TD1 : Rappels de programmation

1 Récursivité

Les coefficients binômiaux satisfont la propriété suivante :

$$\forall k \geq l \in \mathbb{N}\ ,\ \left(\begin{array}{c} k \\ l \end{array} \right) = \left(\begin{array}{c} k-1 \\ l \end{array} \right) + \left(\begin{array}{c} k-1 \\ l-1 \end{array} \right)$$

Ecrire une fonction récursive `bin(k,l)` qui calcule $\left(\begin{array}{c} k \\ l \end{array} \right)$.

2 Boucles

Dans le programme suivant :

```
int i,j;
for (i=0;i<20;i++) {
  for (j=0;j<=20;j++) {
    blup();
  }
}
```

combien de fois la fonction `blup()` est-elle appelée ?

Dans le programme suivant :

```
int k;
for (k=0;k<20;k++) {
  for (k=0;k<10;k++) blup();
}
```

combien de fois la fonction `blup()` est-elle appelée ? Que vaut la variable `k` à la fin ?

3 Pointeurs et tableaux

Un pointeur est une adresse dans la mémoire de l'ordinateur, et pas une valeur. Bien différencier :

```
int a;
a=12;
```

et

```
int *b;
b=new int;
*b=12;
```

`a` est un entier, `b` est l'adresse d'un entier ; il faut réserver un entier, obtenir son adresse (`b=new int ;`) et accéder à l'entier qu'il référence par `*b`. Pour l'ordinateur, tout pointeur est potentiellement l'adresse d'un ou plusieurs entiers consécutifs, ce qu'on appelle usuellement un "tableau" d'entiers.

Après l'exécution du code qui suit :

```
int *t=new int[20];
for (int i=0;i<20;i++) t[i]=i;
```

que vaut `t[18]` ? Que vaut `t[25]` ?

Après l'exécution du code qui suit :

```
int *b=new int;
for (int i=0;i<20;i++) b=&i;
```

que vaut `*b` ? Si on mettait `*b=i ;` au lieu de `b=&i ;`, que vaudrait `*b` ? Et si on mettait `int *b ;` au lieu de `int *b=new int ;` ?

4 Listes

Une liste est une chaîne d'objets terminée par un lien `NULL`. Dans ce TD, on utilisera :

```
typedef struct objet {
  // trucs...
  objet*   suivant;
} objet;
```

Chaque objet est un morceau de mémoire et doit être alloué. Voici une série d'utilisations de listes chaînées plus ou moins fausses. Trouvez les erreurs ou justifiez l'exactitude de l'exemple :

- une fonction qui crée une petite liste de 3 objets :

```
objet* cree_liste(void) {
  objet a,b,c;
  a.suivant=&b;
  b.suivant=&c;
  c.suivant=NULL;
  return &a;
}
```

- une procédure qui modifie les trucs de tous les objets de la liste

```
void bidouille_liste(objet* &liste) {
  while ( liste != NULL ) {
    bidouille(liste); // bidouille l'objet pointé par liste
    liste=liste->suivant;
  }
}
```

Que se passe-t-il si on met `void bidouille_liste(objet* liste)` ?

– une fonction qui coupe une liste en deux au niveau de l'objet `a` et met la deuxième moitié en début de liste :

```
void coupe_liste(objet* liste,objet* a) {
  objet* b=a;
  while ( b->suivant != NULL ) b=b->suivant;
  b->suivant=liste;
}
```

– une fonction pour détruire tous les objets d'une liste

```
void detruit_liste(objet* liste) {
  while ( liste != NULL ) {
    delete liste;  // detruit l'objet pointé par liste
    liste=liste->suivant;
  }
}
```

Ecrire une fonction `objet* duplique_liste(objet* liste)` qui crée un double de la liste. Faire une version itérative et une version récursive.