# Rapport Écrit

# Projet Final Informatique

**Adrian BULNES**

**David FUENMAYOR**

**Vicky LYSANDRA**

# 1. Introduction

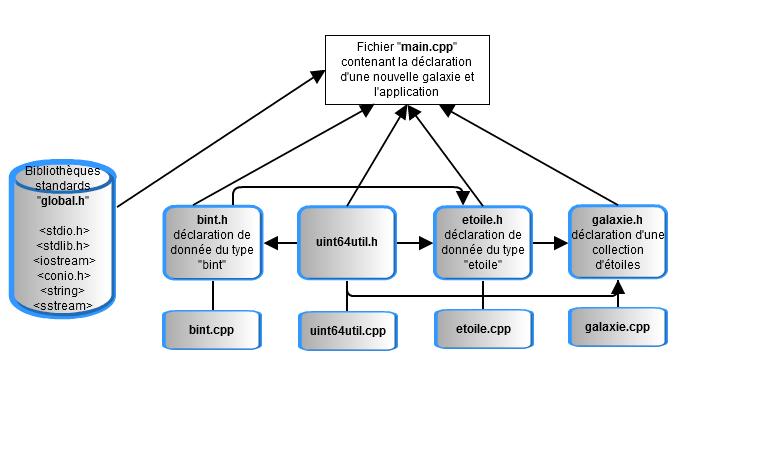
L'objectif principal du projet est de pouvoir traiter des opérations mathématiques avec nombres entiers très grands, de facon à ce qu'on puisse travailler avec une précision  
arbitraire, c'est à dire, ne pas perdre des chiffres dans des nombres très grandes, à cause des limitations technologiques des moyens de calcul.

Un nouveau type de donnée a été créé pour sauvegarder un entier positif de longueur variable en blocs, chacun contenant un nombre entier. Afin d'opérer sur un de ces nombres, des opérations aritmétiques sont éffectués sur chacun de ses blocs composants. On se sert alors de sous-programmes devéloppés pour conduire les opérations aritmétiques de base pour ce type de donnée.

Cet algorithme est ensuite implanté dans la création d'un ensemble appélé "galaxie", qui peut être composé d'un enorme nombre d'étoiles stockées en utilisant une liste chainée et triées à partir de ses distances au centre de la galaxie. En plus, il est aussi possible de modifier les membres d'une galaxie, les ajouter ou les supprimer, selon le cas souhaité, à partir d'un menu principal.

# **2. Développement**

Le projet est composé des modules suivantes:



## **2.1 Fichiers bint.h et bint.cpp**

Un type de donnée appelé "bint" à été créé, qui se compose d'un grand entier et les informations additionnels utiles pour son utilisation.

***//Enregistrement qui represente un entier positive de longueur quelconque (longueur interne dynamique)***

typedef struct bint {

***// bloques (array)***

unsigned long long \*b;

***// longueur. Nombre de bloques utilises***

unsigned int nb;

***// longueur reel. Longeur d'array***

unsigned int internal\_nb;

};

Les sous-programmes utilisés pour traiter "bint" se trouvent dans le fichier "bint.cpp". Là on compte avec les opérations suivantes:

## 2.**1.1. Opérations de base:**

***// creer un bint vide avec la valeur 0***

bint bint\_creerVide();

***// creer bint à partir d'un uint64***

bint bint\_creer(unsigned long long valeur);

***// assigner un uint64 à un bint***

void bint\_assigner(bint &b, unsigned long long n);

***// copier un bint***

bint bint\_copier(const bint &a);

## 2.**1.2. Les fonctions d'affichage**

## ***// Information générale : DEBUG***

void bint\_info(bint &b);

***// Obtient la valeur binaire***

string bint\_getBin(bint &b);

***// Compte le nombre de bits***

unsigned long long bint\_compterBits(bint &b);

***// Retourn une valeur approximative***

string bint\_valEnviron(bint &b);

***// Convert bint en une chaîne***

string bint\_toString(bint &b);

## 2.1.3. Les procédures pour les traitements aritmétiques

***// a=a+b***

void bint\_ajouter(bint &a, int b);

void bint\_ajouter(bint &a, unsigned long long b);

void bint\_ajouter(bint &a, const bint &b);

***// ajoute un nombre en certain position du array***

void bint\_ajouter(bint &a, unsigned long long b, int pos);

***// a=a-b***

void bint\_sustraire(bint &a, int b);

void bint\_sustraire(bint &a, unsigned long long b);

***// sustraire un nombre en certain position du array***

void bint\_sustraire(bint &a, unsigned long long b, int pos);

***// a=a\*b***

void bint\_multiplier(bint &a, unsigned int b);

void bint\_multiplier(bint &a, const bint &b);

***// a=a^b***

void bint\_pow(bint &base, unsigned long long exp);

void bint\_pow(bint &base, const bint &exp);

## 2.1.4. Les fonctions pour faire des comparations entre valeurs

***// est vide or ==0***

bool bint\_estVide(const bint &a);

***// TRUE si a>b. FALSE si a<=b***

bool bint\_plusGrandQue(const bint &a, unsigned long long  b);

bool bint\_plusGrandQue(const bint &a, const bint &b);

***// TRUE si a<b. FALSE si a>=b***

bool bint\_plusPetitQue(const bint &a, unsigned long long  b);

bool bint\_plusPetitQue(const bint &a, const bint &b);

***// TRUE si a==b***

bool bint\_egal(const bint &a, unsigned long long b);

bool bint\_egal(const bint &a, const bint &b);

***// TRUE si a==b et les arrays sont egales aussi***

bool bint\_egalExact(const bint &a, const bint &b);

## 2.1.**5. Changement de valeurs internes du bint**

***// options internes***

***// changer la longeur interieure de bint***

void bint\_setLongInter(bint &a, unsigned int L);

## 2.2 Fichiers etoile.h et etoile.cpp

En utilisant "bint", on a construit un autre enregistrement appelé "etoile" dans le fichier "etoile.h". Où chaque"etoile" posséde l'information suivante:

- La distance au centre de la galaxie

- La taille (rayon)

- Le volume de l'etoile  
  
Dont chaque composant est implementé en tant qu'un type "bint":

***// Un type de donée appele “Etoile”, utilisee pour montrer l'utilisation du type “bint”***

typedef struct etoile {

***// distance au centre de la galaxie (10^12 km)***

bint distCentGal;

***// rayon (radius) de l'etoile (10^3 km)***

bint rayon;

***// volume de l'etoile (10^3 km^3)***

bint volume;

};

En tant que fonctions pour la création et le traitement d'une étoile, l'on compte sur:

***// Cree une etoile a partir de la distance au centre du galaxie et son rayon***

etoile etoile\_creer(unsigned long long distanceCentreGalaxie, unsigned long long rayon);

***// creer une etoile a partir de la distance au centre du galaxie et son rayon***

etoile etoile\_creer(bint distanceCentreGalaxie, bint rayon);

***// creer une etoile a partir de la distance au centre du galaxie, son rayon et volume***

etoile etoile\_creer(bint distanceCentreGalaxie, bint rayon, bint volume);

***// creer une etoile de distance au centre du galaxie et rayon aleatoires, et calculer le volume a partir du rayon***

etoile etoile\_creerRnd();

***// obtenir etoile sur le clavier (demander distance au centre du galaxie et rayon. Calculer volume a partir du rayon)***

etoile etoile\_demander();

***// information generale de l'etoile: DEBUG***

void etoile\_info(etoile &e, string teteCout = "");

## 2.3 Fichiers galaxie.h et galaxie.cpp

Afin de mieux illustrer l'utilisation du type de données “bint”, une collection d'étoiles appelée “galaxie” a été ajoutée :

***// Collection d'etoiles***

typedef struct nodeEtoile\* collEtoiles;

***// Node de la collection comportant une etoile et le pointeur au prochain node***

typedef struct nodeEtoile {

***// l'etoile***

etoile etoile;

***// pointeur a l'etoile prochaine (et aux autres etoiles moins proches)***

collEtoiles suivNodeEtoile;

};

En tant que fonctions de base pour la création et le traitement d'une galaxie (collection d'etoiles), l'on compte dans le fichier ***galaxie.cpp*** avec l'implementation des functions suivantes:

***// Cree collEtoiles vide***

collEtoiles collEtoiles\_creerVide();

***// Cree pointeur au nouveau node de la collection d'etoiles***

nodeEtoile\* nodeEtoile\_creer(etoile &e);

***// Affiche l'information de la collection d'etoiles***

void collEtoiles\_afficherInfo(collEtoiles &c);

***// Affiche tous les etoiles de la collection***

void collEtoiles\_afficherToutes(collEtoiles &c);

***// Affiche le nombre d'etoiles dans la collection***

void collEtoiles\_afficherQuantEtoiles(collEtoiles &c);

***// Compte le nombre d'etoiles en la collection***

unsigned int collEtoiles\_quantiteEtoiles(collEtoiles &c);

***// Ajoute une etoile a la collection***

void collEtoiles\_ajouterEtoile(collEtoiles &c, etoile e);

***// Ajoute en tete une etoile a la collection***

void collEtoiles\_ajouteEnTete(collEtoiles &c, etoile e);

***// Obtient la derniere etoile de la collection***

etoile collEtoiles\_derniereEtoile(collEtoiles &c);

***// Obtient la plus grande etoile de la collection***

etoile collEtoiles\_plusGrandEtoile(collEtoiles &c);

***// Calcule le volume total de la galaxie***

bint collEtoiles\_volumeTotal(collEtoiles &c);

***// Obtient une etoile a partir de son ID***

etoile collEtoiles\_getEtoile(collEtoiles &c, unsigned long long id);

***// Supprime une etoile de la collection***

void collEtoiles\_supprimerEtoile(collEtoiles &c, unsigned long long id);

## 2.4 Sous-programmes à detailler: 2.4.1. bint.cpp

## ***void bint\_ajouter(bint &a, unsigned long long b, int pos){***

***// si l'array interne est tres petit  
if(pos>a.internal\_nb-1) bint\_setLongInter(a, pos+1);  
  
unsigned long long antVal = a.b[pos];  
a.b[pos]+=b;  
  
// si la pos predec pas utilisee  
if(pos > ((signed long)a.nb)-1) a.nb=pos+1;  
  
// si surcharge/overload  
if(a.b[pos]<antVal) bint\_ajouter(a, 1, pos+1);***

***}***

## 2.4.2 etoile.cpp

etoile etoile\_creerRnd() {

unsigned long long rayon = uint64Rnd(500, 100000);

if(rayon < 50000) return etoile\_creer(uint64Rnd(10000, 50000000), rayon);

else {

***// volume ensin rayon, parce-que il est tres grand***

unsigned long long diff = uint64Rnd(100, uint64MaxValue);

bint vol = bint\_creer(uint64MaxValue);

bint\_ajouter(vol, diff);

bint\_ajouter(vol, diff);

return etoile\_creer(bint\_creer(uint64Rnd(10000, 50000000)), bint\_creer(0), vol);

}

}

## 2.4.3. galaxie.cpp

void collEtoiles\_ajouterEtoile(collEtoiles &c, etoile e) {

nodeEtoile\* nouveauNodeEtoile = nodeEtoile\_creer(e); if(c==null) c=nouveauNodeEtoile;

else {

if(bint\_plusGrandQue(c->etoile.distCentGal, nouveauNodeEtoile->etoile.distCentGal)) {

***// le nouveau est plus proche que le node en tete***

collEtoiles\_ajouteEnTete(c, e);

} else {

***// le node en tete es plus proche que le nouveau***

collEtoiles plusProcheNode = c;

while(plusProcheNode->suivNodeEtoile!=null && bint\_plusGrandQue(nouveauNodeEtoile- >etoile.distCentGal, plusProcheNode->suivNodeEtoile- >etoile.distCentGal)){

plusProcheNode=plusProcheNode->suivNodeEtoile;

}

nouveauNodeEtoile->suivNodeEtoile=plusProcheNode- >suivNodeEtoile;

plusProcheNode->suivNodeEtoile=nouveauNodeEtoile;

}

}

}

### 3. Exemple d'utilisation

## 3.1 Affichage du menu principal

## ***Le menu 1 sert à afficher l'information de la galaxie, qui consiste en***

***- La quantité d'étoiles dans la galaxie.***

***- La taille (la distance du centre de la galaxie à l'étoile la plus éloignée)***

***- L'information sur l'étoile la plus grand de la galaxie***

***- Le volume total de la galaxie (la somme totale de tous les volumes des étoiles dans la galaxie)***

Au début de l’exécution du programme, il n y a pas d'étoiles.

Le menu 2 ***sert à*** afficher les informations des étoiles dans la galaxie, automatiquement triées en ordre croissant basée sur la distance au centre de la galaxie.

Le menu 3 ***sert à*** afficher les informations d'une étoile de nombre donnée.

Le menu 4 ***sert à*** ajouter et afficher une étoile de distance et rayon aléatoire et calculer son volume.

Le menu 5 ***sert à*** ajouter une étoile de la distance et rayon précisés par l'utilisateur.

Le menu 6 ***sert à*** ajouter et afficher un nombre des étoiles définis avec la distance et rayon aléatoire.

Le menu 7 ***sert à*** supprimer une étoile de nombre donnée.

## *4. Conclusion*

Ce projet sert comme un exemple d'implementation des structures de données telles que enregistrements et listes chaînées pour aplications démandant l'utilisation des nombres entiers très grandes, dont la capacité de calcul des ordinateurs actuels ne permett pas son traitement en tant que types de données simples.