```cpp
#include<iostream>
#include<cstdio>
#include<string>
using namespace std;
struct Personne
{
   public:
      string nom;
      string prenom;
      int age;
   public:
      Personne(string name, string prenom1, int age1){
         nom = name;
         prenom = prenom1;
         age = age1;

      }
      void setAge(int new_age){
         age = new_age;
         printf("Nouvelle valeur de l'age [%d]\n",age);
      }
      int getAge(){
         return age;
      }
      string affiche(){
         std::string phrase;
         phrase = "Se nomme " + nom + " "+ prenom +" et a " +
    to_string(age) + " ans";
         return phrase;
      }
};
struct Point {
    private :
       float x;
       float y;
    public:
       Point(float px,float py)
       {
         x=px;
         y=py;
       }
       void affichePoint()
       {
         cout << "(" << x << "," << y << ")" << endl;
       }
       void translate(Point b)
       {
         x=x+b.x;
         y=y+b.y;
       }
   };
int main(void)
{
   Personne a("savain","blaneus",20);
   printf("%d ans\n",a.getAge());
   a.setAge(11);
   string phrase = a.affiche();
```

```
57    printf("%s\n",phrase.c_str());
58 }
```

Listing 1: Fichier moyenne2.c

```
1  #include<iostream>
2  #include<cstdio>
3  #include<string>
4  void echange(int a, int b){
5      int c = a;
6      a = b;
7      b = c;
8      printf("a = %d et b = %d\n",a,b);
9  }
10 struct CompteInstance {
11     public:
12        static int nb; //nombre dobjet
13     public:
14         CompteInstance(void)
15         {
16             nb = nb + 1;//nombre dobjet qui sajoute
17         }
18         static int compter(void)
19         {
20            return nb;
21         }
22         ~CompteInstance()
23         {
24         nb = nb - 1;
25         }
26 };
27 int CompteInstance::nb = 0;
28 struct VectorA
29 {
30     public:
31         int taille;
32         int tab[];
33     public:
34         VectorA(int a){
35             taille = a;
36             int tab[taille];
37         }
38         int &access(int index){
39            return tab[index];
40         }
41 };
42 struct VectorB
43 {
44     public:
45         int taille;
46         int* tab;
47     public:
48         VectorB(int a){
49             taille = a;
50             tab = (int*) malloc(taille);
51         }
52         VectorB(VectorB const& nouveau): VectorB(nouveau.taille){
53
```

```
54          }
55          int &access(int index){
56              if (index < taille && index > 0 ){
57                  return tab[index];
58              }
59              else{
60                  throw 1;
61              }
62
63
64          }
65          ~VectorB(){
66              free(tab);
67          }
68  };
69  struct Personne{
70      public:
71          std::string nom;
72          std::string prenom;
73          std::string metier;
74      public:
75          Personne(std::string nom0, std::string prenom0,std::string
    metier0){
76              nom = nom0;
77              prenom = prenom0;
78              metier = metier0;
79              printf("sappelle %s %s et a %d ans",nom,prenom,metier0)
    ;
80          }
81  };
82  int main(){
83      //echange(2,1);
84      VectorB t(5);
85      printf("[%d]->",t.access(3));
86      t.access(3)=2; // met la valeur 2 dans la case d'indice 3
87      printf("[%d]\n",t.access(3));
88
89      VectorB t2(t);
90      printf("[%d]\n",t2.access(3));
91      int a=t.access(4); // initialise a avec le contenu de la case d
    'indice 4
92
93
94      /*Personne c {"Champmathieu","Honore","Boulanger"};
95      Personne d {"Diaz","Rodrigue",4};
96      Personne r {"de Rais","Gilles"};*/
97     CompteInstance as;
98      {
99          CompteInstance b;
100         CompteInstance c;
101         CompteInstance d;
102         printf("%d\n",CompteInstance::compter()); // renvoie 4 (car
     4 objets a b c d)
103     }
104     printf("%d\n",CompteInstance::compter());
105
106
```

```
}
```

Listing 2: Fichier moyenne2.c

```
class Duree{
      private :
         const long int nT; // nombre total de secondes
      public:
      Duree(long int s=0,int m=0,int h=0,int j=0) : nT( s+60* ( m
    +60* ( h+24*j ))){}
      int egale(Duree a) const{
         return a.nT==nT;
      }
      int estPlusLongueQue(Duree a) const {
         return nT>a.nT;
      }
      Duree somme(Duree b) const {
         return Duree(nT+b.nT);
      }
      double duree(void) const {
         return nT/(60.*60*24);
      }
      void afficher(void) const {
         long int s=nT;
         if(s<0) {
            s=-s;
            printf("-");}
         printf("%ld:%ld:%ld:%ld\n",s%60,(s/60)%60,(s/3600)%24,s
    /(24*3600));}
      int operator !=(Duree a) const{
         return !egale(a);}
      int operator <=(Duree a) const{
         return !estPlusLongueQue(a);}
      Duree operator -(Duree a) const{
         return Duree(nT-a.nT);}
   };
```

Listing 3: Durée

```
class Interv {
      private :
      const double m; // extr mit  minimum
      const double M; // extr mit  maximum
      public :
      Interv(double pm, double pM):  m(pm),M(pM){}
      Interv(double s):  m(s),M(s){}
      Interv(void) :  m(1.),M(0.){}  // vide car M<m
      // les destructeurs et constructeur de copie sont inutiles
      int estVide(void) const {
         return m>M;}
      int appartient(double n) const {
         return n>=m && n<=M;}
      Interv intersect(Interv const & a) const {
         return Interv(max(m,a.m),min(M,a.M));}
      Interv reunion(Interv const& a) const {
         if(a.estVide())
            return *this;
         if(estVide())
```

```
            return a;
         if(a.M < m || M < a.m)
            throw 1; // l'union n'est pas un intervalle
         return Interv(min(a.m,m),max(a.M,M));}
      Interv operator &&(Interv const& a) const {
         return intersect(a);}
   };
```

Listing 4: Interv

```
enum {ind_err=0};

class VectorBA{
      int n;//la taille
      int* tab;
   public:
      VectorBA(int taille) : n(taille),tab(new int[n]){}
      VectorBA(const VectorBA& s) : n(s.n),tab(new int[n]){
         int i;
         for(i=0;i<n;i++)
            tab[i]=s.tab[i];
      }
      ~VectorBA(void) {
         delete [] tab;
      }
      int& acess(int a){ // accesseur non constant
         if(a<0 || a>=n)
            throw ind_err;
         return tab[a];
      }
      int  acess(int a) const { // accesseur const
         if(a<0 || a>=n)
            throw ind_err;
         return tab[a];
      }
      void afficher(void) const {
         int i;
         for(i=0;i<n;i++)
            std::cout<<tab[i]<<" "<<std::endl;
      }
};
```

Listing 5: vector BA

```
class CompareLongueur
{
public:
    bool operator()(const string& a, const string& b)
    {
        return a.length() < b.length();
    }
};
int main()
{
  //Une table qui associe le nom d'un animal    son poids
  map<string, double,CompareLongueur> poids;  //On utilise le
    foncteur comme crit re de comparaison

```

```
14
15    //On ajoute les poids de quelques animaux
16    poids["souris"] = 0.05;
17    poids["tigre"] = 200;
18    poids["chat"] = 3;
19    poids["elephant"] = 10000;
20
21    //Et on parcourt la table en affichant le nom et le poids
22    for(map<string, double>::iterator it=poids.begin(); it!=poids.end
        (); ++it)
23    {
24        cout << it->first << " pese " << it->second << " kg." << endl
        ;
25    }
26    return 0;
27 }
```

Listing 6: it

```
1  #include <iostream>
2  #include <iterator>
3  using namespace std;
4
5  int main()
6  {
7      ostream_iterator<double> it(cout, ", ");
8      *it = 3.14;
9      *it = 2.71;
10
11     return 0;
12 }
13 Ce qui donne :
14
15 3.14, 2.71,
```

Listing 7: Fichier moyenne2.c

```
1  #include <vector>
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      vector<double> tab(5, 3.14);  //Un tableau de 5 nombres
       d cimaux
8
9      try
10     {
11         tab.at(8) = 4.;  //On essaye de modifier la 8 me  case
12     }
13     catch(exception const& e)
14     {
15         cerr << "ERREUR : " << e.what() << endl;
16     }
17     return 0;
18 }
```

Listing 8: Fichier moyenne2.c

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    try
    {
        vector<int> a(1000000000,1); //Un tableau bien trop grand
    }
    catch(exception const& e) //On rattrape les exceptions standard de tous types
    {
        cerr << "ERREUR : " << e.what() << endl; //On affiche la description de l'erreur
    }
    return 0;
}
```

Listing 9: Fichier moyenne2.c

```cpp
#include <iostream>
using namespace std;

template <typename T>
T maximum(const T& a,const T& b)
{
   if(a>b)
      return a;
   else
      return b;
}

int main()
{
     double pi(3.14);
     double e(2.71);

     cout << maximum<double>(pi,e) << endl; //Utilise la "version double"de la fonction

     int cave(-1);
     int dernierEtage(12);

     cout << maximum<int>(cave,dernierEtage) << endl; //Utilise la "version int" de la fonction

     unsigned int a(43);
     unsigned int b(87);

     cout << maximum<unsigned int>(a,b) << endl; //Utilise la "version unsigned int" de la fonction.

     return 0;
}
```

Listing 10: Fichier moyenne2.c

```cpp
#include<iostream>
using namespace std;

template<typename T, typename S>
S moyenne(T tableau[], int taille)
{
  S somme(0);                   //La somme des    lments    du tableau
  for(int i(0); i<taille; ++i)
    somme += tableau[i];

  return somme/taille;
}

int main()
{
  int tab[5];
  //Remplissage du tableau

  cout << "Moyenne : " << moyenne<int,double>(tab,5) << endl;

  return 0;
}
```

Listing 11: Fichier moyenne2.c

```cpp
class TestVoyelles
{
public:
    bool operator()(string const& chaine) const
    {
        for(int i(0); i<chaine.size(); ++i)
        {
            switch (chaine[i])   //On teste les lettres une    une
            {
                case 'a':        //Si c'est une voyelle
                case 'e':
                case 'i':
                case 'o':
                case 'u':
                case 'y':
                    return true;  //On renvoie 'true'
                default:
                    break;         //Sinon, on continue
            }
        }
        return false;   //Si on arrive l  , c'est qu'il n'y avait
    pas de  voyelle du tout
    }
};
```

Listing 12: Fichier moyenne2.c