendáře nebo výpočet dne v týdnu. Tyto principy si může jsou demonstrovány na přiložených aplikacích. Aplikace jsou zamýšleny jako pomůcka, k pochopení matematických zákonitostí kalendářů, a proto jsou napsány tak, aby zobrazovaly i jednotlivé dílčí výsledky. Aplikace lze tedy použít při studiu těchto principů, jako pomůcku.

Již odedávna se u lidí vyskytovala potřeba organizace času. Kalendář je systém uspořádání dnů. Čas je odvozen dle různých přírodních cyklů. Například pohybu Měsíce kolem Země. Tyto kalendáře nazýváme lunárními. Další skupinou kalendářů jsou solární kalendáře, které vycházejí z pohybu Země kolem Slunce. Každý kalendářní systém má určitou odchylku od skutečnosti (tropického roku), kterou popisuje. Z tohoto důvodu kalendáře postupně prochází vývojem. Tato práce popisuje matematické principy juliánského a gregoriánského kalendáře. Kalendáře juliánský a gregoriánský kalendář byly vybrány z důvodu historického užívání na našem území. Mayský kalendář byl vybrán pro srovnání jako zástupce kalendářů, které necílí na použití jednoletého cyklu. V rámci praktické části práce byly navrženy a napsány aplikace, které demonstrují nastíněné principy.

2 Historický vývoj kalendářů

2.1 Tropický rok

Tropický rok je doba oběhu Země kolem Slunce – doba nutná k dosažení dvou jarních rovnodenností po sobě. Toto je ideální doba, které by se měl blížit rok kalendáře. Pokud se kalendářní rok a tropický rok liší, tak dochází k postupnému rozcházení kalendáře a skutečnosti. Délka tropického roku je přibližně 365,2422 dne (365 dní, 5 hodin a 49 minut) (Vondrák, 2011). Tato hodnota je obtížně použitelná pro sestavení dlouhodobějšího kalendáře, pokud nebudou používáni korekce kalendáře vůči tropickému roku. Každý z kalendářů tedy aplikuje nějakou formu korekcí, které postupně zvyšují přesnost.

2.2 Mayský kalendář

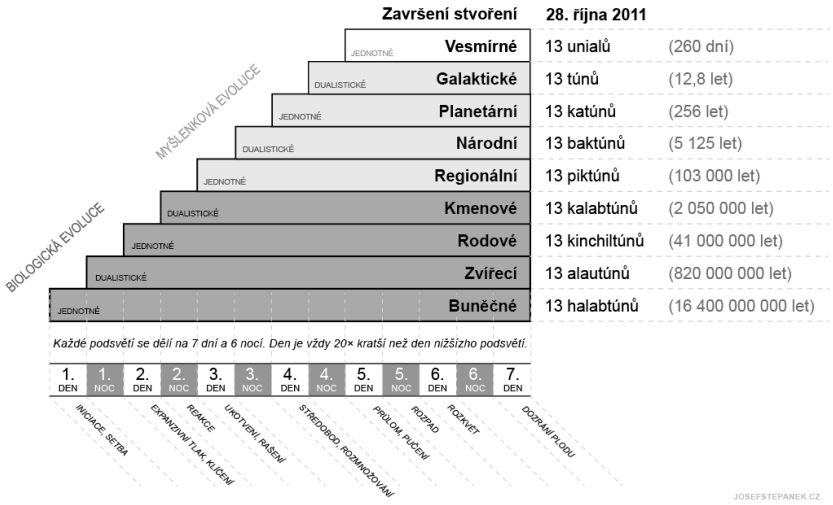
Mayský kalendář primárně nepopisuje jednoleté období. Kalendář využívá tzv tuny a tzolkiny. Tun je 360 dnů, po kterých následuje 5 „nešťastných“ dnů. Tzolkin je období skládající se z 13 tónů („měsíců“) po 20 dnech (viz tabulka 1) (Štěpánek, 2011). Jeden tzolkin tedy popisuje 260 let dlouhé období.52 tunům odpovídá 75 tzolkinů. Mayský kalendář má tedy nejkratší cyklus, po kterém nastane stejný den jak, podle tunu, tak tzolkinu, dlouhý 18720 dnů To je zhruba 51krát delší cyklus než v případě juliánského nebo gregoriánského kalendáře, za předpokladu, že nebudeme uvažovat přestupné roky. V případě uvažování přestupných let se v případě juliánského kalendář dostaneme na 1461 dnů. V případě gregoriánského kalendáře je cyklus 146 097 dnů.

Podobný princip souběhu dvou cyklů lze najít i v souvislosti gregoriánského kalendáře a měsíčních fází. (Smítka, 2012)

Imix	Ik	Akbal	Kan	Chicchan
Cimi	Manik	Lamat	Muluc	Oc
Chuen	Eb	Ben	Ix	Men
Cib	Caban	Etznab	Cuac	Ahau

Tabulka 1 Dny mayského kalendáře (Lucita Inc., 2018)

Výraz měsíc, je potřeba brát s rezervou, neboť nejde o časový úsek v podání, v jakém jej známe v juliánském či gregoriánském kalendáři. Jde o pravidelné střídání sekvence dnů, která začíná 1.Imix,2.Ik,3.Akbal,4.Kan a dále až k 13.Ben, po kterém následuje 1.Ix,2.Men. Pokud budeme pokračovat dále, tak se dostaneme k 8.Imix. Vzhledem k nesoudělnosti čísel 13 a 20 se takto postupně vystřídají všechny kombinace během 260 dnů, což právě odpovídá délce jednoho tzolkinu.



Obrázek 1 Stupně mayského kalendáře (Štěpánek, 2011)

Pokud se podíváme na obrázek 1, tak vpravo uvidíme přepočet na námi používané časové jednotky. Směrem dolu se čas vyjádřený daným stupněm je vždy dvacetinásobkem stupně předchozího. Pokud tedy víme například, že chceme vědět, jak byl v kalendáři zapsán den, který byl 1 153 203. dnem mayského cyklu, tak stačí vhodně aplikovat modulo.

$$1\ 153\ 203 / (20*20*20*18) = 8 \text{ zbytek } 1203 \quad (1)$$

Proběhlo tedy 8 baktůnů. Zbylých 1 203 dní převedeme na katůny.

$$1203 / (20*20*18) = 0 \text{ zbytek } 1203 \quad (2)$$

Neproběhl tedy ani jeden katůn.

$$1203 / (360) = 3 \text{ zbytek } 123 \quad (3)$$

Proběhl třikrát tun.

$$123 / 20 = 6 \text{ zbytek } 3 \quad (4)$$

Toto datum by tedy bylo zapsáno jako 8.0.3.6.3.

Vzhledem k nedostatku podkladu pro věrohodné zasazení mayského kalendáře do kontextu zbylých kalendářů a pravidelnost přechodu mezi jednotlivými vrstvami kalendáře, není součástí práce žádná aplikace pro mayský kalendář.

2.3 Juliánský kalendář

Juliánský kalendář byl zaveden Juliem Caesarem roku 47 př.n.l. Rok se skládá z 12 měsíců nebo také 365 (respektive 366) dnů. Kalendář lze popsat čtyřletým cyklem. První tři roky mají 365 dnů. Čtvrtý má dnů 366. Průměrná délka roku je tedy 365,25 dne. Tabulka 2 popisuje postupný vývoj odchylek roku tropického a roku podle juliánského kalendáře.

Rok	Délka tropického roku	Délka juliánského roku	Odchylka	Kumulovaná odchylka
1	365,2422	365	-0,2422	-0,2422
2	365,2422	365	-0,2422	-0,4844
3	365,2422	365	-0,2422	-0,7266
4	365,2422	366	+0,7578	+0,0312
5	365,2422	365	-0,2422	-0,2110
6	365,2422	365	-0,2422	-0,4532
7	365,2422	365	-0,2422	-0,6954
8	365,2422	366	+0,7578	+0,0624

Tabulka 2 Vývoj odchylky cyklu tropických a juliánských let (zdroj: vlastní)

Rozdíl mezi tropickým rokem a juliánským rokem závisí na roce, který je porovnáván. Pokud je rok nepřestupný, tak absolutní velikost odchylky roste o 0,2422 dne za rok. Přestupný rok poté tento nakumulovaný rozdíl juliánského roku proti tropickému roku kompenzuje, ale zároveň vytváří menší rozdíl kalendářů v opačném směru.

Příloha juliánský_kalendář.xls obsahuje tuto odchylku vypočítanou pro období 3500 let. Příloha je rozšířena o další informace. Obsahuje například informace kolikrát odchylka překročila nějakou z mezních hodnot. Mimo to je zde informace o průměru odchylek (sloupec Q) a jak se tento průměr vyvíjí (sloupec R). Vývoj průměru v prvních 100 letech kalendářního cyklu není monotónní. Od stého roku cyklu je již vývoj monotónní – pouze roste. V stém roce je odchylka již zvětšována přestupným rokem. Bylo by tedy dobré tento přestupný rok vynechat. Bez zajímavosti též není, že trvá 132 let, než odchylka dosáhne jednoho dne. Toto lze využít po zpřesnění, které nabízí gregoriánský kalendář.

Pouze zhruba 5 % let podle juliánského kalendáře má odchylku do 1 dne. Tato hodnota ukazuje tedy na nízkou přesnost kalendáře, kvůli které byl nahrazen gregoriánským.

Z hlediska výpočtu data Velikonoc je důležitý 19letý cyklus, ve kterém se opakuje shoda měsíční fáze a dne týdne. (Juliánský kalendář, nedatováno)

2.4 Gregoriánský kalendář

Gregoriánský kalendář je pojmenován po papeži Řehořovi XIII. - Gregorovi XIII. Tento kalendář byl definován roku 1582 bulou Inter Gravissimas, avšak jeho zavádění probíhalo postupně. (Spencer, 1999) Kalendář je též znám jako juliánský kalendář s gregoriánskou úpravou. Úprava spočívá ve změně systému přestupných roků. Současně se změnou kalendáře došlo i k úpravě postupu výpočtu data Velikonoc. Místo 4letého cyklu gregoriánský kalendář využívá cyklus o délce 400 let. Na začátku jsou pravidelně střídány 3 nepřestupné roky s 1 přestupnými. Jak bylo uvedeno v části věnované juliánskému kalendáři, tak jednodenní odchylka (u juliánského kalendáře) vznikne po 132 letech. Ale již od stého roku cyklu průměrná odchylka juliánského roku a tropického roku (počítaná za 3 čtyřleté cykly) pouze roste. Juliánský kalendář by tedy bylo dobré upravit již po sto letech. Úprava o celý den po sto letech by však otočila směr odchylky. Gregoriánský kalendář tedy odebere přestupné dny v stém, dvoustém a třístém roku cyklu. Vynecháním dvacátého pátého přestupného roku se tato plus jeden den upraví na plus třetinu dne. Další chyba následuje v roce 264. Opět je přičtena chyba plus jeden den. Celková chyba tedy tvoří plus dvě třetiny dne. Vynecháním přestupného dne v dvoustém roce cyklu je odchylka upravena na mínus jednu třetinu dne. Přičtením čtvrtého období by došlo k překročení jak hranice tří set, tak hranice čtyř set let. K mínus jedné třetině dne by byly přidány mínus dvě třtiny (každá za jedno století). Po uplynutí čtyř set let by tedy jeden den scházel, proto je u každého čtyřstého roku přestupný den zachován. Tímto dojde k částečné korekci kalendáře.

Při pohledu do přílohy gregoriánský_kalendář.xls můžeme vidět stejně nasimulovaný cyklus pro 3500 let. Odebrání stého, dvoustého a třístého roku sice zvětší odchylku (sloupec G), ale zároveň upraví monotonii průměru absolutních hodnot odchylek (sloupec T). Použití přestupného dne tedy nebude mít stále stejný účinek, ale bude docházet k postupné vzájemné kompenzaci vyvolaných změn. Pokud se koukneme na buňku H3502, tak uvidíme, že gregoriánský systém kalendáře se od tropického roku odchyluje 3037krát v rozmezí do jednoho dne. Pouze zhruba 13,23 % let má větší odchylku než jeden den.

Počet cyklů	tropický rok	juliánský rok	gregoriánský rok
1	365,2422	(3*365+366) /4 =365,25	(3*(24*(3*365+366) +4*365) +25*(3*365+366)) / 400 = 365,2425
100	36524,22	36525	36524,25
400	146096,88	146100	146097

Tabulka 3 Rozdíly kalendářů (zdroj: vlastní)

Gregoriánská úprava tedy zpřesní kalendář z odchylky (na jeden den) $2{,}14 * 10^{-5}$ na $8{,}21 * 10^{-7}$, tedy 26krát. Juliánský kalendář nakumuluje jednodenní odchylku přibližně za 46 827 dnů. Gregoriánský kalendář odchylku jednoho dne (aniž by tato odchylka opět klesla) nakumuluje přibližně za 1 217 475 dnů.

2.5 Inter gravissimass

Inter gravissimass je bulou papeže Řehoře XIII. Tato bula upravila několik aspektů fungování kalendáře. Důvodem bylo získání lepší návaznosti na přírodní cykly. Cílem bylo získat kalendář, který s přiměřenou matematickou náročností a složitostí a zároveň s dostatečnou přesností.

Jedním z kroků bylo jednorázové vynechání dnů mezi 5. říjnem a 14. říjnem 1582 včetně. Dále bylo vynecháno Další úprava se týká systému střídání přestupných a nepřestupných let. Je zde zahrnulo vynechání přestupného dne stého, dvoustého a třístého roku čtyřsetletého cyklu. V této době byl přestupný den nazýván bissextile, protože v rámci juliánského kalendáře byl původně vkládán šest dní před konec února. Došlo tedy ke „zdvojení“ šestého dne od konce února => „bi“ a „ssextile“. Bulou bylo upraveno, že dnem navíc bude den vložený na konec února.

V této bule je již zmíněno Zlaté číslo, jako prostředek k náhradě systému epakt (význam epakt je vysvětlen dále). (Spencer, 1999)

3 Tvorba kalendáře

3.1 Epakty

Epakty jsou posuny aspektů kalendáře proti jednoduchému modelu. Epakt máme více, podle účelu. Epakta může být využita například k generování tabulky kalendáře. Tabulku můžeme generovat, pokud víme, jakým dnům odpovídá, jaké datum. Vzhledem k střídání dnů v pravidelném 7denním cyklu je potřeba užít korekce (epakty) vůči pořadí dnů v měsíci. První lednový den roku 2018 je pondělí. Tomto případě je tedy epakta 0. Rozložení dnů v únoru již však bude ovlivněno tím, že délka ledna (31 dnů) a délka nejbližšího kratšího sedmi denního cyklu (28 dnů) se liší. V případě výpočtů pro únorové dny bude výpočet zahrnovat tento posun o 3 dny. Únor má 28 dní, tudíž nezpůsobí další posuny. Březen tady také vykazuje posun o 3 dny. Duben již bude posunut o 3 dny nakumulované z ledna a další tři dny z března. Celkově tedy o 6 dnů. Počátek května by byl posunut o další 2 dny, tedy 8 dní celkem. Vzhledem k délce týdne můžeme od výsledku odečíst 7. Tento posun týdne je ekvivalentní posunu o 1 den. Pro stejný výsledek můžeme použít operaci modulo. Celkový přehled těchto posunů ukazují tabulky 4 a 5.

Měsíc	První den	Posun na konci měsíce	Posun následujícího měsíce
Leden	Pondělí	31-28 = 3	3
Únor	Čtvrtek	28-28 = 0	3
Březen	Čtvrtek	31-28 = 3	6
Duben	Neděle	30-28 = 2	8 =1
Květen	Úterý	31-28 = 3	4
Červen	Pátek	30-28 = 2	6
Červenec	Neděle	31-28 = 3	9 = 2
Srpen	Středa	31-28 = 3	5
Září	Sobota	30-28 = 2	7 = 0
Říjen	Pondělí	31-28 = 3	3
Listopad	Čtvrtek	30-28 = 2	5
Prosinec	Sobota	31-28 = 3	8 = 1
Leden (následující rok)	Úterý	31-28 = 3	4

Tabulka 4 Posuny dnů týdne vyvolané měsíci (zdroj: vlastní)

V případě nepřestupného roku následující rok začíná dnem týdne následujícím po 1. dnu týdne roku aktuálního. Rok 2019 začne úterým. Tohoto výsledku lze opět dosáhnout použitím modula, konkrétně

$$365 \; mod \; 7 = 1 \tag{5}$$

Měsíc	První den	Posun na konci měsíce	Posun následujícího měsíce
Leden	Pondělí	31-28 = 3	3
Únor	Čtvrtek	29-28 = 1	4
Březen	Pátek	31-28 = 3	7=0
Duben	Pondělí	30-28 = 2	2
Květen	Středa	31-28 = 3	5
Červen	Úterý	30-28 = 2	7=0
Červenec	Pondělí	31-28 = 3	3
Srpen	Čtvrtek	31-28 = 3	6
Září	Neděle	30-28 = 2	8 = 1
Říjen	Úterý	31-28 = 3	4
Listopad	Pátek	30-28 = 2	6
Prosinec	Neděle	31-28 = 3	9 = 2

Tabulka 5 Posuny dnů v týdnu vyvolané měsíci - přestupný rok (například 2024) (zdroj:vlastní)

V případě přestupného rokuje situace podobná, akorát od měsíce března je posun před použitím modula o 1 den navýšen.

Výpočet pomocí epakt je náročný z hlediska nutnosti mít výchozí bod – den u kterého známe kombinaci dne v týdnu a datumu. Tento nedostatek může být odstraněn použitím jiné metody, například Zellerova[1] algoritmu. Pokud využijeme epakty, můžeme velice snadno následně sestavit tabulku věčného kalendáře.

Pokud budeme znát vzdálenou kombinaci data a dne od data, které chceme počítat, tak výpočet může být zdlouhavý. Tuto zdlouhavost lze eliminovat například vhodnou aplikací zbytkových tříd.

Pokud x-tý den měsíce je pondělí, tak (x+7)-tý je rovněž pondělí, pokud, neboť i x i x+7 spadají do stejné zbytkové třídy.

Obdobným způsobem lze zpracovat i delší časové úseky, například rok. Pokud prvního ledna roku x, který nebude přestupný, bude úterý, tak po uplynutí 365 dnů, tj opět za rok bude týdenní cyklus posunutý o 365 mod 7, tedy o 1 den. Prvním lednem (x+1).tého roku tedy bude středa.

[1] Christian Zeller (1822-1899) německý matematik

Pokud daný rok bude přestupným, tak se týdenní cyklus bude posouvat o 366 mod 7, což značí posun o 2 dny (1.leden 2016 byl pátek, ale 1.leden 2017 byla až neděle). První dny let 1900–2023 uvádí následující tabulka 6.

1.den roku											
PO	1900	1912	1923	1934	1945		1968	1979	1990	2001	
UT	1901		1924	1935	1946	1957		1980	1991	2002	2013
ST	1902	1913		1936	1947	1958	1969		1992	2003	2014
ČT	1903	1914	1925		1948	1959	1970	1981		2004	2015
PÁ	1904	1915	1926	1937		1960	1971	1982	1993		2016
SO		1916	1927	1938	1949		1972	1983	1994	2005	
NE	1905		1928	1939	1950	1961		1984	1995	2006	2017
PO	1906	1917		1940	1951	1962	1973		1996	2007	2018
UT	1907	1918	1929		1952	1973	1974	1985		2008	2019
ST	1908	1919	1930	1941		1964	1975	1986	1997		2020
ČT		1920	1931	1942	1953		1976	1987	1998	2009	
PÁ	1909		1932	1943	1954	1965		1988	1999	2010	2021
SO	1910	1921		1944	1955	1966	1977		2000	2011	2022
NE	1911	1922	1933		1956	1967	1978	1989		2012	2023

Tabulka 6 Rozdělení roků, podle 1. dne kalendáře – gregoriánský systém (zdroj: vlastní)

Podobný vývoj počátečních dnů roku můžeme vytvořit i pro juliánský kalendář. (tabulka 7). V tabulce si mimo posunu můžeme všimnout i vynechání jednoho pole po roce 1900. Tato úprava vznikla vynecháním přestupného roku v roce 1900.

1.den roku											
PO		1907	1918	1929		1952	1963	1974	1985		2008
UT		1908	1919	1930	1941		1964	1975	1986	1997	
ST			1920	1931	1942	1953		1976	1987	1998	2009
ČT		1909		1932	1943	1954	1965		1988	1999	2010
PÁ		1910	1921		1944	1955	1966	1977		2000	2011
SO	1900	1911	1922	1933		1956	1967	1978	1989		2012
NE		1912	1923	1934	1945		1968	1979	1990	2001	
PO	1901		1924	1935	1946	1957		1980	1991	2002	2013
UT	1902	1913		1936	1947	1958	1969		1992	2003	2014
ST	1903	1914	1925		1948	1959	1970	1981		2004	2015
ČT	1904	1915	1926	1937		1960	1971	1982	1993		2016
PÁ		1916	1927	1938	1949		1972	1983	1994	2005	
SO	1905		1928	1939	1950	1961		1984	1995	2006	2017
NE	1906	1917		1940	1951	1962	1973		1996	2007	2018

Tabulka 7 Rozdělení roků, podle 1. dne kalendáře – juliánský systém (zdroj: vlastní)

Další epakta může najít využití při výpočtu data Velikonoc.

3.2 Zellerův algoritmus

Zellerův algoritmus využívá pravidelnost cyklu střídání přestupných a nepřestupných let. Při aplikaci je tedy spočítán počet přestupných let. Vzhledem k změně o 1 den v případě nepřestupného roku a o 2 dny v případě přestupného roku, tak stačí znát počet let od začátku letopočtu, počet přestupných let, měsíc a den, který chceme počítat.

Při použití Zellerova algoritmu přidáváme 29.únor pro zjednodušení na konec roku. Aby toto bylo možné, tak je nutné počítat s únorem jako s posledním měsícem roku. Z tohoto důvodu jsou leden a únor brány jako 13. a 14. měsíc roku předchozího. V případě ledna a února prvního roku století je nutné upravit nejen hodnotu roku, ale i hodnotu století.

Část q představuje posun proti dnu začátku měsíce. Je tedy nutné určit, kdy začíná měsíc.

Dolní celá část [13(m+1) / 5] pomáhá s výpočty začátků měsíců. V případě března, který v roce 2018 začíná ve čtvrtek, vyjde tato hodnota 10. Pro duben, který začíná nedělí vyjde hodnota vyjde 13. Došlo tedy ke zvýšení hodnoty o 3, stejně tak došlo k posunu v sekvenci dnů o 3.

$$(X+10) \bmod 7 = 4 \tag{6}$$

$$(X+13) \bmod 7 = (((X+10) \bmod 7) +3) \bmod 7 \tag{7}$$

$$4 + 3 \bmod 7 = 7 \bmod 7 = 0 \tag{8}$$

Bez pohledu do kalendáře můžeme říci, že duben začal v neděli.

Tabulka 8 demonstruje správnost členu

$$\left\lfloor \frac{13 * (m+1)}{5} \right\rfloor \tag{9}$$

Zellerova vzorce.

Měsíc	První den	Posun na konci měsíce	$\left\lfloor \frac{13 * (m+1)}{5} \right\rfloor$
Březen	Čtvrtek	31-28 = 3	10
Duben	Neděle	30-28 = 2	13
Květen	Úterý	31-28 = 3	15
Červen	Pátek	30-28 = 2	18
Červenec	Neděle	31-28 = 3	20
Srpen	Středa	31-28 = 3	23
Září	Sobota	30-28 = 2	26
Říjen	Pondělí	31-28 = 3	28
Listopad	Čtvrtek	30-28 = 2	31
Prosinec	Sobota	31-28 = 3	33
Leden	Úterý	31-28 = 3	36
Únor	Pátek	28-28 = 0	39

Tabulka 8 výpočet začátku měsíce

Čtvrtý sloupec obsahuje hodnoty, které jsme mohli získat dosazením do výše uvedené části vzorce. Druhý způsob, jak tyto hodnoty získat je k hodnotě předchozího měsíce přičíst počet dnů, které v předchozím měsíci přesahují 28denní cyklus.

3.2.1Juliánský kalendář

$$h = \left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + 5 - J\right) mod\ 7 \qquad (10)$$

q = pořadové číslo dne v měsíci

m = pořadové číslo měsíce – 13 pro leden a 14 pro únor

k = rok století – rok století-1 v případě ledna / února

j = $\lfloor rok/100 \rfloor$

h = den v týdnu počítáno od soboty jako nultého dne

Pokud tedy nepočítáme s lednem či únorem, tak datum má vlastně tvar q.m.jjkk. V případě ledna či února, který není na začátku století se jedná o tvar q.m-12.jjkk+1. Pokud se jedná o první dva měsíce století, které potřebujeme převést na poslední dva měsíce století předchozího, tak jde o tvar q.m-12.jj+1kk+1.

Pokud vezmeme v úvahu 4letý cyklus jednoho přestupného a tří nepřestupných let, tak v juliánském kalendáři cyklus popisuje 3*365+366 = 1461 dnů. 1461 mod 7 = 5, což odpovídá K + dolní celá část (K/4). K je důsledkem

$$365\ mod\ 7 = 1 \qquad (11)$$

Využití dolní celé části je zde z důvodu, že chceme znát počet kroků přes přestupný rok. Pokud je K dělitelné 4, tak při růstu o 4 výsledek dolní celé části se upraví o 1.Toto přidá jeden den za každý přestupný rok.

Pokud by K nebylo dělitelné 4 (například pro rok 2015 K=15), tak $\left\lfloor \frac{15}{4} \right\rfloor$ je 3, ale po úpravě K na K+1, tedy pro přestupný rok, již $\left\lfloor \frac{16}{4} \right\rfloor$ je 4. Pokud budeme pokračovat $\left\lfloor \frac{17}{4} \right\rfloor = 4, \left\lfloor \frac{18}{4} \right\rfloor = 4, \left\lfloor \frac{19}{4} \right\rfloor = 4$,ale $\left\lfloor \frac{20}{4} \right\rfloor = 5$. Pokud tedy dojde ke zvýšení K z 16 na 20, tak dojde ke zvýšení dolní celé části o 1, tj. k připočtení jednoho přestupného dne.

Podobným způsobem můžeme dokázat přítomnost právě jednoho J. J představuje počet století. V juliánském kalendáři má jedno století 25 přestupných let po 366 dnech a 75 nepřestupných let po 365 dnech.

$$století\ ve\ dnech = 25 * 366 + 75 * 365 \quad (12)$$

$$století\ ve\ dnech = 36\,525 \quad (13)$$

$$století\ ve\ dnech\ mod\ 7 = 6 \quad (14)$$

Z tohoto plyne, že každé století způsobí posun o 6 dnů dopředu. Vzhledem k aplikaci modula lze toto upravit

$$(x - 1)\ mod\ 7 = (x + 6)\ mod\ 7 \quad (15)$$

Pro potřeby programování je vhodné použít tvar x+6, neboť u záporných čísel může dojít k nevyhovující interpretaci operace modulo.

3.2.2 Gregoriánský kalendář

$$h = \left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor - 2J\right) mod\ 7 \quad (16)$$

Stejně jako v případě juliánského kalendáře je

q = pořadové číslo dne v měsíci

m = pořadové číslo měsíce – 13 pro leden a 14 pro únor

k = rok století – rok století-1 v případě ledna / února

j = $\lfloor rok/100 \rfloor$

h = den v týdnu počítáno od soboty jako nultého dne

Q, $\left\lfloor \frac{13(m+1)}{5} \right\rfloor, K$ mají stejný efekt jako v případě juliánského kalendáře. Oproti juliánskému kalendáři je potřeba vynechat jeden den, pokud se J zvýší o 4. Z juliánského kalendáře je tedy převzato „-J“, druhé „-J“ je právě ono vynechání, které je opraveno $\left\lfloor \frac{J}{4} \right\rfloor$ v letech 1600, 2000, 2400 a dalších.

Přiložené aplikace umožňují výpočet dne v týdnu, který odpovídá zadanému datu. Toto datum je rozděleno na příslušné hodnoty a následně zpracováno Zellerovým algoritmem.

Již tedy máme k dispozici informaci, jakému dnu odpovídá zvolené datum. Není tedy problém vygenerovat zbytek tabulky kalendáře. V aplikaci je toto ošetřeno vypočtením dne pro první den v měsíci. Tento den je vložen do pole s indexem, který odpovídá správnému dnu. Pokud je prvním dnem například čtvrtek, tak je jednička vložena až do prvku pole s indexem 4. Pole je takto naplněno vložením čísla zadaného dne do příslušného indexu pole. Následně jsou dopočítány hodnoty ostatních polí od prvního dne do posledního dne. Tyto prvky poté slouží jako parametry, které jsou předány jednotlivým objektům, které tvoří tabulku kalendáře.

Ve výše uvedených vzorcích se můžeme mimo sčítání a násobení setkat i s dolní celou částí a modulem. Aplikace obou těchto matematických prostředků má své opodstatnění.

Dolní celá část zajišťuje zjištění počtu cyklů. Například pokud potřebujeme zjistit počet přestupných let v juliánském kalendáři, tak potřebujeme vlastně zjistit, kolik čtyřletých cyklů kalendáře proběhlo. Pokud 2018 vydělíme 4, tak získáme 504.5. Toto ale není vyhovující výsledek pro použití. Výsledkem by bylo, že dopoledne jednoho datumu by patřilo k jednomu dni, zatímco odpoledne by byl den druhý. Pokud ale na tento výsledek použijeme dolní celou část, tak již získáme 504, což odpovídá počtu proběhlých cyklů.

Modulo poté zjišťuje, v které části cyklu zrovna jsme. Jeho využití najdeme například u výpočtu dne v týdnu. Výsledky, které spadají do stejné zbytkové třídy v případě dělení sedmi dají stejný výsledek. Například 1.,8.,15.,22. a 29. den jednoho měsíce, jednoho roku patří stejnému dnu v týdnu, takže pokud vynecháme Zellerově algoritmu mod 7, tak dostaneme čísla, která patří do stejné zbytkové třídy.

$$Z_1 = x + y \quad Z_1 \bmod y = x \bmod y + y \bmod y = x \bmod \tag{17}$$

$$Z_2 = x + 2y \quad Z_2 \bmod y = x \bmod y + y \bmod y = x \bmod \tag{18}$$

$$Z_n = x + ny \quad Z_n \bmod y = x \bmod y + n(y \bmod y) = x \bmod y \tag{19}$$

Jak již bylo uvedeno výše, tak stejný postup lze aplikovat i na delší období.

Podle gregoriánského kalendáře byl první leden roku 1 pondělí, pokud tedy chceme zjistit, které další dny jsou pondělí, tak stačí zjistit vzdálenost kterých dnů od 1.1.1 je dělitelná 7.

„Přirozené číslo N je dělitelné 7, právě když číslo, které vznikne z čísla N odtržením cifry na místě jednotek a odečtením dvojnásobku jednotek od takto „zkráceného čísla“ je dělitelné sedmi.“ (Tlustý, 2006)

29.1.2018 je od 1.1.1 vzdáleno 2017 let a 28 dnů. Z těchto 2017 let bylo 489 let přestupných. Počet dnů lze tedy vyjádřit jako

$$počet\ dnů = 365 * (2018 - 1) + 489 + 28 \quad (20)$$

$$počet\ dnů = 736\ 722 \quad (21)$$

Aby bylo 29.1.2018 pondělí, tak je nutné dokázat, že 7|736 722.

Pro výpočet potřebujeme získat počet jednotek. Můžeme tedy použít modulo:

$$J_1 = počet\ dnů\ mod\ 10 \quad (22)$$

$$J_1 = 736\ 722\ mod\ 10 \quad (23)$$

$$J_1 = 2 \quad (24)$$

Ono zkrácené číslo můžeme získat jako:

$$Z_1 = \left\lfloor \frac{počet\ dnů}{10} \right\rfloor \quad (25)$$

$$Z_1 = \left\lfloor \frac{736\ 722}{10} \right\rfloor \quad (26)$$

$$Z_1 = 73\ 672 \quad (27)$$

Pokračujeme dále:

$$V_1 = Z_1 - 2 * J_1 \quad (28)$$

$$V_1 = 73\ 672 - 4 \quad (29)$$

$$V_1 = 73\ 668 \quad (30)$$

Nyní použijeme V_1 jako vstup pro výpočet J_2 a dalších hodnot:

$$J_2 = 8 \tag{31}$$

$$Z_2 = 7366 \tag{32}$$

$$V_2 = 7350 \tag{33}$$

$$J_3 = 0 \tag{34}$$

$$Z_3 = 735 \tag{35}$$

$$V_3 = 735 \tag{36}$$

$$J_4 = 5 \tag{37}$$

$$Z_4 = 73 \tag{38}$$

$$V_4 = 63 \tag{39}$$

$$J_5 = 3 \tag{40}$$

$$Z_5 = 6 \tag{41}$$

$$V_5 = 0 \tag{42}$$

Číslo 736 722 je dělitelné 7, tudíž 1.leden 2018 v gregoriánském kalendáři také odpovídá pondělí.

3.3 Věčný kalendář

Věčný kalendář je aplikací výše nastíněných principů. Oproti Zellerově algoritmu nevyžaduje úpravy data. Ve své podstatě je jednoduchou tabulkou (viz příloha vecny_kalendar.xls), v které je vybrán rok, pro který chceme počítat datum. Tento výběr určí řádek, s kterým bude dále pracováno. Výběrem měsíce dojde k vybrání pole tabulky. Toto pole obsahuje hodnotu, kterou přičteme k hledanému datu. V pomocné tabulce lze k výsledné hodnotě dohledat odpovídající datum.

Hlavní tabulka má 28 řádků, což odpovídá počtu let, během kterých se vystřídají všechny kombinace dnů, kterými může začít nový rok a zároveň u všech bude jak varianta přestupného, tak nepřestupného roku. V případě gregoriánského kalendáře mají roky 1900 a 2100 mají nepravidelné umístění, protože jsou nepřestupné. Obdobný kalendář pro juliánské uspořádání dnů by měl umístění pravidelné.

Hodnota, která je přečtena z průsečíku měsíce a příslušného roku představuje součet epakt. Není tedy nutné počítat zvlášť posun pro roky, století a měsíce.

Pokud posun začátku 1. měsíce s délkou M1 1. roku označíme jako E1.1, tak epaktu pro 2.měsíc 1.roku E2.1 můžeme zapsat jako:

$$E2.1 = (E1.1 + M1)\ mod\ 7 \tag{43}$$

nebo

$$E3.1 = (E2.1 + M2)\ mod\ 7 \tag{44}$$

Naplněním celé tabulky lze vytvořit věčný kalendář pro jakákoliv data. Pozor ovšem na nepravidelnosti viz roky 1900 a 2100. Dalším omezením může být práce s daty před rokem 1582, kde je potřeba použít tabulku pro juliánský kalendář, případně vzít na vědomí, že pracujeme s jiným kalendářem.

4 Návrhy aplikací

4.1 Výpočet dne v týdnu

Aplikace k výpočtům využívá již výše zmíněný Zellerův algoritmus, který byl modifikován, aby vracel pořadová čísla dnů v týdnu.(Viz tabulka 9)

$$D = (h + 5)\ mod\ 7 + 1 \tag{45}$$

h	D
2	1
3	2
4	3
5	4
6	5
0	6
1	7

Tabulka 9 Převod výsledku Zellerova algoritmu na den v týdnu (zdroj: vlastní)

K vývoji aplikací je využit objektově orientovaný programovací jazyk C#. V aplikacích není využita třída DateTime s jedinou výjimkou – využití metody za účelem získání datumu ze systému. K tomuto kroku slouží metoda, která zavolá DateTime.Now() a je schopna zpracovat jí navrácenou hodnotu.

Vlastní třída obsahuje jen informace, které jsou používány pro výpočet. Dalším důvodem pro využití vlastní třídy je skutečnost, že téměř vše využité aplikací je popsáno ve zdrojovém kódu, a tudíž není pro prostudování studovat další zdroje. Vlastní třída mi také umožnuje používat záporný rok. Nemůže totiž dojít k výpočtu s kladným rokem a následné převedení před náš letopočet (tedy alespoň ne Zellerovým algoritmem).

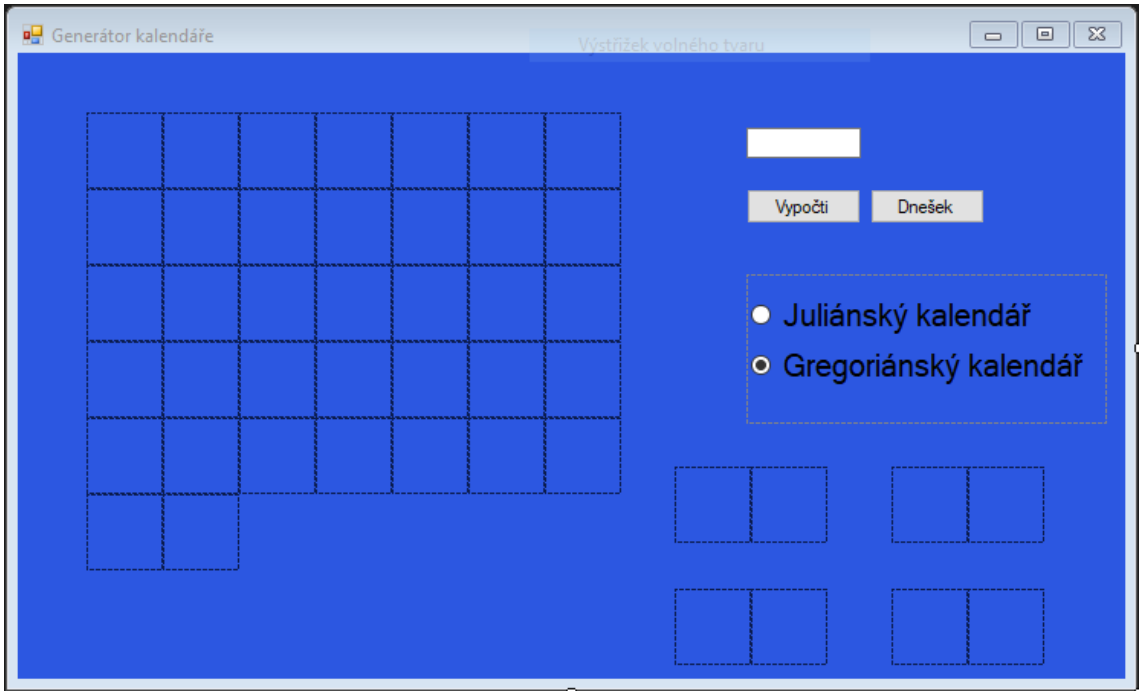
V případě využití třídy DateTime C# neumožňuje používat záporné roky. Nelze tedy automaticky vstupní data uložit a jednoduše ověřit například použitím metody DateTime.tryParse();.

Jednou z možností je tedy spočítat počet dní, které je potřeba připočítat do okamžiku, kdy je možné použít kladný rok. Počet dní mezi požadovaným datumem a rokem našeho letopočtu vydělit sedmi a výslednou hodnotu algoritmu upravit (viz kapitola 3.2.2).

V případě výpočtu modula kladného čísla je vše v pořídku. Ovšem pokud chceme použít modulo na záporné číslo, například -9, tak počítač vrátí -2. Omezení lze obejít úpravou před použitím modula.

V cyklu s podmínkou na konci můžeme přičítat násobky 7, dokud se nedostaneme do kladných čísel.

4.1.1Uživatelské prostředí



Uživatelské prostředí první aplikace se skládá z jednoho okna. V případě některých neplatných vstupů se může zobrazit druhé okno, které na tuto skutečnost upozorní. Stane se tak zejména v případě 32.dne měsíce a podobně. Na chybu, že je vložen text uživatel upozorňován není.

V pravé horní části aplikace je TextBox, který umožňuje uživateli zadat datum, s nímž má být pracováno. Následně lze pomocí RadioButtonů vybrat kalendář, pro který má být tabulka generována. Pod nimi se zobrazují hodnoty, které jsou mezivýsledky. Uživatel tedy může v průběhu studia práce testovat vlastními výpočty zmíněné principy a v aplikaci porovnat mezivýsledky. Pokud se liší, tak uživatel má specifikováno, v které části má hledat chybu.

V levé části jsou následně umístěny objekty, které po spuštění programu vykreslují tabulku kalendáře.

4.1.2 Vlastní návrh třídy

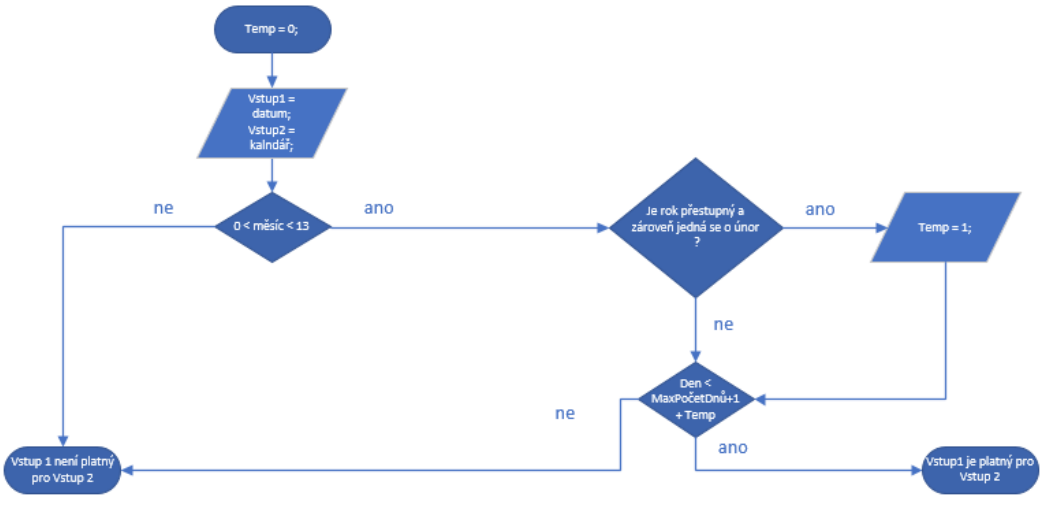
Vzhledem k nevyhovujícím možnostem třídy DateTime jsem se tedy rozhodl vytvořit vlastní třídu MyDate. Tato třída obsahuje následující datové členy:

- den
- měsíc
- rok
- typ kalendáře

Mezi metodami bude samozřejmě konstruktor. Dále jsou vytvořeny metody IsValid(), IsLeapYear(), HMLY(), Now(), Parts().

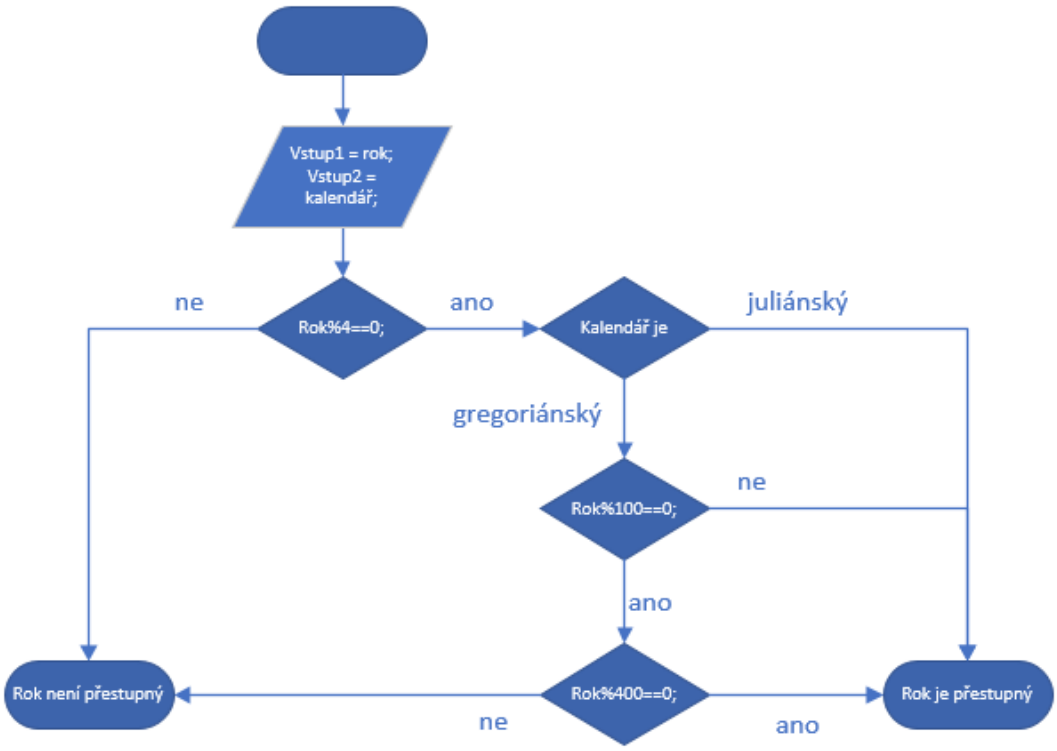
Po kliknutí na tlačítko výpočtu dojde k ověření platnosti datumu. Aplikace rozloží vstup na jednotlivé hodnoty den, měsíc, rok. Zellerův algoritmus pracuje s rokem, jako kdyby začínal 1.března. Díky této změně je v případě přestupného roku 29.únor přidán na konec roku. Leden a únor tvoří třináctý a čtrnáctý měsíc předchozího roku. Z tohoto důvodu je potřeba pro použití hodnoty upravit. Je nutno upravit jak hodnoty měsíců, tak hodnoty roku. V případě, že je rok na počátku století, tak musí dojít i k úpravě století.

Metoda IsValid(MyDateTime, kalendář) má za úkol ověřit, zda bylo zadané platné datum. Především, zda vstupem není text nebo kombinace textu a čísel. Dále je ověřeno, zda je vstup platný ve smyslu, například výpočtu dne pro 40. den 15.měsíce. Jako platná hodnota je uznán jen číselný zápis data platného ve zvoleném kalendáři.(viz obrázek 2)



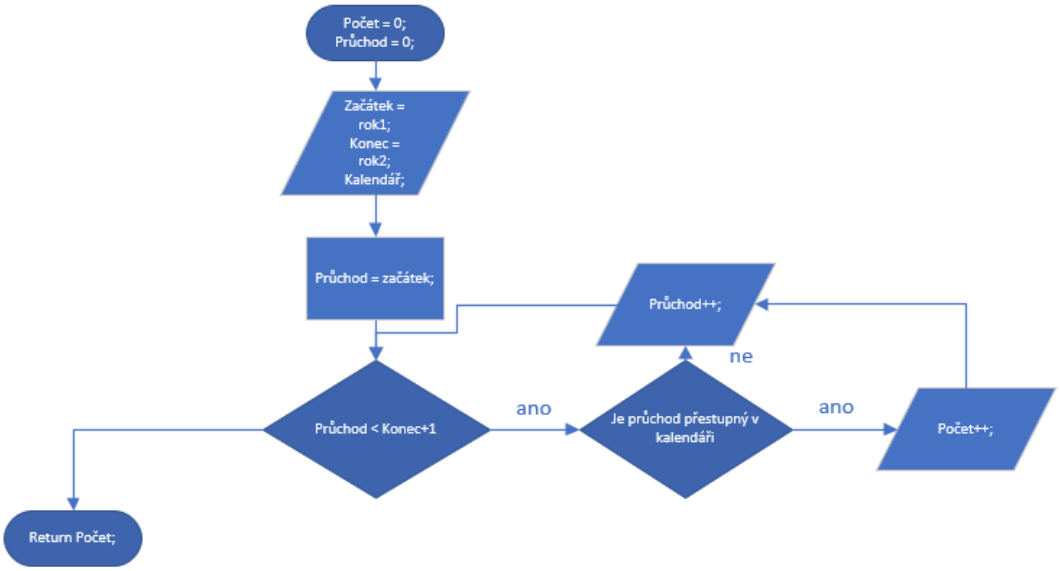
Obrázek 2 Vývojový diagram – IsValid (zdroj:vlastní)

IsLeapYear(rok, kalendář) vrací informaci (viz obrázek 3), zda určitý rok je přestupný, v daném kalendáři, či nikoliv. Metoda má dva argumenty. Prvním je rok, po který chceme určit přestupnost. Druhým je kalendář, v kterém má být přestupnost určena.



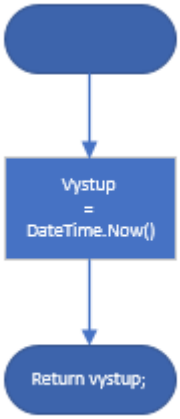
Obrázek 3 Vývojový diagram metody IsLeapYear (zdroj:vlastní)

HMLY – HowManyLeapYears(rok1, rok2, kalendář) – pomocí for-cyklu prochází úsek mezi rokem1 a rokem2 a zjišťuje počet let, které na daném intervalu jsou přestupné. Vrací zjištěný počet. (viz obrázek 4).



Obrázek 4 Vývojový diagram HowManyLeapYears (zdroj:vlastní)

Now() – pomocí zavolání metody DateTime, která je v C# standardně dostupná, umožní vytvořit instanci třídy MyDateTime s aktuální hodnotou data.(Viz obrázek 5)

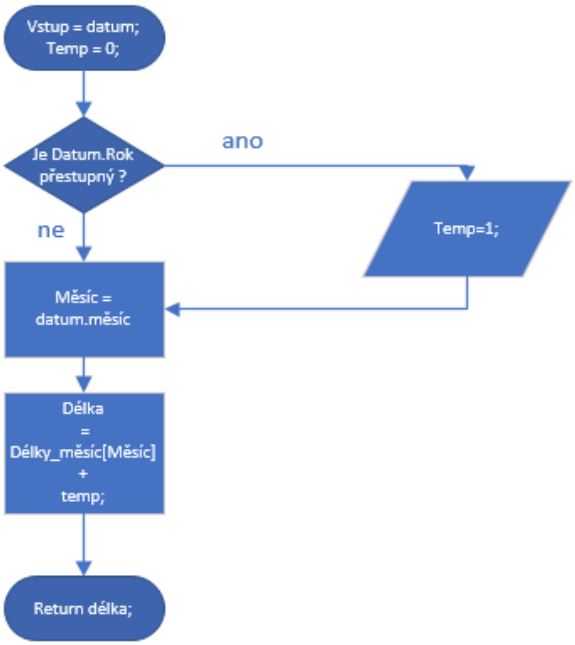


Obrázek 5 Vývojový diagram metody Now (zdroj:vlastní)

Parts (MyDateTime, type) v závislosti na hodnotě type, která může být day, month nebo year vrátí odpovídající hodnotu datumu předaného metodě jako argument MyDateTime.

Obrázek 6 Vývojový diagram metody Parts (zdroj:vlastní)

GetMonthLength(MyDateTime) – tato hodnota vrátí délku měsíce, který je použit v MyDateTime. Tato informace je uchována v třídě MyDateTime.



Obrázek 7Vývojový digram metody GetMonthLength

4.1.3 Testování

V průběhu vývoje aplikace pobíhalo samozřejmě testování. Testovány byly různé vstupy. Byly použity jak vstupy, u kterých lze očekávat korektní výstup aplikace, tak i vstupy, u kterých by aplikace fungovat neměla. Například vstup „ahoj“ by neměl aplikaci shodit. Aplikace by též měla rozpoznat neplatné datumy. Například 31.února, případně datumy 13. měsíce a podobně.

Aplikace očekává datum v platném a plném tvaru. Pokud není zadán vstup, který by bylo možné interpretovat jako datum (dd.mm.rrrr), tak aplikace neprovede výpočet. Pokud je chyba vstupu číselného charakteru – například 30. únor, tak je uživatel upozorněn, že vstup není platný. Pokud se jedná o snahu uživatele vyzkoušet, zda aplikace zvládne neplatný vstup a napíše: „ahoj“, tak aplikace ani nebude vypisovat chybové hlášení (umožňuje rychleji zkoušet nachytat aplikaci).

4.2 Aplikace Velikonoce

Vstupem pro aplikaci Velikonoce je pouze číslo roku, výpočet je tak značně jednoduší a vstup lze velice snadno uložit v dostupných datových typech. Aplikace tedy již není tak složitá na návrh.

Vzorce 46 – 60 jsou odvozeny online aplikace pro výpočet data Velikonoc. (Online výpočet Velikonoční neděle, nedatováno)

Výpočet Velikonoc vychází z tzv. Zlatého čísla. Zlaté číslo získáme:

$$Zlaté\ číslo = rok\ mod\ 19 + 1 \tag{46}$$

Tímto je upraven posun vůči úplňku (viz. kapitola Juliánský kalendář). Výpočet původně předpokládá užití juliánského kalendáře. Z tohoto důvodu jsou zde opět využity epakty. Jednak juliánská epakta, která upraví vztah juliánského kalendáře a tropického roku. Následně je použita gregoriánská epakta, která převede výsledek, aby datum odpovídalo gregoriánskému kalendáři.

$$juliánská\ epakta = (zlaté\ číslo * 11)\ mod\ 30 \tag{47}$$

$$sluneční\ oprava = \left\lfloor \frac{(století - 16) * 3}{4} \right\rfloor \tag{48}$$

$$měsíční\ oprava = \left\lfloor \frac{(století - 15) * 8}{25} \right\rfloor \tag{49}$$

$$gregoriánská\ epakta = (juliánská\ epakta - 10 - sluneční\ oprava + měsíční\ oprava)\ mod\ 30 \tag{50}$$

Dále je potřeba vypočítat datum úplňku (PFM). Pokud je gregoriánská epakta menší než 24, tak

$$PFM = 44 - gregoriánská\ epakta \tag{51}$$

Pokud je epakta větší, než 25, tak:

$$PFM = 74 - 25 \tag{52}$$

V případě, že vyjde epakta 25, tak záleží na hodnotě zlatého čísla. Pokud zlaté číslo je rovno 11, či je menší, tak:

$$PFM = 48 \tag{53}$$

V ostatních případech:

$$PFM = 49 \tag{54}$$

Pokud je PFM roven nebo menší než 31, tak 1. jarní úplněk je B.března. Pokud je PFM větší, než 31, tak 1. jarní úplněk připadá na D.dubna, přičemž platí :

$$B = PFM \tag{55}$$
$$D = PFM - 31 \tag{56}$$

Gregoriánská oprava:

$$Gregoriánská\ oprava = (10 + sluneční\ oprava)\ mod\ 30 \tag{57}$$

Den v týdnu odpovídající PFM (0=neděle):

$$den = \left(\left\lfloor\frac{rok}{4}\right\rfloor + rok - gregoriánská\ oprava + PFM\right) mod\ 7 \tag{58}$$

Velikonoční neděle tedy je N-tý den po začátku března:

$$N = PFM + 7 - den \tag{59}$$

Pokud je N > 31, tak velikonoční neděle připadá na (N-31).duben. Následuje Velikonoční pondělí, které tedy odpovídá:

$$P = PFM + 8 - den \tag{60}$$

4.2.1 Testování

Pro účely testování velikonoční aplikace jsem přidal tlačítka, která umožňují zároveň měnit zadanou hodnotu (o +1 nebo -1) a spustit výpočet.

Dále byly přidány popisky a MessageBoxy, které zobrazují jednotlivé části výpočtů. Původní návrh metody MoveTo30() pomocí While a Do se ukázal jako nešťastný, neboť špatnou posloupností příkazů bylo dosahováno špatných výsledků v případě výpočtu. Těchto důvodů byla metoda upravena na pouhé větvení podmínek a úpravu v případě přesahu mimo požadovaný interval <0;29>. Vzhledem k maximální možné hodnotě vstupu se ukázalo toto řešení také jako vyhovující. Vypočítané hodnoty byly ověřovány pomocí služby Knowledge Graph firmy Google.

5.Závěr

Výstupem této práce jsou aplikace, které umožňují ověřit zde prezentované principy fungování kalendářů. Současně i práce může být zdrojem informací pro další zájemce o tuto problematiku. Tvorba práce byla pro mne přínosná, jelikož jsem potřeboval pochopit principy načtené z literatury, abych byl na základě těchto informací schopen vytvořit přiložené programy.

Použitá literatura

Juliánský kalendář. (nedatováno). Načteno z orthodoxia.cz: http://www.orthodoxia.cz/kalendar/juliansky-kalendar.htm

Lucita Inc. (2018). *The Mayan Tzolkin Calculator | Interactive and animated mayan calendar ringsThe Mayan Calendar Portal*. Načteno z Mayan calendar portal: http://www.maya-portal.net/tzolkin

Online výpočet Velikonoční neděle. (nedatováno). Načteno z http://kalendar.beda.cz/: http://kalendar.beda.cz/vypocet-velikonocni-nedele

Smítka, B. (2012). *Matematické kombinace v dávných tisíciletích, aneb, Tajemství poznání.* České Budějovice: Nová Forma.

Spencer, B. (24-28. Listopad 1999). *English translation of Inter Gravissmas*. Načteno z http://myweb.ecu.edu/mccartyr/intGrvEng.html

Štěpánek, J. (24. Únor 2011). *Maysky kalendar v pravem svetle*. Načteno z josefstepanek.cz: https://josefstepanek.cz/maysky-kalendar-v-pravem-svetle

Tlustý, P. (2006). *Obecná algebra pro učitele.* České Budějovice: Jihočeská univerzita v Českých Budějovicích.

Vondrák, J. (15. Březen 2011). *21. března 2011 začne astronomické jaro | Ostatní | Články | Astronomický informační server astro.cz .* Načteno z astro.cz: http://www.astro.cz/clanky/ostatni/21-brezna-2011-zacne-astronomicke-jaro.html

Seznam obrázků

Seznam příloh

Elektronické přílohy
gregoriánský_kalndář.xls
juliánský_kalendář.xls
věčný_kalndář.xls
aplikace Výpočet dne v týdnu
aplikace Velikonoce