

Gnebehi Bagre ET Perrot Josuah

Compte rendu du TP structure de donnée (Type Set et Bag)

Objectif du TP

L'objectif de ce TP est de fournir une implémentation prouvée (par application une Vérification Formelle) pour les types de structures ensemblistes les plus utilisées en ingénierie du logiciel à savoir, les Set et les Bag.

La méthode qui sera appliquée s'appuie sur les spécifications Casl proposées en cours.

La démarche doit obligatoirement dérouler en suivant les 5 dernières phases du cycle de développement , à savoir :

Phase 1 : Spécification Casl du type abstrait (elle est fournie et doit être éditée sous emacs)

Phase 2 : Validation de la spécification sous Hets Casl.

Phase 3 : Spécification des opérations du type

Phase 4 : Implémentation des opérations du type

Phase 5 : Validation de l'implémentation

C'est le langage C++ qui a été choisi pour implémenter la spécification Casl.

I : Spécification du type abstrait Set et Bag

[illegible]

%%Spécification canonique du type abstrait des bag

spec Bag0[sort Elem] given Int =

generated type Bag[Elem] ::= **bagVide** |
 ajouter(Bag[Elem]; Elem)

op

frequence : Bag[Elem] * Elem -> Int

forall x, y: Elem; M, N: Bag[Elem]

. **frequence**(bagVide, y) = 0

. **frequence**(ajouter(M,x), y) = 1+**frequence**(M, y) when x = y else
frequence(M, y)

. M = N <=> forall x: Elem . **frequence**(M, x) = **frequence**(N, x)

end

%%Spécification enrichie du type abstrait des bag

spec Bag [sort Elem] given Int =

 Bag0 [sort Elem]

then

preds

estVide : Bag[Elem];

appartient : Elem * Bag[Elem];

inclut: Bag[Elem] * Bag[Elem]

ops

enlever: Bag[Elem] * Elem -> Bag[Elem]

forall x,y:Elem; M,N:Bag[Elem]

. **estVide** (M) <=> M = bagVide

. **appartient**(x ,M) <=> **frequence**(M, x) > 0

. **inclut**(M,N) <=> forall x : Elem . **frequence**(M, x) <= **frequence**(N, x)

. **enlever**(M, x) = N <=>

 forall y: Elem. (freq(N, y)= **frequence**(M,x)-1 if x=y) /\
 (**frequence**(N,y)=**frequence**(M,y) if not(x =

y))

end

II: Validation de la spécification

Il existe des outils pour valider les spécifications formelles.

Il nous a été présenté en cours et TP la commande hets (Heterogeneous Tool Set).

Afin de lancer cette analyse de spécification hets il est nécessaire d'utiliser la

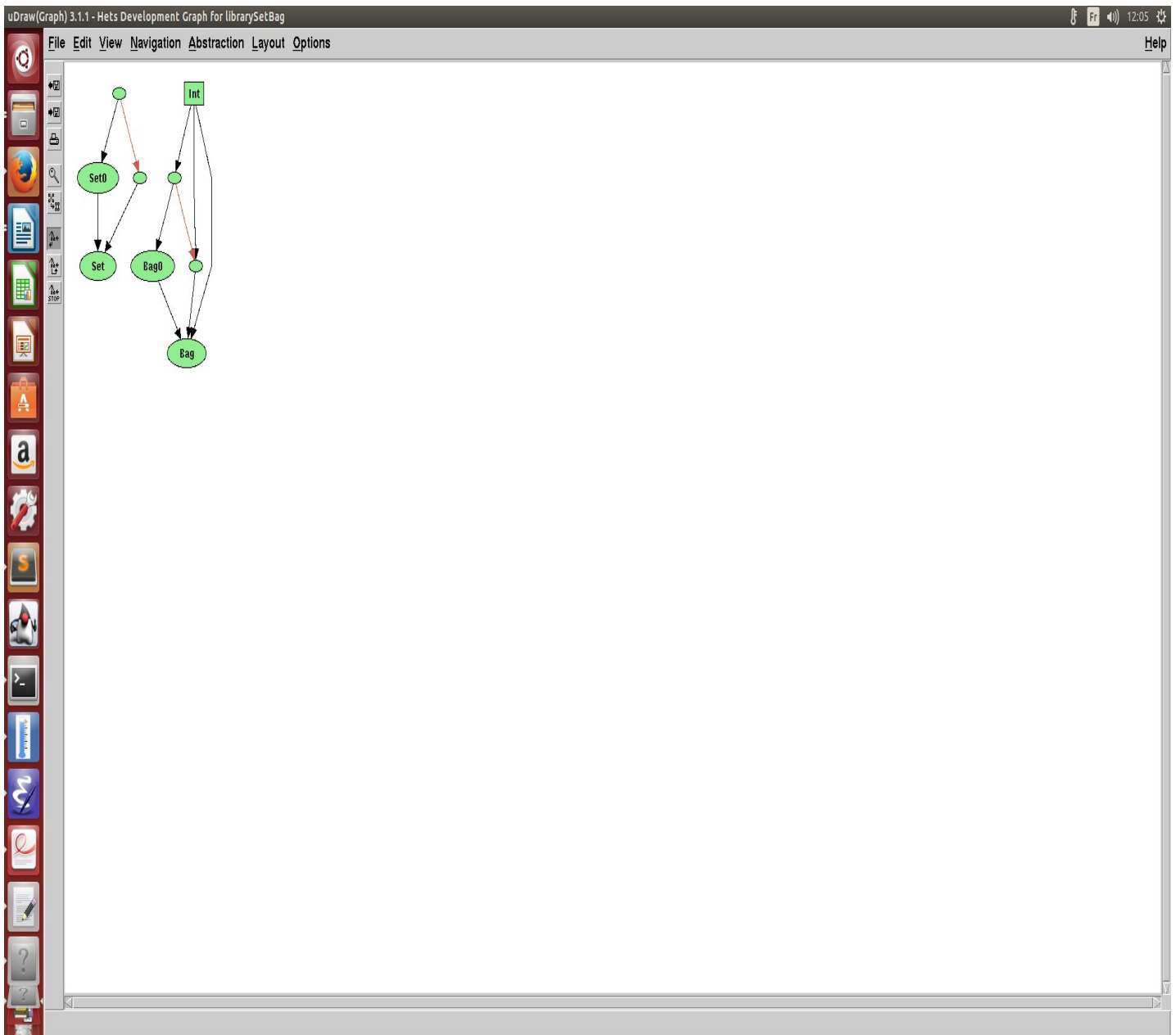
commande suivante via un terminal :

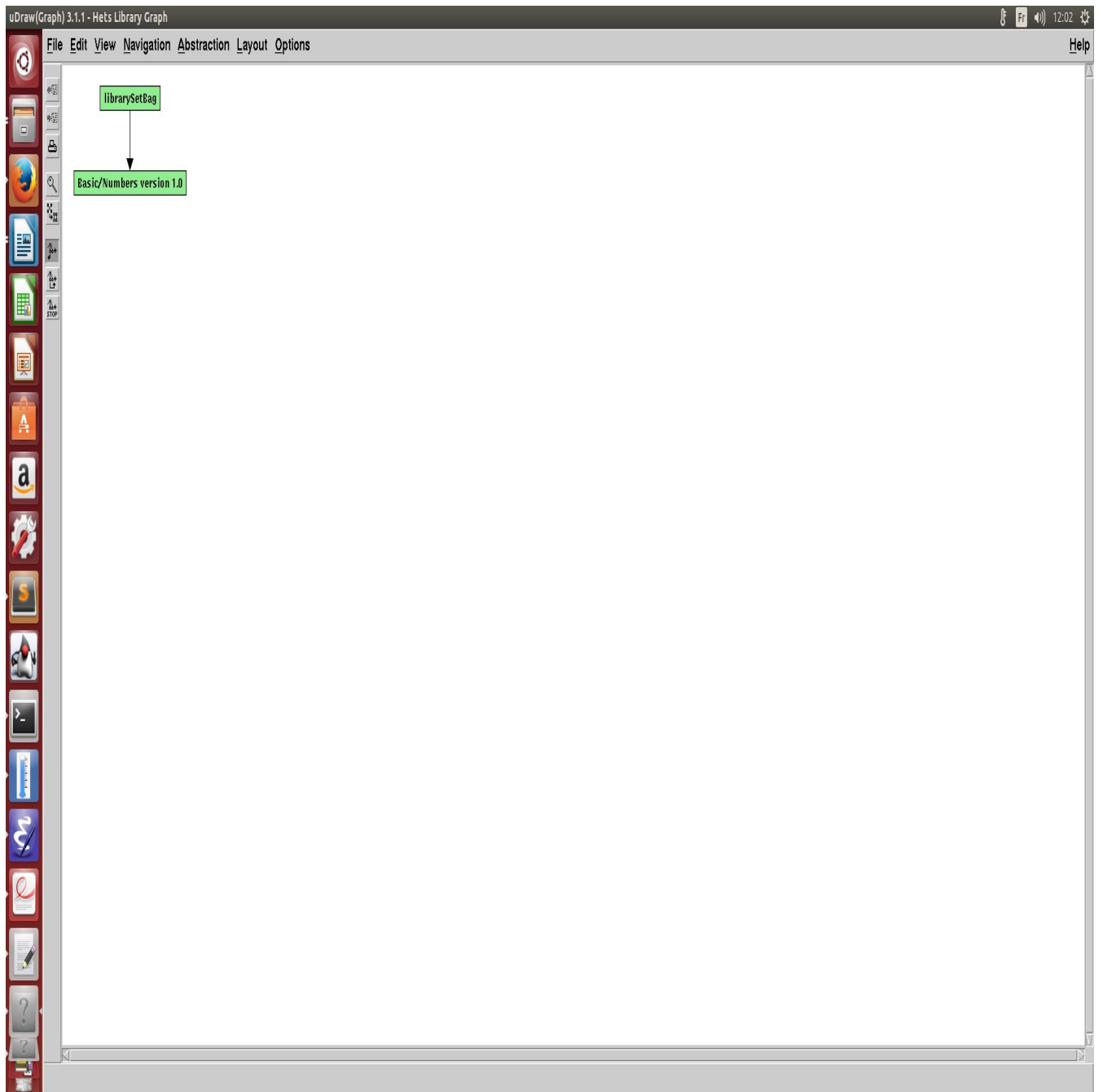
```
hets -g librarySetBag112016.casl
```

où librarySetBag112016 consultable sur le logiciel EMACS, est le fichier contenant le composant

librarySetBag112016.

On obtient les graphes ci dessous :





III- Spécification des opérations du type

Il s'agit ici de définir les observateurs et les constructeurs
En bleu l'on observe les constructeurs et en vert les observateurs.

I- Type abstrait Set

[illegible]

II - Type abstrait Bag

%%Spécification canonique du type abstrait des bag

```
spec Bag0 [sort Elem] given Int =
generated type Bag[Elem] ::= bagVide |
                                ajouter(Bag[Elem]; Elem)
    op
        frequence : Bag[Elem] * Elem -> Int

forall x, y: Elem; M, N: Bag[Elem]
. frequence(bagVide, y) = 0
. frequence(ajouter(M,x), y) = 1+frequence(M, y) when x = y else
  frequence(M, y)
. M = N <=> forall x: Elem . frequence(M, x ) = frequence(N, x )
end
```

%%Spécification enrichie du type abstrait des bag

```
spec Bag [sort Elem] given Int =
  Bag0 [sort Elem]
then
preds
  estVide : Bag[Elem];
  appartient : Elem * Bag[Elem];
  inclut: Bag[Elem] * Bag[Elem]
ops
  enlever: Bag[Elem] * Elem -> Bag[Elem]

forall x,y:Elem; M,N:Bag[Elem]

. estVide (M) <=> M = bagVide
. appartient(x ,M) <=> frequence(M, x ) > 0
. inclut(M,N) <=> forall x : Elem . frequence(M, x ) <= frequence(N, x )

. enlever(M, x) = N <=>
    forall y: Elem. (freq(N, y)= frequence(M,x)-1 if x=y) /\
                    (frequence(N,y)=frequence(M,y) if not(x =
y) )
```

IV - Implémentation

On crée le fichier Bag.h et Set.h qui comprend :

- la définition d'une classe librarysetbag,
- la déclaration, dans cette classe, des constructeurs et des observateurs du type en tant que méthodes de la classe

```
~/Bureau/stockage/Structure de Données/TypeBag/Bag.h - Sublime Text (UNREGISTERED) 11:57

Bag.cpp  Bag.h  main.cpp

1 #define TAILLE_MAX 100
2 #include <iostream>
3 #include <cstdlib>
4
5 struct Bag{
6     int elements[TAILLE_MAX];
7     int nbElements;
8 };
9
10 // Définition des Méthodes
11
12
13
14
15 // Retourne "true" ou "false", permet de savoir si Bag est vide ou pas
16 bool bagVide(Bag unBag);
17 //VALIDE
18
19 //Retourne si un element appartient à l'ensemble
20 bool appartient(Bag unBag , int unElement);
21 //VALIDE
22
23
24 // Affiche le contenu de Bag
25 void afficherBag(Bag unBag);
26 //VALIDE
27
28 // Retourne la fréquence d'éléments de Bag
29 int frequence(Bag unBag, int unElement);
30 //VALIDE
31
32 // Retourne la taille de l'ensemble Bag
33 int obtenirTaille(Bag unBag);
34 //VALIDE
35
36 // Ajouter un element dans Bag
37 Bag ajouter(Bag unBag, int unePosition , int unElement);
38 //VALIDE
39
40 // Supprime un element dans bag
41 Bag enlever(Bag unBag, int unePosition);
42 //VALIDE
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```



On crée le fichier Bag.cpp et Set.cpp
qui contient l'implémentation des fonctions et des procédures

Type Bag

```
1 #include "Bag.h"
2 #include <iostream>
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <cstdlib>
6 #define TAILLE_MAX 100
7
8 using namespace std;
9
10 // Retourne "true" ou "false", permet de savoir si Bag est vide c
11 bool bagVide(Bag unBag){
12     if(unBag.nbElements == 0){
13         return(true);
14     }
15     else{
16         return(false);
17     }
18 }
19 }
```

```
20 //Retourne si un élément appartient à l'ensemble
21 bool appartient(Bag unBag, int unElement){
22     int i;
23     for (i = 0; i < unBag.nbElements; i++)
24     {
25         if (unBag.elements[i] == unElement){
26             return(true);
27         }
28         else{
29             return(false);
30         }
31     }
32 }
33 }
```

```
34 // Affiche le contenu de l'ensemble Bag
35 void afficherBag(Bag unBag){
36     int i;
37     cout << "" << endl;
38     for(i = 0; i < unBag.nbElements; i++){
39         cout << unBag.elements[i] << endl;
40     }
41     cout << "" << endl;
42 }
43
44
45 }
```

```
44
45
46 // Retourne la frequence d'éléments de Bag
47 int frequence(Bag unBag , int unElement){
48     int frequence = 0;
49     int i;
50     for(i=0;i<unBag.nbElements;i++)
51     {
52         if (unBag.elements[i] == unElement){
53             frequence = frequence +1 ;
54         }
55     }
56     return(frequence);
57 }
```

```
59
60
61
62
63 // Retourne la taille de l'emsemble Bag
64 int obtenirTaille(Bag unBag){
65     cout<< "L'emsemble Bag contient : "<<
66     cout<< " "<< endl;
67     return unBag.nbElements;
68 }
69
70
71
```

```
74
75 // Ajouter un element dans Bag
76 Bag ajouter(Bag unBag, int unePosition , int unElement){
77     int i;
78
79     if((unBag.nbElements < TAILLE_MAX) && (unePosition > 0) && (unePo
80         for(i = unBag.nbElements; i >= unePosition; i--){
81             unBag.elements[i] = unBag.elements[i-1];
82         }
83
84         unBag.elements[unePosition-1] = unElement;
85         unBag.nbElements = unBag.nbElements+1;
86
87         return(unBag);
88     }
89     else{
90         cout << "Erreur dans le programme lors de l'ajout !" << endl;
91         return(unBag);
92     }
93 }
```

```
~/Bureau/stockage/Structure de Données/TypeBag/Bag.cpp - Sublime Text (UNREGISTERED)
Bag.cpp
main.cpp
85
86
87 // Supprime un element dans bag
88 Bag enlever(Bag unBag, int unePosition){
89     int i;
90
91     //Si l'élément à supprimer est à la première position
92     for(i = unePosition-1 ; i != obtenirTaille(unBag) ; i++){
93         unBag.elements[i] = unBag.elements[i+1];
94     }
95     unBag.nbElements = unBag.nbElements-1;
96     return(unBag);
97 }
98
```

Recherchez sur le Web...

E:\makefile - Sublime Text (UNREGISTERED)

```
File Edit Selection Find View Goto Tools Project Preferences Help
set.cpp
makefile
1 monProgrammeBag : Bag.o main.o
2   g++ -o Programme Bag.o main.o
3
4 Bag.o : Bag.h Bag.cpp
5   g++ -c Bag.cpp
6
7 main.o : Bag.h main.cpp
8   g++ -c main.cpp
```

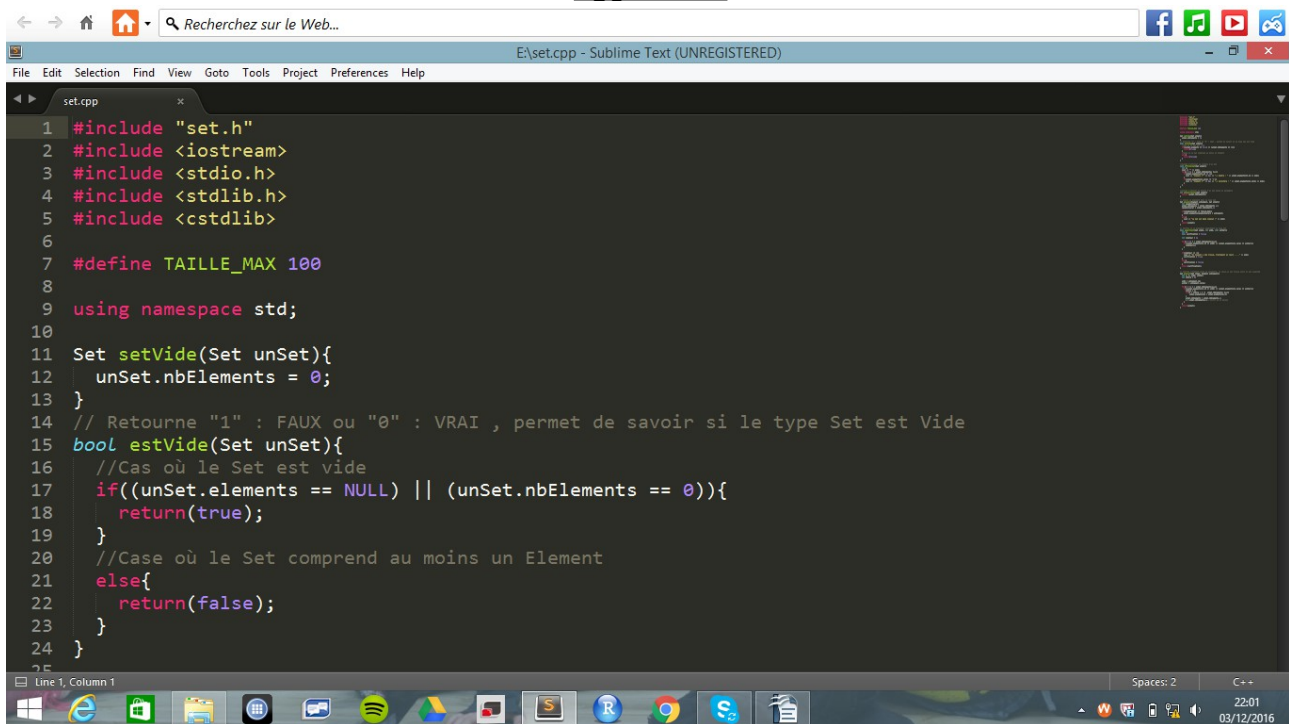
Line 1, Column 1

Tab Size: 4

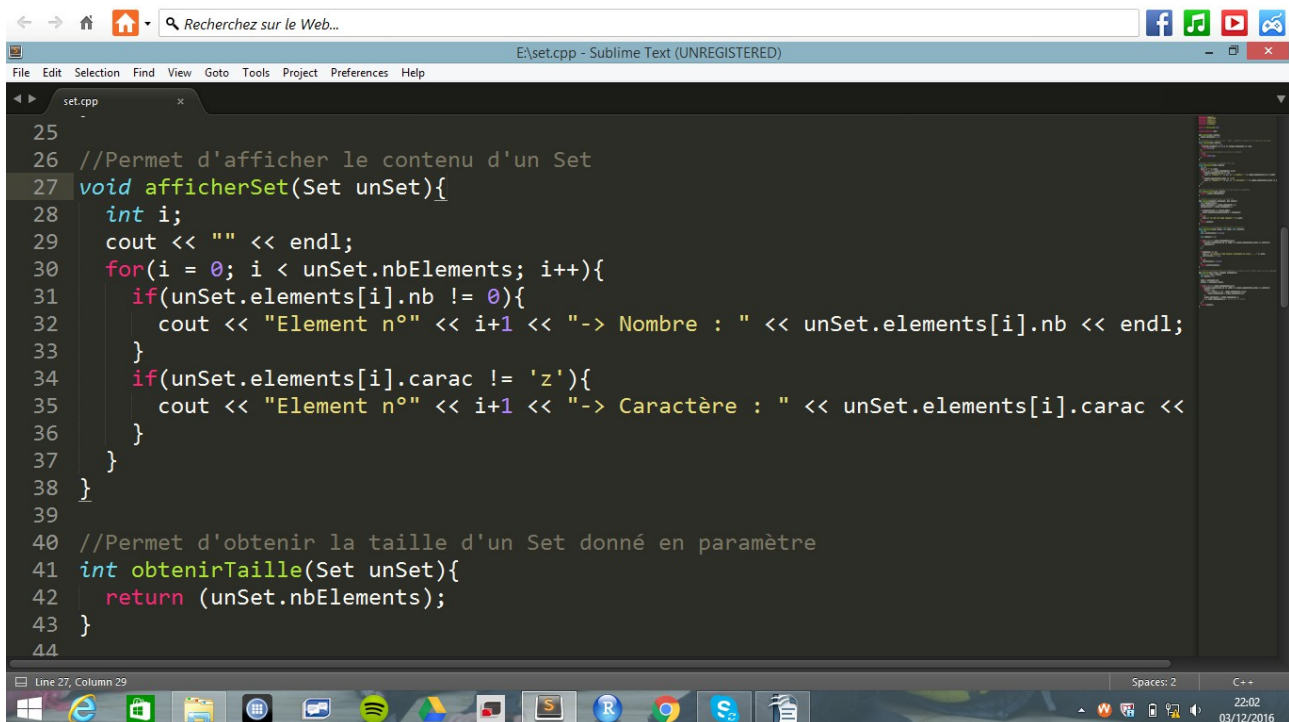
Makefile

22:10 03/12/2016

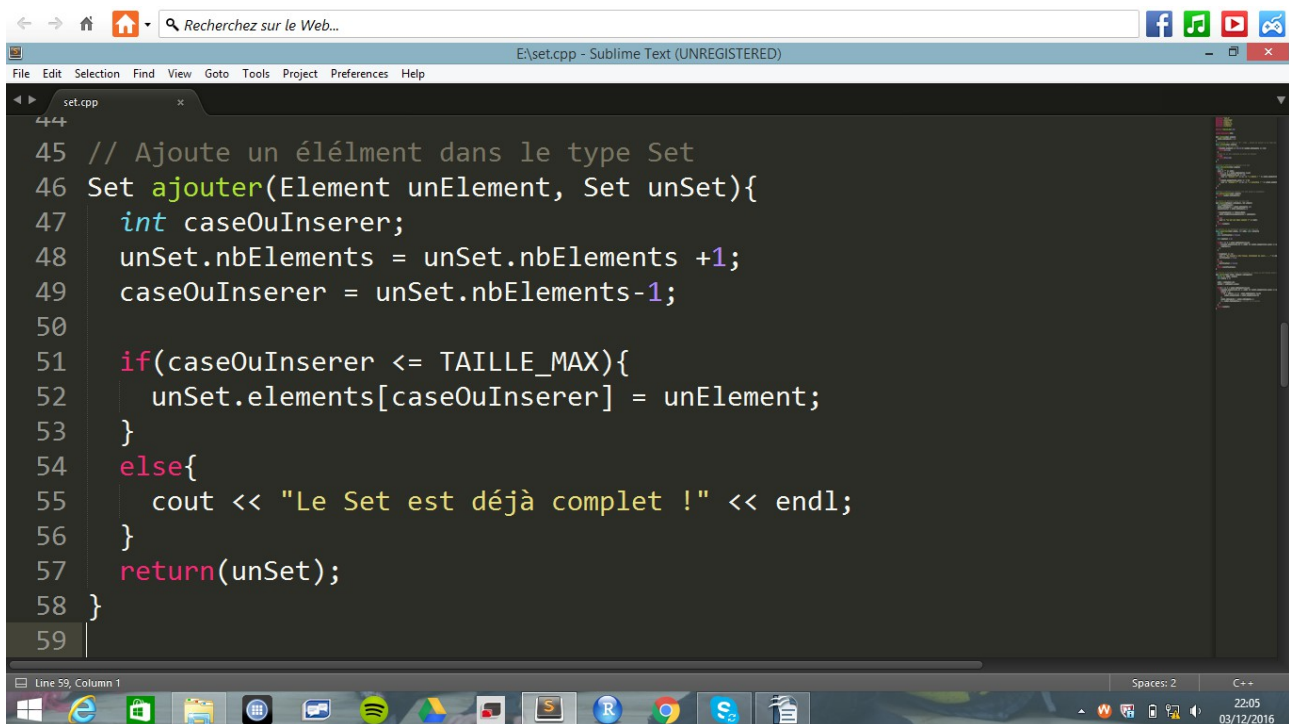
Type Set



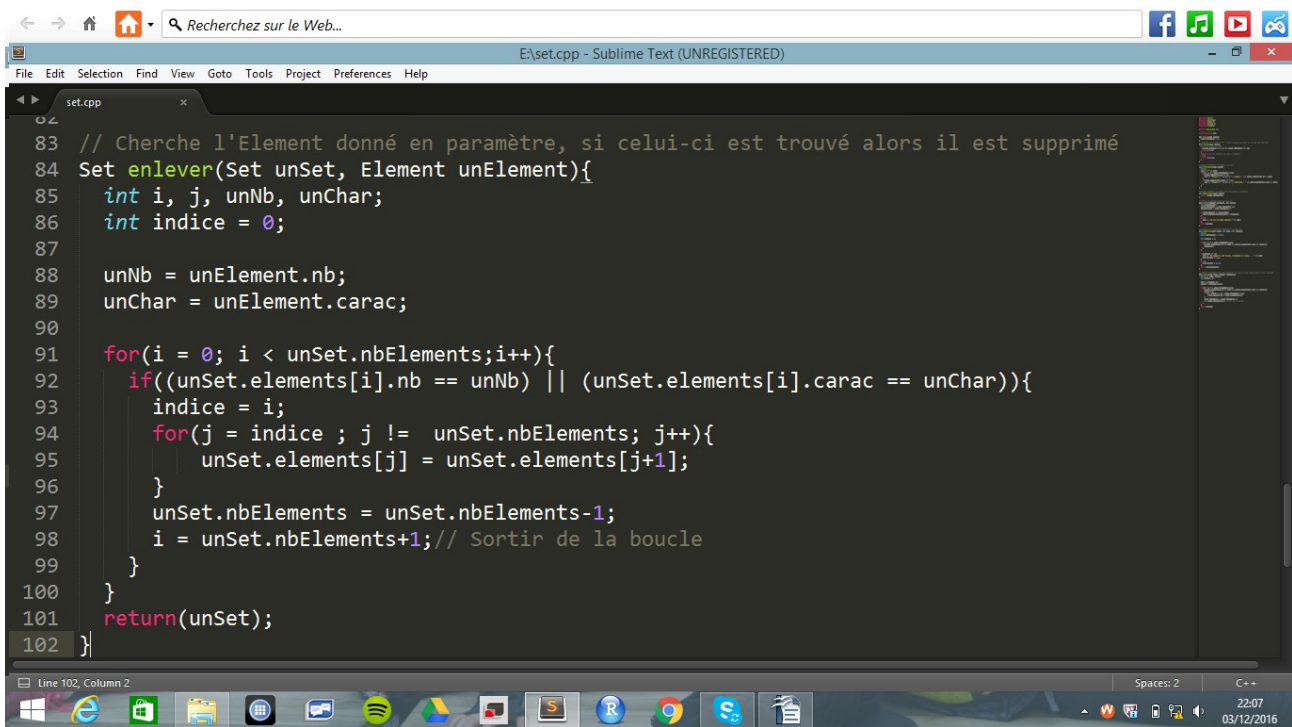
```
1 #include "set.h"
2 #include <iostream>
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <cstdlib>
6
7 #define TAILLE_MAX 100
8
9 using namespace std;
10
11 Set setVide(Set unSet){
12     unSet.nbElements = 0;
13 }
14 // Retourne "1" : FAUX ou "0" : VRAI , permet de savoir si le type Set est Vide
15 bool estVide(Set unSet){
16     //Cas où le Set est vide
17     if((unSet.elements == NULL) || (unSet.nbElements == 0)){
18         return(true);
19     }
20     //Case où le Set comprend au moins un Element
21     else{
22         return(false);
23     }
24 }
25
```



```
25
26 //Permet d'afficher le contenu d'un Set
27 void afficherSet(Set unSet){
28     int i;
29     cout << "" << endl;
30     for(i = 0; i < unSet.nbElements; i++){
31         if(unSet.elements[i].nb != 0){
32             cout << "Element n°" << i+1 << "-> Nombre : " << unSet.elements[i].nb << endl;
33         }
34         if(unSet.elements[i].carac != 'z'){
35             cout << "Element n°" << i+1 << "-> Caractère : " << unSet.elements[i].carac << endl;
36         }
37     }
38 }
39
40 //Permet d'obtenir la taille d'un Set donné en paramètre
41 int obtenirTaille(Set unSet){
42     return (unSet.nbElements);
43 }
44
```

```
45 // Ajoute un élément dans le type Set
46 Set ajouter(Element unElement, Set unSet){
47     int caseOuInserer;
48     unSet.nbElements = unSet.nbElements +1;
49     caseOuInserer = unSet.nbElements-1;
50
51     if(caseOuInserer <= TAILLE_MAX){
52         unSet.elements[caseOuInserer] = unElement;
53     }
54     else{
55         cout << "Le Set est déjà complet !" << endl;
56     }
57     return(unSet);
58 }
59
```



```
83 // Cherche l'Element donné en paramètre, si celui-ci est trouvé alors il est supprimé
84 Set enlever(Set unSet, Element unElement){
85     int i, j, unNb, unChar;
86     int indice = 0;
87
88     unNb = unElement.nb;
89     unChar = unElement.carac;
90
91     for(i = 0; i < unSet.nbElements; i++){
92         if((unSet.elements[i].nb == unNb) || (unSet.elements[i].carac == unChar)){
93             indice = i;
94             for(j = indice; j != unSet.nbElements; j++){
95                 unSet.elements[j] = unSet.elements[j+1];
96             }
97             unSet.nbElements = unSet.nbElements-1;
98             i = unSet.nbElements+1; // Sortir de la boucle
99         }
100     }
101     return(unSet);
102 }
```

V-Vérification de l'implémentation

Type Bag

```
#include "Bag.h"
#include <iostream>
#include <stdio.h>
#include <cstdlib>

using namespace std;

int main(){
    int i,tailleBag,taille;
    Bag monBag;

    //Déclaration d'un Vecteur vide
    monBag.nbElements = 0;

    //Verification de estVide (doit renvoyer true)
    if(bagVide(monBag) == true){
        cout << "L'ensemble Bag est vide (initialement) ,
verification correcte de 'bagVide()' " <<endl;
        cout<< " " << endl;
    }
    else{
        cout << "Erreur dans la vérification n°1" << endl;
        cout<< " " << endl;
    }

    //L'emsemnle Bag va contenir n elements
    cout << "Combien de cases souhaitez-vous que cet ensemble
contienne ?" << endl;
    cin >> monBag.nbElements;

    //Vérification :    frequence()
    cout<< "Appel de 'frequence()' & vérification . . ." << endl;
    cout<< "L'ensemble Bag contient : " << monBag.nbElements << "
elements" << endl;
    cout<< " " << endl;
    tailleBag = obtenirTaille(monBag);

    //L'utilisateur va saisir manuellement les 10 valeurs
    for (i=0 ; i < tailleBag ; i++){
        cout << "Saisir une valeur pour la case " << i+1 << " ."
<<endl;
        cin >> monBag.elements[i];
    }
```

```

//Seconde vérification de estVide (doit renvoyer false)
if(bagVide(monBag) == true){
    cout << "Erreur dans la vérification n°2" << endl;
    cout << " " << endl;
}
else{
    cout << "L'ensemble Bag n'est pas vide (complété
précédemment), vérification correcte de 'bagVide()' " << endl;
    cout<< " " << endl;
}

//Affichage
cout << "Affichage du contenu de l'ensemble : " << endl;
afficherBag(monBag);
system("pause");
system("clear"); //Effacer l'affichage des instructions
précédente dans la console

//Vérification :frequence()
int element;
cout << "Donner l'élément dont vous souhaitez connaitre la
Fréquence " << endl;
cin >> element;
taille = frequence(monBag , element);
cout << "La Fréquence est " << taille << " " << endl;

//Vérification appartient()
int autreElement;
cout << "Donner l'élément dont vous souhaitez savoir s'il
appartient ou non à l'ensemble " << endl;
cin >> autreElement;
if(appartient(monBag, autreElement) == true){
    cout << "L'élément saisi appartient à l'ensemble" << endl;
    cout<< " " << endl;
}
else{
    cout << "Erreur dans la vérification n°1" << endl;
    cout<< " " << endl;
}

//Vérification : ajout()
int maPosition;
int monElement;

cout << "Saisissez un nouvel élément à ajouter : " << endl;
cin >> monElement;

```

```

        cout << "A quelle position souhaitez vous insérer ce nouvel
élément ?" << endl;
        cin >> maPosition;

        monBag = ajouter(monBag, maPosition ,monElement);

        //Affichage de l'emsemble : Vérification de l'Element ajouté
        cout << "Affichage de l'emsemble après l'ajout" << endl;
        afficherBag(monBag);
        cout << "Taille de l'emsemble : " << obtenirTaille(monBag) <<
endl;
        cout << "" << endl;

        //Vérification de : enlever()
        cout << "A quelle position souhaitez vous enlever la valeur ?
(verification enlever() )" << endl;
        cin >> maPosition;
        monBag = enlever(monBag , maPosition);
        afficherBag(monBag);
        cout<< "L'emsemble Bag contient : "<< monBag.nbElements << "
elements" << endl;

    }

```

Type Set

```

#include "set.h"
#include <iostream>
#include <stdio.h>
#include <cstdlib>

using namespace std;

int main(){
    int i,tailleSet;
    Set monSet;

    //Déclaration d'un Vecteur vide
    setVide(monSet);

    //Verification de estVide (doit renvoyer true)
    if(estVide(monSet) == true){
        cout << "Le Set est vide (initialement) , verification
correcte de 'estVide()' " <<endl;
        cout<< " " << endl;
    }
    else{
        cout << "Erreur dans la vérification n°1" << endl;
        cout<< " " << endl;
    }

    //Le tableau va contenir n elements

```



```

    cout << "Combien de cases souhaitez-vous que ce vecteur
contienne ?" << endl;
    cin >> monSet.nbElements;

    //Vérification : obtenirTaille()
    cout<< "Appel de 'obtenirTaille()' & vérification . . ."<<
endl;
    cout<< "Le Set contient : " << monSet.nbElements << "
elements" << endl;
    cout<< " " << endl;
    tailleSet = obtenirTaille(monSet);

char charAAjouter;
int nbAAjouter;

    bool verifAppartenir;

    for(i=0; i< monSet.nbElements; i++){
        verifAppartenir = false;
        cout << "(Laissez 0 pour rien) Saisir un nombre pour
l'Element à la case " << i+1 << " ." << endl;
        cin >> nbAAjouter;

        cout << "Saisir un caractère pour l'Element à la
case " << i+1 << " ." << endl;
        cin >> charAAjouter;

        verifAppartenir = appartient(monSet, nbAAjouter,
charAAjouter);
        if(verifAppartenir == true){
            i--;
        }
        else{
            monSet.elements[i].nb = nbAAjouter;
            monSet.elements[i].carac = charAAjouter;
        }
    }

    //Seconde vérification de estVide (doit renvoyer false)
    if(estVide(monSet) == true){
        cout << "Erreur dans la vérification n°2" << endl;
        cout << " " << endl;
    }
    else{
        cout << "Le Set n'est pas vide (complété précédement),
vérification correcte de 'estVide()' " <<endl;
        cout<< " " << endl;
    }

    //Affichage
    cout << "Affichage du contenu du Vecteur : " << endl;
    afficherSet(monSet);

```

```

//Vérification : ajouter()
Element monElement;

    cout << "Saisissez un nombre pour le nouvel élément à
ajouter : " << endl;
    cin >> monElement.nb;
    cout << "Saisissez une caractère pour le nouvel élément à
ajouter : " << endl;
    cin >> monElement.carac;

    verifAppartenir = false;
    verifAppartenir = appartient(monSet, monElement.nb,
monElement.carac);
    if(verifAppartenir == false){
        monSet = ajouter(monElement, monSet);
    }
    else{
        cout << "L'element existe déjà dans le Set !" << endl;
    }

//Affichage du Set : Vérification de l'Element inséré
cout << "Affichage du tableau après insertion" << endl;
afficherSet(monSet);
cout << "Taille : " << obtenirTaille(monSet) << endl;
cout << "" << endl;

//Vérification de : enlever()
    cout << "Quel Element souhaitez-vous supprimer ?" <<
endl;
    cout << "Saisissez un nombre pour l'Element à rechercher
puis supprimer : " << endl;
    cin >> monElement.nb;
    cout << "Saisissez désormais le caractère : " << endl;
    cin >> monElement.carac;

    verifAppartenir = false;
    //Boucle tant que l'Element est trouvé dans le SET pour
supprimer toutes les occurences et non une seule
    do{
        verifAppartenir = appartient(monSet, monElement.nb,
monElement.carac);
        if(verifAppartenir == true){
            monSet = enlever(monSet,monElement);
        }
    }while(verifAppartenir == true);
    afficherSet(monSet);
    cout<< "Le vecteur contient : "<< monSet.nbElements << "
elements" << endl;

    system("pause");
    return(0);

```

```
}
```

Conclusion

Nous avons réappris dans ce Tp à manipuler des Tableaux dans toute leur règle .

L'utilisation de la commande **hets** nous a permis de mieux comprendre le graphe généré .