Réalisation d'une carte permettant de visualiser une conjonction remarquable,
en utilisant le langage de programmation ADA

I. Ce que l'on a l'intention de faire

Le lundi 18 février 2013, une conjonction spectaculaire réunit la Lune, Jupiter, l'astéroïde Vesta et ceci sur un fond stellaire remarquable : l'amas des Hyades dans la constellation du Taureau.
Vesta, de magnitude visuelle 7,8 au moment de la conjonction n'est pas visible à l'oeil nu, mais accessible à des jumelles mêmes modestes.
Encore faut-il savoir où le chercher.
La conjonction offre une opportunité de lever le nez vers le ciel et le repérer.
L'auteur dispose d'un appareil photo numérique avec un court téléobjectif et veut vérifier ce que l'on peut en faire dans pareille circonstance.

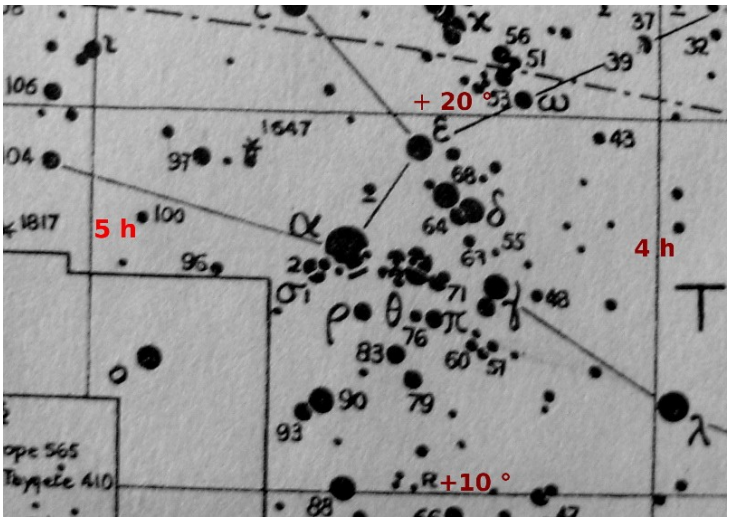
I.1. Quel champ devra couvrir notre carte ?

La carte devra couvrir un champ un peu suppérieur à celui couvert par l'appareil photo. Celui-ci est équipé d'un capteur mesurant **23 mm x 16 mm**.
L'objectif a une focale fixe (pas de zoom) de 105 mm et un rapport d'ouverture avantageux : 2,5.
Le champ couvert qui en découle est approximativement de 13 degrés sur 9 degrés.
Comme Vesta va se déplacer par la suite, on va réaliser une carte un peu plus grande : les dimensions retenues sont de **18 degrés sur 12 degrés.**

I.2. Utilisation d'un investissement intellectuel antérieur

Le présent compte-rendu se base sur un acquis antérieur : le travail effectué pour réaliser une carte de la constellation Orion en utilisant le langage graphique SVG.
Le lecteur aura peut-être intérêt à s'y reporter.

II. Une carte ancienne pour rêver un peu



L'image ci-contre est tirée d'un ouvrage édité par la Société Astronomique de France, « La revue des constellations ».

Ce travail collectif, réalisé avec les moyens des astronomes amateurs des années 50-60, a été un des ouvrages de référence de cette génération. Il est encore recherché dans les sites de vente en ligne, à un prix sans rapport avec celui de l'époque.

Vesta mais aussi Céres vont se trouver « par là » en février 2013.

III. Vers la suite

Par rapport au projet « carte d'Orion », on trouve des similitudes mais aussi des différences :

- Utilisation d'un fichier intermédiaire séquentiel qui ne contient que les données utiles.
- Coordonnées exprimées en degrés et parties décimales de degrés (donc en « flottant » au sens de ADA).
- Les calculs intermédiares seront, le plus souvent possible, délégués à des fonctions ou procédures.
- La structuration des données doit permettre de simplifier le passage des paramètres.
- Le petit catalogue d'étoiles utilisé pour le projet « Orion » est limité à la magnitude 5 alors qu'il faut atteindre au moins la magnitude 8.
  Le catalogue **Hipparcos** a été retenu. Il est constitué de plusieurs fichiers : un « principal » ainsi que des « complémentaires » dédiés aux étoiles doubles, aux variables...
  Dans un premier temps, le catalogue principal seul sera utilisé. Il comprend 118218 enregistrements.
  L'ensemble des catalogues est décrit, en anglais, dans une annexe jointe au présent document.

IV. Création d'un fichier séquentiel intermédiaire

IV.1. Cahier des charges :

Ouvrir le fichier Hipparcos principal. Le lire et vérifier si les données sont interprétables. Si oui, pour chaque ligne du fichier texte, recopier les données suivantes dans autant d'enregistrements structurés.
Chaque enregistrement devra contenir :

- le numéro dans le catalogue Hiparcos,
- la déclinaison,
- l'ascension droite,
- la magnitude.

Analyse du code du programme vesta05.adb

Seuls les éléments jugés importants sont commentés.

Voyons déjà ce qui est simple :

Code	Commentaires

`type Et is record`
`  hip_number  : string(1..6) ;`
`  hip_decl  : float;`
`  hip_ad  : float;`
`  hip_mag  : float;`
`end record ;`
	Définition d'un type pour la structure de l'enregistrement.
	En ADA, il faut « typer » les données.

`hip  : Et ;`	Création d'une variable sur le type.
`Source_Hipparcos  : File_Type;`	L'identifiant Source_Hipparcos est défini pour permettre l'accès au fichier texte.

`package Fichier_IO is new ada.sequential_io(Et);`
`use Fichier_IO ;`
`Sequentiel : Fichier_IO.file_type ;`
	Adaptation d'un paquetage générique au type particulier de donnée défini plus haut.

`Open (Source_Hipparcos,In_File, "hip_main.dat" );`	Ouverture, en lecture seulement, du fichier de données Hipparcos.
`Create(Sequentiel,out_file, "Hip_restreint.dat");`	Création d'un accès sur ce qui deviendra le fichier séquentiel.

`while not End_of_file(Source_Hipparcos) loop`
`….`
 	
`end loop ;`
	Parcours du fichier texte, du début jusqu'à la fin.

`Get_line(Source_Hipparcos,Ligne_fichier_TXT,lg);`	Lecture d'une ligne du fichier texte et transfert du contenu dans la chaîne de caractères « ligne_fichier_TXT »
`write(Sequentiel,hip);`	Ecriture d'un enregistrement dans le fichier séquentiel.
`Close(Source_Hipparcos);` `Close(Sequentiel);`	Fermeture des deux fichiers à la fin du traitement.

Le fichier texte contient quelques enregistrements mal structurés ou incomplets, par absence de point décimal dans certains des trois champs qui m'intéressent.

Code	Commentaires

`if ligne_fichier_TXT(55..55) = "."`
`and ligne_fichier_TXT(68..68) = "."`
`and ligne_fichier_TXT(44..44) = "."`
`  then...`
	Le point décimal a une position parfaitement définie. S'il est présent dans les trois zones, l'enregistrement peut être utilisé.

`hip.hip_number :=` `Ligne_Fichier_TXT(9..14);`	Simple : le numéro de l'étoile est une chaîne de caractères. On le transfère dans une chaîne de dimension identique.

Plus de « valeur ajoutée » dans ce qui suit :

La valeur de la déclinaison est contenue dans une chaine de 11 caractères.
Par exemple `67.1234567 ou -43.1234567`

Cette valeur doit être transformée en valeur numérique (de type float ici) et chargée dans la zone prévue de l'enregistrement (cette zone est une variable de type float).
L'instruction d'appel de la fonction est codée ainsi :
`hip.hip_decl  := declinaison(Ligne_fichier_TXT(65..76));`
(ce qui est entre parenthèses est le paramètre « expédié » à la fonction).
Mais comment la fonction le reçoit-elle ?

Code	Commentaires

`function declinaison(sc : string)`
`return float is`
`begin`
 	
`end ;`
	La fonction reçoit, sous le nom « sc » ce qui lui a été envoyé plus haut. Pas d'obligation d'utiliser le même nom. Celui-ci est plus court et sera plus facile à taper.
	Elle doit **retourner** un « float » (nombre décimal).

`valeur : float ;  …`
 	
`return valeur ;`
	La variable « valeur » sert à recueillir le résultat du travail de la fonction, afin de pouvoir le retourner.

`valeur:=float'value(...) ;`	Transformation d'une chaîne en float. Remarquer la syntaxe particulière à ADA.

La suite est très ADA'iste et mérite un commentaire plus complet.
Si l'on reprend les valeurs données en exemple, la chaine « sc » contient :

position	1	2	3	4	5	6	7	8	9	10	11
ex.1		6	7	.	1	2	3	4	5	6	7
ex.2	-	4	3	.	1	2	3	4	5	6	7

En position 1, on trouve soit un espace, soit le signe moins.
La chaîne allant de la position 2 à la position 11 peut être convertie en nombre flottant dans les deux cas ci-dessus.
Si la chaîne contient le signe « - » en position 1, le résultat doit être multiplié par -1 (pour changer le signe du nombre).
Dans le langage ADA une chaîne est pourvue de deux attributs particuliers (voire plus???) :

- l'attribut `'first` désigne le premier caractère
- l'attribut `'last` désigne le dernier.
- Pour la chaîne « sc » tous les caractères sont compris entre `sc'first` et `sc'last`, ce qui s'écrira `sc(sc'first..sc'first)`
- Si l'on ne veut pas prendre le premier caractère (pour éviter le signe), on écrira `sc(sc'first+1..sc'last)` (ce qui est assez subtil, n'est-ce pas?).

Vous avez maintenant tout ce qu'il faut pour comprendre le code suivant.

Code	Commentaires

`begin`
`valeur:=float'value(sc(sc'first+1..`
`sc'last));`
`if sc(sc'first..sc'first)="-"`
`  then valeur := valeur * (-1.0) ;`
`end if ;`
`return valeur ;`
`end declinaison ;`
	Cela doit vous sembler limpide. Sinon relire ce qui précède.
	Un point délicat :
	la multiplication par -1 se code `* (-1.0)`
	Sans les parenthèses, cela ne va pas.

Explication supplémentaire :

En ADA, une chaîne de caractère n'est terminée par rien , alors que dans certains langages, le C par exemple, un caractère particulier avertit que l'on est arrivé à la fin.
La fonction déclinaison utilise un paramètre « `sc` » dont le type « `string` » ne précise pas le nombre de caractères de la chaîne, ce qui va à l'encontre du typage fort.
Pourquoi l'auteur du code a-t-il fait ce choix ? Parce qu'il voulait que cette fonction « fasse preuve de souplesse » et puisse accepter des paramètres de longueurs différentes.
Imaginons que l'on utilise, dans le même programme, deux catalogues d'étoiles qui codent la déclinaison avec deux chaînes de longueurs différentes. Ce serait rageant d'avoir à coder deux versions de la fonction déclinaison.

Catalogues d'origine	**Exemples de contenus**
Catalogue 1 (sur 11 caractères)	`- 4 2 . 1 2 3 4 5 6 7 8`
Catalogue 2 (sur 8 caractères)	`- 1 2 . 1 2 3 4`

Dans les deux cas, `(sc(sc'first+1` donnera le caractère suivant le signe (marqué en rouge).
Dans les deux cas, `(sc(..sc'last` donnera le dernier caractère suivant le signe (marqué en rbleu).
On obtient ainsi un code plus adaptable, pour le prix d'un petit apprentissage.
La fonction magnitude utilise le même raisonnement, aussi elle n'est pas expliquée en détail.

Le code complet du programme est joint en **annexe 1**.

V. Quelques élements sur la trigonométrie

Jusqu'à présent, on a évité de se confronter au problème posé par la « projection » d'une portion de sphère sur une surface plane : on découpe une zone « rectangulaire » dans un ballon et puis, ensuite, on essaie de l'aplatir sur une table... Franche rigolade.
Tant que l'on reste dans les constellations proches de l'équateur céleste, le résultat n'est pas trop éloigné de la réalité. Mais on va aller vers les pôles célestes, et la qualité va se dégrader.
Sachons que, quelque soit l'effort, on ne résoudra pas parfaitement la difficulté initiale : la portion de ballon sur la table. Intuitivement, on pense à un ballon de caoutchouc que l'on va pouvoir déformer...
Déformer, le mot est trouvé. Mais comment ? Par une sorte d'opération « magique » effectuée grâce à des calculs qui sont difficiles à imaginer par le commun des mortels. Ces calculs étaient faits autrefois à la main, puis avec des calculatrices mécaniques. L'outil informatique facilite grandement les choses... mais, renforçant le côté magique, n'aide pas à comprendre.
Les formulaires de calcul s'appuient sur une discipline mathématique appelée trigonométrie.

V.1. Origine de la trigonométrie[1]

L'étymologie est éclairante :
trigon : suggère les **triangles** (trois angles) ; métron : suggère **mesure**.

Chacun sait que des triangles, il en existe de plusieurs « sortes ». De celle-ci, isolons les triangles qui ont un angle droit.

La suite, page suivante s'applique aux triangles tracés sur un plan (une feuille de papier sur une table, une tablette, un écran d'ordinateur).

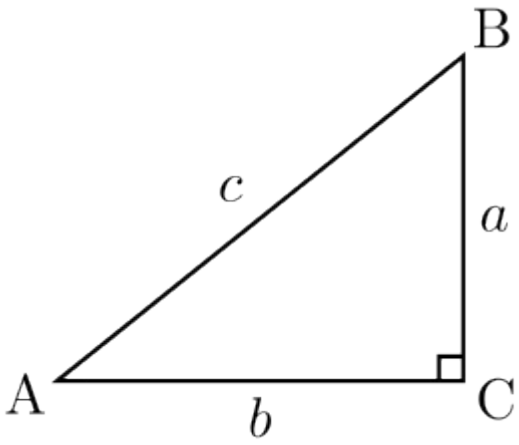
L'astronome et mathématicien grec Hipparque de Nicée (-190 ; -120) construisit les premières tables trigonométriques sous la forme de tables de cordes : elles faisaient correspondre à chaque valeur de l'angle au centre (avec une division du cercle en 360°), la longueur de la corde interceptée dans le cercle, pour un rayon fixe donné. Ce calcul correspond au double du sinus de l'angle moitié, et donne donc, d'une certaine façon, ce que nous appelons aujourd'hui une table de sinus. Toutefois, les tables d'Hipparque n'étant pas parvenues jusqu'à nous, elles ne nous sont connues que par le grec Ptolémée, qui les publia, vers l'an 150, avec leur mode de construction ...

1 Voir http://fr.wikipedia.org/wiki/Trigonom%C3%A9trie

(*Remarque : l'image ci-contre est issue de Wikipedia*).
L'angle droit est marqué par la lettre C.
On suppose que l'on connaît deux valeurs :

- un angle : par exemple l'angle en A.
- un côté : par exemple le « grand côté » appelé hypothénuse et marqué « petit c ».

Avec cela et une table de valeurs précalculées, on peut retrouver tout le reste.



On appliquera une formule du genre :

Mesure du segment **b** = valeur précalculée pour l'angle **A** **multipliée** par la longueur du grand côté.

Remarque :

Dans le cas du triangle rectangle, la trigonométrie, ce n'est pas plus difficile que cela.
Les mathématiciens étant des gens pointilleux, aimant leur vocabulaire à eux, ont remplacé l'expression « valeur précalculée pour l'angle A » par *cosinus A*. Cela donnera :
b = c * cosinus A
Si l'angle A mesure 20 degrés, cosinus **A** vaudra 0,939693... et **c** = 10
b = 10 * 0,939693 = 9,39693
Soit mais l'astronome non mathématicien ne voit pas où cela peut le mener.

Un exemple plus concret :

Voici un exemple plus pratique : au début de l'article, j'ai cité les dimensions du capteur : 23 millimètres pour la largeur ainsi que la longueur focale de l'objectif 105 millimètres.
Si **A** est au centre de l'objectif, **b** est égal à 105 mm et **a** est égal à la moitié de 23 mm, soit 11,5 mm
Connaissant **a** et **b**, on peut calculer les rapport des longueurs **a / b = 0,109524**.
Cette valeur est celle dite des **tangentes**. Si l'on va voir quelle valeur d'angle correspond à 0,109524, on trouve 6,25 degrés.
L'angle couvert par l'objectif, en largeur, mesure 6,25 degrés * 2 soit 12,5 degrés et j'avais arrondi cette valeur en l'augmentant à 13 degrés (page 1).

En résumé et à retenir

- Connaissant certains éléments bien choisis d'un triangle rectangle, il est possible de calculer les autres en utilisant des tables dites trigonométriques. Ces tables peuvent être remplacées par une calculatrice scientifique ou un ordinateur.
- Il est possible d'étendre la trigonométrie à des triangles sans angle droit, au prix de formules plus compliquées.
- Il est également possible de l'appliquer à des triangles tracés à la surface d'un ballon (une sphère), au prix d'un formulaire plus compliqué encore.
- Et, même, il est possible, toujours via un formulaire et une table trigonométrique, de **reporter** des positions d'étoiles repérées sur la sphère céleste **sur** un plan (une carte).
  Cette opération est appelée **projection**.

VI. Les projections

La première fois que j'ai entendu parler de cette opération, c'était par un professeur de géographie, qui se

devait de nous expliquer de quelle façon les cartes qu'il utilisait (planes) pouvaient s'appliquer au monde réel (sphérique).
Il n'y a pas une projection, mais des types de projections, selon ce que l'on veut privilégier...
J'ai utilisé l'article suivant http://sam.electroastro.pagesperso-orange.fr/dossiers/projection/project.htm comme base de mes essais.
Dans les pages suivantes, je vais montrer les résultats obtenus selon le choix du formulaire.

VII. Améliorer la précision des cartes

Le tableau ci-dessous présente les choix qui ont été faits.

Ce que l'on veut faire	Conséquences
Calculs plus exacts	Utilisation du type `long_float` qui permet plus de précision dans les calculs. La transmission des paramètres à « `fairesvg` » devra se faire en flottant et non en entiers arrondis. → modifier « `fairesvg` ».
Faire apparaître les variables à longue période, en les marquant d'un signe distinct.	Récupérer l'information dans le catalogue et la transformer en boolean `TRUE` ou `FALSE`.
Faire apparaître les différences de couleurs des étoiles.	Récupérer l'indice `B-V` dans le catalogue. Trouver moyen d'en tirer parti. Imprimer les étoiles en couleur.
Corriger les coordonnées (ascension droite et déclinaison) des valeurs du mouvement propre, pour la période actuelle (février 2013).	Récupérer ces infos dans le catalogue. Effectuer les calculs et enregistrer les valeurs corrigées.
Effectuer deux calculs de cosinus et les enregistrer de façon à gagner du temps par la suite pour la réalisation de la carte.	Ajouter deux champs, et y écrire les calculs. C'est sans nécessité, mais c'est le choix actuel.

Ces choix, souvent de bon sens, vont s'appliquer sur un fichier intermédiaire, en respectant le fichier initial Hipparcos. La première conséquence, par rapport au programme précédent (`vesta05.adb`) est que l'on va construire un enregistrement (record) avec plus de champs (zones distinctes identifiées).

VII.1. Structure de l'enregistrement

La position de début et de fin dans la ligne lue depuis Hipparcos a été signalée en remarque.

```
type Et is record
-- structure recevant les données transformée depuis le catalogue Hipparcos
-- seules certaines informations sont conservées
   number       : string(1..6) ;  -- (9..14)
   -- ascension droite, déclinaison et magnitude
   ad           : long_float;     -- (52..63) point en  55
   decl         : long_float;     -- (65..76) point en  68
   mag          : float;          -- (42..46) point en  44
   -- mouvement propre en ascension droite et déclinaison en mas / année
   mvp_ad       : long_float ;    -- (88..95)
   mvp_decl     : long_float ;    -- (97..104)
   -- indice de couleur en Bleu moins Vert selon Johnson
   ind_couleur : float ;          -- (246..251)
   -- variable périodique
```

```
   periodique  : boolean ;       -- si (322) = "P" alors TRUE
   -- valeurs calculées pour accélérer la projection de la carte
   sin_decl    : long_float ;    -- sinus de la déclinaison
   cos_decl    : long_float ;    -- cosinus de la déclinaison
end record ;
```

Conséquence sur la partie déclaration du programme ADA

En plus de ces deux lignes :

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Sequential_IO ;
```

Il va falloir ajouter la bibliothèque qui donne accès aux fonctions trigonométriques en `long_float`.

```
with Ada.Numerics.Long_Elementary_Functions;
use Ada.Numerics.Long_Elementary_Functions;
```

Quelques particularités du programme

Le mouvement propre, en déclinaison et en ascension droite est exprimé (et présent dans le fichier Hipparcos) en MAS par an. MAS = millième de seconde d'arc.
Les coordonnées, déclinaison et en ascension droite, exprimées en degrés et partie décimale, sont définies pour l'année 1991,25 (le 0,25 représentant 3 mois). On considére qu'aujourd'hui (fin février 2013), 22 années se sont écoulées.
Le mouvement propre par an multiplié par 22 et exprimé en degrés va être ajouté à la valeur de la coordonnée pour 1991,25. Or 1 MAS = 1 degré divisé par 3600000.
Pour éviter les erreurs dues aux fautes de frappe, deux constantes ont été définies :

```
kdiv   : constant long_float := 3_600_000.0 ;
kannees : constant long_float := 22.0 ;
```

– pour les 22 ans

Les fonctions sinus et cosinus s'appliquent par défaut pour des angles exprimés en radian. Or nos valeurs sont en degrés. Dans l'appel de la fonction, on doit passer le paramètre `360` qui oblige le calcul sur les degrés :

```
 hip.sin_decl := sin(hip.decl, 360.0) ;
 hip.cos_decl := cos(hip.decl, 360.0) ;
```

Autre remarques :

Code	**Remarques**

`if ligne_fichier_TXT(248..248) = "."`
`  then hip.ind_couleur :=`
`float'value(ligne_fichier_TXT(246..251));`
`   else hip.ind_couleur := 0.0 ;`
`end if ;`
	Si le champ contenant l'indice de couleur contient un point, c'est qu'il y a une valeur (sinon il n'y a rien) et dans ce cas, on convertit.
	Sinon on met zéro dans la variable.

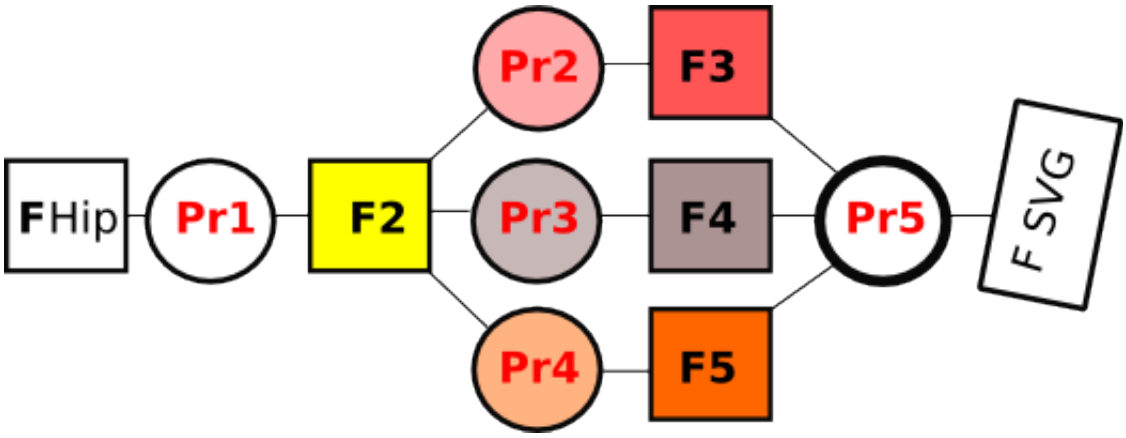
`if ligne_fichier_TXT(322..322) = "P"`
`     then  hip.periodique := true;`
`     else hip.periodique := false;`
`end if ;`
	Si le champ contient la lettre P, alors l'étoile est une périodique.
	Sinon elle ne l'est pas.

`hip.mvp_ad :=`
`long_float'value(ligne_fichier_TXT(88..95`
`));`
	La conversion utilise la syntaxe `long_float'value` pour spécifier que le résultat sera du type `long_float`

L'ensemble du code du programme est joint en **annexe 2.**

VIII. L'organisation du travail en étapes successives

Il a été dit précédemment (voir compte-rendu Orion) que le traitement du problème passerait, si nécessaire, par la génération de fichiers intermédiaires.
Pour le cas du « projet Vesta », voyons à quoi cela pourra ressembler.



Les fichiers sont représentés par des rectangles. Les programmes sont représentés par des cercles.
Le traitement du projet s'effectue de gauche à droite : on part du fichier Hipparcos pour aboutir au fichier SVG qui représente la carte.

Noms	**Ce que c'est**
F Hip	Fichier Hipparcos initial : ici `Hip_main.dat`
Pr1	Premier traitement par programme. Dans ce qui précède, nous avons vu deux versions possibles : `Vesta05.adb` puis `Carte01.adb`. Pour la suite, nous considérons ce dernier comme le bon traitement.
F2	Premier fichier intermédiaire résultant du travail de **Pr1**. Ici, il s'agit de `Hip_etat1.dat`
Pr2, Pr3, Pr4	Programmes effectuant chacun une projection, selon des formulaires différents. Le programme **Pr2** génère le fichier **F3**, le programme **Pr3** génère le fichier **F4**, le programme **Pr4** génère le fichier **F5**.
F3, F4, F5	Fichiers intermédiaires contenant les positions X, Y pour la carte – selon un formulaire de projection particulier, la magnitude, la couleur, la variabilité.
Pr5	Programme réalisant la carte SVG à partir du contenu d'un des fichiers **F3, F4, F5...**

Pourquoi procéder ainsi ?

- Cela permet d'avoir, à chaque fois, des programmes plus courts, plus simples, plus faciles à expliquer (et à concevoir = moindre complexité).
- Chacun des programmes effectue une opération logique facile à identifier : transformation de fichier, tri, projection, réalisation de la carte.
- Le travail effectué par Pr5 est clairement défini et borné.
- Si par la suite, on veut « comparer informatiquement » une photographie avec les données d'un fichier, cela pourra être fait en utilisant le fichier intermédiaire (F3,F4...) qui représente le mieux « la réalité plane de la photo ».

IX. Projection stéréographique à partir du centre de la carte.

Le formulaire utilisé (pris dans le site donné en URL[2]) est le suivant :

$$Y = \frac{2\ (\sin\delta\cos\delta_0 - \cos\delta\sin\delta_0\cos(\alpha-\alpha_0)}{\sin\delta\sin\delta_0 + \cos\delta\cos\delta_0\cos(\alpha-\alpha_0)+1}$$

$$X = -\frac{2\cos\delta\sin(\alpha-\alpha_0)}{\sin\delta\sin\delta_0 + \cos\delta\cos\delta_0\cos(\alpha-\alpha_0)+1}$$

Symboles	**Explications**
Y	Ordonnée sur la carte, **à partir du point central** de celle-ci, pour une **sphère céleste de rayon égal à 1**
X	Absisse sur la carte, **à partir du point central** de celle-ci, pour une **sphère céleste de rayon égal à 1**
Delta zéro	Déclinaison du centre de la portion de la sphère céleste retenue, exprimée en degrés.
Alpha zéro	Ascension droite du centre de la portion de la sphère céleste retenue, exprimée en degrés.
Delta	Déclinaison de l'étoile sur la sphère céleste, exprimée en degrés.
Alpha	Ascension droite de l'étoile sur la sphère céleste, exprimée en degrés.
2 et + 1	Liés à la notion de foyer de la projection (voir URL)

Chacun des mots a été pesé avec soin. Les points importants ou délicats sont en gras.

Points délicats :

« **à partir du point central** » : les valeurs X et Y qui résultent du traitement ne pourront pas être utilisées telles que, parce que les coordonnées en SVG ne partent pas du centre, mais du point haut-gauche.
« **sphère céleste de rayon égal à 1** » : Si l'on veut obtenir une carte de 1200 pixel de large (par exemple), il va falloir appliquer à X et Y un coefficient multiplicateur qui les adaptera aux dimensions de la carte.
C'est aussi une des raisons pour avoir prévu un fichier intermédiaire (**F3, F4, F5)**, de façon à séparer les traitements.

Organiser le calcul de X et de Y

Examinons les différents termes. Le fichier intermédiaire est `Hip_etat1.dat`

Termes	**Remarques**	**Variables stockant cette valeur**
Sinus delta	Présent dans le fichier intermédiaire	Hip.sin_decl
Cosinus delta zéro	A calculer une seule fois	Cos_decl0
Cos (alfa – alfa zéro)	Utilise la différence alfa – alfa zéro qui sera réutilisée pour le calcul de Y. → Mettre la différence dans une variable avant de calculer le cosinus	Cos_ad_ad0 ad_ad0
Cosinus delta	Présent dans le fichier intermédiaire	Hip.cos_decl
Sinus delta zéro	A calculer une seule fois	Sin_decl0
Sinus (alfa – alfa zéro)	Utilise la différence ad_ad0	Sin_ad_ad0

2 http://sam.electroastro.pagesperso-orange.fr/dossiers/projection/project.htm

IX.1. La structure de données

Pour la lecture du fichier

C'est un « `record` » identique à celui qui a permis l'écriture du fichier intermédiaire.
Revoir la description pages 7 et 8.
Seuls changements au niveau de l'ouverture : on ouvre le fichier en lecture (et non en écriture)

```
Open(Dequentiel, in_file, « Hip_etat1.dat ») ;
```

ainsi qu'au niveau de la lecture : le contenu est transféré dans le « record » qui va bien.

```
Read(Sequentiel, hip) ;
```

Pour l'écriture du nouveau fichier intermédiaire

Un enregistrement « `proj` » destiné à contenir les informations est défini, à partir du type « `Tproj` » et le paquet générique « `ada.sequential_io` » est décliné sur le type. Un nom fonctionnel est affecté au fichier « `coordonnees` ».

```
type Tproj is record
   number      : string(1..6) ;
   X           : long_float;
   Y           : long_float;
   mag         : float;
   couleur     : float ;
   periodique  : boolean ;
end record ;
proj           :  Tproj ;
package Fichier_cible is new ada.sequential_io(Tproj);
use Fichier_cible ;
coordonnees : Fichier_cible.file_type ;
```

Pour l'ouverture et l'écriture dans ce nouveau fichier intermédiaire

Il faut créer le fichier :

```
Create(Coordonnees,out_file,"proj_stereo.dat");
```

Pour écrire dedans, on retrouve la syntaxe vue dans les deux programmes précédents :

```
write(Coordonnees,proj);
```

Pour la gestion des contraintes

Il y a tout d'abord les définitions du centre de la zone, les dimensions du champ couvert, la magnitude maximale souhaitée. Ceci est défini par des **constantes** (commençant par la lettre K).

Code	**Ce que cela veut dire**
KcentreX : constant long_float := 67.0 ;	Ascencion droite du centre de la portion de sphère céleste.
KcentreY : constant long_float := 16.0 ;	Déclinaison du centre de la portion de sphère céleste.
Klarg : constant long_float := 18.0 ;	Largeur en degrés de la zone couverte.
Khaut : constant long_float := 12.0 ;	Hauteur en degrés de la zone couverte.
Kmaglim : constant float := 9.0 ;	Magnitude maximale des étoiles de la carte

A partir de ces contraintes, il faut effectuer divers calculs sur la position des « coins » de la portion de sphère céleste, utiles pour filter dans le fichier des étoiles.
Une structure regroupant l'ensemble des contraintes a été définie sous la forme d'un « record ».

```
type Tcontraintes is record
   c_AD           : long_float ;    -- Ascension droite du centre
   c_Decl         : long_float ;    -- Déclinaison du centre
   larg           : long_float ;    -- largeur de l'image en degrés
   haut           : long_float ;    -- hauteur de l'image en degrés
   Ad_limite_W    : long_float ;    -- ascension droite limite à l'ouest
   Ad_limite_E    : long_float ;    -- ascension droite limite à l'est
   Decl_limite_N  : long_float ;    -- limite de la déclinaison Nord
   Decl_limite_S  : long_float ;    -- limite de la déclinaison Sud-
   mag_limite     : float ;
end record ;
contraintes              : Tcontraintes ;
```

Ce choix peut paraître un peu « lourd », mais il a été fait en pensant au passage de paramètres : si l'on passe « `contraintes` » à une procédure ou une fonction, on donne accès à l'ensemble des données.
Cette structure, il faut l'initialiser.
Cela se fait via une procédure :

```
Procedure initialisation_des_contraintes is
Begin
     contraintes.c_AD   := KcentreAD ;
     contraintes.c_Decl  := KcentreDecl ;
     contraintes.larg := Klarg ;
     contraintes.haut := Khaut ;
     contraintes.Ad_limite_W   := contraintes.c_AD   +
                                  (contraintes.larg / 2.0 ) ;
     contraintes.Ad_limite_E   := contraintes.c_AD   -
                                  (contraintes.larg / 2.0 ) ;
     contraintes.Decl_limite_N := contraintes.c_Decl +
                                  (contraintes.haut / 2.0 ) ;
     contraintes.Decl_limite_S := contraintes.c_Decl -
                                  (contraintes.haut / 2.0 ) ;
     contraintes.mag_limite    := kmaglim ;
     sin_decl0  := sin(contraintes.c_Decl,360.0);
     cos_decl0  := cos(contraintes.c_Decl,360.0);
  End initialisation_des_contraintes ;
```

Les « coins » `Ad_limite_W , Ad_limite_E, Decl_limite_N, Decl_limite_S` sont définis à partir du centre et de la largeur et hauteur de la carte exprimés en degrés.
Remarque importante : dès que l'on s'éloigne de l'équateur céleste, les « coins » sont trop resserrés par rapport au champ d'un appareil photo.
Pour le moment, on va s'accomoder de l'approximation, mais il faudra y revenir.

La recherche des étoiles qui sont « dans le cadre » :

Elle est déléguée à une fonction.

```
function est_dans_cadre(sour : Et ; loc : Tcontraintes) return boolean
is
ret : boolean ;
begin
   ret := false ;
   if sour.mag < loc.mag_limite
      then if sour.decl < loc.Decl_limite_N and sour.decl >
                                                 loc.Decl_limite_S
           then if sour.ad < loc.Ad_limite_W and sour.ad >
```

```
                                                        loc.Ad_limite_E
                    then  ret := true ;
                  end if ;
            end if ;
      end if ;
      return ret ;
   end est_dans_cadre ;
```

Si l'étoile est entre les 4 « coins » et que sa magnitude est inférieure à la limite, on la garde.

Le calcul de X et Y à partir du formulaire :

Il s'agit là d'un moment déplaisant, où il faut éviter toute perte de concentration.
Je laisse le lecteur lire le code en se référant au formulaire.
Des calculs de vérification ont été faits avec crayon, papier et calculette. C'était indispensable.

```
procedure calculeYY is
begin
    YY :=2.0 * ((Hip.sin_decl * cos_decl0) -
                  (Hip.cos_decl * sin_decl0 * cos(ad_ad0,360.0)))
                  /
               ((Hip.sin_decl * sin_decl0) +
               (Hip.cos_decl * cos_decl0 * cos(ad_ad0,360.0)) + 1.0 ) ;
end calculeYY ;

procedure calculeXX is
   begin
      XX := 2.0 * (Hip.cos_decl * sin(ad_ad0,360.0))
                /
            ((Hip.sin_decl * sin_decl0) + (Hip.cos_decl*cos_decl0 *
cos(ad_ad0,360.0) +1.0)) ;
      XX := XX *(-1.0) ;
end calculeXX ;
```

Remarquer que ADA tolère le code « aéré ».

Remarques sur ce code :

Il a été pénible à mettre en place, étant à la fois ressemblant et différent du précédent.
A relire ces trois pages, je me demande pourquoi j'ai tant peiné.
Code complet du programme `carte02.adb` en **annexe 3.**

X. Réalisation de la carte SVG à partir du fichier intermédiaire

Quelques finesses ici, mais un programme court : le plus dur a été fait avant.

Quelques constantes faute de mieux :

Code	**Explications**
`Klarg_2 : constant long_float := 540.0 ;`	Demi largeur de la carte en pixels
`Khaut_2 : constant`	Demi-hauteur de la carte en pixels

`long_float := 360.0;`	
`kmulti : constant long_float :=3452.0 ;`	Le demi champ couvert par la carte = 9 degrés. Sinus (9)= 0.1564... La demie largeur / sinus(9) = le facteur de multiplications
`xx, yy : long_float ;` `r : integer ;`	Variables provisoires utiles.

Le seul point un peu délicat est le suivant : les valeurs de X et Y contenues dans le fichier sont valables pour une projection obtenue d'une sphère de rayon 1. Or notre carte correspond à une projection pour une largeur de 1080 pixels (540 x 2). Pour passer de l'un à l'autre, il faut multiplier par un facteur de... ici 3452.

Cette façon de coder, très primitive est liée au fait que je voulais boucler ce soir. C'est à améliorer .

La fonction `calcule_r` est un petit bricolage pour trouver une « bonne » taille des cercles en fonction de la magnitude.

Code	**Explications**

`xx := proj.x * kmulti ;`
`yy := proj.y * kmulti ;`
`xx := xx + Klarg_2 ;`
`if yy > 0.0`
`then`
`yy :=Khaut_2 -yy ;`
`else yy :=(yy *(-1.0)) + Khaut_2 ;`
`end if ;`
	Les coordonnées sont mise à la taille de la carte puis les points sont positionnés en tenant compte du fait que l'origine est dans le coin haut-gauche dans les dessins SVG.

Quelques finasseries encore à l'essai :

- Ecrire le numéro de l'étoile dans la bonne couleur et selon la magnitude :

Si l'étoile est une variable périodique, on écrit son numéro en rouge. Pour les autres, on n'écrit le numéro – en bleu – que si leur magnitude est inférieure à 7. (et c'est largement fouillis comme cela).

```
if proj.periodique
    then texte(integer(xx-10.0) ,integer(yy+10.0),5,"red",proj.number);
      else  if proj.mag < 7.0
                    then texte(integer(xx-10.0)
,integer(yy+10.0),5,"blue",proj.number);
               end if ;
end if ;
```

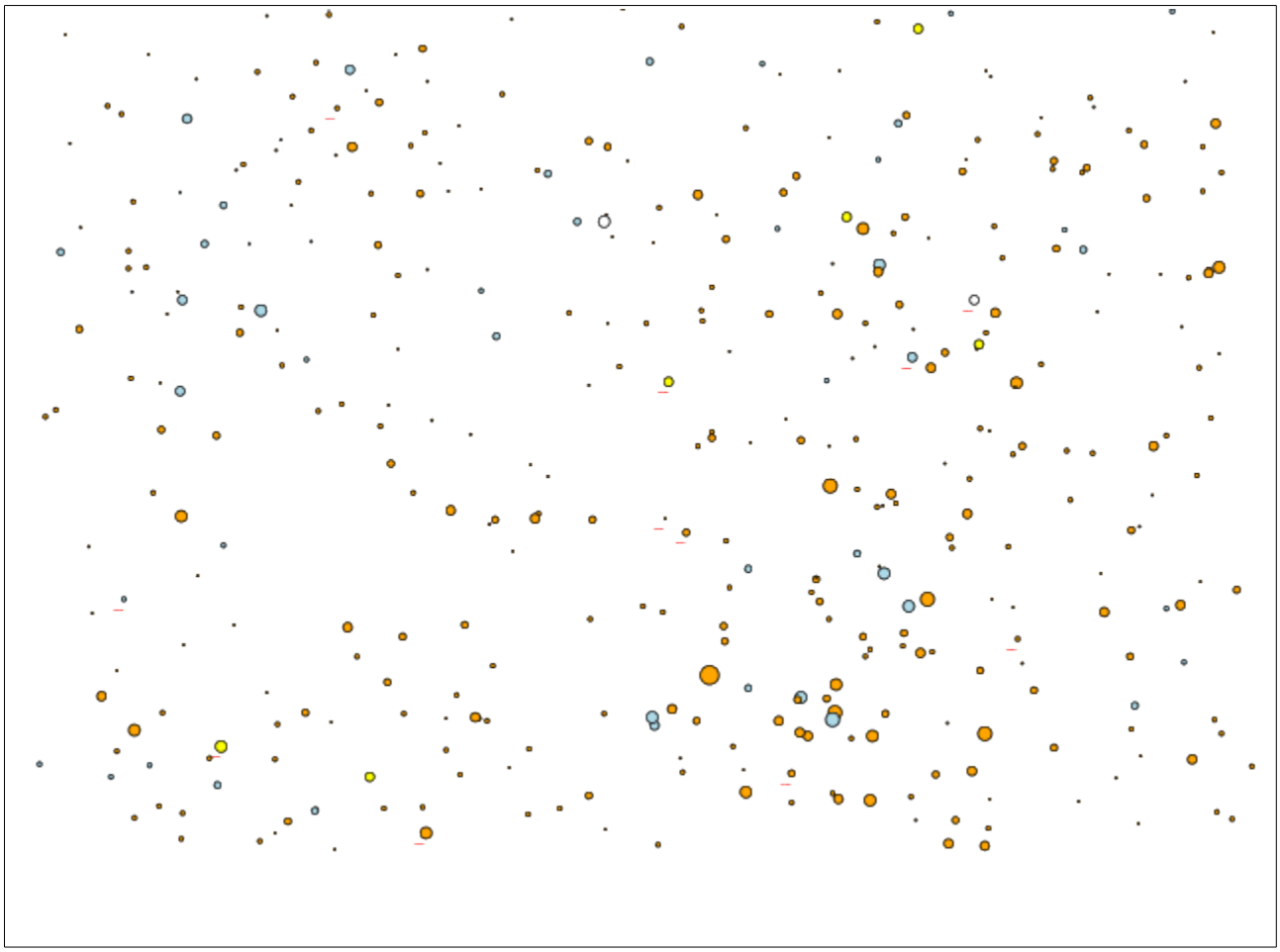
Tracer le cercle à la taille de l'étoile (selon la magnitude) et le remplir de la « bonne couleur » en fonction de l'indice B-V. Très approximatif, mais donne un joli résultat.

```
 if proj.couleur >0.2
     then cercle(integer(xx),integer(yy),r,"black",1,"orange");
     else if proj.couleur > -0.1 and proj.couleur <= 0.0
               then cercle(integer(xx),integer(yy),r,
                                        "black",1,"yellow");
               else if proj.couleur <= -0.1 and proj.couleur > -0.2
                        then cercle(integer(xx),integer(yy),r,
                                        "black",1,"white");
```

```
                    else cercle(integer(xx),integer(yy),r,
                                        "black",1,"lightblue");
                end if ;
            end if ;
        end if ;
```

Le code source du programme `carte03.adb` **est en annexe 4**.

Cette partie devra être relue.
En attendant, une carte de la tête du Taureau, avec des couleurs.



ANNEXES

Annexe 1 : code source du programme Vesta05.adb

```
-- programme vesta05.adb
-- lerautal 02 / 2013
-- lecture du fichier texte Hipparcos
-- création d'un fichier séquentiel contenant toutes les étoiles

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Sequential_IO ; -- use Ada.Sequential_IO ;

Procedure vesta05 is
-- types
type Et is record
   hip_number     : string(1..6) ;
   hip_decl       : float;
   hip_ad         : float;
   hip_mag        : float;
end record ;
-- ---------------- variables
lg                : natural ;
Ligne_fichier_TXT : string (1 .. 449);
hip               : Et ;
Source_Hipparcos  : File_Type; -- attention !!!
-- mettre cette déclaration AVANT celle du fichier séquentiel pour éviter
-- toute ambiguité
-- --              déclarations spécifiques au fichier séquentiel
package Fichier_IO is new ada.sequential_io(Et);
use Fichier_IO ;
Sequentiel : Fichier_IO.file_type ;
------------------ fin de ces déclarations --------

   function declinaison(sc : string) return float is
      valeur : float ;
   begin
      valeur:=float'value(sc(sc'first+1..sc'last));
               if sc(sc'first..sc'first)="-"
                  then valeur := valeur * (-1.0) ;
               end if ;
      return valeur ;
    end declinaison ;

   function magnitude (sc : string) return float is
   valeur :  float ;
   begin
      if sc(sc'first..sc'first) = "-"    then
               valeur :=float'value(sc(sc'first+1..sc'last))*(-1.0) ;
         else  if sc(sc'first..sc'first) = "1"  then
                  valeur :=float'value(sc(sc'first..sc'last)) ;
                  else
                  valeur := float'value(sc(sc'first+1..sc'last)) ;
         end if ;
      end if ;
      return valeur ;
   end magnitude ;
```

```
Begin
   Open (Source_Hipparcos,In_File,"hip_main.dat" );
   Create(Sequentiel,out_file,"Hip_restreint.dat");
   while not End_of_file(Source_Hipparcos) loop
      Get_line(Source_Hipparcos,Ligne_fichier_TXT,lg);
      if ligne_fichier_TXT(55..55) = "." and ligne_fichier_TXT(68..68) = "." and
ligne_fichier_TXT(44..44) = "."
      then
         hip.hip_number := Ligne_Fichier_TXT(9..14);
         hip.hip_mag    := magnitude(Ligne_fichier_TXT(42..46));
         hip.hip_decl   := declinaison(Ligne_fichier_TXT(65..76));
         hip.hip_ad     := float'value(ligne_fichier_TXT(52..63));
         write(Sequentiel,hip);
      end if ;
    end loop ;
   Close(Source_Hipparcos);
   Close(Sequentiel);
   Put("Catalogue Hipparcos écrit dans le fichier annexe : ");

End vesta05 ;
```

Annexe 2 : Code du programme carte01.adb

```
-- nouvelle base pour réaliser des cartes de champs stellaires
-- à partir du catalogue Hipparcos
-- lerautal 02 / 2013
-- les coordonnées ad et decl sont corrigées du mouvement propre
-- pour l'année 2013 (soit 22 ans après l'année de référence du catalogue)
-- les valeurs sinus et cosinus sont calculées à partir de la valeur
-- corrigée de la déclinaison

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Sequential_IO ; -- use Ada.Sequential_IO ;
with Ada.Numerics.Long_Elementary_Functions;  -- fonctions trigonométriques
use Ada.Numerics.Long_Elementary_Functions;

Procedure carte01 is

-- ******** Ce qui se rapporte au fichier texte lu (Hip_main.dat)
lg                : natural ;
Ligne_fichier_TXT : string (1 .. 449);
Source_Hipparcos  : File_Type;

-- ********* ce qui se rapporte au fichier séquentiel créé
type Et is record
-- structure recevant les données transformée depuis le catalogue Hipparcos
-- seules certaines informations sont conservées
   number      : string(1..6) ;  -- (9..14)
   -- ascension droite, déclinaison et magnitude
   ad          : long_float;     -- (52..63) point en  55
   decl        : long_float;     -- (65..76) point en  68
   mag         : float;          -- (42..46) point en  44
   -- mouvement propre en ascension droite et déclinaison en mas / année
   mvp_ad      : long_float ;    -- (88..95) point en
   mvp_decl    : long_float ;    -- (97..104) point en
   -- indice de couleur en Bleu moins Vert selon Johnson
   ind_couleur : float ;         -- (246..251) point en
   -- variable périodique  et periode
   periodique  : boolean ;       -- si (322) = "P" alors TRUE
   -- valeurs calculées pour accélérer la projection de la carte
   sin_decl    : long_float ;    -- sinus de la déclinaison
   cos_decl    : long_float ;    -- cosinus de la déclinaison
end record ;
hip               : Et ;
package Fichier_cible is new ada.sequential_io(Et);
use Fichier_cible ;
Sequentiel : Fichier_cible.file_type ;

kdiv : constant long_float := 3_600_000.0 ;   -- pour convertir les MAS en degrés
kannees : constant long_float := 22.0 ;       -- 22 ans depuis 1991

   function magnitude (sc : string) return float is
   valeur :  float ;
   begin
      if sc(sc'first..sc'first) = "-"    then
               valeur :=float'value(sc(sc'first+1..sc'last))*(-1.0) ;
         else  if sc(sc'first..sc'first) = "1"  then
                  valeur :=float'value(sc(sc'first..sc'last)) ;
```

```
                else
                valeur := float'value(sc(sc'first+1..sc'last)) ;
          end if ;
       end if ;
       return valeur ;
    end magnitude ;

    function declinaison(sc : string) return long_float is
       valeur : long_float ;
    begin
       valeur:=long_float'value(sc(sc'first+1..sc'last));
                if sc(sc'first..sc'first)="-"
                   then valeur := valeur * (-1.0) ;
                end if ;
       return valeur ;
     end declinaison ;

Begin
   Open (Source_Hipparcos,In_File,"hip_main.dat" );
   Create(Sequentiel,out_file,"Hip_etat1.dat");
   while not End_of_file(Source_Hipparcos) loop
      Get_line(Source_Hipparcos,Ligne_fichier_TXT,lg);
      if ligne_fichier_TXT(55..55) = "." and ligne_fichier_TXT(68..68) = "." and
                ligne_fichier_TXT(44..44) = "."
         then
            hip.number := Ligne_Fichier_TXT(9..14);
            hip.ad     := long_float'value(ligne_fichier_TXT(52..63));
            hip.decl   := declinaison(Ligne_fichier_TXT(65..76));
            hip.mvp_ad := long_float'value(ligne_fichier_TXT(88..95));
            hip.ad     := hip.ad  + ((hip.mvp_ad * kannees) / kdiv) ;
            hip.mvp_decl := long_float'value(ligne_fichier_TXT(97..104));
            hip.decl   := hip.decl + ((hip.mvp_decl * kannees) / kdiv) ;
            hip.mag    := magnitude(Ligne_fichier_TXT(42..46));
            if ligne_fichier_TXT(248..248) = "."
               then hip.ind_couleur := float'value(ligne_fichier_TXT(246..251));
               else hip.ind_couleur := 0.0 ;
            end if ;
            if ligne_fichier_TXT(322..322) = "P"
               then  hip.periodique := true;
               else hip.periodique := false;
            end if ;
            hip.sin_decl := sin(hip.decl, 360.0) ;
            hip.cos_decl := cos(hip.decl, 360.0) ;
            write(Sequentiel,hip);
      end if ;
   end loop ;
   Close(Source_Hipparcos);
   Close(Sequentiel);
   Put("Catalogue Hipparcos écrit dans le fichier annexe : ") ;

End carte01 ;
```

Annexe 3 : code du fichier carte02.adb

```
-- tri dans le fichier intermédiaire pour extraire les étoiles utiles
-- lerautal 02 / 2013
-- on va calculer les coordonnées X et Y
-- en utilisant un formulaire de projection

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO ; use Ada.Integer_Text_IO ;
with Ada.Sequential_IO ; -- use Ada.Sequential_IO ;
with Ada.Long_Float_Text_IO; use Ada.Long_Float_Text_IO ;
with Ada.Numerics.Long_Elementary_Functions;  -- fonctions
trigonométriques
use Ada.Numerics.Long_Elementary_Functions;


Procedure carte02 is

-- ********* ce qui se rapporte au fichier séquentiel lu
type Et is record
   number      : string(1..6) ;
   ad          : long_float;        decl        : long_float;
   mag         : float;
   mvp_ad      : long_float ;       mvp_decl    : long_float ;
   ind_couleur : float ;            periodique  : boolean ;
   sin_decl    : long_float ;
   cos_decl    : long_float ;
end record ;
hip               : Et ;
package Fichier_source is new ada.sequential_io(Et);
use Fichier_source ;
Sequentiel : Fichier_source.file_type ;

-- ********* ce qui se rapporte au fichier séquentiel écrit
type Tproj is record
   number      : string(1..6) ;
   X           : long_float;
   Y           : long_float;
   mag         : float;
   couleur     : float ;
   periodique  : boolean ;
end record ;
proj           :  Tproj ;
package Fichier_cible is new ada.sequential_io(Tproj);
use Fichier_cible ;
coordonnees : Fichier_cible.file_type ;

-- ********* ce qui se rapporte aux contraintes
KcentreAD   :  constant long_float := 30.0 ;
KcentreDecl :  constant long_float := 35.0 ;
Klarg    :  constant long_float := 18.0 ;
Khaut    :  constant long_float := 12.0 ;
Kmaglim  :  constant float      :=  9.0 ;  -- magnitude limite
```

```
type Tcontraintes is record
   c_AD            : long_float ;   -- coordonnée x du centre
   c_Decl            : long_float ;   -- coordonnée y du centre
   larg           : long_float ;   -- largeur de l'image
   haut           : long_float ;   -- hauteur de l'image
   Ad_limite_W    : long_float ;   -- ascension droite limite à l'ouest
   Ad_limite_E    : long_float ;   -- ascension droite limite à l'est
   Decl_limite_N  : long_float ;   -- limite de la déclinaison Nord
   Decl_limite_S  : long_float ;   -- limite de la déclinaison Sud-
   mag_limite     : float ;
end record ;
contraintes               : Tcontraintes ;

-- **** variables pour calculs intermédiaires
sin_decl0         : long_float ; -- calculé une seule fois
cos_decl0         : long_float ; -- calculé une seule fois
ad_ad0            : long_float ; -- calculé une fois dans la boucle
utilisé deux fois
-- coordonnées de la projection
XX                : long_float ;
YY                : long_float ;
compteur : integer := 0 ;

  Procedure initialisation_des_contraintes is
   Begin
      contraintes.c_AD  := KcentreAD ; contraintes.c_Decl  :=
KcentreDecl ;
      contraintes.larg := Klarg ;      contraintes.haut := Khaut ;
      contraintes.Ad_limite_W   := contraintes.c_AD   +
(contraintes.larg / 2.0 ) ;
      contraintes.Ad_limite_E   := contraintes.c_AD   -
(contraintes.larg / 2.0 ) ;
      contraintes.Decl_limite_N := contraintes.c_Decl +
(contraintes.haut / 2.0 ) ;
      contraintes.Decl_limite_S := contraintes.c_Decl -
(contraintes.haut / 2.0 ) ;
      contraintes.mag_limite    := kmaglim ;
      sin_decl0  := sin(contraintes.c_Decl,360.0);
      cos_decl0  := cos(contraintes.c_Decl,360.0);
   End initialisation_des_contraintes ;

   function est_dans_cadre(sour : Et ; loc : Tcontraintes) return
boolean is
      ret : boolean ;
   begin
      ret := false ;
      if sour.mag < loc.mag_limite
      then if sour.decl < loc.Decl_limite_N and sour.decl >
loc.Decl_limite_S
            then if sour.ad < loc.Ad_limite_W and sour.ad >
loc.Ad_limite_E
                  then
                    ret := true ;
```

```
                  end if ;
            end if ;
      end if ;
      return ret ;
   end est_dans_cadre ;

   procedure calculeYY is
   begin
    YY :=2.0 * ((Hip.sin_decl * cos_decl0) -
                (Hip.cos_decl * sin_decl0 * cos(ad_ad0,360.0)))
                /
              ((Hip.sin_decl * sin_decl0) +
              (Hip.cos_decl * cos_decl0 * cos(ad_ad0,360.0)) + 1.0 ) ;
   end calculeYY ;

   procedure calculeXX is
   begin
      XX := 2.0 * (Hip.cos_decl * sin(ad_ad0,360.0))
                /
           ((Hip.sin_decl * sin_decl0) + (Hip.cos_decl*cos_decl0 *
cos(ad_ad0,360.0) +1.0)) ;
      XX := XX *(-1.0) ;
   end calculeXX ;


Begin
   initialisation_des_contraintes ;
   Open(Sequentiel,in_file,"hip_etat1.dat");
   Create(Coordonnees,out_file,"proj_stereo_TRI.dat");

   while not End_of_file(Sequentiel) loop
     Read(Sequentiel,hip);
     if est_dans_cadre(hip, contraintes)
         then
         ad_ad0 := hip.ad - contraintes.c_AD ;
         CalculeYY ;
         CalculeXX ;
         proj.number    := hip.number ;
         proj.x         := XX ;
         proj.y         := YY ;
         proj.mag       := hip.mag ;
         proj.couleur   := hip.ind_couleur ;
         proj.periodique := hip.periodique ;
         compteur := compteur + 1 ;
         write(coordonnees,proj);
      end if ;

   end loop ;
   Close(Sequentiel);
   Close(Coordonnees);
   Put("Fichier pour projection réalisé : ");
   put(compteur);

End carte02 ;
```

Annexe 4 : programme carte03.adb

```
-- générer fichier SVG à partir d'une projection
-- lerautal 02 / 2013

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO ; use Ada.Integer_Text_IO ;
with Ada.Sequential_IO ; -- use Ada.Sequential_IO ;
with Ada.Long_Float_Text_IO; use Ada.Long_Float_Text_IO ;
with Ada.Numerics.Long_Elementary_Functions;  -- fonctions
trigonométriques
use Ada.Numerics.Long_Elementary_Functions;
with fairesvg ; use fairesvg ;

procedure carte03 is

type Tproj is record
   number      : string(1..6) ;
   X           : long_float;
   Y           : long_float;
   mag         : float;
   couleur     : float ;
   periodique  : boolean ;
end record ;
proj           :  Tproj ;
package Fichier_cible is new ada.sequential_io(Tproj);
use Fichier_cible ;
coordonnees : Fichier_cible.file_type ;

Klarg_2    :  constant long_float := 540.0 ;
Khaut_2    :  constant long_float := 360.0;
kmulti   :  constant long_float :=3452.0 ;
xx, yy   : long_float ;
r        : integer ;

   function calcule_r(pr : Tproj) return integer is
   locale : integer ;
   begin
      locale :=integer((pr.mag - 10.0) *(-0.9));
      return locale ;
   end calcule_r ;

begin
   Open(Coordonnees,in_file,"proj_stereo.dat");
   create(Fichier,Out_file,"Taureau_couleur.svg");
   en_tete;
   cadre(1080,800);
   -- encadrement ;
   while not End_of_file(Coordonnees) loop
     Read(Coordonnees,proj);
      xx := proj.x * kmulti ;
      yy := proj.y * kmulti ;
      xx := xx + Klarg_2  ;
```

```
      if yy > 0.0
         then
               yy :=Khaut_2 -yy ;
         else  yy :=(yy *(-1.0)) + Khaut_2 ;
      end if ;
      r:= calcule_r(proj);
      if proj.couleur >0.2
         then cercle(integer(xx),integer(yy),r,"black",1,"orange");
         else if proj.couleur > -0.1 and proj.couleur <= 0.0
               then
cercle(integer(xx),integer(yy),r,"black",1,"yellow");
               else if proj.couleur <= -0.1 and proj.couleur > -0.2
                        then
cercle(integer(xx),integer(yy),r,"black",1,"white");
                        else
cercle(integer(xx),integer(yy),r,"black",1,"lightblue");
                    end if ;
             end if ;
      end if ;
      if proj.periodique
         then texte(integer(xx-10.0)
,integer(yy+10.0),5,"red",proj.number);
          else  if proj.mag < 7.0
                    then texte(integer(xx-10.0)
,integer(yy+10.0),5,"blue",proj.number);
                end if ;
      end if ;
   end loop ;
   Close(Coordonnees);
   fin;
   Close(Fichier);
   Put("Fichier pour projection lu : ");

end  carte03 ;
```

Annexe ??? : Documentation du catalogue Hipparcos

I/239 The Hipparcos and Tycho Catalogues (ESA 1997)

The Hipparcos and Tycho Catalogues
ESA 1997
<ESA, 1997, The Hipparcos Catalogue, ESA SP-1200>
<ESA, 1997, The Tycho Catalogue, ESA SP-1200>
=1997HIP...C.....0E

Liste des fichiers composant le catalogue Hipparcos :

FileName	Records		Explanations
ReadMe	80	.	This file
hip_main.dat	450	118218	The Hipparcos Main Catalogue
h_dm_com.dat	238	24588	Double and Multiples: Component solutions -COMP
h_dm_cor.dat	238	12591	Double and Multiples: Component solutions -CORR
hip_dm_g.dat	195	2622	Double and Multiples: Acceleration solutions
hip_dm_o.dat	337	235	Double and Multiples: Orbital solutions
hip_dm_v.dat	144	288	Double and Multiples: VIM solutions
hip_dm_x.dat	22	1561	Double and Multiples: Stochastic solutions
hip_va_1.dat	142	2712	Variability Annex: Periodic variables
hip_va_2.dat	142	5542	Variability Annex: Unsolved variables
solar_ha.dat	64	5609	Solar System Annex: Astrometric catalogue
solar_hp.dat	63	2639	Solar System Annex: Photometric catalogue
solar_t.dat	95	291	Solar System Annex: Tycho astrometry/photometry
hd_notes.doc	97	2622	Hipparcos notes: Double and multiple systems
hg_notes.doc	97	3898	Hipparcos notes: General notes
hp_notes.doc	97	2444	Hipparcos notes: Photometric notes
hp_refs.doc	19	33769	References Hipparcos stars
hp_auth.doc	80	4335	References of hp_notes.doc
dmsa_o.doc	80	118	References of hip_dm_o.dat
tyc_main.dat	350	1058332	The main part of Tycho Catalogue

Description de la structure d'une ligne du fichier texte hip_main.dat

Bytes	Format	Units	Label	Explanations	
1	A1	---	Catalog	[H] Catalogue (H=Hipparcos)	(H0)
9- 14	I6	---	HIP	**Identifiant :** numéro Hipparcos	(H1)
16	A1	---	Proxy	*[HT] Proximity flag	(H2)
18- 28	A11	---	RAhms	Right ascension in h m s, ICRS (J1991.25)	(H3)
30- 40	A11	---	DEdms	Declination in deg ' ", ICRS (J1991.25)	(H4)
42- 46	**F5.2**	**mag**	**Vmag**	**? Magnitude in Johnson V**	(H5)
48	I1	---	VarFlag	*[1,3]? Coarse variability flag	(H6)
50	A1	---	r_Vmag	*[GHT] Source of magnitude	(H7)
52- 63	**F12.8**	**deg**	**RAdeg**	***? alpha, degrees (ICRS, Epoch=J1991.25)**	(H8)
65- 76	**F12.8**	**deg**	**DEdeg**	***? delta, degrees (ICRS, Epoch=J1991.25)**	(H9)
78	A1	---	AstroRef	*[*+A-Z] Reference flag for astrometry	(H10)
80- 86	F7.2	mas	Plx	? Trigonometric parallax	(H11)
88- 95	**F8.2**	**mas/yr**	**pmRA**	***? Proper motion mu_alpha.cos(delta),** ICRS	(H12)
97-104	**F8.2**	**mas/yr**	**pmDE**	***? Proper motion mu_delta, ICRS**	(H13)
106-111	F6.2	mas	e_RAdeg	*? Standard error in RA*cos(DEdeg)	(H14)

```
 113-118  F6.2  mas    e_DEdeg    *? Standard error in DE                    (H15)
 120-125  F6.2  mas    e_Plx       ? Standard error in Plx                   (H16)
 127-132  F6.2 mas/yr e_pmRA      ? Standard error in pmRA                   (H17)
 134-139  F6.2 mas/yr e_pmDE      ? Standard error in pmDE                   (H18)
 141-145  F5.2  ---     DE:RA     [-1/1]? Correlation, DE/RA*cos(delta)      (H19)
 147-151  F5.2  ---     Plx:RA    [-1/1]? Correlation, Plx/RA*cos(delta)     (H20)
 153-157  F5.2  ---     Plx:DE    [-1/1]? Correlation, Plx/DE                (H21)
 159-163  F5.2  ---     pmRA:RA   [-1/1]? Correlation, pmRA/RA*cos(delta)    (H22)
 165-169  F5.2  ---     pmRA:DE   [-1/1]? Correlation, pmRA/DE               (H23)
 171-175  F5.2  ---     pmRA:Plx  [-1/1]? Correlation, pmRA/Plx              (H24)
 177-181  F5.2  ---     pmDE:RA   [-1/1]? Correlation, pmDE/RA*cos(delta)    (H25)
 183-187  F5.2  ---     pmDE:DE   [-1/1]? Correlation, pmDE/DE               (H26)
 189-193  F5.2  ---     pmDE:Plx  [-1/1]? Correlation, pmDE/Plx              (H27)
 195-199  F5.2  ---     pmDE:pmRA [-1/1]? Correlation, pmDE/pmRA             (H28)
 201-203  I3    %       F1         ? Percentage of rejected data            (H29)
 205-209  F5.2  ---     F2        *? Goodness-of-fit parameter              (H30)
 211-216  I6    ---     ---        HIP number (repetition)                  (H31)
 218-223  F6.3  mag     BTmag      ? Mean BT magnitude                      (H32)
 225-229  F5.3  mag    e_BTmag     ? Standard error on BTmag
(H33)
 231-236  F6.3  mag     VTmag      ? Mean VT magnitude                      (H34)
 238-242  F5.3  mag    e_VTmag     ? Standard error on VTmag                (H35)
     244  A1    ---    m_BTmag    *[A-Z*-] Reference flag for BT and VTmag  (H36)
 246-251  F6.3  mag      B-V       ? Johnson B-V colour                     (H37)
 253-257  F5.3  mag    e_B-V       ? Standard error on B-V                  (H38)
     259  A1    ---    r_B-V       [GT] Source of B-V from Ground or Tycho  (H39)
 261-264  F4.2  mag      V-I       ? Colour index in Cousins' system        (H40)
 266-269  F4.2  mag    e_V-I       ? Standard error on V-I                  (H41)
     271  A1    ---    r_V-I      *[A-T] Source of V-I                      (H42)
     273  A1    ---      CombMag   [*] Flag for combined Vmag, B-V, V-I     (H43)
 275-281  F7.4  mag      Hpmag    *? Median magnitude in Hipparcos system   (H44)
 283-288  F6.4  mag    e_Hpmag    *? Standard error on Hpmag                (H45)
 290-294  F5.3  mag      Hpscat    ? Scatter on Hpmag                       (H46)
 296-298  I3    ---    o_Hpmag     ? Number of observations for Hpmag       (H47)
     300  A1    ---    m_Hpmag    *[A-Z*-] Reference flag for Hpmag
(H48)
 302-306  F5.2  mag      Hpmax     ? Hpmag at maximum (5th percentile)      (H49)
 308-312  F5.2  mag      HPmin     ? Hpmag at minimum (95th percentile)     (H50)
 314-320  F7.2  d        Period    ? Variability period (days)              (H51)
     322  A1    ---      HvarType *[CDMPRU]? variability type               (H52)
     324  A1    ---      moreVar  *[12] Additional data about variability   (H53)
     326  A1    ---      morePhoto [ABC] Light curve Annex                  (H54)
 328-337  A10   ---      CCDM      CCDM identifier                          (H55)
     339  A1    ---    n_CCDM     *[HIM] Historical status flag
(H56)
 341-342  I2    ---      Nsys      ? Number of entries with same CCDM       (H57)
 344-345  I2    ---      Ncomp     ? Number of components in this entry     (H58)
     347  A1    ---      MultFlag *[CGOVX] Double/Multiple Systems flag     (H59)
     349  A1    ---      Source   *[PFILS] Astrometric source flag          (H60)
     351  A1    ---      Qual     *[ABCDS] Solution quality                 (H61)
 353-354  A2    ---    m_HIP       Component identifiers                    (H62)
 356-358  I3    deg      theta     ? Position angle between components      (H63)
 360-366  F7.3  arcsec   rho       ? Angular separation between components  (H64)
 368-372  F5.3  arcsec   e_rho     ? Standard error on rho                  (H65)
 374-378  F5.2  mag      dHp       ? Magnitude difference of components     (H66)
 380-383  F4.2  mag    e_dHp       ? Standard error on dHp                  (H67)
     385  A1    ---      Survey    [S] Flag indicating a Survey Star        (H68)
     387  A1    ---      Chart    *[DG] Identification Chart                (H69)
     389  A1    ---      Notes    *[DGPWXYZ] Existence of notes             (H70)
```

```
 391-396  I6    ---     HD        [1/359083]? HD number <III/135>          (H71)
 398-407  A10   ---     BD        Bonner DM <I/119>, <I/122>
(H72)
 409-418  A10   ---     CoD       Cordoba Durchmusterung (DM) <I/114>      (H73)
 420-429  A10   ---     CPD       Cape Photographic DM <I/108>             (H74)
 431-434  F4.2  mag     (V-I)red  V-I used for reductions                  (H75)
 436-447  A12   ---     SpType    Spectral type                            (H76)
     449  A1    ---   r_SpType   *[1234GKSX]? Source of spectral type      (H77)
--------------------------------------------------------------------------------
```

Interprétation des notes.

Celles-ci ne me paraissant pas bien structurées, j'ai réorganisé leur contenu de façon à m'y retrouver.

Code renvoi	Explications
H02	Valeur **H ou T** : Note on Proxy: this flag provides a coarse indication of the presence of nearby objects within 10arcsec of the given entry. If non-blank, it indicates that 'H' there is one or more distinct Hipparcos Catalogue entries, or distinct components of system from h_dm_com.dat T' there is one or more distinct Tycho entries If 'H' and 'T' apply, 'H' is adopted. The 'T' flag implies either an inconsistency between the Hipparcos and Tycho catalogues, or a deficiency in one or both of the catalogues.
H06	Note on **VarFlag** : the values are 1: < 0.06mag ; 2: 0.06-0.6mag ; 3: >0.6mag
H07	Note on **r_Vmag** : the source is G = ground-based, H=HIP, T=Tycho
H08 et H9	Note on **RAdeg**, **Dedeg** : right ascension and declination are expressed in degrees for epoch J1991.25 (**JD2448349.0625** (TT)) in the ICRS (International Celestial Reference System, consistent with J2000) reference system. There are 263 cases where these fields are missing (no astrometric solution could be found)
H10	Note on AstroRef: this flag indicates that the astrometric parameters in H3-4 and H8-30 refer to: A to Z : the letter indicates the component of a double or multiple system * : the photocentre of a double or multiple system + : the centre of mass
H12 et H13	Note on **pmRA**, **pmDE** : The proper motions refer to the ICRS and to the epoch J1991.25.
H14 et H15	Note on **e_RAdeg**, **e_DEdeg** : The standard errors refer to the epoch J1991.25, and represent a minimum of the error on the position. The actual standard error on the positions is increasing for epochs increasingly differing from the nominal J1991.25 epoch.
H30	Note on F2: values exceeding +3 indicate a bad fit to the data.
H36	Note on m_BTmag: this flag indicates the component or combined photometry: A to Z : the letter indicates the component measured in Tycho (non-single star) * : the photometry refers to all components of the Hipparcos entry - : single-pointing triple or quadruple system
H42	Note on **r_V-I** : the origin of the V-I colour, in summary: 'A' for an observation of V-I in Cousins' system; 'B' to 'K' when V-I derived from measurements in other bands/photoelectric systems 'L' to 'P' when V-I derived from Hipparcos and Star Mapper photometry 'Q' for long-period variables 'R' to 'T' when colours are unknown

H49	Note on **Hpmag , e_Hpmag** : the Hipparcos magnitude could not be determined for 14 stars.
H48	Note on **m_Hpmag** : this flag indicates for double or multiple entries: A to Z : the letter indicates the specified component measured * : combined Hpmag of a double system, corrected for attenuation - : combined Hpmag of a multiple system, not corrected for attenuation
H52	Note on **HvarType** : Hipparcos-defined type of variability (a blank entry signifies that the entry could not be classified as variable or constant): C : no variability detected ("constant") D : duplicity-induced variability M : possibly micro-variable (amplitude < 0.03mag) P : periodic variable R : V-I colour index was revised due to variability analysis U : unsolved variable which does not fall in the other categories
H53	Note on **moreVar** : more data about periodic variability are provided
H56	Note on **n_CCDM** : the flag takes the following values: H : determined multiple by Hipparcos, previously unknown I : system previously identified as multiple in HIC <I/196> (annex1) M : miscellaneous (system identified after publication of HIC)
H59	Note on **MultFlag** : indicates that further details are given in the Double and Multiple Systems Annex: C : solutions for the components G : acceleration or higher order terms O : orbital solutions V : variability-induced movers (apparent motion arises from variability) X : stochastic solution (probably astrometric binaries with short period)
H60	Note on **Source** : qualifies the source of the astrometric parameters H8-30 with a 'C' in MultFlag: P : primary target of a 2- or 3-pointing system F : secondary or tertiary of a 2- or 3-pointing 'fixed' system (common parallax and proper motions) I : secondary or tertiary of a 2- or 3-pointing 'independent' system (no constraints on parallax or proper motions) L : secondary or tertiary of a 2- or 3-pointing 'linear' system (common parallax) S : astrometric parameters from 'single-star merging' process.
H61	Note on **Qual** : Reliability of the double or multiple star solution: A=good, B=fair, C=poor, D=uncertain, S=suspected non-single
H69	Note on **Chart** : the chart was produced: D : from the STScI Digitized Sky Survey G : from the Guide Star Catalog
H70	Note on **Notes** : the flag has the following meaning: D : double and multiple systems note only (note in hd_notes.doc file) G : general note only (note in hg_notes.doc file) P : photometric notes only (note in hp_notes.doc file) W : D + P X : D + G Y : G + P Z : D + G + P

H77	Note on **r_SpType** : the flag indicates the source, as: 1 : Michigan catalogue for the HD stars, vol. 1 (Houk+, 1975) <III/31> 2 : Michigan catalogue for the HD stars, vol. 2 (Houk, 1978) <III/51> 3 : Michigan Catalogue for the HD stars, vol. 3 (Houk, 1982) <III/80> 4 : Michigan Catalogue for the HD stars, vol. 4 (Houk+, 1988) <III/133> G : updated after publication of the HIC <I/196> K : General Catalog of Variable Stars, 4th Ed. (Kholopov+ 1988) <II/214> S : SIMBAD data-base <http://cdsweb.u-strasbg.fr/Simbad.html> X : Miscellaneous A blank entry has no corresponding information.

En astronomie, l'**indice de couleur** d'une étoile désigne la différence entre les magnitudes apparentes obtenue dans deux bandes spectrales différentes. Par exemple, l'indice de couleur *B-V* indique la différence entre la magnitude apparente dans la bande spectrale *B* (c'est-à-dire *bleue*, autour de 436 nm) et la bande spectrale *V* (c'est-à-dire *verte*, autour de 545 nm).

Définition

La magnitude apparente étant directement liée à la distance de l'étoile et à sa luminosité dans la bande considérée, la différence entre les deux magnitudes apparentes fait que le terme de distance disparaît, et ne reste que le rapport effectif des luminosités dans les deux bandes. Ce rapport indique par conséquent la quantité de flux qui est émis dans chaque bande, et donc la couleur.

Il s'agit bien d'un rapport des luminosités car la magnitude est une échelle logarithmique : et donc la différence des logarithmes est égale au logarithme du ratio :

$$B-V \equiv m_B-m_V = -2.5\log(L_B/(4\pi D^2.F_B^0))+2.5\log(L_V/(4\pi D^2.F_V^0)) = 2.5\log\left(\frac{L_V.F_B^0}{L_B.F_V^0}\right)$$

où L désigne la luminosité intrinsèque, dans la bande indiquée, et D la distance de l'objet.F_V^0 et F_B^0 sont des valeurs de normalisation des flux servant à fixer l'origine zéro des magnitudes dans chaque bande : conventionnellement, elles sont choisies de façon à annuler la magnitude de l'étoile Alpha Lyrae (Vega) dans toutes les bandes optiques. L'indice $B-V$ de Vega est donc également zéro par définition.

Cahier des charges

Dans le fichier texte contenant les coordonnées de Vesta, mais aussi dans certains catalogues d'étoiles consultés, les coordonnées sont contenues dans des chaînes de caractères (premier tableau page 1). Premier travail : réaliser deux fonctions qui permettront de transformer ces valeurs en nombres à virgules (flottants).

Conversion à la main pour comprendre (et vérifier)

Dans le programme, on va utiliser l'ascension droite suivante :

```
04 heures 33 minutes de temps et  52.1 secondes de temps.
```

Le tableau ci-dessous nous fournit les correspondances.

Heure en degrés	Minute en degrés	Secondes en degrés
* 15	* 15 / 60 = * 0.25	* 15 / 3600

Soit :

heures	:	4 * 15	=	60
minutes	:	33 * 0.25	=	8,25
secondes	:	(52.1 * 15) / 3600	=	0,21708333

Total : 60 + 8.25 + 0,21708333 = 68,467083333

Remarque : cette vérification est très importante. En effet, elle permet de se rendre compte que le type float est trop court : il ne donne pas le résultat calculé à la main (ce qui est un comble!).

Il faut utiliser un type flottant plus long : long_float

Conversion de la déclinaison

Dans le programme, on va utiliser la déclinaison suivante

```
+19 degrés 38 minutes d'arc  03 secondes d'arc.
```

Le tableau suivant nous donne les correspondances :

Degrés	Minute d'arc en degrés	Secondes d'arc en degrés
On ne fait rien	/ 60	/ 3600

Cela donne :

degrés	:	19	=	19
minutes	:	38 / 60	=	0,633333333
secondes	:	3 / 3600	=	0,000833333
Total : 19 + 0,633333333 + 0,000833333			=	19,6341666...7

Le code source du programme réalisant ce travail.

Il est joint en **annexe 1**.

```
Compte-tenu de ce qui précède, il est assez lisible.
```