SDA1 Travaux Pratiques n°2 Année scolaire 2020-2021

Remarque : Ce TP est noté. Vous devez montrer votre travail à l'enseignant au fur et à mesure et finir le TP en dehors de la séance, avant la séance suivante et déposer votre compte-rendu dans la zone de dépôt ouverte par votre enseignant.

Objectifs

- Manipulation des tableaux, monodimensionnel et multidimensionnel
- Connaissance des méthodes de tris simples

Compétences attendues

- Comprendre les étapes de développement d'une application informatique
- Etre capable de coder en langage C une analyse en pseudocode
- Débogage d'une application
- Evaluation de l'efficacité d'une méthode de tri

Etapes de développement d'une application en programmation procédurale

L'approche classique de développement repose sur un cycle en V. Elle passe par 3 phases : analyse, codage et tests.

Phase 1 : L'analyse

L'analyse consiste dans un premier temps à comprendre le besoin afin d'envisager une solution pouvant y répondre. La compréhension nécessite souvent de faire des schémas qui permettent de mieux comprendre le problème posé. La solution se traduit ensuite dans le choix de structures de données et la définition d'un algorithme par exemple sous forme de pseudocode.

Dès que l'analyse a été faite, il faut immédiatement définir les tests à effectués pour vérifier le code qui sera obtenu. C'est la compréhension de la solution qui permet d'envisager les différents cas à tester : le cas général, mais également des cas limites. Dans tous les cas les tests devront répondre à trois objectifs : valider le besoin (ou le cahier de charges), vérifier la robustesse de programme (pas de plantage intempestif), vérifier l'ergonomie du programme.

Phase 2 : Codage

On peut ensuite passer à la phase de codage. C'est dans cette phase que l'on choisit la plateforme de développement et le langage de programmation.

- La plateforme de développement est basée sur la distribution **linux Debian**. Ce choix est justifié par des raisons pédagogiques : vous habituez à utiliser un système libre qui favorise l'insertion de l'outil informatique dans les PME. Il est également à la base des applications embarquées que l'on retrouve dans de nombreux systèmes multimédias grand-public mais également de plus en plus dans des applications industrielles.
- Le langage de programmation et le **langage C** pour son utilisation dans tous les secteurs de l'entreprise, notamment pour le développement des applications industrielles.

La phase de codage est d'autant plus rapide que l'analyse a bien été réalisée. Le code doit être écrit de manière à faciliter la maintenance de l'application. La relecture par une personne tierce doit être facilitée. Cet objectif est atteint par :

- ***un nommage judicieux des variables correspondant à leur rôle***,
- ***l'indentation du programme***,
- ***l'insertion de commentaires dans le code***.

La phase de codage comprend 3 étapes :

Etape 1 : Vous éditerez votre fichier source à l'aide d'un éditeur pleine page de type « kate » ou « gedit ».
armand@toguyeni#gedit <source>.c &

Etape 2 : Compilation et édition de liens du fichier source en code exécutable avec gcc
armand@toguyeni#gcc <source>.c –o <executable.exe>

La compilation sert à transformer le code source en code objet. Elle permet de vérifier la syntaxe. L'édition de lien sert à inclure le code des fonctions des bibliothèques et à transformer le code objet en code exécutable. Avec gcc, **l'option « -o » permet de faire en une seule étape la compilation suivi de l'édition de lien**.

Etape 3 : Lancer l'exécution de votre programme
armand@toguyeni#./<executable.exe>

Phase 3 : Les tests
Une fois le code réalisé, il faut effectuer les tests définis dans la phase d'analyse. Ces tests doivent donner des jeux d'essais qui doivent être commentés pour indiquer en quoi il vérifie la conformité du code ou besoins ou ils prouvent sa robustesse.

Comment faire des jeux d'essais
Tous les cas doivent être testés. Les tests doivent être commentés si possibles en renvoyant à des parties de code.

Capture du test :
***armand@toguyeni* #script <fichier_resultats>**
Script démarré

=> que tout ce qui sort à l'écran est redirigé vers ce fichier. A la fin arrêter le script.

***armand@toguyeni* #exit**

Partie I – Transcription d'une analyse en pseudo-code en code C

Remarque : Vous devez envisager à chaque fois tous les cas de tests prouvant la robustesse et le respect du cahier de charge.

Exercice 1 : Saisie et affichage d'un tableau monodimensionnel

Ecrire un programme qui permet de demander à l'utilisateur le nombre de composantes d'un vecteur d'entiers, et de saisir ce nombre. Ce nombre devra être positif. Tant que l'utilisateur n'entre pas un nombre positif, redemander la saisie à l'utilisateur. Arrêtez le programme au bout de trois saisies erronées. Si la saisie s'est bien effectuée, saisir les composantes du vecteur et afficher l'ensemble du vecteur en colonne avec pour chaque composante un affichage du type **V[7]=31**. Les vecteurs auront au plus 20 composantes.

Exercice 2 : Saisie et affichage d'un tableau multidimensionnel

Ecrire un programme qui permet de demander à l'utilisateur le nombre de lignes et de colonnes d'un tableau à 2 dimensions (matrice). Ces nombres devront être positifs. Tant que l'utilisateur n'entre pas un nombre positif, redemander la saisie à l'utilisateur. Arrêtez le programme au bout de trois saisies erronées. Si la saisie des deux dimensions a été bien effectuée, saisir les composantes du tableau et l'afficher ligne par ligne. Le nombre de lignes et de colonnes devra être inférieur ou égale à 20.

Partie II – Implémenter les méthodes de tris dans le programme fourni en tant que squelette.

Exercice 3 : Tri sélection

1. Développer un programme qui permet de trier un tableau monodimensionnel par la méthode du tri sélection. Cette méthode pourra être testée à l'aide du programme de l'exercice 1 qui sera adapté.
2. Faire un jeu d'essai pour prouver le bon fonctionnement de votre programme.
3. Adapter votre programme à l'aide du code donné par l'enseignant. Ce code vous permet de générer de manière aléatoire des tableaux de grandes tailles et calculer la durée de tri. Faire des jeux d'essais pour illustrer le bon fonctionnement de votre programme et évaluer la complexité de la méthode. On tracera la courbe durée de la méthode de tri en fonction nombre de composants à trier.
4. Chercher sur internet la complexité de la méthode utilisée. Est-ce que vos tests vous donnent la même complexité ?

Exercice 4 : Tri à bulles

1. Développer un programme qui permet de trier un tableau monodimensionnel par la méthode du tri à bulle. Cette méthode pourra être testée à l'aide du programme de l'exercice 1 qui sera adapté.
2. Faire un jeu d'essais pour illustrer le bon fonctionnement de votre programme.
3. Peut-on améliorer la méthode précédente afin de réduire la durée de tri sur des tableaux partiellement triés ?
4. Adapter votre programme à l'aide du code donné par l'enseignant. Ce code vous permet de générer de manière aléatoire des tableaux de grandes tailles et calculer la durée de tri. Faire des jeux d'essais pour illustrer le bon fonctionnement de votre programme et évaluer la complexité de la méthode. On tracera la courbe durée de la

méthode de tri en fonction nombre de composants à trier.

5. Chercher sur internet la complexité de la méthode utilisée. Est-ce que vos tests vous donnent la même complexité ?

Exercice 5 : Tri insertion

1. Développer un programme qui permet de trier un tableau monodimensionnel par la méthode du tri par insertion. Cette méthode pourra être testée à l'aide du programme de l'exercice 1 qui sera adapté.
2. Faire un jeu d'essais pour prouver le bon fonctionnement de votre programme.
3. Adapter votre programme à l'aide du code donné par l'enseignant. Ce code vous permet de générer de manière aléatoire des tableaux de grandes tailles et calculer la durée de tri. Faire des jeux d'essais pour illustrer le bon fonctionnement de votre programme et évaluer la complexité de la méthode. On tracera la courbe durée de la méthode de tri en fonction nombre de composants à trier.
4. Chercher sur internet la complexité de la méthode utilisée. Est-ce que vos tests vous donnent la même complexité ?