Chapitre 4 – Algorithmes mathématiques – Exercice Algo 4.4

Méthode Monte Carlo (calcul de pi)
Mathematische Algorithmen – Monte Carlo Methode (Berechnung von pi)

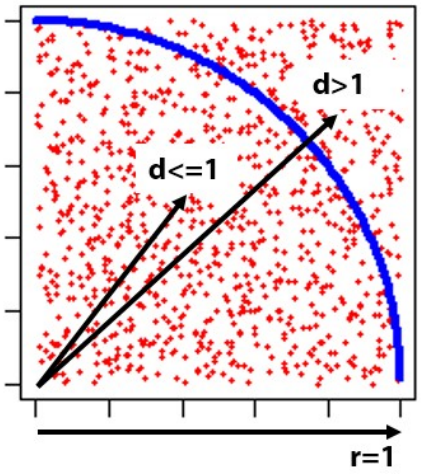
Surface du carré
Fläche des Quadrats $A_r = r^2$

Surface du quart de cercle:

Fläche des Viertelkreises: $A_q = \frac{\pi * r^2}{4}$

Ratio entre les deux:

Verhältnis der beiden : $\frac{A_q}{A_r} = \frac{\pi}{4}$ = π/4 ou π = $\pi = 4 * \frac{A_q}{A_r}$



Maintenant, on jette des boules d'une manière aléatoire dans le carré et on compte, combien de boules sont placés dans le quart ce cercle. Selon la formule en haut, le nombre de boules avec d<=1 divisé par le nombre de boules jeté et multiplié par 4 donne le nombre π.
Jetzt werfen wir nach dem Zufallsprinzip Bälle in das Quadrat und zählen, wie viele Bälle im Viertelkreis auftreffen. Nach der obigen Formel ergibt die Anzahl der Bälle mit d<=1 geteilt durch die Anzahl der geworfenen Bälle und multipliziert mit 4 die Zahl π.

a) Écrivez un programme qui calcule π avec la méthode expliquée ci-dessus et avec 100'000 itérations. Lancez 20 fois le programme et calculez la moyenne ainsi que la déviation standard de ces 20 valeurs.
Schreiben Sie ein Programm, das π mit der oben beschriebenen Methode und mit 100'000 Iterationen berechnet. Führen Sie das Programm 20 Mal aus und berechnen Sie den Mittelwert und die Standardabweichung dieser 20 Werte.

b) Modifiez le programme pour que l'algorithme arrête si le résultat de pi s'approche à 0.1‰ de la valeur littérature 3.14159265.Est-ce que l'analyse de ce résultat confirme votre résultat de a.) ?
Modifizieren Sie das Programm so, dass der Algorithmus stoppt wenn das Pi-Ergebnis 0,1‰ des Literaturwertes 3.14159265 erreicht. Bestätigt die Analyse dieses Ergebnisses Ihr in a.) erzieltes Ergebnis?

Notes: La fonction `int rand();` retourne un int entre 0 et la constante ***RAND_MAX***.
Pour initialiser le générateur des nombres aléatoires, il est bien de lancer au début de programme la fonction `void srand(unsigned int seed);` avec une constante 'aléatoire'. Très souvent, on prend une variable représentant la date et l'heure actuelle comme seed (fonction `time_t`

time(*NULL);*)

Bemerkung: Die Funktion int ***rand();*** gibt ein int zwischen 0 und der Konstanten ***RAND_MAX*** zurück. Um den Zufallszahlengenerator zu initialisieren ist es ratsam, zu Beginn des Programms die Funktion *void* ***srand****(unsigned int seed)* mit einer 'Zufalls'-Konstanten zu starten. Sehr oft nehmen wir eine Variable als Seed, die das aktuelle Datum und die aktuelle Uhrzeit repräsentiert (Funktion *time_t* ***time****(NULL);*)

Le template disponible sur moodle est le suivant:
Das auf moodle verfügbare Template ist wie folgt:

```
1.  /**
2.    ******************************************************************************
3.    * @file    main.c
4.    * @author  Wolfram Luithardt / Roland Scherwey
5.    * @date    10 December 2020
6.    * @brief   Computation of pi using the method of Monte Carlo
7.    ******************************************************************************/
8.
9.  #include <stdio.h>  // for printf()
10. #include <stdlib.h> // for srand() / rand() / RAND_MAX
11. #include <time.h>   // for time()
12. #include <stdint.h> // for uint32_t
13. #include <math.h>   // for fabs()
14.
15. #define PI 3.14159265358979323846264338327795
16.
17. typedef uint32_t IterationType;
18. IterationType const cNbrOfIterations = (10*100);
19.
20. int main()
21. {
22.   srand(time(NULL)); // seeds the random number generator used by rand()
23.
24.   // create one random point in coordinate system with range x[0..1] / y[0..1]
25.   double const cNumX = (double)rand() / RAND_MAX;
26.   double const cNumY = (double)rand() / RAND_MAX;
27.
28.   // Display coordinates of random point
29.   printf("Test number is (%.10f,%.10f)!\n",cNumX, cNumY);
30.
31.   // Dummy indication for iterations
32.   printf("Computing pi with %u iterations...\n",cNbrOfIterations);
33.
34.   return 0;
35. }
36.
37.
```

Solution a)

```
#include <stdio.h>  // for printf()
#include <stdlib.h> // for srand() / rand() / RAND_MAX
#include <time.h>   // for time()
#include <stdint.h> // for uint32_t
#include <math.h>   // for fabs()

#define PI 3.14159265358979323846264338327795

typedef uint32_t IterationType;
IterationType const cNbrOfIterations = (1000*1000);

int main()
{
   srand(time(NULL)); // initialize random number generator using seed of OS time
   // a) 100'000'000 iterations
   printf("a) Computing pi with %u iterations...\n",cNbrOfIterations);

   IterationType nbrOfTimesWithinCircle = 0;
   double computedPi = 0.0;
   for (IterationType i=1; i<=cNbrOfIterations; i++)
   {
     // create one random point in coordinate system with range x[0..1] / y[0..1]
     double const cNumX = (double)rand()/RAND_MAX;
     double const cNumY = (double)rand()/RAND_MAX;
     // compute corresponding vector
     double const cDiag = cNumX*cNumX + cNumY*cNumY;

     // check if vector length is smaller or equal the radius of the circle (<= 1.0)
     if (cDiag<=1.0) {
       nbrOfTimesWithinCircle++;
     }
   }
   // compute pi at the end of the iterations
   computedPi =  (    (double)(nbrOfTimesWithinCircle)
                   / (double)(cNbrOfIterations      ))
                   * 4.0;

   printf("    Result pi=%.10f\n", computedPi);
  return 0;
}
```

Solution b)

```c
#include <stdio.h>  // for printf()
#include <stdlib.h> // for srand() / rand() / RAND_MAX
#include <time.h>   // for time()
#include <stdint.h> // for uint32_t
#include <math.h>   // for fabs()

#define PI 3.14159265358979323846264338327795

typedef uint32_t IterationType;
IterationType const cNbrOfIterations = (1000*1000);

int main()
{
   srand(time(NULL)); // initialize random number generator using seed of OS time
   // b) stops if error is smaller than 0.01%
   printf("b) Computing pi until given precision of 0.01%%\n");

   // Next variables are necessary to declare outside for(),
   // as they will be accessed later on
   IterationType nbrOfTimesWithinCircle = 0;
   double computedPi = 0.0;
   IterationType i;
   double const cPiErrAllowed = PI / 10000;      // 0.01%
   double errPiAct = 0.0;
   for(i=1; i<=cNbrOfIterations; i++)
   {
     // create one random point in coordinate system with range x[0..1] / y[0..1]
     double const cNumX = (double)rand()/ RAND_MAX;
     double const cNumY = (double)rand()/ RAND_MAX;
     // compute corresponding vector
     double const cDiag = cNumX*cNumX + cNumY*cNumY;

     // check if vector length is smaller or equal the radius of the circle (<= 1.0)
     if (cDiag <= 1.0) {
       nbrOfTimesWithinCircle++;
     }

     // compute pi each time
     computedPi = 4.0 * (double)nbrOfTimesWithinCircle / (double)i;

     // compute absolute value of difference
     errPiAct = fabs(computedPi - PI);

     // Check if desired precision has already been achieved
     if (errPiAct < cPiErrAllowed) {
         printf(" Yes ! Min error reached - break after %u iterations\n",i);
         break; // exists for-loop
     }

   }
   printf("    Result pi=%.10f, error is %.8f (allowed %.8f)\n",
                  computedPi , errPiAct, cPiErrAllowed);

   printf("%u times within circle out of %u iterations \n",nbrOfTimesWithinCircle, i);
  return 0;
}
```