Exercices C++ - Code complet (01 à 10)

Exercice 01:

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

Exercice 02:

```
#include <iostream>
using namespace std;
int fct1(int x){
if(x%2==0)
    return 1;
else
    return 0;
}
int fct2(int y){
if(y%3==0)
    return 1;
else
    return 0;
}
int main(){
int n;
cout<<"entrer un nombre entier"<<endl;
cin>>n;
if(fct1(n)==1)
    cout<<n<<"est pair"<<endl;
if(fct2(n)==1)
    cout<<n<<"est un multiple de 3"<<endl;
if(fct1(n)==1 && fct2(n)==1 )
    cout<<n<<"est un multiple de 6"<<endl;
if(fct1(n)==0 && fct2(n)==0 )
    cout<<n<<" ni multiple de 2 ni multiple de 3"<<endl;
return 0;
}
```

Exercice 03:

a-

```
#include <iostream>
using namespace std;
int main(){
int T[10];
int x,y;
cout<<"remplir le tableau"<<endl;
for(int i = 0; i < 10; i++) {
    cout << "T[" << i << "] = ";
    cin >> T[i];
}
```

```
x=T[0];
for (int i = 1; i < 10; i++) {
    if (T[i] > x)
        x = T[i];
}
cout<<x<<"est le plus grand element du tableau"<<endl;
y=T[0];
for (int i = 1; i < 10; i++) {
    if (T[i] < y)
        y = T[i];
}
cout<<y<<"est le plus petit element du tableau"<<endl;
return 0;
}
```

b-

```
#include <iostream>
using namespace std;
int main() {
    int T[10];
    int *p = T;
    int x, y;

    cout << "Entrez 10 nombres :" << endl;
    for (int i = 0; i < 10; i++) {
        cout << "T[" << i << "] = ";
        cin >> *(p + i);
    }

    x = *p;
    y = *p;

    for (int i = 1; i < 10; i++) {
        if (*(p + i) > x)
            x = *(p + i);
        if (*(p + i) < y)
            y = *(p + i);
    }

    cout << "Le plus grand element est:" << x << endl;
    cout << "Le plus petit element est:" << y << endl;
    return 0;
}
```

Exercice 04:

```
#include <iostream>
using namespace std;
int main (){
    int n;
    cout<<"entrer la taille du tableau "<<endl;
    cin>>n;
    int*A= new int[n];
    for(int i=0;i<n;i++){
        cout<<"A["<<i<<"]=";
        cin>>A[i];
    }
    int*B=new int[n];
    for(int i=0;i<n;i++){
        B[i]=A[i]*A[i];
    }
    delete[] A;
    for(int i=0;i<n;i++)
```

```
        cout<<"B["<<i<<"]="<<B[i]<<endl;
    delete[] B;
    return 0;
}

Exercice 05:
#include <iostream>
using namespace std;
int main(){
 int a;
 int &ref_a=a;
 int *p_a=&a;
 cout<<"entrer la valeur de a "<<endl;
 cin>>a;
 cout<<"valeur de a :"<<a<<endl;
 cout<<"valeur de a via ptr  :"<<*p_a<<endl;
 cout<<"valeur de a via ref  :"<<ref_a<<endl;
 cout<<" l'adresse  de a :"<<&a<<endl;
 cout<<" l'adresse de a via ptr  :"<<p_a<<endl;
 cout<<" l'adresse de a via ref  :"<<&ref_a<<endl;
 return 0;
}

Exercice 06:
Methode 1 (les pointeurs)
int incrementer(int *x){
 (*x)++;
 return *x;
}
void permutter(int *a ,int *b){
 int c=*b;
 *b=*a;
 *a=c;
}
int main(){
 int x,y,z;
 int *a=&x;
 int *b=&y;
 int *c=&z;
 cout<<"veuillez entrer la valeur a incrementer et les valeurs a permuter"<<endl;
 cin>>z>>x>>y;
 cout<<"la valeur est incremente "<<incrementer(c)<<endl;
 cout<<"avant permutation : "<<x<<" "<<y<<endl;
 permutter(a,b);
 cout<<"apres permutation :"<<*a<<" "<<*b<<endl;
 return 0;
}

Methode 2 (utilisant les références)
int incrementer(int &x){
 x++;
 return x;
}
void permutter(int &a ,int &b){
 int c=b;
 b=a;
 a=c;
}
int main(){
 int x,y,z;
 cout<<"veuillez entrer la valeur a incrementer et les valeurs a permuter"<<endl;
 cin>>z>>x>>y;
```

```
 cout<<"la valeur est incremente "<<incrementer(z)<<endl;
 cout<<"avant permutation : "<<x<<" "<<y<<endl;
 permutter(x,y);
 cout<<"apres permutation :"<<x<<" "<<y<<endl;
 return 0;
}
```

Exercice 07:

```
#include <iostream>
using namespace std;
void echanger(char &a, char &b) {
 char temp = a;
 a = b;
 b = temp;
}
void permuter(char str[], int debut, int fin) {
 if (debut == fin) {
     cout << str << endl;
 } else {
     for (int i = debut; i <= fin; i++) {
         echanger(str[debut], str[i]);
         permuter(str, debut + 1, fin);
         echanger(str[debut], str[i]);
     }
 }
}
int main() {
 char chaine[100];
 cout << "Entrez une chaine de caracteres : ";
 cin >> chaine;
 int longueur = 0;
 while (chaine[longueur] != '\0') longueur++;
 cout << "Les permutations possibles sont :" << endl;
 permuter(chaine, 0, longueur - 1);
 return 0;
}
```

Exercice 08:

```
class Voiture {
private:
 string marque;
 string modele;
 int annee;
 float kilometrage;
 float vitesse;
public:
 Voiture() {
     marque = "Inconnue";
     modele = "Inconnu";
     annee = 0;
     kilometrage = 0.0;
     vitesse = 0.0;
 }
 Voiture(string m, string mod, int a, float km, float v) {
     marque = m;
     modele = mod;
     annee = a;
     kilometrage = km;
     vitesse = v;
 }
 void accelerer(float valeur) {
     vitesse += valeur;
```

```
    cout << "La voiture accelere de " << valeur << " km/h" << endl;
 }
 void freiner(float valeur) {
    if (vitesse - valeur < 0)
        vitesse = 0;
    else
        vitesse -= valeur;
    cout << "La voiture freine de " << valeur << " km/h" << endl;
 }
 void avancer(float distance) {
    kilometrage += distance;
    cout << "La voiture avance de " << distance << " km" << endl;
 }
 void afficherInfo() {
    cout << "Marque: " << marque << "\nModele: " << modele << "\nAnnee: " << annee << "\nKilome
 }
 ~Voiture() {
    cout << "La voiture " << marque << " " << modele << " est detruite." << endl;
 }
};

Exercice 09:
class Vecteur3D {
private:
 float x, y, z;
public:
 Vecteur3D(float x = 0, float y = 0, float z = 0) {
    this->x = x;
    this->y = y;
    this->z = z;
 }
 void afficher() const {
    cout << "(" << x << ", " << y << ", " << z << ")" << endl;
 }
 Vecteur3D somme(const Vecteur3D& v) const {
    return Vecteur3D(x + v.x, y + v.y, z + v.z);
 }
 float produitScalaire(const Vecteur3D& v) const {
    return x * v.x + y * v.y + z * v.z;
 }
 bool coincide(const Vecteur3D& v) const {
    return (x == v.x && y == v.y && z == v.z);
 }
 float norme() const {
    return sqrt(x*x + y*y + z*z);
 }
 static Vecteur3D normax_valeur(const Vecteur3D& v1, const Vecteur3D& v2) {
    return (v1.norme() >= v2.norme()) ? v1 : v2;
 }
 static Vecteur3D* normax_adresse(Vecteur3D* v1, Vecteur3D* v2) {
    return (v1->norme() >= v2->norme()) ? v1 : v2;
 }
 static Vecteur3D& normax_reference(Vecteur3D& v1, Vecteur3D& v2) {
    return (v1.norme() >= v2.norme()) ? v1 : v2;
 }
};

Exercice 10:
#include <iostream>
using namespace std;
class Complexe {
private:
```

```
 double reel;
 double imag;
public:
 Complexe(double r = 0, double i = 0) {
     reel = r;
     imag = i;
 }
 void afficher() const {
     if (imag >= 0)
         cout << reel << " + " << imag << "i";
     else
         cout << reel << " - " << -imag << "i";
 }
 Complexe operator+(const Complexe& c) const {
     return Complexe(reel + c.reel, imag + c.imag);
 }
 Complexe operator-(const Complexe& c) const {
     return Complexe(reel - c.reel, imag - c.imag);
 }
 Complexe operator*(const Complexe& c) const {
     return Complexe(reel*c.reel - imag*c.imag, reel*c.imag + imag*c.reel);
 }
 Complexe operator/(const Complexe& c) const {
     double denom = c.reel*c.reel + c.imag*c.imag;
     return Complexe((reel*c.reel + imag*c.imag)/denom, (imag*c.reel - reel*c.imag)/denom);
 }
 bool operator==(const Complexe& c) const {
     return (reel == c.reel && imag == c.imag);
 }
};

int main() {
 double r1, i1, r2, i2;
 cout << "Entrez le premier nombre complexe (reel imaginaire) : ";
 cin >> r1 >> i1;
 cout << "Entrez le deuxième nombre complexe (reel imaginaire) : ";
 cin >> r2 >> i2;
 Complexe c1(r1, i1);
 Complexe c2(r2, i2);
 int choix;
 do {
     cout << "\nMenu des operations :\n";
     cout << "1. Addition 2. Soustraction 3. Multiplication 4. Division 5. Egalite 0. Quitter";
     cout << "\nChoisissez une operation : ";
     cin >> choix;
     switch (choix) {
         case 1: { Complexe resultat = c1 + c2; cout << "Resultat : "; resultat.afficher(); cout
         case 2: { Complexe resultat = c1 - c2; cout << "Resultat : "; resultat.afficher(); cout
         case 3: { Complexe resultat = c1 * c2; cout << "Resultat : "; resultat.afficher(); cout
         case 4: { if (r2 == 0 && i2 == 0) cout << "Erreur : Division par zero !"; else { Comple
         case 5: { if (c1 == c2) cout << "Les deux nombres complexes sont egaux."; else cout <<
         case 0: cout << "Au revoir !"; break;
         default: cout << "Choix invalide !"; break;
     }
 } while (choix != 0);
 return 0;
}
```