

Projet d'algorithmique CIR2 — 2020

À l'issue de ce projet, vous aurez acquis des compétences :

- **d'architecte logiciel** car vous aurez élaboré une structure de programme précise et solide, et vous l'aurez découpée en sous-ensembles plus simples à gérer ;
- **de développeur logiciel** car vous aurez codé une architecture logicielle élaborée par vous-même ;
- **d'intégrateur logiciel** car vous aurez réassemblé les sous-ensembles logiciels pour en faire une application complète ;
- **de testeur logiciel** car vous aurez réalisé un ensemble de tests sur votre code afin de s'assurer que l'ensemble correspond aux attentes
- **d'intégrateur système** car vous aurez testé la cohérence du programme final en conditions normales de fonctionnement.

Phases rencontrées
Conception du logiciel
Développement du logiciel
Phase de tests
Finalisation du produit

Votre objectif doit être de fournir avec votre groupe, un produit qui fonctionne et conforme aux exigences initiales spécifiées et correctement testé. Gardez toujours cet objectif-là à l'esprit, quoi qu'il vous soit demandé.

Table des matières

I.	Présentation générale de projet	2
II.	Objectifs par rapport au cursus	2
III.	Modalités de constitution des groupes	2
IV.	Modalités de notation	2
V.	Modalités d'organisation du travail	5
VI.	Déroulement du projet	5
VII.	Récapitulatif des documents et livrables à rendre	9
VIII.	Quelques conseils et remarques	10
IX.	Annexe – démarche de spécification	12

I. Présentation générale de projet

Chaque groupe est en situation de concurrence avec les autres groupes. La qualité du produit et son positionnement qualitatif par rapport aux autres produits aura une influence notable sur la note finale. Les aspects documentaires et la rigueur de la démarche projet interviennent aussi dans la note.

Chaque groupe commencera par élaborer l'architecture d'un programme sur son sujet. Il fournira ensuite plus tard ses documents de conception sous-traitance à un autre groupe qui assurera la partie implémentation des modules de persistance (voir annexes) - et recevra de même d'un autre groupe des documents de conception sous-traitance qu'il aura à suivre pour en faire l'implémentation.

Chaque groupe jouera ainsi au début le rôle d'un architecte logiciel, puis d'un sous-traitant spécialisé dans le développement logiciel. Il implémentera ensuite le « moteur de calcul » et les interfaces de saisie et consultation de son projet. Il assemblera le tout, partie sous-traitée et partie développée en propre, et procèdera à la recette globale du projet.

II. Objectifs par rapport au cursus

Il s'agit de mettre en application les compétences acquises dans le cours d'algorithmique. Il vous est demandé de travailler en équipe pour concevoir dans toutes ses phases un projet informatique (mise en situation identique à un projet en entreprise):

- analyse ;
- conception ;
- codage ;
- intégration ;
- validation.

III. Modalités de constitution des groupes

Un groupe est constitué de quatre étudiants (ou exceptionnellement trois ou cinq, décision prise par l'encadrant), dont un (le chef de projet) gèrera la coordination au sein du groupe ;

- les interactions avec les enseignants ;
- la gestion de la documentation ;
- les sauvegardes des différentes versions de fichiers sources ;
- les tests de l'application.

IV. Modalités de notation

Le projet d'algorithmique sera noté sur vingt :

- quatre points seront alloués à l'élaboration de l'architecture du programme ainsi qu'à la documentation (dont le rapport final) accompagnant ce travail d'architecture ;

- six points seront alloués pour la qualité du codage réalisé, tant sur le projet du groupe qu'en sous-traitance, ainsi que sur la qualité de la documentation fournie au sous-traitant afin que celui-ci fasse correctement le travail qui lui est demandé ;
- cinq points seront alloués à l'élaboration de rapports d'avancement décrivant les tâches réalisées par chaque membre du groupe ainsi que les documents de tests, et à tout ce qui pourra montrer le niveau de maîtrise du groupe vis-à-vis de l'avancement de son projet ;
- cinq points seront alloués lors de la recette finale du programme.

Pour chacune de ces parties de notation, une partie de la note viendra du résultat obtenu globalement par le groupe, une autre partie viendra de la part apportée individuellement par chaque étudiant.

Notation	Part du groupe	Part individuelle
Architecture	qualité du document d'architecture	contribution personnelle au résultat global, respect des horaires, attitude en groupe
Recette sous-traitance & donneur d'ordre	qualité du codage / qualité des documents fournis au sous-traitant	
Avancement	Pertinence des choix, qualité du codage et de l'évolution, gestion et évolution des documents/ rigueur de la démarche de test	
Recette finale	qualité du produit fini / richesse informative des documents.	

1. Architecture

Le document d'architecture est noté par l'enseignant. Son contenu est décrit plus loin dans ce document.

2. Sous-traitance

La sous-traitance consiste à faire développer par un autre groupe la partie « persistance » (sauvegarde et relecture de données sur disque) de son projet.

- Chaque groupe « développeur sous-traitant » fournira à l'enseignant une fiche d'appréciation des documents de conception qui lui auront été fournis ; la partie donnée à sous-traiter portera sur la « persistance des données » (voir ci-après) ;
- Chaque groupe « donneur d'ordre » fournira le cahier de recette et l'évaluation du travail de son sous-traitant.

Chaque fois que les étudiants sont amenés à juger le travail d'un autre groupe, il leur est demandé de trouver un équilibre entre objectivité et d'impartialité d'une part, et mansuétude et compréhension d'autre part

3. Avancement

Le rapport d'avancement sera noté par l'enseignant. Il fait un point précis sur ce qui est fait, ce qu'il reste à faire, les problèmes résolus et ceux restant à résoudre. Il fournit une nouvelle version du document d'architecture éventuellement modifié et complété.

4. Pitches et recette finale

Plusieurs pitches intermédiaires auront lieu, le premier (fin de première semaine) portant sur la validation des choix d'architecture et une première vision du projet, les suivants portant sur la présentation totale de ce qui devra être fourni ainsi que l'avancement actuel du projet (difficultés rencontrées...).

Le produit fini sera noté par l'enseignant. La note tiendra compte des objectifs atteints (impératifs et/ou optionnels), de la présentation et démonstration de l'application intégrant les modules sous-traités et les modules développés en interne, de la démarche globale de gestion du projet avec sa partie validation et tests.

En ce qui concerne la notation individuelle, chaque membre de groupe devra contribuer à rédiger une fiche récapitulative de son travail à *chaque séance*. Elle devra être systématiquement consignée par le chef de projet, qui devra pouvoir les fournir sur simple demande d'un enseignant, et les fournira dans le rapport final du projet (voir le chapitre suivant).

5. Autres dispositions

En cas de retard non-justifié d'un étudiant vis-à-vis des horaires de projet, un cumul des retards sera calculé personne par personne et servira à appliquer un malus personnel à l'étudiant concerné.

Il en est de même pour les départs en avance.

Il est donc à la charge de chaque étudiant :

- de justifier un quelconque retard à son encadrant de projet, en plus du secrétariat ;
- de venir se signaler à son encadrant s'il n'était pas présent lors de l'appel de début de séance, pour être sûr d'être pointé présent.

V. Modalités d'organisation du travail

Le projet se déroulera en plusieurs phases, avec certains documents à rendre à des dates précises.

Afin de mener à bien ce projet, il vous est demandé de faire des « cycles courts » de développement. Cette méthode de travail (inspirée des « méthodes agiles », pour ceux qui veulent faire des recherches sur le sujet) permet une bonne réactivité ainsi qu'une meilleure répartition de la charge de travail.

Il vous est donc demandé à chaque séance de faire une courte réunion de 10 minutes, consistant à réunir tous les membres du groupe pour que chacun puisse s'exprimer :

1. faire le bilan de ce que chacun a réalisé depuis la dernière réunion ;
2. faire le bilan des difficultés éventuellement rencontrées par chacun depuis la dernière réunion, de manière à ce que le groupe essaie de débloquer ces situations difficiles ;
3. faire le planning des choses à faire par chacun jusqu'à la prochaine réunion ;
4. évaluer globalement tout ce qui reste à faire pour atteindre le prochain jalon.

La réunion est obligatoire et donne lieu à un compte rendu rédigé par le chef de projet.

La fiche de compte-rendu sera systématiquement archivée par le chef de projet, et devra pouvoir être fournie sur simple demande d'un enseignant. Tout chef de projet, ou son suppléant si celui-ci est absent, devra être en mesure de fournir ces fiches, sous peine de pénalités dans la notation. Elles seront toutes jointes au rapport final.

VI. Déroulement du projet

Le projet se déroule en six phases :

1. présentation des sujets ;
2. analyse et conception générale ;
3. codage en sous-traitance et recette ;
4. point d'avancement ;
5. codage / intégration - intégration sous-traitance ;
6. validation recette.

1. Présentation des sujets :

Lors de la séance de présentation des sujets, vous sont proposés différents sujets de projet. Chaque groupe devra classer les sujets selon sa préférence, et fournir ce classement à son enseignant. Un chef de projet aura dû être désigné pour chaque groupe.

2. Analyse et conception générale : 1ere semaine

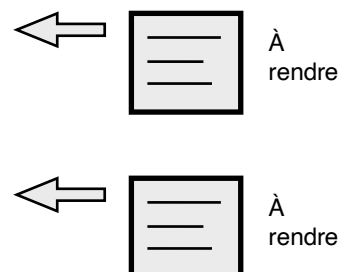
Objet : travail en salle de cours par groupe, qui conduit :

- à une compréhension totale du sujet (lire ce qui est demandé, replacer la demande dans son contexte, aller chercher des exemples similaires, etc.) ;
- à lister de manière formelle et aussi courte que possible (sujet-verbe-complément) chaque *exigence* du programme, qu'elle concerne une caractéristique à très haut niveau d'abstraction ou un fin détail de fonctionnement (voir la notion d'exigence en annexe);
- pour chacune des exigences exprimées, à indiquer dans un second document comment tester que chaque exigence listée a été prise en compte dans votre programme (cahier de recette) ;
- à l'élaboration de scénarios qui décriront comment fonctionne le produit fini ;
- étude générale des modules fonctionnels du logiciel de manière à garantir l'absence de « trou de conception » ;
- définition des *structures de données* (en langage intermédiaire), des *actions sur ces données* (quels calculs/transformations vont leur être appliqués), de la *visibilité entre ces données* (quelle donnée englobe telle autre, quelle donnée est fabriquée à partir de telle autre, etc.) ;
- définition et justification de l'architecture des *modules* : nature, contenu et interfaces avec les autres modules ;
- *arbre des fonctions* définissant l'arborescence des appels entre fonctions, les fonctions pouvant ici appartenir à un même module ou non ;
- description des fonctions principales et des fonctions « compliquées » en langage familier mais formel (à quoi servent ces fonctions, quels sont les prérequis, les résultats ainsi que les méthodes/moyens qu'elles doivent employer).

Pour de plus amples informations sur les notions présentées voir les annexes.

Rapport : cette phase donne lieu à la rédaction :

- d'un « rapport de conception générale » contenant tous les éléments cités ci-dessus (liste des exigences, modules, fonctions et structures) ;
- d'un « cahier de recette » qui donne la liste des tests validant les exigences du projet final dans son ensemble.



Durée : les documents sus-cités devront être finalisés avant la fin de la première semaine, puis alors validés par votre encadrant, avant de passer aux phases suivantes. Ils seront tous passés au groupe qui assurera l'implémentation d'une partie de votre projet. Veuillez

donc à les rédiger de manière à montrer sans ambiguïté à votre sous-traitant ce à quoi vous voulez aboutir.

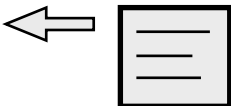
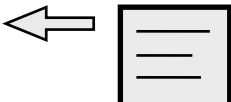
Notez aussi que les groupes sur un même sujet de projet sont en concurrence. Il s'agit donc ici de choisir des caractéristiques qui soient atteignables par vous et par votre sous-traitant. Mais si vous vous cantonnez à des trivialités, votre produit sera certainement peu concurrentiel. Il vous faudra donc vous appuyer sur ce qui vous a été enseigné durant le semestre pour essayer d'élaborer des caractéristiques plus complexes mais plus séduisantes pour le client, tout en faisant en sorte que ces fonctionnalités soient concrètement réalisables. Veillez donc à rédiger vos documents de manière à montrer sans ambiguïté à votre sous-traitant comment fonctionne à haut niveau d'abstraction votre logiciel.

3. Codage en sous-traitance et recette :

Objet (pour le compte du groupe donneur d'ordre) :

- Compréhension du travail de codage de la partie « persistance » sur la base des documents fournis ;
- codage des différentes fonctions composant la partie « persistance » en sous-traitance ;
- test unitaire de chaque fonction et du module « persistance » ;
- description de toutes les interfaces de vos fonctions (paramètres et retours), afin que celles-ci puissent être facilement appelées par les fonctions de votre donneur d'ordre ;
- description du résultat attendu du module de sous-traitance (ex. création d'un fichier .txt) et également indiquer de quelle façon seront enregistrées les données dans le fichier ;
- fournir un programme « main » au groupe « sous-traitant » afin que celui-ci puisse tester et valider le codage de son côté ;
- corrections nécessaires au niveau des différentes fonctions ;

=> cette partie s'achève sur la recette de la partie « persistance » avec le groupe donneur d'ordre.

- Le groupe donneur d'ordre évalue le travail de son sous-traitant sur la base du cahier de recette qu'il a fourni.  À rendre
- Le groupe sous-traitant rendra en même temps un rapport contenant l'évaluation des documents qui lui ont été fournis pour effectuer sa tâche.  À rendre

4. Point d'avancement

Objet (point d'avancement, recadrage, évaluation du "reste à faire") :

- Un point d'avancement est fait incluant éventuellement une reprise partielle de l'analyse et de la conception générale.
- Une évaluation de ce qui reste à faire,
- Une de la répartition des tâches,



=> le document point d'avancement est à rendre dans la journée.

5. Codage complet et intégration : journées projet du 8 au 12 janvier 2018

Objet (codage des modules IHM et moteur de calcul, assemblage des modules) :

- travail de codage de l'IHM et du moteur de calcul à réaliser (sur la base de votre document de conception générale, éventuellement modifié) ;
- codage des différentes fonctions du logiciel ;
- test unitaire de chaque fonction et de chaque module du logiciel ;
- intégration de toutes les parties du logiciel écrites pendant la phase de codage et de sous-traitance ;
- corrections nécessaires au niveau des différents modules (repasser les tests unitaires sur les modules corrigés) ;
- réalisation du programme final opérationnel.

Vous devez pouvoir tester *chaque module, voire chaque fonction de module, de manière indépendante*. Il n'est absolument pas question de tout écrire puis d'essayer de déboguer l'ensemble du projet : vous devez pouvoir effectuer régulièrement chaque test pour valider les évolutions correctives.

Cela deviendra impératif lors des évolutions liées à l'intégration qui ne doivent pas se solder par l'introduction de nouvelles erreurs (il y aurait alors ce que l'on appelle une *régression*). Pour ce faire, votre équipe devra se répartir les tâches, chaque membre codant et testant une ou plusieurs fonctions du projet (tests unitaires).

Puis les fonctions écrites et testées sont mises en commun et intégrées dans des fonctions plus complexes, qui sont alors elles-mêmes testées, et le cycle se perpétue jusqu'à ce que tous les morceaux soient assemblés et donc le logiciel fini (différents niveaux d'intégration).

Durée nominale : la phase se terminera avec la recette finale du projet.

Lorsque le logiciel est terminé et que toutes les fonctions ont passé positivement leurs tests unitaires ou d'intégration, on utilise le cahier de recette pour passer en revue tout ce qu'il a été demandé explicitement de tester.

6. Recette finale du logiciel

Durée : ½ heure maximum par groupe.

Objet : présentation par le chef de projet et son équipe du programme fonctionnant sous UNIX à un enseignant/client final, pour en vérifier la conformité avec le cahier de recette défini en phase d'analyse. Tous les membres du groupe participent à la présentation du résultat.

- Une démonstration est attendue
- La présentation des documents en version intermédiaire est attendue

Modalités : recette du projet avant le vendredi 12 janvier 2018 12h. Une recette durant environ 30 mn, tenez-en compte pour vos plannings.

VII. Récapitulatif des documents et livrables à rendre

La livraison finale est unique par groupe. Elle a pour but en premier lieu de finaliser le projet global. Elle a lieu pendant la recette.



À
rendre

1. Rapport final et livrables

Il doit contenir les dernières versions des documents suivants :

1. document de conception générale ;
2. cahier de recette ;
3. fiche d'évaluation du document de conception du donneur d'ordre ;
4. fiche d'évaluation de l'implémentation du sous-traitant.
5. document de point d'avancement
6. une notice d'utilisation rédigée de la façon la plus simple possible ;
7. des exemples d'exécution montrant que toutes les fonctionnalités du programme sont vérifiées ; ces exemples pourront être insérés dans la notice d'utilisation et feront partie intégrante de la validation du logiciel ;
8. un bilan général pour le groupe, *ainsi qu'un bilan individuel pour chaque membre du groupe* ; chaque bilan mettra idéalement en évidence les points positifs rencontrés, les solutions d'amélioration employées pour résoudre les problèmes rencontrés, ainsi que les solutions d'amélioration à employer pour réussir encore mieux dans le futur.
9. en plus de ces documents, chaque groupe devra fournir le listing complet (développement « maison » + développement sous-traité) de l'ensemble du programme dûment commenté, + un makefile. Le projet complet doit pouvoir être recompilé à partir de zéro par votre encadrant.

7. Note sur la dernière version

La date finale pour la remise du rapport final vous sera communiqué ultérieurement. Tout retard aura une influence négative sur la note finale, sous la forme d'une pénalité d'un point par jour calendaire entamé de retard, voire de note à 0/20 pour la partie document.

8. Gestion des documents

Les documents doivent être rendus dans un format lisible par l'encadrant. À vous de vous concerter avec eux sur le sujet. Choisissez lorsqu'il est nécessaire de compresser le format ZIP, format universellement reconnu (au lieu des rar et autres formats du genre).

Votre groupe aura un numéro et un trigramme correspondant au projet. Par exemple le groupe 1 se verrait attribué le projet de présentation des tris (TRI). Les documents suivront la convention de nommage numéro du groupe, trigramme projet, nature du document, version du document :

NOM DOCUMENT	REFERENCE	VERSION
01TRI_DCG_V3	Document de Conception Générale	Version 3
01TRI_DCR_V1	Document Cahier de Recette	Version 1
01TRI_EDD_V1	Evaluation du Donneur D'ordre	Version 1
01TRI_EST_V1	Evaluation du Sous-Traitant	Version 1
01TRI_DPA_V1	Document de Point d'Avancement	Version 1
01TRI_NU_V1	Notice d'Utilisation	Version 1
01TRI_BGI_V1	Bilan Général et Individuel	Version 1
01TRI_CR01_V1	Compte Rendu n°1	Version 1

Structure des documents : chaque document devra avoir un en-tête et un pied de page reprenant le nom du document et le numéro de la page. Le document comportera une table des matières. La première page comportera un cartouche indiquant la version et les modifications apportées au cours des différentes versions.

Expression écrite : bien entendu la plus grande attention doit être portée à l'expression écrite. La clarté du propos est importante, les fautes (orthographe et grammaire) doivent être corrigées.

Un total de huit documents séparés est à rendre au final. Chacun de ces documents devra être rendu selon les modalités décrites précédemment.

Les bilans globaux et personnels du rapport final seront le lieu privilégié d'une prise de recul nécessaire à une bonne capitalisation de tout ce qui aura été vécu durant le projet.

Il est important que le projet soit bien entendu vu comme un projet technique d'algorithmique. Mais il est aussi important que l'étudiant perçoive que la réussite complète va bien au-delà de la simple application de « recettes de programmation », et que cette réussite est liée à la mise en application de connaissances issues d'autres domaines que l'informatique pure : ouverture à d'autres domaines scientifiques, gestion de projet, gestion humaine, etc.

VIII. Quelques conseils et remarques

- Réalisez avec le plus grand sérieux possible les documents de conception générale afin de faciliter le travail de votre sous-traitant, mais aussi votre travail d'implémentation ! ;
- basez-vous principalement sur ce qui vous a été enseigné en cours : votre sous-traitant a obligation de résultat uniquement si ce que vous lui demandez est accessible avec les techniques vues en cours ;
- si quelque chose est possible à faire avec ce qui a été vu en cours/TD/TP (graphiques, fichiers, etc.) vous êtes *obligés de l'implémenter avec ce qui vous a été proposé => il est interdit de remplacer* ;

- si quelque chose est impossible à faire avec ce qui a été vu (ajout de son, par exemple), *il est alors autorisé d'ajouter*.

IX. Annexe – démarche de spécification

Cette partie vient en complément des "règles du jeu" du projet. Elle contient les informations techniques précisant certaines notions d'architecture, de conception et de documentation.

1. Notions et exemples « d'exigences »

Une exigence décrit une et une seule caractéristique d'un produit, qui devra être implémentée ET testée. Elle est écrite en langage naturel, si possible avec un sujet, un verbe et un complément uniquement (pas de subordonnée, etc. : restez simple).

Par exemple, on ne dira pas :

« On fait une voiture verte », mais :

- « Le système sera une voiture automobile » puis :

- « Le système sera de couleur verte »,

avec éventuellement des précisions sur la nature de la couleur verte... ;

Autre exemple, pour un programme censé lire un fichier texte contenant des nombres, trier ces nombres et écrire le résultat sur disque, tout en l'affichant à l'écran, nous aurons les exigences suivantes (liste non-exhaustive) :

1. Le programme demandera le nom du fichier texte à ouvrir.
2. Le système lira le fichier texte.
3. Le fichier texte sera constitué de nombres au format décimal entier.
4. Les nombres du fichier seront séparés par des séparateurs classiques (espace, tabulation, retour à la ligne)
5. En cas d'erreur de lecture un message sera affiché.
6. Les nombres seront triés par ordre croissant.
7. Le résultat du tri sera affiché à l'écran.
8. Le résultat du tri sera sauvegardé dans un fichier.
9. Le fichier de sauvegarde portera le même nom que le fichier d'entrée préfixé de "tri_".
10. En cas d'erreur d'écriture un message sera affiché.
11. ...

À vous de voir à quel niveau de détail vous arrêter, mais au plus ce niveau sera fin, au plus le produit final sera maîtrisé.

Les exigences doivent être numérotées, et préférablement regroupées en différentes classes (exigences d'implémentation, exigences de design, exigences d'interface, exigences de sécurité, etc.)

Vous pouvez aussi différencier les exigences impératives des exigences facultatives.

Chaque exigence doit donner naissance à une ou plusieurs procédures de test, pour vérifier et valider que l'exigence est bien remplie.

2. Exemples de « modules »

Rappel : un module est un groupe de fonctions qui rendent un service identifié (par exemple : service d'affichage, service de lecture de fichiers...) et qui doit pouvoir être testé unitairement (c'est-à-dire indépendamment des autres modules du projet).

Découper votre problème en sous-problèmes, qui eux-mêmes pourront être découpés en sous-sous-problème, etc. est une manière efficace :

- de gérer la complexité d'une réalisation ;
- d'améliorer la qualité de son code, car les petits bouts de code sont plus faciles à tester et à corriger que les longues fonctions de plusieurs dizaines de lignes ;
- d'augmenter ses capacités et sa productivité, en se focalisant sur un seul petit problème à la fois ;
- de se répartir les tâches afin de maximiser la performance d'une équipe et de gagner en temps de développement.

C'est pourquoi il vous est conseillé et demandé :

- d'identifier les données que vous allez manipuler ;
- d'identifier les différentes parties de votre programme ;
- d'identifier quelles parties de votre programme manipulent quelles données.

Ainsi, pour notre programme qui lit des données dans un fichier et les trie, nous aurons besoin pratiquement d'une seule donnée, définie comme suit :

```
Structure TableauDeValeurs
    taille du tableau (entier)
    données du tableau (entiers)
```

En langage C, nous aurons écrit cette structure :

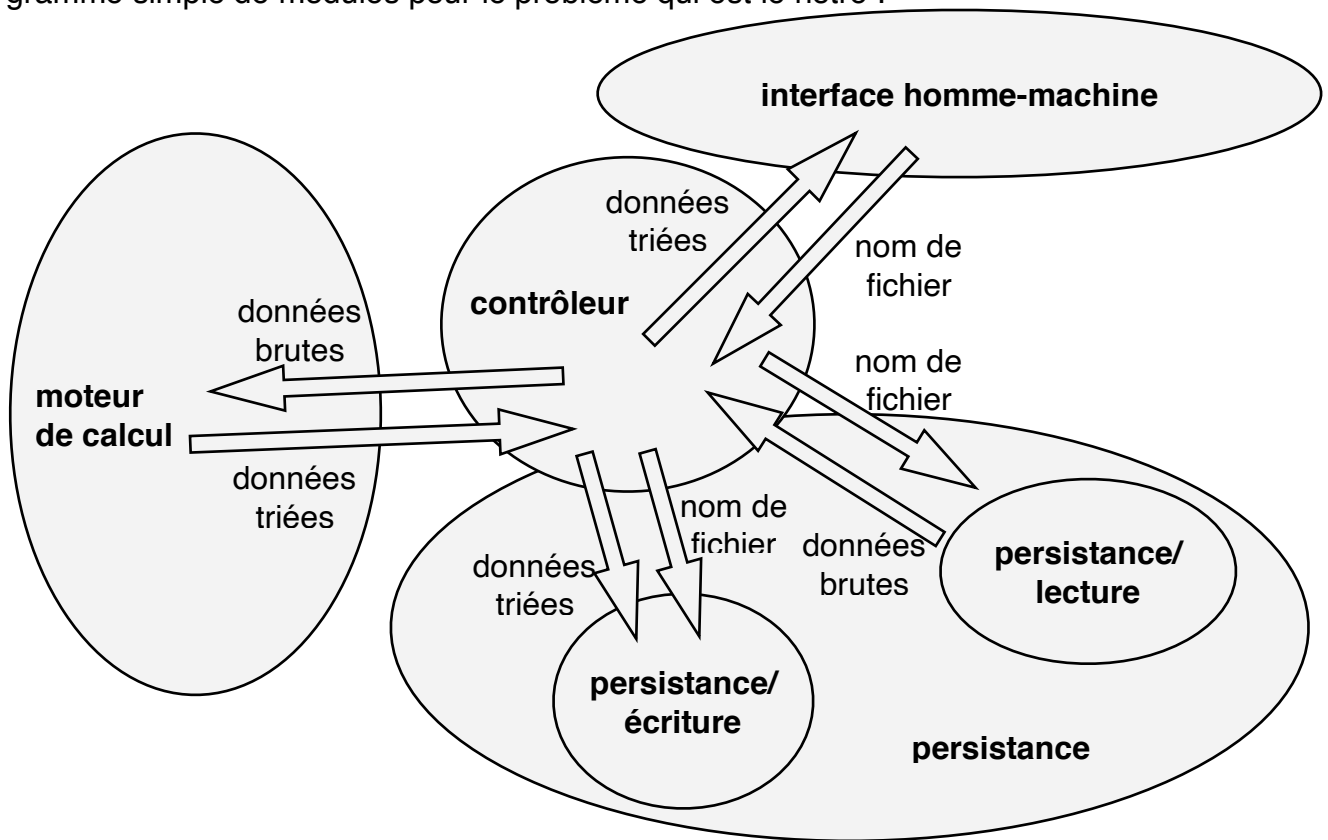
```
typedef struct
{
    int taille;
    int *donnees;
} TableauDeValeurs;
```

Au vu des exigences listées au chapitre précédent, nous pouvons identifier trois modules différents :

- le module « moteur de calcul », qui va faire le travail de tri ;
- le module « persistance », qui va lire et écrire les données sur disque ;
- le module « interface homme-machine » qui va permettre de dialoguer avec l'utilisateur ;
- le module « contrôleur », qui joue le rôle de chef d'orchestre.

Ces modules peuvent être découpés en sous-modules, eux-mêmes découposables, etc. Nous pourrions par exemple avoir un module lecture de fichier et un module écriture de fichier, un module lecture des informations données par l'utilisateur et un module affichage des informations, etc.

Il est important de faire figurer les informations que s'échangent les modules. Toujours pour donner un exemple (qui pourrait être amélioré), voici ce à quoi pourrait ressembler un diagramme simple de modules pour le problème qui est le nôtre :



1. Interface d'une fonction

Les prototypes des fonctions devront être commentés. Vous utiliserez un style "à la doxygen" pour commenter vos prototypes directement dans le code.

Par exemple, une fonction `indValeurMinTableau` qui fournit l'indice de la case du tableau contenant la plus petite valeur (à partir d'un indice donné) se décrirait comme suit :

```
/**
 * \fn int indValeurMinTableau(TableauDeValeurs t, int i)
 * \brief Fonction qui renvoie l'indice de la plus petite valeur à
 * partir de l'indice i
 * \param t Le tableau de valeurs
 * \param i L'indice (y compris) à partir duquel il faut rechercher
 * \return L'indice (à partir de la position i) contenant la plus
 * petite valeur
 */
```

2. Exemple « d'arbre de fonctions »

Chaque module sera constitué d'une ou de plusieurs fonctions. Il vous est demandé d'écrire le prototype de chacune de ces fonctions, c'est-à-dire de décrire leurs « interfaces ».

Ceci est très important, car cela va permettre de donner des informations suffisantes à vos collègues codeurs et/ou à votre sous-traitant.

Un arbre de fonction est une représentation arborescente dans laquelle chaque nœud est une fonction racine, et chaque sous-arbre est une fonction appelée par cette fonction racine.

Ainsi, la fonction *trieDonnees* du moteur de calcul pourrait avoir comme arbre de fonction (voir vos cours/TD pour le fonctionnement de *triSelectionIteratif*).

