FACULTÉ DES SCIENCES ET TECHNOLOGIES | Université de Lille

Parcours PEIP
semestre 4

Programmation Structurée

Travaux Pratiques

TP 1 – Affichage et saisie au clavier

TP 2/3 – Structures conditionnelles et structures de boucles

TP 4/5 – Applications des structures de boucles

TP 6/7 – Les Tableaux et applications

Pour tout problème ou question : xavier.melique@ univ-lille.fr

TP 1 : Affichage et saisie au clavier

<u>PREPARATION :</u> Vous établirez les arbres programmatiques des exercices de la séance.

Connaissance : `printf` et `scanf`.

<u>Exercice 1 :</u> Ecrire un programme qui demande un entier A, un réel B et un caractère Car à l'utilisateur, puis qui les affiche à l'écran

<u>Exercice 2 :</u> Ecrire un programme qui saisit le nombre de pièces et calcule le contenu d'une tirelire contenant des pièces de 2€, 1€, 50 cents et 10 cents. Vous afficherez le résultat avec 2 chiffres après la virgule.

Connaissance : printf, scanf., modulo

<u>Exercice 3 :</u> Ecrire un programme qui convertit le nombre de secondes que vous lui donnerez en heures, minutes, secondes. Pour cet exercice, le nombre de secondes doit être inférieur à 65000.

Exemple :

```
Entrez le nombre de secondes (< 65000) : 10000 ↵
10000 secondes correspondent à 2 h , 46 mn et 40 s
```

<u>Exercice 4 :</u> Ecrire un programme qui demande à l'utilisateur le rayon puis calcule la surface et le volume de la sphère ($S=4.\pi.r^2$ et $V=4.\pi.r^3/3$)

TP 2/3 : Structures conditionnelles et structures de boucles

PREPARATION : Vous établirez l' arbres programmatique de l' exercice 1.

Exercice 1 :

1) A partir de la saisie du prix unitaire d'un produit (Pu) et de la quantité commandée (QtCom), calculer le prix total (PTot) du produit. Afficher le prix à payer (Pap), en détaillant le port (Port) et la remise (Rem) , sachant que :

   - Le port est gratuit si le prix des produits (PTot) est supérieur à 70€. Dans le cas contraire, le port est de 2% du PTot.
   - La remise est de 5% si PTot est compris entre 30€ et 150€ et de 10% au-delà.

2) Testez votre programme avec des données de départ illustrant tous les cas de figure de possibles.

Exercice 2 : Affichage Ordonné :

Etablir un programme qui :
- demande à l'utilisateur de rentrer 3 entiers
- qui affiche les entiers dans l'ordre croissant.

1) A partir de l'arbre défini en travaux dirigé, effectuez le codage du programme en langage C
2) Testez votre programme en considérant tous les cas de figure de possibles.

Exercice 3 : Suite de Fibonacci

Etablir un programme qui :
- *Demande à l'utilisateur un nombre entier N*
- *Calcule et affiche les N premiers termes de la suite de Fibonnaci $u_n = u_{n-1} + u_{n-2}$ avec $u_0 = 0$ et $u_1 = 1$.*

1) A partir de l'arbre défini en travaux dirigé, effectuez le codage du programme en langage C
2) Testez votre programme sur les 10 premières valeurs de la suite.

Exercice 4 : Racine d'un entier

Etablir un programme qui :
- *Demande à l'utilisateur un nombre entier N.*
- *Donne la partie entière de $\sqrt{N}$ sans utiliser la fonction sqrt.*

1) A partir de l'arbre défini en travaux dirigé, effectuez le codage du programme en langage C
2) Testez votre programme sur les plusieurs valeurs.

Exercice 5 : Calcul de moyenne de 5 valeurs

Etablir un programme qui :
- *Demande à l'utilisateur de rentrer jusqu'à 5 notes*
- *Affiche la moyenne sauf si une note vaut zéro. Le programme s'arrête alors et affiche « note éliminatoire. »*

1) A partir de l'arbre défini en travaux dirigé, effectuez le codage du programme en langage C
2) Testez votre programme sur les plusieurs valeurs.
3) Etendre le programme à N valeurs.

Exercices complémentaires :

<u>PREPARATION :</u> Vous établirez les arbres programmatiques de chaque exercice.

<u>Exercice 6 :</u>

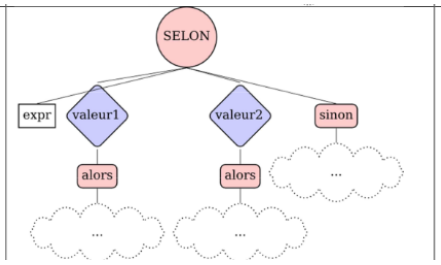
1 - Ecrire un programme se comportant comme une calculatrice (+,-,/,*), c'est-à-dire qui réalise

1. La saisie d'un entier, un opérateur et un entier (ex : 2 + 2),
2. Le calcul
3. L'affichage du résultat reprenant l'expression saisie (Ex : 2 + 2 =4)

Vous ferez attention à la division par 0.

Vous pourrez pour concevoir l'arbre et le programme utiliser la structure switch case (Selon le cas) suivante :

Selon expr
cas valeur1 : ...
cas valeurN : ...
cas sinon : ...
finSelon




```
switch (expr_numérique)
{
  case valeur1 : … ; [break ;]
  case valeurN : … ; break ;
  default : …; break;
}
```

2 - Ecrire un programme qui propose indéfiniment un calcul. Le programme devra demander à l'utilisateur s'il souhaite à nouveau un calcul. L'utilisateur saisira 'O' pour réaliser un nouveau calcul, 'N' pour arrêter.

<u>Exercice 7 :</u>

Afficher les N premiers nombres premiers, N étant saisi par l'utilisateur

Rappel : Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs. Ces deux diviseurs sont 1 et le nombre considéré.

<u>Exercice 3 :</u>

1Ecrire un programme qui propose à l'utilisateur de dessiner à l'écran certaines figures composées d'étoiles comme un sapin, un carré ou un sablier.

Les figures seront choisies grâce à un menu (sapin ou 'T', carré ou 'C', sablier ou 'S') et une lettre saisie. La hauteur de la figure sera saisie par l'utilisateur.

Exemple : hauteur=4

```
   *
  ***
 *****
*******
```

```
****
****
****
****
```

```
*******
 *****
  ***
   *
   *
  ***
 *****
*******
```

TP 4/5 : Applications des structures de boucles

Exercice 1 :

Préparation : Etablir les arbres programmatiques des programmes proposés.

1 - Ecrire un programme qui permet la saisie d'un nombre puis écrit la table de multiplication correspondante à l'écran. Le nombre saisi doit être compris entre 1 et 9 sinon un message d'erreur est affiché et la saisie recommence.

2 - Ecrire un programme qui effectue la saisie et la vérification d'un numéro de table. Les réponses seront saisies et vérifiées. Si le résultat est correct passage à la ligne suivante sinon un message d'erreur est affiché. La réponse est saisie une nouvelle fois. En fin de table le nombre d'erreurs est affiché.

Exemple: (en gras les réponses de l'utilisateur)

```
Table à réviser: 12↵
Erreur !  donnez un nombre entre 1 et 9
Table à réviser: 8↵
OK Révision de la table par 8
1X8 =  8↵
2X8 =  18↵
Erreur !
2X8 = 16↵
/.../
10 X 8 = 80↵
vous avez fait 2 erreur(s)
```

Exercice 2 : Intégration par la méthode de Simpson.

1 – Etablir le code à partir de l'arbre programmatique établi en travaux dirigés, exp(x) sera déclarée comme une fonction f(x). Ne pas oublier d'inclure la bibliothèque math.h.

2 – Tester et valider votre programme sur l'exemple du TD : $I = \int_0^4 e^x \, dx$ avec 8 intervalles.

3 – Tester ce même exemple mais avec 10,100, 1000 et 10000 intervalles. Comparez ces résultats à la valeur exacte de l'intégrale, indiquez dans chaque cas la précision obtenue. Regroupez les résultats dans un tableau. Conclusion.

4 – Calculer $I = \int_1^4 Ln(x^2) \, dx$ avec une précision de 10^{-8} près. Combien d'intervalles faut-il prendre ? Expliquez votre démarche. (La fonction logarithme népérien s'écrit log(x) en langage C.)

Exercice 3 : Recherche de racines par la méthode de Dichotomie.

1 – Etablir le code à partir de l'arbre programmatique établi en travaux dirigés.

2 – Tester et valider votre programme sur l'exemple du TD : résoudre l'équation $e^x-2 = 0$ dans l'intervalle [0,2] avec une précision de 0.1. Donner la valeur de la racine et le nombre d'itérations nécessaires au calcul. Comparez à la solution exacte. Conclusion.

3 – Résoudre par cette méthode l'équation $x^5-2x-5 = 0$ dans l'intervalle [1,3] pour atteindre les précisions successives suivantes : 10^{-6}, 10^{-8}, 10^{-10}. Donner dans chaque cas la valeur de la racine et le nombre d'itérations correspondantes.

Exercices complémentaires :

PREPARATION : Vous établirez les arbres programmatiques de chaque exercice.

Exercice 4 : Extension du programme Simpson

On désire améliorer le programme sur la méthode de Simpson afin que l'utilisateur n'ait pas à rentrer le nombre d'intervalles de calcul mais seulement la précision qu'il souhaite sur le calcul de l'intégrale.

Le programme renvoie alors le résultat du calcul ainsi que le nombre d'intervalles nécessaires.

Vous testerez votre programme sur le calcul de $I = \int_0^4 e^x \, dx$ avec une précision de 1^{E-6} et 1^{E-8} vous donnerez le nombre d'intervalles nécessaires dans chaque cas et vous comparerez vos résultats à la valeur exacte de l'intégrale.

Vous ferez le calcul de $I = \int_1^4 Ln(x^2) \, dx$ pour les précisions 1^{E-4} , 1^{E-8} et 1^{E-12}, donner dans chaque cas le nombre d'intervalles N nécessaires au calcul.

Conseils :

- Passez le programme Simpson de l'exercice 2 en une fonction à 3 paramètres d'entrée : a, b et N.
- Pour la variation du nombre d'intervalles N, doubler son nombre à chaque itération.

Exercice 5 : Recherche de racines par la méthode de Newton.

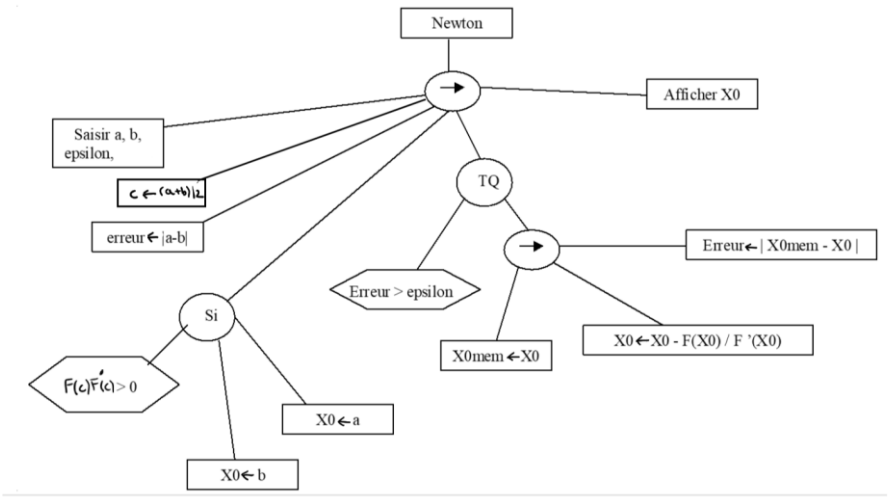
On désire dans cette exercice utiliser la méthode de Newton (méthode de la tangente) pour rechercher la racine de l'équation F(x) = x^5-2x-5 = 0 dans l'intervalle [1,3].

Pour cela on donne l'arbre programmatique ci-dessous où :

- F'(x) est la dérivée de F(x)
- Epsilon représente la précision du calcul

Coder et exécuter le programme pour atteindre les précisions successives suivantes : 10^{-6}, 10^{-8}, 10^{-10}. Donner dans chaque cas la valeur de la racine et le nombre d'itérations correspondantes (on intégrera au programme un compteur d'itérations).

Comparez les résultats à la méthode de Dichotomie. Conclusion.



TP 6/7 : Les Tableaux et applications

Exercice 1 : Tri à bulle

1) En reprenant l'arbre programmatique défini en cours, effectuer le codage de l'algorithme
2) Tester le programme sur plusieurs exemples de tableau (non trié, trié ,….)

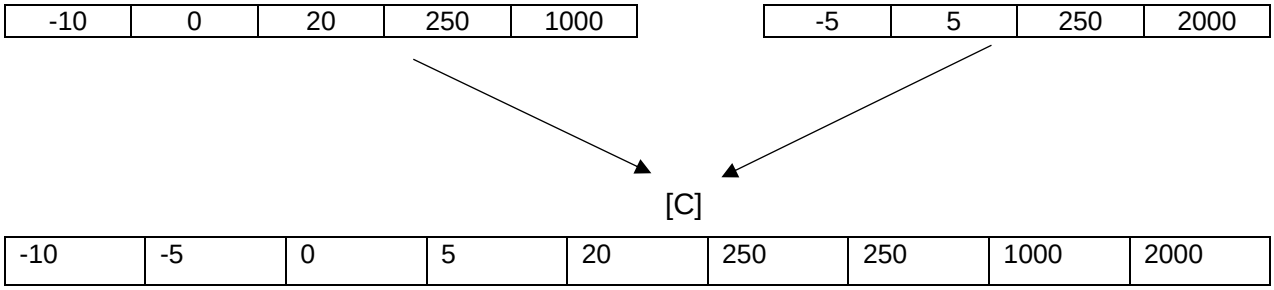
Exercice 2 : Tri par fusion

Le but de cet exercice est d'effectuer la fusion de 2 tableaux ordonnés en un seul tableau ordonné. Pour plus de facilité, on initialisera les 2 tableaux directement dans le programme sans demander à l'utilisateur de rentrer ces valeurs. Après avoir afficher ces tableaux à l'écran, le programme effectuera la fusion et affichera le nouveau tableau trié.

Principe du tri par fusion :

Soient 2 tableaux à une dimension [A] et [B] contenant des nombres entiers déjà ordonnés par ordre croissant. L'idée est de fusionner [A] et [B] dans un tableau [C] dont les éléments seront triés par ordre croissant lors de la fusion. (on n' aura pas recours au tri à bulle pour effectuer le tri !)

Exemple : [A] [B]



Préparation : Etablir l'arbre programmatique de la méthode

1) Effectuer le codage de l'algorithme
2) Tester le programme sur plusieurs exemples
    a. [A] de dimension 10 et [B] de dimension 15
    b. [B] de dimension 10 et [A] de dimension 15
    c. [A] et [B] de même dimension
    d. [A] et [B] possèdent certains éléments de mêmes valeurs (doublon dans [C])

Exercice 3 : Suppression des doublons

A partir du programme de tri par fusion, élaborer un programme qui demande à l'utilisateur s'il veut retirer les doublons du tableau [C]. Dans ce cas, le programme créera un nouveau tableau [D] qui recopiera les valeurs de [C] en supprimant les doublons.

On affichera le tableau ainsi créé ainsi que le nombre de doublons effacés.

Préparation : Etablir l'arbre programmatique complémentaire

1) Effectuer le codage
2) Tester l'exemple sur un cas où [C] contient des doublons et un autre où [C] ne contient pas de doublons.