esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

TP n°2 LAB1413

Fonctions

Nicolas Bodin
Arnaud Bannier
Matthieu Le Berre

Ce TP est consacré à la réalisation de fonctions. Lorsqu'une saisie utilisateur est demandée, elle doit être réalisée dans le `main()`. Les valeurs ainsi récupérées sont ensuite passées en arguments de la fonction.

Exercices d'introduction

Si vous pensez avoir besoin de réaliser ces exercices pour vous mettre en confiance, faites-le avant le reste du TP.

Exercice 1. Passage par valeur

1.1) Créez la fonction `void Swap (int a, int b)` qui échange les valeurs des variables `a` et `b`.

1.2) Appelez cette fonction depuis le `main()` et affichez les valeurs des variables avant et après l'appel à `Swap`. Expliquez le résultat à votre chargé de TP.

Exercice 2. Nombre miroir

Une fonction miroir d'un entier est une fonction qui retourne l'entier symétrique de l'entier en entrée. Par exemple, le miroir de 1337 est 7331. Rédigez les fonctions suivantes.

2.1) `int Miror(int N)` : qui retourne le nombre miroir de l'entier `N`.

2.2) `int NbDigit(int N)` : qui retourne le nombre de chiffres du nombre `N`.

2.3) `int Palindromize(int N)` : qui retourne le nombre palindrome de l'entier `N`. On rappelle que le palindrome du nombre 1234 est 1234321.

Exercice 3. Soirée arrosée

Un étudiant de première année décide de réaliser un punch pour fêter son anniversaire. Un peu trop optimiste, il décide de vider une bouteille de 2L de rhum à 40 degrés dans 2L de jus de fruits. Ses amis le trouvant trop fort, et voulant faire la fête longtemps, ils décident de le diluer en buvant chaque heure 1L de punch, et en y ajoutant à la fin de l'heure 1L de jus de fruits. La soirée ayant commencé à 20h, quel est le degré d'alcool du punch à 23h30 ? à 00H15 ?

Résolution d'équation du second degré

Retour au lycée pour cet exercice ... Nous vous demandons de créer un programme qui résout une équation du second degré dans $\mathbb{R}$. Nous rappelons que pour résoudre l'équation $ax^2 + bx + c = 0$ avec $a, b, c \in \mathbb{R}$, il faut tout d'abord calculer la valeur du déterminant $\Delta = b^2 - 4ac$. Les solutions sont données par :

$$\begin{cases} \text{Si } \Delta > 0 \text{ alors} & x_1 = \frac{-b-\sqrt{\Delta}}{2a} \text{ , } x_2 = \frac{-b+\sqrt{\Delta}}{2a}. \\ \text{Si } \Delta = 0 \text{ alors} & x = \frac{-b}{2a}. \\ \text{Si } \Delta < 0 \text{ alors} & \text{pas de solution.} \end{cases}$$

Exercice 4.

Vous devez donc dans un premier temps demander à l'utilisateur de saisir les valeurs a, b et c avant d'afficher la solution. Vous devrez afficher la solution de deux façons différentes : la première sous forme de nombre réel (par exemple `x1 = 3.38 et x2 = -1.74`) et la deuxième sous forme de fraction (par exemple `x=(-2-rac(5))/14`). Le calcul du déterminant et l'affichage des solutions doit être réalisé au travers d'une fonction prenant en argument les trois réels `a`, `b` et `c`.

Approximation de racine

Continuons dans les mathématiques ! L'objectif est de calculer les N premiers termes d'une suite numérique. Votre programme doit être capable de demander à l'utilisateur quelle suite générer à l'aide d'un menu, et de demander jusqu'à quel rang de la suite afficher les valeurs. Voici la suite en question :

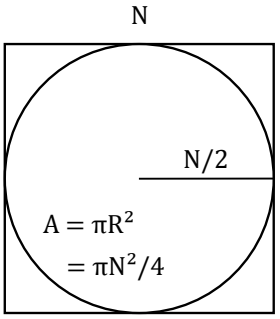
$$\text{Calcul de } \sqrt{A} : \begin{cases} u_0 & = c \\ u_{n+1} & = \frac{1}{2}\left(u_n + \frac{A}{u_n}\right) \end{cases}$$

Exercice 5.

5.1) Créez la fonction permettant d'approcher la valeur de $\sqrt{A}$ lorsque u_n tend vers l'infini, et pour un A fixé. En plus de demander le rang de la suite à l'utilisateur, il faut aussi lui demander la valeur de A. La valeur réelle c est une valeur qui peut être choisie par l'utilisateur.

5.2) Modifiez par la suite votre fonction ainsi que son prototype pour que la fonction s'arrête d'elle même lorsque la différence entre u_n et u_{n+1} est inférieure à un seuil donné par l'utilisateur.

Approximation de π



De nombreuses méthodes existent afin d'approximer la valeur de π en utilisant une fonction ou un algorithme mathématique. Nous allons utiliser la méthode dite de *Monte Carlo*, basée sur le principe de placer un grand nombre de points aléatoirement dans un carré, et vérifier combien appartiennent au cercle inscrit dans ce carré. Afin de bien fixer les idées, le schéma ci-contre vous donne l'explication visuelle de l'approximation.

L'aire du cercle inclus dans un carré de dimension N est $A = \pi N^2/4$. En supposant que $N = 2$, on retrouve $A = \pi$. L'idée est alors de placer des points aléatoirement dans le carré et de compter le nombre de points inscrits dans le cercle. En normalisant avec la surface du carré (dans le cas $N = 2$, la surface du carré est de 4), il est alors possible de trouver une approximation convenable (et probabiliste) de la valeur π.

Exercice 6.

6.1) Dans un premier temps, codez la fonction `float TirageAleatoire (float N)` qui renvoie une valeur flottante aléatoire entre 0 et `N`.

6.2) Afin de vérifier si le point placé aléatoirement est inscrit dans le cercle, il suffit de mesurer sa distance par rapport au centre du cercle, et de vérifier qu'elle est bien inférieure au rayon du cercle. En supposant que le centre c du carré (et donc du cercle) soit situé en $(N/2, N/2)$, et que l'on ait tiré un point aléatoire p en position (x, y), sa distance au centre est :

$$d = \sqrt{\left(x - \frac{N}{2}\right)^2 + \left(y - \frac{N}{2}\right)^2}.$$

Si $d \leq N/2$, cela signifie que le point est inscrit dans le cercle. Codez alors la fonction `int EstInscrit (float N)` qui réalise le tirage d'un point aléatoire inscrit dans le carré de dimension N dont le coin en bas à gauche est situé en $(0, 0)$, et renvoie 1 si le point est inscrit dans le cercle, 0 sinon.

6.3) Il ne reste plus qu'à coder dans le `main()` le reste du programme permettant de demander à l'utilisateur la taille du carré, et d'appeler un grand nombre de fois la fonction `EstInscrit()` (demandez le nombre d'itérations à l'utilisateur). La dernière étape est d'effectuer le ratio (nombre de points inscrits) / (nombre de points tirés) et de multiplier le tout par 4 pour obtenir l'approximation de π.

Temps de réaction

Nous allons réaliser un programme qui va nous permettre de mesurer le temps de réaction d'un individu. Nous avons toujours entendu dire qu'il était en moyenne d'une seconde pour un adulte, nous allons vérifier cela dans le contexte d'un réflexe simple face à un stimulus. L'objectif de ce programme est de générer un signal visuel (stimulus) et de capter le temps que l'utilisateur a mis avant d'appuyer sur une touche du clavier (réaction). Voici le contexte précis du programme.

— Lors du lancement du programme, vous devrez afficher un compte à rebours qui va permettre de préparer l'utilisateur à réagir au stimulus.

— À la fin de ce compte à rebours, un temps d'attente aléatoire devra être établi avant d'afficher le stimulus. Le côté aléatoire de ce temps empêche l'utilisateur de connaitre précisément le moment de l'apparition du stimulus.
— Le stimulus est alors composé de caractères choisis apparaissant dans la console. Nous fixons un temps de réaction limite (par exemple 3 secondes) au delà duquel le programme s'arrête et affiche un message indiquant que l'utilisateur est soit endormi, soit vraiment lent. Durant ces N secondes, le programme affiche à l'écran 20 caractères, à raison de 1 caractères tous les $N/20$ secondes. Cela permet de visualiser clairement le temps de réaction humain.
— Durant cet affichage, le programme se prépare à recevoir la réaction de l'utilisateur via la saisie d'une touche du clavier.
— Lorsqu'une touche est saisie (n'importe laquelle), le programme cesse son affichage et indique à l'utilisateur son temps de réaction.

Pour mener à bien cet exercice, vous allez donc créer les fonctions suivantes.

L'usage de la fonction `scanf()` est strictement interdit.

Exercice 7.

7.1) `int main()` : cette fonction, point d'entrée de notre programme, va uniquement initialiser l'aléatoire et appeler les fonctions nécessaires au bon déroulement du programme. Plus le `main()` est court, plus le code est clair !

7.2) `void CompteARebours (int duree)` : cette fonction permet d'afficher un compte à rebours sur `duree` secondes. Pour cela, vous aurez besoin d'utiliser une fonction permettant de mesurer le temps. Vous connaissez déjà la fonction `int time()` mais celle-ci n'offre une précision qu'à la seconde, suffisante pour la génération de nombre aléatoires avec la fonction `srand()`. Nous allons donc lui préférer la fonction `clock_t clock()` qui offre une précision beaucoup plus importante puisque qu'elle compte le nombre de tics d'horloge du processeur depuis le lancement de la machine. En divisant cette valeur par la constante `CLOCKS_PER_SEC` – qui comme son nom l'indique, nous donne le nombre de tics d'horloge par seconde – nous obtenons le nombre de secondes écoulées depuis l'allumage de l'ordinateur. L'objectif est alors d'appeler la fonction `clock()` dès le début de la fonction `CompteARebours()` afin d'obtenir un point de départ pour le compte à rebours.

7.3) `double TempsAleatoire (int max)` : cette fonction permet de générer un temps d'attente aléatoire dans le programme, d'une durée supérieure à une seconde, mais inférieure à `max` secondes. Les fonctions décrites précédemment nous seront une nouvelle fois utiles. Le processus pour bloquer l'exécution d'un programme est simple, il suffit d'exécuter une boucle d'instructions vide durant le temps souhaité.

7.4) `double DeroulerJeu (int duree)` : cette fonction nous permet de dérouler le programme mesurant le temps de réaction. Elle affiche à l'écran la série de symbole représentant le temps de réaction écoulé. Durant cet affichage, l'appel à la fonction `int kbhit()` (fournie) nous permettra de vérifier si une touche clavier a été saisie par l'utilisateur. Lorsque nous détectons cette dernière, le programme doit arrêter d'afficher les symboles et retourner le temps de réaction de l'utilisateur.