**Travail 2 de traitement numérique des données : Calculatrice**

****

**Par**

**Nicolas Bauthier,**

**Jonathan Lachapelle,**

**Jennifer Denis**

**Table des matières**

1 Enoncé reçu ……………………………………………………………… 4

2 Généralités………………………………………………………………...

3 Fonction Exponentielle……………………………………………………

3.1 Analyse…………………………………………………………...

3.1.1 Explications en français………………………………….

3.1.2 Détails mathématiques……………………………..

3.2 Diagramme d’action……………………………………………...

3.3 Code……………………………………………………………....

3.4 Testing……………………………………………………………

4 Fonction ln(x+1)…………………………………………………………..

4.1 Analyse…………………………………………………………...

4.1.1 Explications en français………………………………….

4.1.2 Détails mathématiques……………………………..

4.2 Diagramme d’action……………………………………………...

4.3 Code……………………………………………………………....

4.4 Testing……………………………………………………………

5 Fonction sin (x)…………………………………………………………...

5.1 Analyse…………………………………………………………...

5.1.1 Explications en français………………………………….

5.1.2 Détails mathématiques……………………………..

5.2 Diagramme d’action……………………………………………...

5.3 Code……………………………………………………………....

5.4 Testing……………………………………………………………

6 Polynôme P(x) de degré max 10………………………………………….

6.1 Analyse…………………………………………………………...

6.1.1 Explications en français………………………………….

6.1.2 Détails mathématiques……………………………..

6.2 Diagramme d’action……………………………………………...

6.3 Code……………………………………………………………....

6.4 Testing……………………………………………………………

7 Probabilité que X <= x sachant que ………………………...

7.1 Analyse…………………………………………………………...

7.1.1 Explications en français………………………………….

7.1.2 Détails mathématiques……………………………..

7.2 Diagramme d’action……………………………………………...

7.3 Code……………………………………………………………....

7.4 Testing……………………………………………………………

**1.Enoncé reçu**

TRAVAIL 2 : CALCULATRICE

Le but de ce travail est de réaliser une calculatrice comportant les fonctionnalités suivantes :

* Calculer et afficher exp(x)
* Calculer et afficher ln(x+1)
* Calculer et afficher sin(x)
* Calculer et afficher sachant que
* Calculer et afficher une racine d’un polynôme de degré max 10.

L’utilisateur devra choisir dans un menu reprenant ces différentes propositions.  
NB :

- les noms écrits en gras sont des noms de variables à respecter ;  
- réutiliser la librairie réalisée dans le chapitre « erreur ».

# Calculer et afficher exp(x)

*In* : **x** // **nbDecimales** : le nombre de décimales exactes sur la solution.  
*Out* : **resultat** = exp(x) avec nbDecimales exactes.

Moyen : partir du polynôme de Mac Laurin.

# Calculer et afficher ln(x+1)

*In* : **x** // **nbDecimales** : le nombre de décimales exactes sur la solution.  
*Out* : **resultat** = ln(x+1) avec nbDecimales exactes.

Moyen : partir du polynôme de Mac Laurin, prendre X0 =0

# Calculer et afficher sin(x)

*In* : **x** , x exprimé en degrés // **nbDecimales** : le nombre de décimales exactes sur la solution.  
*Out* : **resultat** = sin(x) avec nbDecimales exactes.

Moyen : partir du polynôme de Mac Laurin, prendre X0=0

# Afficher sachant que

*In* : **x**.  
*Out* : avec 5 décimales exactes.

Pour ce faire, vous devez utiliser la librairie « simpson.h » se trouvant sur le commonProfsEtudiants/Charlier.c/biblioInteg.

Voici le contenu de « simpson.h » :

typedef double(\*Fonction)(double);

double calculSimpson(int , double, double, Fonction);

paramètres d’entrée : le nombre de points (nbPoints), la borne inférieure (borneInf), la borne supérieure (borneSup) ainsi qu’un pointeur vers la fonction dont on veut calculer l’intégrale ;

paramètre de sortie : la valeur de l’intégrale calculée par la méthode de Simpson

# Calculer et afficher une racine d’un polynôme P(x) de degré max 10

*In* : **X0**, **nbDecimales**, les coefficients des différentes puissances de x avec degré de .  
*Out* : **resultat**= racine du polynôme avec nbDecimales exactes

Moyen : Newton Raphson.

Les rôles.

Le client : c’est moi. J’ai donné ci-dessus mes spécifications. Si un doute subsiste (le client n’est pas toujours précis sur ce qu’il veut), il faut que le chef de projet contacte le client.

Vous réalisez ce projet par groupes de 3. Chaque groupe aura un chef de projet et deux programmeurs.

Le chef de projet :

* ne peut pas programmer ;
* réalise l’analyse du projet afin qu’il soit pensé au mieux pour le clean code (lisibilité – fonctionnalité supplémentaire – multifichiers etc… ) ;
* réalise le pp général et laisse le détail des modules prévus (avec arguments I/O) aux programmeurs ;
* donne les directives et le timing aux programmeurs (selon la méthode AGILE – veiller à ce qu’ils aient du travail en permanence) ;
* supervise l’avancement, le clean code, le DA ;
* prévoit les tests unitaires à effectuer au fur et à mesure de l’avancement du projet ;
* gère l’avancement du rapport final
* gère la communication.

Les programmeurs :

* réalisent le DA qui leur est demandé ;
* codent en clean code ;
* respectent les consignes du chef de projet ;
* utilisent le deboggeur de visual studio ;

Le rapport

Votre dossier paginé doit comprendre :

* une page de garde,
* une table des matières paginée,
* l’énoncé reçu (pas les calendriers),
* par fonction :
  + l’analyse détaillée (\*),
  + le diagramme d’actions,
  + le code(\*\*\*),
  + le testing(\*\*).

Le programme (codes : source, compilé et exécutable) – sous le nom TNDD\_IG2\_Groupe\_NomPrenom (un seul nom du groupe suffit) - sera placé sur le portail avant le blocus de Pâques.

Rappel : vous avez signé une charte. Dès lors, n’oubliez pas de la relire ainsi que le fichier donnant les consignes pour la rédaction d’un rapport.

(\*)Analyse :

Pour ce faire, réaliser d’abord un travail d’analyse qui consiste à répondre aux questions ci-dessous pour chacune des fonctions à implémenter grâce à Mac Laurin :

1. Déterminer le domaine de définition de la fonction.
2. Etablir la formule du polynôme de Mac Laurin de la fonction. Détailler votre réponse et, si possible, établir une formule de récurrence.
3. Etablir la formule du reste, la majorer pour estimer l’erreur commise. Détailler votre réponse.
4. A partir des propriétés de la fonction considérée, établir le calcul permettant d’évaluer celle-ci en n’importe quel point du domaine de définition. Détailler votre réponse et donner des exemples.
5. Prendre en compte la propagation des erreurs due à l’utilisation des propriétés.
6. Etablir le diagramme d’actions qui permettra d’implémenter la fonction.

Prévoir ce type d’analyse pour les autres fonctionnalités.

Code.(\*\*\*)

Le programme doit présenter un menu offrant le choix des fonctions ou quitter.

*Spécifications :*

* affichage du résultat convivial avec uniquement le nombre de décimales demandé,
* une attention particulière sera apportée à la qualité de votre code source, notamment :
  + variables significatives,
  + pas de variables globales,
  + commentaires,
  + découpe adéquate du programme en multi-fichiers,
  + emploi de fonctions quand les mêmes tâches sont attendues,
  + lisibilité, clarté et diagramme d’actions simple et bien pensé.

(\*\*)Testing :

Pour chaque fonction (y compris le programme principal), prévoir un jeu de tests complet disposé sous forme de tableau de la façon suivante :

Titre : Fonctionnalité testée

Libellé Argument nombre de résultat résultat conclusion

Du test Encodé décimales attendu obtenu (ok ou non)

L’objectif n’est pas de corriger les erreurs éventuelles mais de s’exercer à penser à tous les cas.

Objectifs pédagogiques & méthodologie :

La bonne réalisation d’un projet informatique ne se limite pas à produire du code par essais/erreurs.

Un projet bien mené est un projet dans lequel

* une analyse fouillée est réalisée dès le départ (1/2 à 2/3 du temps);
* des diagrammes d’actions précis sont mis en œuvre dans un travail d’équipe ;
* toute l’équipe suit les consignes ;
* la communication entre les membres de l’équipe est efficace ;
* la communication avec le client aussi ;
* le deboggage est effectué de manière précise et efficace ;
* chaque temps de travail a objectif précis dont il ne faut pas se détourner ;
* un timing précis pour mener à bien l’ensemble du projet est prévu (pas trop-ni trop peu précis) ;
* le code est facilement lisible par tous ;
* AGILE.

Objectif mathématique : comprendre et savoir expliquer les notions mathématiques correspondantes.

Organisation côté client :

A chaque cours, je reçois les équipes pour faire le point sur l’avancement du travail.

Evaluation :

L’évaluation sera basée sur les entretiens réguliers avec l’équipe (c’est l’évolution qui compte), le rapport rendu ainsi qu’une évaluation orale qui « remontera ou descendra » la note attribuée. Si l’étudiant atteint au moins 14/20 à cette évaluation, il pourra être dispensé de cette partie de cours à l’examen mais devra faire au moins 8/20 à l’examen pour l’autre partie.

Préparation de l’oral :

**2.Généralités**

Dans beaucoup des fonctions implémentées, le **polynôme de Mac-Laurin** sera utilisé. Il s’agit de l’application d’un théorème disant que toute fonction peut être décomposée en une somme de termes obtenus de la manière suivante :

**f(x)=f(0) +f′(0)x+f′′(0)x2/2!+f′′′(0)x3/3!+...+fn(0)xn/n!**

Cette formule est théoriquement infinie. Bien évidemment, dans les programmes, nous devrons arrêter l’itération après un certain temps afin d’éviter une boucle infinie. Cette condition d’arrêt correspondra au moment à partir duquel l’ajout de termes ne modifiera plus le résultat de manière significative, c’est-à-dire le moment à partir duquel la nbDecimales (argument passé aux fonctions) décimale sera correcte.

La valeur du terme à partir duquel on arrête d'additionner les termes dépend donc de la précision que l'on veut obtenir, ce qui est la raison pour laquelle le nombre de décimales souhaité est demandé en argument.

**3.Fonction exp(x)**

**3.1 :Analyse**

**3.1.1Explications en français**

L'exponentielle prend deux arguments :

- "x", qui sera exposant de e;

- "nbDecimales", qui correspond au nombre de decimales que l'on veut obtenir;

Elle renvoie "résultat", qui correspond à l'approximation de e^x arrondi à nbDecimales près.

Pour calculer ce résultat, on utilisera la méthode de taylor/mac-laurin.

**3.1.2 Détails mathématiques**

Le **domaine de définition**  de cette fonction est mathématiquement R. Cependant, il est évident que si x est trop extrême, il ne sera plus possible de représenter la valeur de exp(x) sur un nombre de bits fini.

Le **polynôme de Mac-Laurin** représentant cette fonction est de forme

**exp(x) =1+ x+(x²/2 !)+(x^3/3 !)+(x^4/4 !)/+….+(x^n/n !)**

Cela veut dire que pour trouver le terme suivant à additionner, il suffit de multiplier le terme précédent par x/n, n étant le degré (on commence à 0).

**L’image de la fonction** appartiendra toujours à R+. En effet, plus l’exposant sera négatif, plus la valeur tendra vers 0.

(exemple : exp(-10) = 0.00004539993).

Plus la valeur de x augmentera, plus la valeur de l’image augmentera aussi. En 0, L’image vaudra 1 (en effet, un nombre exposé en 0 donnera toujours 1). Pour une valeur de x positive, on aura une image également positive, dont la valeur augmente exponentiellement avec la valeur de x.

**4. Fonction ln(x+1)**

**4.1 Analyse**

**4.1.1 Explications en français**

La fonction de logarithme népérien de la calculatrice permet de calculer ln(x+1) et prend deux arguments:

-"x", à remplacer dans l'expression ln(x+1) par sa valeur;

-"nbDecimales", qui correspond au nombre de décimales que l'on veut obtenir;

Le retour de la fonction est "résultat", qui correspond à une approximation obtenue pour ln(x+1).

Pour calculer le résultat, on utilisera la méthode de Taylor et Mac-Laurin. La condition d’arrêt sera la même que celle utilisée pour exp(x), à savoir le moment à partir duquel l’ajout de termes n’influence plus de manière significative le résultat.

**4.1.2 Détails mathématiques**

**Le domaine de définition** de cette fonction est égal à R, sauf les valeurs inférieures ou égales à -1. En effet, à partir de cette valeur, on demande le logarithme népérien de 0 ou d’un négatif, ce qui n’est pas possible. Cette condition devra être vérifiée avant de lancer le calcul.

Le **polynôme de Mac-Laurin** représentant cette méthode est de forme

**Ln(1+x) = x – x^2/2 + x^3/3  – x^4/4 + x^5/5 -…+x^n/n !**

On peut observer une récurrence pour un terme T:

**T(n+1) = (T(n)\*(-x))/(degré+1)**

**L’image de la fonction**  tend vers – l’infini quand x se rapproche de -1. C’est logique, étant donné qu’en -1, on demande le logarithme népérien de 0, qui n’est pas défini. Du côté des x positifs, les images évoluent de moins en moins pour finir par former ce qui ressemble à une ligne droite.

**5. La fonction sin x**

**5.1 Analyse**

**5.1.1 Explications en français**

La fonction de logarithme népérien de la calculatrice permet de calculer sin(x) et prend deux arguments:

-"x", à remplacer dans l'expression sin(x) par sa valeur;

-"nbDecimales", qui correspond au nombre de décimales que l'on veut obtenir;

Le retour de la fonction est "résultat", qui correspond à une approximation obtenue pour sin(x).

La méthode employée est celle de Taylor et Mac-Laurin en prenant un x0 = 0 comme point de départ.

**5.1.2 Détails mathématiques**

**Le domaine de définition** de la fonction sin(x) est R.

**Le polynôme de Mac-Laurin** de la fonction est :

**Sin(x) = x – x3/3 ! + x5/5 ! – x7/7 ! + x9/9 ! -…**

Ce polynôme s’explique par le fait que la dérivée de sin(x) est cos(x) et que la dérivée de cos(x) est –sin(x). Cela explique l’inversion régulière de signe. Seuls les termes de degré impair sont conservés car sin(0) = 0. Les termes de degré pair étant multipliés par 0, on peut les retirer du polynôme.

Dans ce polynôme, on peut donc établir par récurrence que pour chaque terme T,

**T(n+1) = T(n)\*(-x2/((n+1)\*(n+2)))**

**L’image de la fonction** oscille entre -1 et 1, décrivant ainsi une sinusoïde.

# **6.Calculer et afficher une racine d’un polynôme P(x) de degré max 10**

* 1. **Analyse**
     1. **Explications en français**
     2. **Détails mathématiques**