

Génie Logiciel TD n° 1, 2

Makefile, Jeux de tests

Objectifs du TD

Ce(s) TD, premier(s) de l'UE, est une sorte de révision sur C++. Ses deux objectifs sont :

- Savoir faire un `Makefile`.
- Comprendre le but d'un jeu de tests.

En annexe figure le programme C++ Chimie permettant de créer, associer, dissocier et détruire atomes et molécules. Le but des questions suivantes est d'écrire le `Makefile`, de **tester** et **corriger** ce programme.

Question 1

Le programme Chimie est structuré en fichiers `.h` et `.cpp` : `chimie.cpp`, `atome.cpp`, `entite.cpp`, `molécule.cpp`, `liste.cpp`, `atome.h`, `entite.h`, `molécule.h` et `liste.h`.

- L'annexe ne donne pas les instructions de compilation à mettre au début de chaque `.h`. Quelle instruction de compilation doit-on mettre au début de chaque fichier `.h` ?
- Quelle est la différence entre `#include "truc.h"` et `#include <truc.h>` ?
- Quelle est la différence entre `#include "truc.h"` et `class Truc;` ? Quand a-t-on besoin de cette différence ?
- L'annexe ne donne pas les inclusions de fichiers applicatifs (`#include "truc.h"`), ni les déclarations (`class Truc;`). Pour chaque fichier `.h` et `.cpp`, donner les inclusions et déclarations nécessaires et absentes de l'annexe.
- Quelles sont les commandes, leurs entrées et leurs sorties :
 - de compilation sans édition de liens ?
 - d'édition de liens ?
- Dans quel répertoire met-on les différents type de fichiers `*.cpp` `*.h` `*.o`, exécutable Chimie ?
- A quoi sert la compilation séparée ? la commande `make` ? un `Makefile` ?
- De quels type de fichiers, dépend l'exécutable ? les `*.o` ? Si on modifie `bidule.cpp`, faut-il recompiler `truc.cpp` ?
- Peut-on éviter la compilation séparée ?
- Ecrivez une première version du `Makefile` correspondant à ce programme.
- Ajouter une règle pour détruire les fichiers compilés.
- Comment connaître les dépendances de `*.o` par rapport aux `*.h` ?
- On mettra les règles de compilation fonction des `.h`.

Question 2

- a) Quelles sont les méthodes redéfinies ?
- b) Quelle est la différence entre les méthodes `toNom()` et `toString()` ?

Question 3

- a) C++ ne peut compiler `menuPrincipal()`. Pourquoi ? Rajouter le mot-clé `static` là où cela est nécessaire.
- b) Rajouter les définitions et initialisations de membres statiques qui manquent.

Jeux de tests

Le programme `Chimie` contient encore des erreurs et chaque question 4 à 8 va permettre de déceler une erreur.

Pour chaque question suivante, on donne un jeu de test effectué par un utilisateur, et on demande :

- a) le but du test,
- b) la sortie *erronée* du programme,
- c) l'explication de l'erreur,
- d) les lignes corrigeant l'erreur dans le source C++,
- e) la sortie *correcte* du programme.

Un jeu de test est composé d'une suite de caractères tapés au clavier par l'utilisateur du logiciel `Chimie`. Un espace ou un saut de ligne de la suite de caractères correspond à la touche "entrée" du clavier.

Le but du test (a) et l'explication de l'erreur (c) seront donnés en quelques mots. Dans les sorties du programme (b, e), on supprimera l'affichage des menus `Atome`, `Molecule` et `Chimie`. L'erreur (b) et sa correction (e) seront mises en évidence clairement en soulignant la partie de la sortie qui est erronée ou corrigée. On indiquera clairement le lieu des lignes C++ corrigeant l'erreur (d).

Question 4

```
A C cplusplus A P Q
```

Question 5

```
M C genie P A C cplusplus P M J genie cplusplus A P
A A D cplusplus A P M A P Q
```

Question 6

```
M C genie P A C cplusplus P M J genie cplusplus A P
A A P M D genie A P A A P Q
```

Question 7

```
M C genie C logiciel P A C cplusplus C uml P
M J genie cplusplus J logiciel cplusplus A P A A P Q
```

Question 8

```
M C genie C logiciel P A C cplusplus C uml P
M J genie cplusplus J logiciel uml A R genie uml A P A A P Q
```

Annexe

```
// chimie.cpp

char menuPrincipal() {
    cout << "Chimie :" << endl;
    cout << "\t M. Molecule" << endl;
    cout << "\t A. Atome" << endl;
    cout << "\t Q. Quitter" << endl;
    char r; cin >> r;
    switch(r) {
        case 'M': Molecule::boucle(); break;
        case 'A': Atome::boucle(); break;
    }
    return r;
}

void main() {
    cout << "Chimie, bonjour," << endl;
    while (menuPrincipal()!='Q');
    cout << "Chimie, au revoir." << endl;
}
```

```
// liste.h

class Liste {
private:
    // non visible
public:
    int longueur;
    Liste();
    ~Liste();
    void ajouter(Entite *);
    int enlever(Entite *);
    Entite * premier();
    Entite * prochain();
    Entite * getInstance(char *);
    char * toNom();
    char * toString();
};
```

```
// liste.cpp

// non visible
```

```
// entite.h

class Entite {
public:
    char nom[32];
    Entite(char *);
    virtual char * toString();
    virtual char * toNom();
};

// entite.cpp

Entite::Entite(char * n) {
    strcpy(nom, n);
}

char * Entite::toString() {
    return toNom();
}

char * Entite::toNom() {
    char * s = new char[32];
    strcpy(s, nom);
    return s;
}

// atome.h

class Atome : public Entite { public:
    static Liste * mesInstances;
    Molecule * maMolecule;
    Atome(char *);
    ~Atome();
    char * toString();
    char * toMolecule();
    void boucle();
    static char menu();
    static void creation();
    static void destruction();
};

// atome.cpp

Atome::Atome(char * n) : Entite (n) {
    maMolecule = NULL;
    mesInstances->ajouter(this);
}

Atome::~~Atome() {
    mesInstances->enlever(this);
}

char * Atome::toString() {
    char * s = new char[128];
    strcat(s, toNom());
    strcat(s, toMolecule());
    return s;
}
```

```

char * Atome::toMolecule() {
    char * s = new char[64];
    strcat(s, " ma molecule: ");
    strcat(s, maMolecule->toNom());
    strcat(s, " ");
    return s;
}

void Atome::boucle() {
    while (menu()!='P');
}

char Atome::menu() {
    cout << "Atome :" << endl;
    cout << "\t C. Creation" << endl;
    cout << "\t D. Destruction" << endl;
    cout << "\t A. Affichage" << endl;
    cout << "\t P. Precedent" << endl;
    char r; cin >> r;
    switch(r) {
    case 'C': creation(); break;
    case 'D': destruction(); break;
    case 'A': {
        cout << "Les atomes sont: " << endl;
        cout << mesInstances->toString();
    } break;
    }
    return r;
}

void Atome::creation() {

    cout << "Creation d'un atome: " << endl;
    cout << "Nom de l'atome ? ";
    char s[32]; cin >> s;

    Atome * a = new Atome(s);
    cout << endl << "Atome '" << s << "' cree." << endl;
}

void Atome::destruction() {

    cout << "Destruction d'un atome: " << endl;
    cout << "Nom de l'atome ? ";
    char s[32]; cin >> s;

    Atome * a = (Atome*) mesInstances->getInstance(s);
    if (a!=NULL) {
        delete a;
        cout << endl << "Atome '" << s << "' detruit." << endl;
    }
    else cout << endl << "Pas d'atome portant le nom '" << s << "'." << endl;
}

```

```

// molecule.h

class Molecule : public Entite { public:
    static Liste * mesInstances;
    Liste * mesAtomes;
    Molecule(char *);
    ~Molecule();
    void detruire();
    char * toString();
    void boucle();
    static char menu();
    static void creation();
    static void destruction();
    static void ajoutAtome();
    static void retraitAtome();
};

// molecule.cpp

Molecule::Molecule(char * n) : Entite(n) {
    mesAtomes = new Liste();
    mesInstances->ajouter(this);
}

Molecule::~Molecule() {
    mesInstances->enlever(this);
    delete mesAtomes;
}

char * Molecule::toString() {
    char * s = new char[1024];
    strcat(s, toNom());
    strcat(s, " Mes atomes: ");
    strcat(s, mesAtomes->toNom());
    return s;
}

void Molecule::boucle() {
    while (menu()!='P');
}

char Molecule::menu() {
    cout << "Molecule :" << endl;
    cout << "\t C. Creation" << endl;
    cout << "\t J. aJouter un atome" << endl;
    cout << "\t R. Retirer un atome" << endl;
    cout << "\t D. Destruction" << endl;
    cout << "\t A. Affichage" << endl;
    cout << "\t P. Precedent" << endl;
    char r; cin >> r;
    switch(r) {
    case 'C': creation(); break;
    case 'J': ajoutAtome(); break;
    case 'R': retraitAtome(); break;
    case 'D': destruction(); break;
    case 'A': {
        cout << "Les molecules sont: " << endl;
        cout << mesInstances->toString();
    } break;
    }
    return r;
}

```

```

void Molecule::creation() {
    cout << "Creation d'une molecule: " << endl;
    cout << "Nom de la molecule ? "; char s[32]; cin >> s;
    Molecule * m = new Molecule(s);
    cout << endl << "Molecule '" << s << "' creee." << endl;
}

void Molecule::destruction() {
    cout << "Destruction d'une molecule: " << endl;
    cout << "Nom de la molecule ? "; char s[32]; cin >> s;
    Molecule * m = (Molecule*) mesInstances->getInstance(s);
    if (m!=NULL) {
        delete m;
        cout << endl << "Molecule '" << s << "' detruite." << endl;
    }
    else
        cout << endl << "Pas de molecule avec le nom '" << s << "'." << endl;
}

void Molecule::ajoutAtome() {
    cout << "Ajout d'un atome a une molecule: " << endl;
    cout << "Nom de la molecule ? "; char sm[32]; cin >> sm;
    Molecule * m = (Molecule*) mesInstances->getInstance(sm);
    if (m==NULL) {
        cout << endl << "Pas de molecule avec le nom '" << sm << "'." << endl;
        return;
    }
    cout << endl << "Nom de l'atome ? "; char sa[32]; cin >> sa;
    Atome * a = (Atome*) Atome::mesInstances->getInstance(sa);
    if (a==NULL) {
        cout << endl << "Pas d'atome avec le nom '" << sa << "'." << endl;
        return;
    }
    m->mesAtomes->ajouter(a);
    a->maMolecule = m;
    cout << endl << "Atome '" << sa << "' ajoute a la molecule '"
        << sm << "'." << endl;
}

void Molecule::retraitAtome() {
    cout << "Retrait d'un atome d'une molecule: \n";
    cout << endl << "Nom de la molecule ? "; char sm[32]; cin >> sm;
    Molecule * m = (Molecule*) mesInstances->getInstance(sm);
    if (m==NULL) {
        cout << endl << "Pas de molecule avec le nom '" << sm << "'." << endl;
        return;
    }
    cout << endl << "Nom de l'atome ? "; char sa[32]; cin >> sa;
    Atome * a = (Atome*) Atome::mesInstances->getInstance(sa);
    if (a==NULL) {
        cout << endl << "Pas d'atome avec le nom '" << sa << "'." << endl;
        return;
    }
    m->mesAtomes->enlever(a);
    a->maMolecule = NULL;
    cout << endl << "Atome '" << sa << "' retire de la molecule '"
        << sm << "'." << endl;
}

```