Generátory náhodných čísel

V. Bílý
Fakulta jaderná a fyzikálně inženýrská, Břehová 7, 115 19 Praha 1
bilyvit@fjfi.cvut.cz

Abstrakt

Během svého experimentu jsem se zajímal a porovnával různé generátory náhodných/ pseudonáhodných čísel, jejich využití ale především jejich testování a zjišťování jejich kvality.

1 Úvod

Za cíl jsem si stanovil sestavit funkční domácí a co nejjednoduší generátor pravých náhodných čísel a porovnat jeho kvalitu s běžně dostupnými generátory (především softwarovými generátory pseudonáhodných čísel). K tomu jsem ale nejdříve potřeboval nějaké testy, na jejichž základě bych mohl o náhodnosti a případně i o kvalitě jednotlivých generátorů rozhodnout.

2 Teorie

Při generování čísel se rozlišují dva základní typy generátorů (vygenerovaných čísel) - generátory pseudonáhodných čísel se zkratkou PRNG (pseudo random number generator) a generátory pravých náhodných čísel se zkratkou TRNG (true random number generator).

Pseudonáhodná čísla nejsou čistě náhodná. Jsou totiž vypočtena softwarem pomocí funkce, zpravidla to bývá tzv. kongruentní generátor, který čísla vypočítává pomocí rekurentního zadání funkce:

$$x_0 = Z \tag{1}$$

$$x_{n+1} = (A * x_n + C) mod M \tag{2}$$

kde Z, A, C, M jsou konstanty na jejichž volbě závisí nagenerovaná posloupnost, kdy Z je první číslo posloupnosti a modulo M je zbytek po celočíselném dělení M. Z toho jasně vyplývá že volba konstanty M hraje v nagenerované posloupnosti zcela zásadní roli. To proto, že základním rozdílem mezi pravými a pseudonáhodnými čísli je ten, že
u pseudonáhodných se po nagenerování větší posloupnosti začne objevovat perioda. A právě velikost periody závísí především (ale ne zcela) na volbě konstanty M (taky se občas nazývá perioda). Asi nejznámější v knihovně C/C++ je funkce Rand(). Ta má pevně určeny všechny konstanty. Z toho vyplývá, že nagenerovaná posloupnost bude pokaždé úplně stejná, což může být výhodou, ale i nevýhodou - závisí na využití.
K nagenerování pokaždé jiné posloupnosti pseudonáhodných čísel, se v jazyku C/C++

nejčastěji používá funkce Srand(). Ta si při každém zavolání vypočítá všechny konstanty na základě světového času a aktuálním datumu. Tím se docílí toho, že konstanty budou pokaždé jiné a tím i vygenerovaná posloupnost. Obě uvedené funkce navíc výsledná čísla vydělí maximálním rozsahem, čímž získáme výsledná čísla v intervalu $\langle 0 - 1)$. To je výhodné, protože uživatel si poté může jednoduchým vynásobením libovolně nastavit rozsah a jestli chce výsledná čísla celá nebo reálná.
Ryze náhodná čísla tedy narozdíl od pseudonáhodných nejsou vypočtená a nemají žádnou periodu. Proto k jejich generování nelze použít software, ale je třeba nalézt nějaký zcela náhodný fyzikální jev - proto se také můžeme setkat s pojmem hardwarový generátor čísel. Asi úplně nejjednoduším může být třeba házení dobře kalibrovanou kostkou, mincí,... Je ale jasné, že tato metoda je z hlediska zpracování a nagenerování většího množství čísel nepoužitelná. Ve skutečnosti se využívají různé fyzikální jevy třeba v kvantové fyzice, ale nejčastěji se používá radioaktivní zářič. To proto, že počet vyzářených částic za jednotku času je relativně velké číslo, což umožňuje generovat velké množství náhodných čísel za jednotku času, a dále protože pravděpodobnost rozpadu částice za poločas rozpadu je přesně 50 procent, což je velmi výhodné pro další zpracování dat počítačem.

3 Využití generátorů čísel

Generátory náhodných/pseudonáhodných čísel se hodně využívají například v šifrování a nejrůznějších počítačových simulacích. Zde si většinou vystačíme i s kvalitnějšími pseudonáhodnými generátory. Kde je ovšem pro přesné výsledky nutné použít ryze náhodná čísla, je výpočetní metoda Monte Carlo.
Metoda Monte Carlo je vhodná k výpočtu například obsahu plochy nejrůznějších objektů neznámé velikosti, která je jen obtížně spočítatelná matematicky. Funguje na tom principu, že v počítačové simulaci "umístíme"objekt neznámé velikosti na objekt známé velikosti (např. čtverec) a provádíme náhodné výstřeli - tzn., že si zvolíme zcela náhodnou souřadnici a zkoumáme, zda jsme se trefili do neznámého objektu nebo ne.
Při provedení velkého počtu takovýchto výstřelů můžeme jednoduše z poměru zásahů a celkového počtu výstřelů přibližně dopočítat plochu neznámého tělesa. A právě volba souřadnic pro dosažení vysoké přesnosti musí být zcelá náhodná, proto jsou zde generátory pseudonáhodných čísel zcela nevhodné.

4 Testování náhodnosti

Testování náhodnosti nagenerované posloupnosti čísel je poměrně komplexní problém. Existuje celá řada nejrůznejších statistických testů a dost záleží na jejich interpretaci - nelze totiž nikdy s naprostou jistotou rozhodnout o náhodnosti či nenáhodnosti dané posloupnosti. Během svého pokusu jsem pracoval s volně přístupným programem ENT [1] (pseudorandom number sequence test), který obsahuje celou řadu testů:

1. Entropie - Popisuje informační hustotu souboru v bitech na znak. Čím vyšších hodnot dosahuje, tím je soubor informačně hustší. Hodnoty pro komprimované soubory (téměř maximálně husté, téměř náhodný vzorek) se blíží 8. Například entropie psaného textu většinou pohybuje okolo 5.

2. Možnost komprese - Udává v procentech o kolik by byl soubor menší, pokud by se optimálně zkomprimoval. Počítá se zde s kousky dat o velikosti jednoho bytu, výsledky je proto třeba brát trochu s rezervou, protože některé algoritmy, které pracují s většími rámci čísel mohou dosahovat lepších výsledků, pokud jsou v datech opakující se sekvence více bytů. Obecně platí, že čím menší možnost komprese tím větší náhodnost čísel.

3. Chí kvadrát test dobré shody - Poměrně důležitý statistický test, který je citlivý na běžné neduhy nenáhodných posloupností čísel. Test funguje tak, že si v našem případě rozdělí posloupnost na více navazujících disjunktních intervalů a spočítá četnost jednotlivých znaků na těchto intervalech. Dále si na těchto intervalech spočítá teoretickou četnost pro náhodná čísla a následně porovnáním teoretické a skutečné četnosti spočítá testovací kritérium podle rovnice (3).

$$\chi^2 = \sum_{i=0}^{k} \frac{(X_i - P_i)^2}{P_i} \tag{3}$$

   kde X_i je skutečná četnost a P_i očekávaná četnost.
   Program dále pro sruzumitelnost testu dopočítá v kolika procentech případů by tuto hodnotu překonala skutečně náhodná data. Hodnota by se tedy měla pro zcela náhodná data blížit k 50 procentům. Ale vzhledem k velké citlivosti testu lze výsledky testu přibližně interpretovat jak je uvedono v Tabulce 1.

4. Aritmetický průměr - Tento test spočítá aritmetický průměr všech bytů i bitů.

5. Hodnota Monte Carlo pro π - Využije čísla posloupnosti k výpočtu π pomocí metody Monte Carlo. Test bere dvojice po sobě jdoucích čísel a interpretuje je jako x-ové a y-ové souřadnice. Takto získané náhodné body vynaší na čtverec s vypsanou čtvrtkružnicí o poloměru rovném délce strany. Pokud je vzdálenost náhodného bodu od počátku menší než poloměr kružnice považujeme to za zásah. Pak již jednoduše dopočítá hodnotu π z rovnice (4) a uvede i jeho odchylku od skutečného π v procentech.

$$S_1 = \frac{\pi r^2}{4}; \quad S_2 = r^2; \; odkud: \quad \pi = 4\,\frac{S_1}{S_2} = 4\,\frac{X_1}{X_c} \tag{4}$$

   kde S_1 je obsah čtvrtkružnice, S_2 je obsah čtverce, X_1 je počet zásahů a X_c je počet celkových výstřelů.

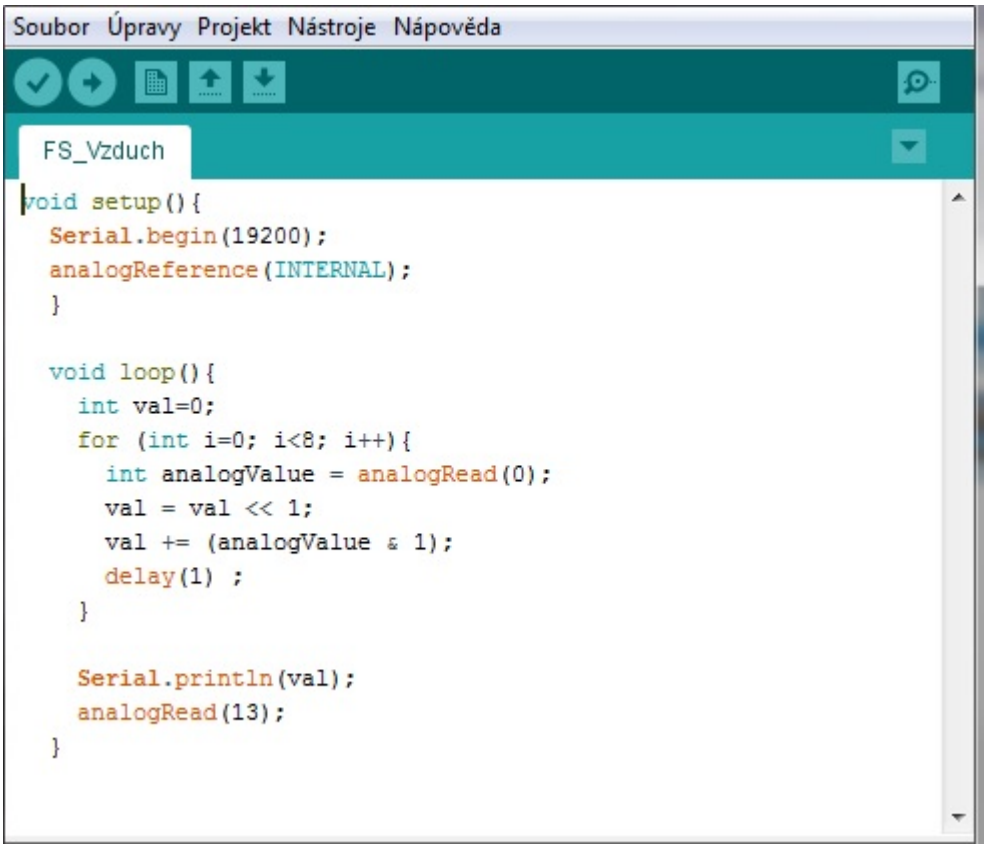
5 Vlastní pokus

Během vlastního pokusu jsem chtěl dvěma různými způsoby nagenerovat posloupnosti náhodných čísel, otestovat a porovnat s softwarovými generátory. Ve všech případech jsem generoval 8-bitová čísla (rozsah 0 - 255.) Jako první způsob jsem použil Arduino se sbíráním

Výsledná hodnota[%]	**Interpretace**
0 - 1	téměř jistě nenáhodná
1 - 5	podezředlá (asi nenáhodná)
5 - 10	trochu podezřelá (možná nenáhodná)
10 - 90	pravděpodobně náhodná data
90 - 95	trochu podezřelá (možná nenáhodná)
95 - 99	podezředlá (asi nenáhodná)
99 - 100	téměř jistě nenáhodná

Tabulka 1: Interpretace výsledků Chí kvadrát testu

el. mag. šumu ze vzduchu viz. Obrázek 1. Důležité bylo, aby se s Arduinem a hlavně se zvoleným portem nemanipulovalo - stačilo přes port přejet prstem a čísla se zcela zjevně stala nenáhodnými. Následný výstup na sériovém portu jsem si ukládal
do textového souboru (stejně tak posloupnosti nagenerované funkcemi Rand() a Srand ()). Následným otestováním posloupností získaných Arduinem a funkcemi Rand() a Srand() jsem dostal tyto výsledky: Při testu Entropie posloupnosti funkcí Rand() i Srand() dosáhly skoro stejné hodnoty, přibližně 7.19, zatímco Arduinem získané posloupnosti se pohybovali mezi 7.18 až 7.22. Při testu možnosti komprese posloupnosti získané všemi třemi způsoby pohybovali mezi 60 - 67 %. Takto vysoké číslo je způsobeno především zvoleným formátem a jeho absolutní hodnota je pro nás nezajímavá. Zajímá nás odlišnost v možnosti komprese u jednotlivých posloupností, tu jsem ale i při opakování testů neobjevil. Při Chí kvadrát testu fce Rand() dosáhla hodnoty pouze 0.01%, funkce Srand od 0.01 - 2 % zatímco posloupnost získaná Arduinem dosahovala až 20 %. Při testu aritmetickým průměrem dosahovali všechny posloupnosti shodně 127.5 pro byty a 0.5 pro bity. Při testu výpočtem π pomocí metody Monte Carlo opět nebyl vidět žádný větší rozdíl mezi jednotlivými způsoby. Přesnost vypočteného π závisela na počtu bytů - při 10 000 znacích jsem získával π s chybou $\pm$ 0.2 - s přesností na 1 desetinné místo. Pro přesnost na 2 desetinná místa bylo třeba posloupnost o řádech minimálně miliónů - desetimiliónů čísel.
Shrnutím lze tedy říct, že se mi nejspíš povedlo sestavit generátor náhodných čísel, anebo alespoň kvalitnější generátor pseudonáhodných čísel než byly oba testované, z nichž lépe v třetím testu vycházela funkce Srand().



Obrázek 1: Nastavení arduina

Poděkování

Chtěl bych poděkovat ing. Vojtěchovi Svobodovi, CSc. za zapůjčení součástek potřebných k pokusu a dále bych chtěl poděkovat Tomášovi Truhlářovi za poskytnutí zdrojového kódu, který jsem v upravené verzi použil během svého experimentu.

Literatura

[1] J. Walker, ENT, http://www.fourmilab.ch/random/
[2] M. Malý, http://www.root.cz/clanky/hardwarovy-generator-nahodnych-cisel-aneb-nahoda-z-atomu/