Dokumentace k projektu pro předmět Programování

Rozdíl kalendářních dat

vzorový projekt

24. srpna 2023

Autor: Ing. David Martinek, martinek@gvid.cz
Gymnázium Brno, Vídeňská, příspěvková organizace

Obsah

Kapitola 1

Úvod

Tato dokumentace popisuje návrh a implementaci programu pro výpočet rozdílu dvou kalendářních dat zadaných v přesně specifikovaném formátu. Jde o netriviální problém, protože gregoriánský kalendář, který se u nás dnes používá, obsahuje množství nepravidelností. V tomto dokumentu jsou prezentovány dva návrhy, jak lze problém výpočtu rozdílu dvou dat řešit. Jedno z těchto řešení bylo implementováno a výsledný program byl úspěšně otestován.

V kapitole 2 se nachází stručná charakteristika řešeného problému, jeho analýza a z ní vyplývající princip řešení. Zvláště jsou zde diskutovány problémy s počátkem letopočtu a s datem (daty) zavedení gregoriánského kalendáře.

Kapitola 3 se zabývá návrhem vlastního řešení, zejména pak algoritmem výpočtu přechodných roků a vlastního výpočtu rozdílu zadaných kalendářních dat. Při návrhu těchto algoritmů byly zároveň odvozeny jejich mezní stavy, které byly použity pro návrh testovacích hodnot aplikace (kapitola 3.5). Implementace (kapitola 4) přímo vyplývá z provedené analýzy a návrhu řešení.

V závěrečné kapitole (5) je zhodnoceno celkové řešení programu a jsou zde vyhodnoceny skutečně provedené testy.

Kapitola 2

Analýza problému a princip jeho řešení

Protože kalendářní aritmetika je netriviální problém, podívám se na něj v této kapitole podrobněji. Pro pochopení, jak dnešní kalendář vypadá a proč, je užitečné vědět něco o jeho historii. Dále se v této kapitole zaměřím na různé možnosti, které se pro výpočet rozdílu kalendářních dat nabízí.

2.1 Zadání problému

Cílem tohoto projektu je vytvoření programu v jazyce C, který vypočte počet dnů mezi dvěma kalendářními daty. Program musí zohlednit přestupné roky. Dále bylo požadováno, aby program načítal oba kalendářní údaje ze standardního vstupu ve formátu *dd.mm.rrrr-dd.mm.rrrr*. Výsledný počet dnů musí být vypisován na standardní výstup.

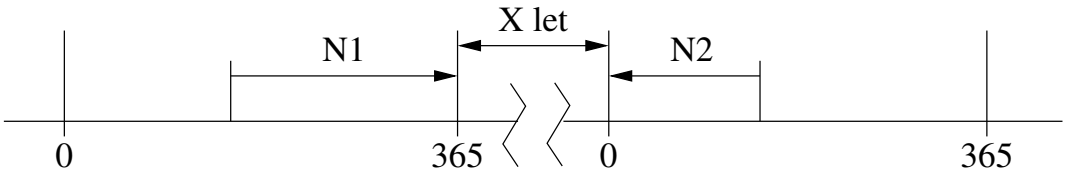
2.2 Gregoriánský kalendář

V dnešním světě se používá několik různých kalendářů (čínský, židovský, indický, blíže viz (Wikipedie, 2023b)). V částech světa ovlivněných křesťanstvím se dnes používá takzvaný gregoriánský kalendář (Wikipedie, 2023a). Vznikl nařízením papeže Řehoře XIII. a nahradil starší juliánský kalendář zavedený Juliem Caesarem. Gregoriánský kalendář byl vyhlášen 5. října 1582 podle juliánského kalendáře, což je podle gregoriánského kalendáře 15. října 1582. Nový kalendář byl vytvořen zpřesněním juliánského kalendáře, který se za jedno a půl tisíciletí fungování zpožďoval oproti astronomickému roku o téměř deset dní.[1] V novém kalendáři přibylo další pravidlo pro výpočet přestupných let, které přidává mezi přestupné roky i ty letopočty, které jsou dělitelné 400 (podle starého kalendáře byly přestupné roky dělitelné čtyřmi a roky dělitelné stem přestupné nebyly).

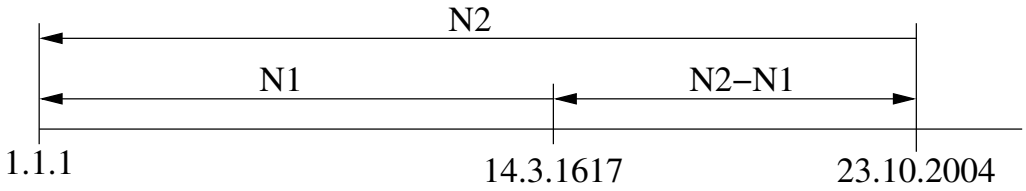
Gregoriánský kalendář nebyl přijat okamžitě. Různé státy v Evropě i ve světě tento kalendář přijímaly postupně, takže například v Čechách byl přijat na začátku roku 1584, na Moravě o půl roku později, na Slovensku až roku 1857, v protestantských státech až kolem roku 1700, v Rusku v roce 1918 a v Řecku dokonce až roku 1923 (viz (Wikipedie, 2023a)).

Zavedením nového kalendáře vznikl problém nespojitosti v počítání času. Vinou různých dat přijetí kalendáře v různých částech světa není možné jednoduše stanovit počáteční datum používání. Z těchto důvodů jsem se rozhodl akceptovat při řešení všechna data od počátku letopočtu a úvahu, zda datum dává v dané zemi smysl ponechávám na uživateli.

[1] Astronomický rok nebo také tropický je dán dobou oběhu Země kolem Slunce. V současné době to činí 365,24219 dne.



Obrázek 2.1: Řešení s postupným výpočtem počtu dní mezi zadanými daty.



Obrázek 2.2: Řešení s výpočtem dní od počátku letopočtu.

2.3 Počátek letopočtu

Křesťanský kalendář začíná rokem předpokládaného narození Ježíše Krista. Spornost přesnosti tohoto údaje není pro řešení této úlohy podstatná. Podstatnější je, že v této době se ještě v matematice nepoužíval pojem nula, takže letopočet začíná od roku 1 (to je také důvod, proč desetiletí a století nezačínají rokem dělitelným 10 nebo 100, ale až rokem následujícím). První akceptovatelné datum v našem letopočtu je tedy 1. ledna roku 1.

2.4 Problém přechodných letopočtů

V gregoriánském kalendáři má běžný rok 365 dní a únor 28 dní. Protože sluneční rok je asi o čtvrt dne delší, má gregoriánský kalendář každé čtyři roky den navíc – jedná se o přestupný rok. V takovém roce má únor 29 dní a celý rok pak 366 dní. Aby se kalendář co nejméně odchyloval od astronomického roku, byla stanovena tato pravidla pro výpočet přestupných let:

- Přestupný je každý rok, který je beze zbytku dělitelný 4,
- kromě těch let, které jsou beze zbytku dělitelné 100,
- s výjimkou těch let, které jsou beze zbytku dělitelné 400, které jsou také přestupné.

2.5 Možná řešení výpočtu rozdílu dvou dat

Na obrázku 2.1 je demonstrován způsob výpočtu, kdy se nejprve vypočte počet dní od menšího data do konce roku a pak počet dní od začátku druhého roku do druhého zadaného data. Poté se zjistí, kolik let leží mezi zadanými daty a přepočítá se to na počet dnů. Součet těchto tří údajů dá požadovaný rozdíl.

Druhou možnost demonstruje obrázek 2.2. U každého data se spočítá počet dnů od počátku letopočtu a tyto dva údaje se pak odečtou.

Bližší zkoumání obou možností ukázalo, že první varianta není vhodná, protože vyžaduje ošetření příliš mnoha výjimečných stavů. Příkladem je situace, kdy obě data leží ve stejném roce. Dalším problémem jsou přestupné dny, které se mohou vyskytnout uvnitř nebo vně počítaných intervalů, což vyžaduje ošetření hned čtyř možností. Druhá varianta výpočtu je výhodnější i s ohledem na započítávání přestupných let, protože obě data lze před vlastním odečtením zpracovat stejným způsobem[2] a tedy jediným podprogramem. Z těchto důvodů jsem toto řešení vybral pro implementaci problému.

[2]Výpočet počtu dnů od začátku letopočtu bude stejný pro obě data, nezávisle na tom, které je starší.

Kapitola 3

Návrh řešení problému

Po analýze problému s počátkem letopočtu a počátkem gregoriánského kalendáře jsem se rozhodl pro akceptování všech dat od počátku letopočtu (tedy od 1.1.1), jako by se i na ně tento kalendář vztahoval. Výpočet zahrnující i počátek gregoriánského kalendáře by totiž vyžadoval zadání údaje o zeměpisné poloze místa, pro které se tento výpočet provádí.

3.1 Výpočet rozdílu

Princip zvoleného řešení je na obrázku 2.2. V tomto obrázku ovšem není vyřešen problém se zahrnutím přechodných let. Zvolil jsem takové řešení, že se nejprve počítá počet dnů od počátku letopočtu do zadaného data, přičemž se neberou přestupné roky v úvahu. K tomuto údaji se přičte počet přestupných let mezi počátkem letopočtu a zadaným datem. Výsledkem je počet dnů od počátku letopočtu včetně přestupných let. Před vlastním výpočtem rozdílu je potřeba správně přehodit hodnoty, aby výsledek nevyšel záporně, protože výpočet probíhá v bezznaménkových číslech.

Poznámka: Protože jde o počítání rozdílu dat, počítá se počet dnů od počátku fiktivního roku nula. Výpočet se tím zjednoduší, protože není potřeba neustále odečítat 365 dní.

3.2 Výpočet přechodných let

Pro navržený algoritmus je potřeba vzorec pro výpočet počtu přechodných let od počátku letopočtu. V tomto vzorci se uplatní všechna tři pravidla pro detekci přestupných let. Pro výpočet počtu dní od roku 0 do konce zadaného roku jsem navrhnul tento vzorec:

$$dni = 365 * rok + \frac{rok}{4} - \frac{rok}{100} + \frac{rok}{400} \tag{3.1}$$

Podmínku pro detekci, zda je rok přestupný, lze vytvořit pomocí operace modulo (%):

$$prestupny = (rok\%4 == 0)\&\&((rok\%100! = 0)||(rok\%400 == 0)) \tag{3.2}$$

3.3 Analýza vstupních dat

V zadání je přesně specifikován formát vstupních dat včetně oddělovačů mezi jednotlivými složkami jednotlivých dat. V jazyce C lze tento typ formátovaných dat analyzovat pomocí funkce `scanf`, takže je zbytečné vymýšlet speciální algoritmus. Po zavolání funkce `scanf` je potřeba detekovat případné chyby rozsahu (např. datum menší než 1.1.1) a detekovat nesmyslné údaje (např. 29.2.2003, 32.18.2000, atd.)

3.4 Volba rozsahu

Při testech předpokládám, že datový typ `int` má na dnešních počítačích alespoň 32 bitů. Pokud by bylo potřeba aplikaci přenést na platformu s menší délkou slova, bude aplikace větší letopočty považovat za nelegální hodnoty. Maximální hodnota typu `int` bez znaménka je 4294967296. Když tuto hodnotu vydělíme délkou kalendářního roku (365,25 dní[1]), vyjde maximální počet let zobrazitelných pomocí počtu dnů. Pro 32 bitovou architekturu tato hodnota vychází něco přes 11 miliónů let.

V programu je nutné tuto hodnotu znát, protože se každé datum převádí na počet dní od imaginárního roku nula. Při zadání většího letopočtu než je toto maximum, by se tento počet dní nevešel do proměnné typu `int`. Toto omezení je daň za eleganci zvoleného řešení. Pokud by z nějakého důvodu bylo nutné počítat s většími letopočty, bylo by potřeba zvolit jiný algoritmus výpočtu.

3.5 Specifikace testů

Z návrhu řešení vyplývá několik rizikových oblastí, které je potřeba otestovat – chybný rozsah vstupních hodnot, chybně zadané datum (neodpovídá předepsané syntaxi), nesmyslné datum a chyby při výpočtu (hlavně s přestupnými roky).

Test 1: Chybná syntaxe ⟶ Detekce chyby.

```
01.01.2000+02.01.2000
01,01,2000-02,01,2000
02 . 01 . 2000 - 1 . 1 . 2000
aleluja
```

Test 2: Nesmyslná data ⟶ Detekce chyby.

```
29.02.2001-29.2.2000
01.15.2001-31.4.2000
```

Test 3: Data mimo povolený rozsah hodnot ⟶ Detekce chyby.

```
1.15.2001-15.2.0
1.1.1-31.12.110000001
```

Test 4: Správnost výpočtu ⟶ Předpokládaná správná hodnota.

```
02.01.2000-1.1.2000 -> 1
1.1.2000-01.01.2000 -> 0
28.02.2000-28.2.2001 -> 366
29.2.2000-28.02.2001 -> 365
29.02.2000-1.03.2001 -> 366
1.03.2000-28.02.2001 -> 364
01.03.2001-29.02.2000 -> 366
31.12.11000000-15.10.1582 -> 4017089764
31.12.11000000-1.1.1 -> 4017667499
17.00004.1978-7.3.24063 -> 8066340
```

[1] Jde o zjednodušení, protože tato hodnota zanedbává onu stoletou korekci přestupných let.

Kapitola 4

Popis řešení

Při implementaci jsem vycházel ze závěrů popsaných v předchozích kapitolách. Vlastní výpočet rozdílu dvou dat je implementován podle vzorců 3.1 a 3.2.

4.1 Ovládání programu

Program funguje jako konzolová aplikace, má tedy pouze textové ovládání. Při spouštění program reaguje na jediný parametr `-h`. Pokud je s tímto parametrem zavolán, neprovádí žádný výpočet, ale vypíše obrazovku s nápovědou.

Po spuštění program očekává na standardním vstupu řádek s údaji v zadaném formátu. Pokud tam tento řetězec nenalezne, nebo není dodržen vstupní formát, tiskne chybové hlášení. To se tiskne i v případě, že zadaná data jsou sice syntakticky správně, ale představují neexistující datum. V případě korektního vstupu se vypíše výsledný počet dnů na jediný řádek.

Výhodou takto strohého ovládání je, že program může být použit ve skriptech (dávkových souborech) a jím produkovaný výsledek může být použit jiným programem pro další výpočet.

4.2 Volba datových typů

Pro uložení hodnot výsledku jsem zvolil datový typ `unsigned int` (viz 3.4). Pro uložení jednotlivých dat slouží struktura `TDate`, která obsahuje tři položky typu `unsigned int` pro den, měsíc a rok. Datum je ve své podstatě heterogenní útvar, takže by nebylo vhodné ukládat jej například do pole. Struktura navíc poskytuje prostor pro případné budoucí rozšíření například o časový údaj.

4.3 Vlastní implementace

Parametry příkazové řádky zpracovává funkce `doParams`, která je spouštěna jako první ve funkci `main`. Poté se ze standardního vstupu přečte textový řetězec se zadanými daty a předá se funkci `readDates`. Ta pomocí volání `scanf` analyzuje vstupní řetězec a detekuje v něm případné chyby. V případě, že je vstup v pořádku, naplní dvě struktury `TDate`, které vrátí pomocí parametrů předávaných odkazem. Pro otestování správnosti zadaných dat se volá funkce `isDateOk`, která vrací logickou hodnotu.

Pro vlastní výpočet rozdílu slouží funkce `getDiff`. Tato funkce dvakrát volá funkci `getDays` a provede odečtení. Funkce `getDays` pomocí funkcí `getDaysToYear`, `isLeapYear` vypočte podle výše popisovaného vzorce počet dní od roku nula. Pro výpočet počtu dní od počátku roku do zadaného data slouží tabulka (pole) `daysToMonth`, která obsahuje dvanáct hodnot, jež představují počet dní od začátku roku do začátků všech dvanácti měsíců.

Kapitola 5

Závěr

Navrhl jsem dva způsoby, jakými lze zadaný problém řešit. Vybral jsem řešení, které umožňuje vytvořit elegantní algoritmus ovšem za cenu omezení obecnosti. Toto omezení se projeví při počítání s letopočty v řádu desítek miliónů, což je vzhledem k charakteru úlohy akceptovatelné. Pokud by bylo potřeba pracovat s takto velkými letopočty, bylo by potřeba použít jiný algoritmus.

Program počítá s daty od začátku letopočtu podle gregoriánského kalendáře. Kvůli výše zmíněným problémům s identifikací jeho počátečního data, nebylo do řešení zahrnuto omezení na data mladší než rok 1582 (1583, 1700, ...) – ostatně zadání nic takového nepožaduje. Tyto problémy by mohly být řešeny v dalších verzích programu jako volitelná rozšíření.

Program byl otestován se všemi navrženými testovacími hodnotami a odladěn tak, aby všechny testy proběhly správně podle předpokladů. Program přesně dodržuje požadavky kladené na formát vstupních a výstupních dat, takže může být bezproblémově používán spolu s dalšími programy ve skriptech nebo jiných programech.

Navržené řešení je bez problémů přenositelné na všechny platformy, které používají alespoň 32 bitové registry. Při přenosu na platformu s menší velikostí registru by program byl schopen zpracovat menší rozsah dat.

Program byl úspěšně otestován v prostředí operačních systémů Linux a MS Windows.

Příloha A

Metriky kódu

Počet souborů: 1 soubor

Počet řádků zdrojového textu: 240 řádků

Velikost statických dat: 744 B (statické a globálních proměnné a konstanty – položky `bss` a `data` získané programem `size`)

Velikost spustitelného souboru: 14 528 B (systém Linux, 64 bitová architektura, při překladu bez ladicích informací se zapnutým parametrem -O3)

Bibliografie

WIKIPEDIE, 2023a. *Wikipedie/Gregoriánský kalendář* [online]. [cit. 2020-08-23]. Dostupné z: https://cs.wikipedia.org/wiki/Gregori%C3%A1nsk%C3%BD_kalend%C3%A1%C5%99.

WIKIPEDIE, 2023b. *Wikipedie/Kategorie:Kalendářní systémy* [online]. [cit. 2020-08-23]. Dostupné z: https://cs.wikipedia.org/wiki/Kategorie:Kalend%C3%A1%C5%99n%C3%AD_syst%C3%A9my.