AGREGAT

EXERCICES

1. Utiliser *strcat* implique qu'on a réservé assez de mémoire pour représenter la premier argument auquel est ajouté le deuxième argument. On demande d'écrire une fonction *conc* qui libère le programmeur de l'obligation de cette réservation.

```
char * conc(char * & a, char * b)
{   char * l;
   l=new char [strlen(a)+strlen(b)+1];
   strcpy(l, a); strcat(l,b);
   delete a; a=l;
   return a;
}
```

2. Ecrire la définition d'une structure *complexe* pour représenter des nombres complexes. Surcharger les opérateurs +, -, * et / pour cette structure. On représentera un nombre complexe de deux façons :
   - en coordonnées cartésiennes : partie réelle et partie imaginaire.
   - en coordonnées polaires : module et argument.

   Soient x et y les parties réelle et imaginaire, m et a le module et l'argument, on a :

$$x = m \times \cos(a) \qquad m = \sqrt{x^2 + y^2}$$
$$y = m \times \sin(a) \qquad a = \arctan(y/x)$$

3. Ecrire la définition d'une fonction qui retourne vrai si un tableau est un palindrome et faux sinon.

4. Ecrire la définition d'une fonction qui additionne deux tableaux à une dimension dans un troisième.

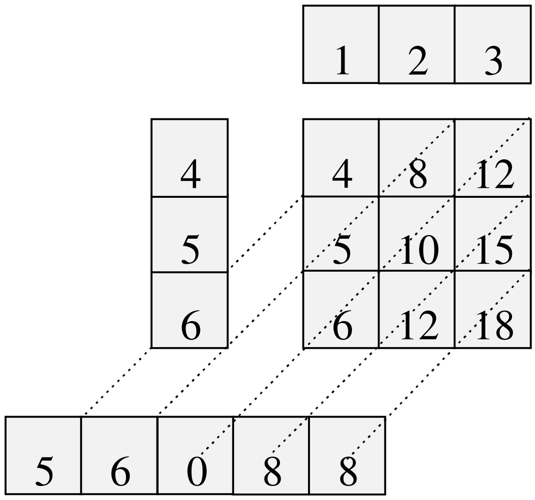
5. Ecire la définition d'une fonction qui affiche le contenu d'un tableau à deux dimensions ligne par ligne.

6. Ecrire la définition d'une fonction qui initialise un tableau d'entiers à 0.

7. Ecrire la définition d'une fonction qui initialise la première diagonale d'un tableau d'entiers à 1.
8. Soit T un tableau rempli de 0 et de 1. Ecrire la définition de fonction qui pour i et j retourne le nombre de 1 dans le tableau entourant la case t[i][j].

9. Ecrire une définition de fonction qui transpose un tableau à deux dimensions.

10. Une des méthodes introduites par Fibonnacci pour multiplier (à la main) deux entiers est la suivante : pour multiplier 123 par 456 on calcule un tableau à deux dimensions, puis on fait des sommes sur les diagonales en reportant la retenue sur la diagonale suivante.



11. Quel est le résultat du petit programme suivant ?

```
int * p;
int t[5];
int i;
p=t;
for(i=0; i<3; ++i) t[i]=10+i;
for(i=0; i<3; ++i, ++p)
cout<<*(&t[i])<<" === "<<*(t+i)<< " === "<<i[t]<< " === "<<*p<<"\n";
```

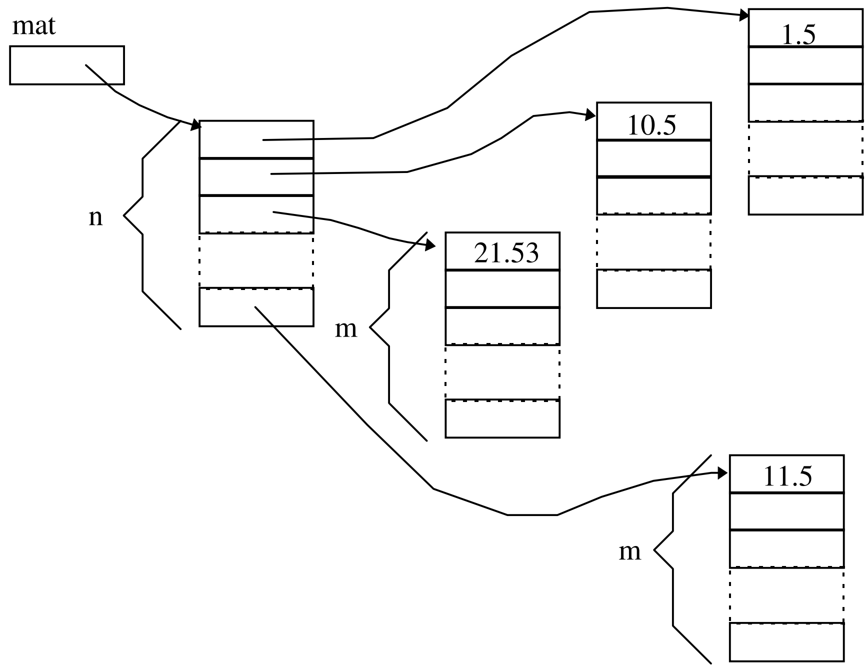
12. Soit les déclarations suivantes :

int i;
int * pi;
int * const cp = &i;
const int ci = 7;
const int * pci;
const int * const cpc = &ci;

Parmi les affectations suivantes quelles sont celles qui sont légales et expliquer ce qu’elles font, et quelles sont celles qui ne sont pas légales et expliquer pourquoi.

i = ci;
*cp = ci;
++pci;
pci = cpc;
ci = 0;
ci--;
*pci = 3;
cp = &ci;
cpc++;
pi = pci;

13. Pour représenter un tableau à deux dimensions déléments de type *float* nous aimerions utiliser la structure suivante :



Ecrire la séquence d'instruction qui construit la structure (en réservant l'espace) et la séquence d'instructions qui libère l'espace occupé.

Ecrire la séquence d'instructions qui permet de lire des valeurs réelles au clavier et les range dans la structure précédente.

14. Quel est l'effet des instructions suivantes :

```
int t [20];

void f( int t[] ){
    t[1]=111;
}

    t[10] =10;
    *(&t[10]+1) = 11;
     *(t+12) = 12 ;
     13[t]=13;
     (10+4)[t]=14;
     f(t);
```

15. Quel est le message d'erreur du compilateur lors de la compilation de :

```
void f1(const int  t[] ){
    t[1]=111;
}
```