Université Paris Diderot **Langages à objets avancés**
M1 **Année 2015-2016**

TP n° 3

Type référence, tableaux, pointeurs

Exercice 1 Proposez une fonction `countDown` dans les cas suivants, de sorte que le programme termine et affiche `5, 4, 3, 2, 1, done !`

— cas 1 :

```
int main(){
    int n=5;
    while (n>0) countDown(& n);
    cout << "done !" << endl;
    return 0;
}
```

— cas 2 :

```
int main(){
    int n=5;
    while (n>0) countDown2(n);
    cout << "done !" << endl;
    return 0;
}
```

— cas 3 :

```
int main(){
    int n=5;
    while (n>0) countDown3(n)--;
    cout << "done !" << endl;
    return 0;
}
```

Exercice 2 N'écrivez pas tout de suite le code suivant, mais devinez d'abord ce qu'il affiche. Puis vérifiez.

```
int a = 2;
int b = 4;
int & ref1 = a;
int & ref2 = b;

ref1 -= ref2;
ref2 += ref1;
cout << "ref1 = " << ref1 << ", ref2 = "<< ref2 << endl;

a    -= b;
b    += a;
cout << "a = " << a << ", b = "<< b << endl;
```

Exercice 3 Quelle est la nature du type type `int * & y` ? Ecrivez une séquence qui permette d'incrémeter un entier par l'intermédiaire d'une variable y ainsi définie. Pourquoi le type `int & * y` n'a pas vraiment de sens ?

Exercice 4 (bizarreries du type référence)

1. Supposons que vous écriviez un accesseur d'une classe A qui retourne un type référence vers un de ses champs privés. Est ce que votre programme compile ? Est ce qu'il s'exécute ? Vérifiez le.
2. On considère le code suivant :

```
int main() {
 Rien a(3);
 Rien & r= testReturn();
 Rien b(4);
 return 0;
}
Rien & testReturn() {
 Rien y(1);
 Rien x(-1);
 Rien z(2);
 return x;
}
```

avec

```
class Rien {
 public :
  Rien(int);
  virtual ~Rien();
 private :
  int champs;
}

Rien::Rien(int x):champs(x){}
Rien::~Rien(){
 cout << "destruction du no : " << champs << endl;
}
```

Est ce que le programme compile ? Quand est censé être détruite la variable dont le champs vaut -1 ? Vérifier.

Exercice 5 On s'intéresse à une implémentation des listes chainées d'entiers.

1. Définissez une classe `CelluleChainee` d'entiers suivis d'autres cellules chainées avec simplement les méthodes `ajouter(int)` et `afficher()`. On rappelle que le pointeur nul est appelé `nullptr` en `C++`.
2. Définissez une classe `ListeChainee` qui contienne uniquement un pointeur vers une `celluleChainee`

3. Construisez à la main une liste chainée de 3 éléments.
4. On souhaite faire particulièrement attention aux ressources mémoire, modifiez le destructeur de vos cellules pour qu'il affiche par exemple "destruction de la cellule de valeur 1". Assurez vous que la mémoire soit bien totalement restituée en fin de programme.

Exercice 6 Pour modéliser un graphe on peut utiliser la représentation dites par listes d'adjacences. On rappelle qu'il s'agit d'un tableau de listes chainées, indexé par les numéros de sommets, et chaque liste représente les successeurs du dit sommet.

— Ecrivez un objet graphe qui utilise cette représentation
— Ecrivez une méthode qui génère un graphe complet de taille n
— Assurez vous que la destruction libère bien toute la mémoire