## Exercice 1

Afficher le nombre d'appels :

#include <iostream>  
using namespace std;  
  
void compterAppels() {  
 static int compteur = 0;  
 compteur++;  
 cout << "appel numero " << compteur << endl;  
}  
  
int main() {  
 compterAppels();  
 compterAppels();  
 compterAppels();  
 return 0;  
}

## Exercice 2

Tester si un nombre est pair ou multiple de 3 :

#include <iostream>  
using namespace std;  
  
int multiple2(int n) { return n % 2 == 0; }  
int multiple3(int n) { return n % 3 == 0; }  
  
int main() {  
 int n;  
 cout << "donnez un entier : ";  
 cin >> n;  
  
 if (multiple2(n)) cout << "il est pair" << endl;  
 if (multiple3(n)) cout << "il est multiple de 3" << endl;  
 if (multiple2(n) && multiple3(n)) cout << "il est divisible par 6" << endl;  
  
 return 0;  
}

## Exercice 3

Trouver le plus grand et le plus petit dans un tableau :

#include <iostream>  
using namespace std;  
  
int main() {  
 int tab[10];  
 for (int i = 0; i < 10; i++) cin >> tab[i];  
  
 int min = tab[0], max = tab[0];  
 for (int i = 1; i < 10; i++) {  
 if (tab[i] < min) min = tab[i];  
 if (tab[i] > max) max = tab[i];  
 }  
 cout << "min = " << min << ", max = " << max << endl;  
 return 0;  
}

## Exercice 4

Tableaux dynamiques et carrés :

#include <iostream>  
using namespace std;  
  
int main() {  
 int n;  
 cout << "taille du tableau : ";  
 cin >> n;  
  
 int\* t = new int[n];  
 for (int i = 0; i < n; i++) cin >> t[i];  
  
 int\* carre = new int[n];  
 for (int i = 0; i < n; i++) carre[i] = t[i] \* t[i];  
  
 delete[] t;  
 for (int i = 0; i < n; i++) cout << carre[i] << " ";  
 delete[] carre;  
 return 0;  
}

## Exercice 5

Afficher variable, référence et pointeur :

#include <iostream>  
using namespace std;  
  
int main() {  
 int a = 10;  
 int& ref\_a = a;  
 int\* p\_a = &a;  
  
 cout << "a = " << a << endl;  
 cout << "ref\_a = " << ref\_a << endl;  
 cout << "p\_a pointe vers = " << \*p\_a << endl;  
  
 cout << "&a = " << &a << endl;  
 cout << "&ref\_a = " << &ref\_a << endl;  
 cout << "p\_a = " << p\_a << endl;  
 return 0;  
}

## Exercice 6

Incrémenter et permuter (adresse et référence) :

#include <iostream>  
using namespace std;  
  
void incrementer(int\* x) { (\*x)++; }  
void permuter(int\* a, int\* b) { int tmp = \*a; \*a = \*b; \*b = tmp; }  
  
void incrementerRef(int& x) { x++; }  
void permuterRef(int& a, int& b) { int tmp = a; a = b; b = tmp; }  
  
int main() {  
 int a = 5, b = 8;  
 incrementer(&a);  
 cout << "a = " << a << endl;  
  
 permuter(&a, &b);  
 cout << "a = " << a << ", b = " << b << endl;  
  
 incrementerRef(a);  
 permuterRef(a, b);  
 cout << "a = " << a << ", b = " << b << endl;  
 return 0;  
}

## Exercice 7

Afficher toutes les permutations d'une chaîne :

#include <iostream>  
using namespace std;  
  
void permuter(char &a, char &b) { char tmp = a; a = b; b = tmp; }  
  
void permutations(string s, int i, int n) {  
 if (i == n) cout << s << endl;  
 else {  
 for (int j = i; j <= n; j++) {  
 permuter(s[i], s[j]);  
 permutations(s, i + 1, n);  
 permuter(s[i], s[j]);  
 }  
 }  
}  
  
int main() {  
 string mot;  
 cin >> mot;  
 permutations(mot, 0, mot.size() - 1);  
 return 0;  
}

## Exercice 8

Classe Voiture :

#include <iostream>  
using namespace std;  
  
class Voiture {  
private:  
 string marque, modele;  
 int annee;  
 float kilometrage, vitesse;  
public:  
 Voiture() { marque="Inconnue"; modele="Inconnu"; annee=0; kilometrage=0; vitesse=0; }  
 Voiture(string m,string mod,int a,float km,float v){marque=m;modele=mod;annee=a;kilometrage=km;vitesse=v;}  
 void accelerer(float v){vitesse+=v;}  
 void freiner(float v){vitesse-=v;if(vitesse<0)vitesse=0;}  
 void avancer(float d){kilometrage+=d;}  
 void afficherInfo(){  
 cout<<"Marque:"<<marque<<"\nModele:"<<modele<<"\nAnnee:"<<annee<<"\nKm:"<<kilometrage<<"\nVitesse:"<<vitesse<<endl;  
 }  
 ~Voiture(){cout<<"Voiture detruite"<<endl;}  
};  
  
int main(){  
 Voiture v1("Toyota","Corolla",2020,50000,0);  
 v1.accelerer(50);  
 v1.avancer(120);  
 v1.freiner(30);  
 v1.afficherInfo();  
 return 0;  
}

## Exercice 9

Classe Vecteur3d :

#include <iostream>  
#include <cmath>  
using namespace std;  
  
class Vecteur3d {  
public:  
 float x,y,z;  
 Vecteur3d(float a=0,float b=0,float c=0){x=a;y=b;z=c;}  
 void afficher(){cout<<"("<<x<<","<<y<<","<<z<<")"<<endl;}  
 Vecteur3d somme(Vecteur3d v){return Vecteur3d(x+v.x,y+v.y,z+v.z);}  
 float produitScalaire(Vecteur3d v){return x\*v.x+y\*v.y+z\*v.z;}  
 bool coincide(Vecteur3d v){return(x==v.x&&y==v.y&&z==v.z);}  
 float norme(){return sqrt(x\*x+y\*y+z\*z);}  
 Vecteur3d normax(Vecteur3d v){return(this->norme()>=v.norme()?\*this:v);}  
};  
  
int main(){  
 Vecteur3d v1(1,2,3),v2(3,4,5);  
 cout<<"v1=";v1.afficher();  
 cout<<"v2=";v2.afficher();  
 cout<<"Somme:";v1.somme(v2).afficher();  
 cout<<"Produit scalaire="<<v1.produitScalaire(v2)<<endl;  
 cout<<"Coincide?"<<(v1.coincide(v2)?"Oui":"Non")<<endl;  
 cout<<"Norme v1="<<v1.norme()<<endl;  
 cout<<"Vecteur avec plus grande norme:";v1.normax(v2).afficher();  
 return 0;  
}

## Exercice 10

Classe Complexe :

#include <iostream>  
using namespace std;  
  
class Complexe {  
public:  
 float re,im;  
 Complexe(float r=0,float i=0){re=r;im=i;}  
 Complexe addition(Complexe c){return Complexe(re+c.re,im+c.im);}  
 Complexe soustraction(Complexe c){return Complexe(re-c.re,im-c.im);}  
 Complexe multiplication(Complexe c){return Complexe(re\*c.re-im\*c.im,re\*c.im+im\*c.re);}  
 Complexe division(Complexe c){  
 float denom=c.re\*c.re+c.im\*c.im;  
 return Complexe((re\*c.re+im\*c.im)/denom,(im\*c.re-re\*c.im)/denom);  
 }  
 bool egal(Complexe c){return(re==c.re&&im==c.im);}  
 void afficher(){cout<<re<<"+"<<im<<"i"<<endl;}  
};  
  
int main(){  
 Complexe c1,c2;cout<<"Entrer re et im du 1er:";cin>>c1.re>>c1.im;  
 cout<<"Entrer re et im du 2eme:";cin>>c2.re>>c2.im;  
 int choix;cout<<"1.Addition 2.Soustraction 3.Multiplication 4.Division 5.Egalite:";cin>>choix;  
 Complexe res;  
 switch(choix){  
 case 1:res=c1.addition(c2);break;  
 case 2:res=c1.soustraction(c2);break;  
 case 3:res=c1.multiplication(c2);break;  
 case 4:res=c1.division(c2);break;  
 case 5:cout<<(c1.egal(c2)?"Egaux":"Differents")<<endl;return 0;  
 default:cout<<"Choix invalide";return 0;  
 }  
 res.afficher();  
 return 0;  
}