Université Mohamed Boudiaf, M'sila
Faculté de technologie
Socle commun

Module : TP-informatique 3
2ième année ST

TP1 - Rappels : Introduction à la conception d'un programme

1. Introduction

La programmation peut être faite par intuition ou par préparation d'un plan de travail. La meilleure méthode pour aborder un projet de programmation est d'appliquer l'intuition sous le contrôle d'une structure bien conçue.
Le premier principe de bonne programmation est la persistance. Si on est intéressé par ce qu'on fait, il ne serait pas difficile de se créer une passion pour rester avec la tâche quelle que soit sa complexité.
Le second principe de bonne programmation est l'utilisation de structure, les programmes sont construits du général au spécifique.
La conception est la procédure de créer quelque chose à l'aide d'un plan de travail. Durant la conception de la chose, on doit avoir une idée de comment le produit final va être. Le troisième principe est la technique : "diviser et conquérir". Selon le philosophe Anglais du quatorzième siècle, William d'Ockham, n'importe quel problème peut être divisé en parties de telle manière que la résolution de toutes les parties séparément, résoudra tout le problème. La conception du langage Matlab est principalement guidée par cette idée. Par exemple en langage C (notre cas le Matlab), on est encouragé à diviser le programme en une collection de fonctions plus ou moins isolées qui sont conçues et testées séparément. Cette caractéristique est la propriété fondamentale de la conception connue sous le nom de l'approche "de haut en bas ou descendante" de programmation.
L'approche descendante commence avec la définition du problème. Il est impossible de connaître comment commencer la conception d'une application sans connaître les objectifs qu'on souhaite accomplir. La définition du problème contient elle même la solution du problème. On peut voir que ceci est vrai dans notre vie quotidienne. Pour résoudre un problème, on a besoin seulement de se concentrer sur les petits détails du problème, pour obtenir une compréhension exacte de tout ce qui concerne ce problème. Si on garde les objectifs visés clairs dans notre esprit à travers toute la procédure de conception, nos applications seront propres et convenables. En réalité, la conception d'une application est faite du haut en bas.

2. Résolution des problèmes

Depuis le temps où on était à l'école primaire, on était appelé à résoudre des problèmes. Quels uns de ces problèmes, utilisent des calculs tels que ceux rencontrés en un cours de mathématique ou de science, par exemple, quelle est la somme des dix premiers nombres impairs. D'autres problèmes traitent la manipulation du texte, par exemple, quelles sont les principales ressources d'Algérie.
A travers nos années scolaires, on a accumulé des techniques pour résoudre une large variété de problèmes.

Notre objectif serait d'apprendre à traduire ces capacités de résolutions des problèmes, pour qu'on puisse utiliser un ordinateur, pour exploiter les détails mécaniques des solutions de ces problèmes. Ceci est l'essence de la programmation: Avoir une technique de résolution des problèmes constituée d'une série d'étapes que l'ordinateur peut exécuter. En décrivant une technique de résolution d'un problème pour un ordinateur, il est nécessaire de procéder d'une façon très organisée. La description doit inclure toutes les étapes nécessaires pour résoudre ce problème et les étapes doivent être dans le propre ordre. N'importe quelle information intermédiaire qui est exigée dans une étape doit être donnée dans l'étape précédente.

3. Conception d'un programme

L'approche de conception utilisée consiste de trois étapes fondamentales

Étape 1 : analyser le problème.
Étape 2 : faire le profil du programme.
Étape 3 : mettre en oeuvre le programme.
a. coder le programme.
b. tester et déboguer le programme.
c. documenter le programme.

3.1. Analyse du problème

Dans cette étape, on doit:
a. Déterminer les données qui vont être produites (sorties) comme solution du problème. Dans cette sous-étape, on doit définir les variables qui sont exigées pour représenter la sortie.
b. Déterminer les données (entrées) nécessaire pour produire les sorties. Dans cette sous-étape, on doit définir les variables exigées pour représenter les entrées.
c. Développer un algorithme pour obtenir les sorties des entrées en un nombre fini étapes. Un algorithme est défini comme une séquence organisée d'opérations pour résoudre un problème en un nombre fini d'étapes. Dans le cas des problèmes numériques, l'algorithme est constitué d'une série de calculs nécessaire pour obtenir les sorties. Mais, dans le cas des problèmes non numériques, les algorithmes peuvent inclure des opérations variées de manipulations de texte et graphique avec les opérations numériques.
Le but principal de cette étape est d'être sûr que le problème soit bien compris. N'importe quelle erreur commise dans l'analyse peut être transférée au programme.

3.2. Profil du programme

Dans cette étape, on fait le profil du programme utilisant des phrases symboliques. Chaque phrase correspond à une simple opération du programme. Pour un simple programme, il est possible de produire le code symbolique en listant les différentes tâches que le programme doit exécuter, dans l'ordre avec lequel elles seront exécutées.
Mais pour les programmes complexes, il est nécessaire d'organiser la procédure du profil. Pour cela, on utilisera la méthode d'organisation appelée la conception de haut en bas ou descendante.
Pour concevoir un programme utilisant l'approche descendante, on divise le programme en un nombre de tâches, la liste des tâches est un profil du programme appelé module principal. Dans le module principal, la tâche est seulement identifiée par son nom. Il n'y a pas d'indication de comment la tâche est actuellement accomplie. Ce niveau de détail est laissé pour l'étape suivante dans la conception du programme. La division, du programme en tâches, constitue la conception initiale du programme.

3.3. Ecriture du programme

La phase finale de la conception du programme est d'écrire le code source pour le programme. Dans cette étape, le langage symbolique ou pseudo-code pour les modules est traduit en instructions d'un langage de programmation, tels que PASCAL, C, ou Matlab etc...
Comme partie du code source, on doit inclure la documentation sous forme de commentaire qui décrit ce que les différentes sections du programme sont supposées faire. En plus, le code source peut inclure un code de débogage pour tester opération du programme et permettre de trouver les erreurs (bogues) de programmation.
Une fois que le programme fonctionne correctement, on doit éliminer le code de débogage. Mais la documentation devra rester comme une partie permanente du code source pour faciliter la maintenance et la modification du code source à un autre programmeur.

4. Exemple de conception d'un programme

Problème :
Ecrire un programme qui calcule la surface d'un triangle dont les dimensions sont spécifiées par l'utilisateur.

Analyse du problème :
Un triangle est spécifié d'habitude par deux dimensions, la hauteur et la base. Celles-ci sont les quantités fournies par l'utilisateur. Définissons les variables correspondantes :
Hauteur = hauteur du triangle,
Base = base du triangle.
Le problème demande l'écriture d'un programme pour calculer la surface du triangle.

Définissons une variable :
La sortie : Aire = surface du triangle.
Remarquer qu'on a utilisé des noms complets pour les variables (Aire au lieu de A, Hauteur au de H, etc.). En mathématiques, il y a une tendance à utiliser des abréviations. Mais en programmation, il serait préférable d'utiliser des noms descriptifs de variables, aux dépens de formules plus longues et peut être plus complexes. En utilisant des noms descriptifs de variables, le code sera plus documenté.

Résume de l'analyse
Entrées du programme : Hauteur, Base.
Sortie du programme : Aire.
Algorithme : Obtenir Hauteur et Base de l'utilisateur.
Calculer la surface utilisant la formule Aire = 0.5 * Hauteur * Base.
Afficher la valeur d'Aire.
Le programme doit accomplir trois opérations:
Obtenir les dimensions du triangle de l'utilisateur.
Calculer la surface.
Afficher le résultat.
Ceci correspond à un profil du programme constitué de trois tâches:

Programme triangle
Module principal: Triangle
Module 1: Obtenir les dimensions.

Module 2: Calculer la surface.
Module 3: Afficher le résultat.
Le profil ci-dessus donne le pseudo-code pour le module principal, mais il ne contient pas les détails des différentes tâches. Ceux-ci sont décrits par le pseudo-code des sous-modules suivants :
Module 1: Obtenir les dimensions
Afficher 'Entrer les dimensions:'
Afficher 'Hauteur = '
Afficher 'Base = '
Entrer la valeur de Hauteur
Entrer la valeur de Base
Module 2: Calculer la surface
Aire = 0.5 * Hauteur * Base
Module 3: Afficher le résultat
Afficher 'La surface est égale'
Afficher la valeur d'Aire
La conception exige deux niveaux de détail, le niveau module principal et le niveau sous-module.

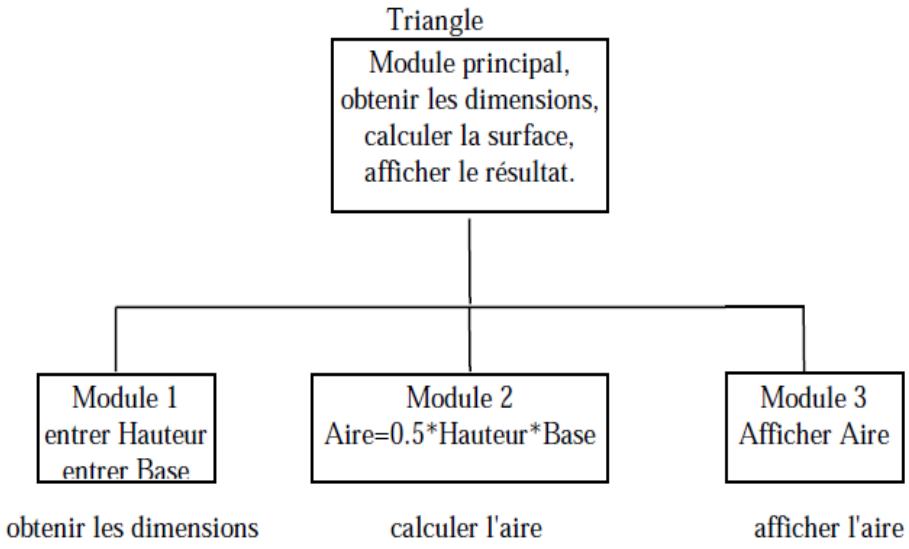

Schéma de la structure du programme triangle