Proiect 1

Formatori:
Tutor: Stiegelbauer Paul
Tutor: Iovanovici Alexandru
+1

Data de începere a cursului:
19.02.2024
Utilizatori înscriși
Calendar
Note

Proiect 1

Scopul acestei prime teme de proiect este determinarea experimentala a claselor de complexitate algoritmica si dezvoltarea unei biblioteci de functii pentru masurarea timpilor de executie a subrutinelor de program C.

1. Generarea de numere pseudoaleatoare

Se foloseste funcata rand() din stdlib.h care returneaza la fiecare apel un numar natural cuprins intre [0, RAND_MAX).

Initializarea generatorului de numere pseudoaleatoare se poate face folosind functia srand(unsigned) care primeste un *seed* si permite generarea de secvente pseudoaleatoare distincte, intre rulari succesive. Apelul la srand() trebuie facut o singura data, ca parte a rutinei de initializare, inainte de orice apel la rand().

O practica uzuala este utilizarea rezultatului functiei time(0), care returneaza o data de tipul time_t, cu valoare distincta la fiecare apel (timpul curge unidirectional) si ganranteaza ca la fiecare apel se obtine o alta secventa pseudoaleatoare. Astfel se va folosi srand(time(0));

```
// genereaza o secventa de numrere pseudoaleatoare, distincta la fiecare apel
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
    // Foloseste ora curenta pentru initializarea PRNG-ului
    srand(time(0));
    for(int i = 0; i<4; i++){
        printf(" %d ", rand());
        }
    return 0;
}
```

2. Masurarea timpului de executie

2.1 Utilizarea utilitarului time

Pe sisteme Linux/UNIX aveti "by default" utilitarul time care permite executarea unui program (primit ca si argument in linie de comanda), masurarea si afisarea timpilor: total de executie, utilizat de sistemul de operare, si timpul efectiv de rulare al programului analizat.

Spre exemplu un posibil output pe MacOS este

```
./a.out  0.02s user 0.00s system 79% cpu 0.031 total
```

2.2 Utilizarea functiilor din time.h

Pentru o analiza mai amanuntita (spre exemplu a unei singure functii sau chiar a unei portiuni dintr-o functie) se poate folosi functia clock, declarata in time.h.
In mod uzual se apeleaza clock() la inceputul si sfarsitul portiunii de analizat, se scad valorile si converteste in timp-real, prin impartire la CLOCKS_PER_SEC (numarul de "cloci" ai procesorului), astfel:

```
/* preluat din referinta [2] */
#include <time.h>

clock_t start, end;
double cpu_time_used;

start = clock();
… /* Do the work. */
end = clock();
cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
```

Rezolutia tipica oferita cf. [2] este oriunde intre 1e-2 si 1e-6 dintr-o secunda. C++ ofera in std::chrono o serie de functie cu precizie sporita. Deoarece pentru aplicatii practice de multe ori timpii de executie, pentru seturi reduse de date, procesoare performante si algoritmi efiecienti, sunt in zona pragului de zgomot/eroare se recomanda masurarea unui set mai mare de executii ale aceluiasi algoritm (spre exemplu 100 de apeluri ale functiei de sortare analizate) si impartirea timpului gasit la numarul de apeluri.

Aspecte ce tin de hardware (mecanisme de memorie cache, lucrul cu discul) si/sau sistem de operare (multithreading) pot influenta semnificativ rezultatele, deci se recomanda reluarea analizei in anii superiori dupa parcurgerea disciplinelor cu accent pe arhitectura sistemelor de calcul si/sau sisteme de operare.

3. Aplicatii

1. Implementati o functie **unsigned* makeRandArray(unsigned seed, unsigned n)**, care primeste un *seed* si un numar natural **n** si returneaza un vector alocat dinamic, continand **n** numere naturale pseodoaleatoare, uniform distribuite

2. Implementati o functie **int* makeRandLimitArray(int seed, unsigned n, int a, int b)** care primeste un seed, si numerele naturale **n**, **a** si **b** si returneaza un vector alocat dinamic, continand **n** numere intregi pseudoaleatoare, uniform distribuite, cuprinse intre **a** si **b**

3. Implementati o functie cu antetul **int* makeRandFlexiArray(unsigned n, int (*getNewElem)(int*, unsigned))** care genereaza un vector alocat dinamic cuprinzand elemente numere psoedoaleatoare. Pentru generarea unui nou element din vector se va face apel la o functie "concreta" primita sub forma unui pointer la o functie care primeste drept argumente un tablou si numarul sau curent de elemente. Implementati functii "concrete" pentru generarea unor vectori monoton crescatori si monoton descrescatori.

4. Resurse

1. https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/
2. https://www.gnu.org/software/libc/manual/html_node/CPU-Time.html

◄ Inregistrare curs Saptamana 5, anul 2021-2022!!!

Sari la...

Predare Proiect 1 ►