Devoir TD 2:

exercice 1 :

```
#include <iostream>
using namespace std;
void compter(){
    int i;
    ++i;
        cout<<"appel numéro"<<i<<endl;
    }
int main(){
    compter();
    compter();
    compter();
    return 0;
}
```

Exercice 2:

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

int multiple2(int n ){
    if(n%2== 0)
    cout<<n<<"pair"<<endl;
    return n ;  }
int multiple3(int n ){
    if(n%3==0)
        cout<<n<<"est de multiple de 3"<<endl;
    return n ; }
int main (){
    int x ;
    int i;
    for(i=0;i<2;i++){
        cout<<"Donner un entier  : ";
        cin>>x;
        x=multiple2( x );
        x=multiple2( x );
        if(x % 6 == 0)
            cout<<x<<"il est divisibe par 6"<<endl;}
        cout<<endl; return 0;}
```

Exercice 3:

```cpp
#include <iostream>
using namespace std;
   int main(){
       int t[10];//lecture
       cout<<" les elements de tableau :"<<endl;
       for(int i=0;i<10;i++){
           cout<<"element :"<<i<<endl;
           cin>>t[i];
       }//afichage
       for (int i = 0; i < 10; i++) {
           cout << t[i] << " ";
       }
       cout <<endl;// determiner le plus perit et le plus grand ;
       int max=0,min=0;
       for(int i=0;i<10;i++)
       if(t[i]>max){
           max=t[i];
       }else min=t[i];

       cout<<"le plus grand elemt est :"<<max<<endl;
       cout<<"le plus petit element est : "<<min<<endl;
       return 0;
   }
```

Exercice 4:

```cpp
#include <iostream>
using namespace std;
     int main() {
          Allouer dynamiquement un tableau d'entiers
         int n;
         cout << "Entrez la taille du tableau : ";
         cin >> n;
         int* t = new int[n];
         cout << "Entrez " << n << " nombres entiers :\n";
         for (int i = 0; i < n; i++) {
             cin >> t[i];}
          la creation un nouveau tableau pour les carrés des nombres
        int* te = new int[n];
        for (int i = 0; i < n; i++) {
            te[i] = te[i] * te[i];}
         affichage Afficher les valeurs du deuxième tableau
         cout << "Les carrés des nombres sont : ";
         for (int i = 0; i < n; i++) {
         cout << te[i] << " ";}

         return 0;
     }
```

Exercice 5 :

```cpp
#include <iostream>
using namespace std;
int main() {
         int a ;
```

```
    int &ref_a = a;
        int *p_a = &a;

            A=3 ;
        cout << "La variable a : " << a << endl;
    cout << "L'adresse de a : " << &a << endl;
    cout << "La référence ref_a : " << ref_a << endl;
    cout << "L'adresse de ref_a : " << &ref_a << endl;
    cout << "La valeur pointée par p_a : " << *p_a << endl;
    cout << "L'adresse contenue dans p_a : " << p_a << endl;
    return 0;
}
```

Exercice 7 :

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


int trier(int t[]){
int temp;
int i=0;
int j=0;
while(i<10){
       while(j<10){
             if(t[j]>t[i]){
            temp=t[i];
            t[i]=t[j];
            t[j]=temp;}j++;
}j=0;i++;};
    for(i=0;i<10;i++){ cout<<t[i]<<endl;}return temp;}

int main(){int i;
    int tableau[10]={1,10,20,8,9,05,40,210,0,23};
    for(i=0;i<10;i++){
        cout<<tableau[i]<<endl;
    }
    cout<<endl;
    cout<<"apres trie   :"<<endl;
   //for(i=0;i<10;i++){ cout<<tableau[i]<<endl;}

 trier(tableau);
    return 0;}
```

exercice 8:

```
class NombreComplexe {
    double Reelle;
    double Imaginaire;
public:
    NombreComplexe(double re, double im) : Reelle(re), Imaginaire(im) {};
    NombreComplexe add( NombreComplexe &c) const {
        return NombreComplexe(Reelle + c.Reelle, Imaginaire +
c.Imaginaire);
    }
    NombreComplexe soustraction(const NombreComplexe &c) const {
```

```
        return NombreComplexe(Reelle - c.Reelle, Imaginaire -
c.Imaginaire);
    }
    NombreComplexe multiplication(const NombreComplexe &c) const {
        return NombreComplexe(Reelle * c.Reelle - Imaginaire *
c.Imaginaire,
                              Reelle * c.Imaginaire + Imaginaire *
c.Reelle);
    }
    NombreComplexe division(const NombreComplexe &c) const {
        double diviseur = c.Reelle * c.Reelle + c.Imaginaire *
c.Imaginaire;
        return NombreComplexe((Reelle * c.Reelle + Imaginaire *
c.Imaginaire) / diviseur,
                              (Imaginaire * c.Reelle - Reelle *
c.Imaginaire) / diviseur);
    }
    void afficher() const {
        if (Imaginaire > 0) {
            cout << Reelle << " + " << Imaginaire << "i";
        } else {
            cout << Reelle << " -" << Imaginaire << "i";
        }
    }
};

int main() {
    double r, i, r2, i2;
    std::cout << "Entrez lle reel et  imaginaire de premier nombre complexe
: ";
    std::cin >> r >> i;
    std::cout << "Entrez la partie réelle et la partie imaginaire du
deuxième nombre complexe : ";
    std::cin >> r2 >> i2;

    NombreComplexe c1(r, i);
    NombreComplexe c2(r2, i2);

    int choix;
    switch (choix) {
        case 1:
            cout << "Addition : ";
            c1.add(c2).afficher();
            break;
        case 2:
            cout << "Soustraction : ";
            c1.soustraction(c2).afficher();
            break;
        case 3:
            cout << "Multiplication : ";
            c1.multiplication(c2).afficher();
            break;
        case 4:
            cout << "Division : ";
            c1.division(c2).afficher();
            break;
        default:
            cout << "n'existe pas " << endl;
    }
    return 0;}
```

Exercice 9 :

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

class Animal {

    string name;
    int age;

public :
    Animal (string n , int a ): name(n),age(a){}

    virtual void set_value( ) {

        cout << "nom : " << name << endl;
        cout << " age : " << age << endl;
    }
};
class Zebra : public Animal {

    string lieu;
public :
    Zebra(string n, int a, string l) :Animal(n, a), lieu(l){}
     void set_value  ()  {


        Animal::set_value();
        cout << "lieu de Zebra  : " << lieu << endl;
    }
};
class Dolphin : public Animal {
private:
    string lieu;
public :

    Dolphin(string n, int a, string l) :Animal(n, a), lieu(l) {}
    void set_value() {

        Animal::set_value();
        cout << "lieu de Dolphin : " << lieu << endl;
    }

};

int main()
{
    Zebra z("annexe",  10 , "australe. ");
    Dolphin D("DELFIN.",  2 , "maroc");

    z.set_value();
    cout <<endl;
    D.set_value();
    return 0;
}
```

Exercice 10 :

```
nclude <iostream>
#include <string>

using namespace std;

class Personne {

    string nom;
    string prenom,datedenaissance;
public:
    Personne(string n, string p, string d) : nom(n), prenom(p),
datedenaissance(d) {}
    virtual void Afficher() {
        cout << "Nom: " << nom << endl;
        cout << "Prénom: " << prenom << endl;
        cout << "Date de Naissance: " << datedenaissance << endl;}
};
class Employe : public Personne {
    double salaire;
public:
    Employe(string _nom, string _prenom, string _dateDeNaissance, double
_salaire) : Personne(_nom, _prenom, _dateDeNaissance), salaire(_salaire) {}

    void Afficher() {
        Personne::Afficher();
        cout << "Salaire: " << salaire << endl;
    }
};

class Chef : public Employe {
    string service;

public:
    Chef(string _nom, string _prenom, string _dateDeNaissance, double
_salaire, string _service) : Employe(_nom, _prenom, _dateDeNaissance,
_salaire), service(_service) {}

    void Afficher()  {
        Employe::Afficher();
        cout << "Service: " << service << endl;
    }
};

class Directeur : public Chef {
    string societe;
public:
    Directeur(string _nom, string _prenom, string _dateDeNaissance, double
_salaire, string _service, string _societe) : Chef(_nom, _prenom,
_dateDeNaissance, _salaire, _service), societe(_societe) {}

    void Afficher() {
        Chef::Afficher();
        cout << "Société: " << societe << endl;
    }
};

int main() {
    Personne p("Mokhtar", "Mokhtar", "01/01/1999");
```

```
    Employe e("zineb", "zineb", "05/05/1980", 90000);
    Chef c("BEN", "abdeslam", "10/10/1974", 100000, "hd");
    Directeur d("Ben Mokhtar", "yousra", "1942/02/03", 300000, "sante",
"ccc");

    p.Afficher();
    cout << endl;
    e.Afficher();
    cout << endl;
    c.Afficher();
    cout << endl;
    d.Afficher();

return 0;
```

Exercice 12 :

```
#include <iostream>
using namespace std;
class Test {
    static int count;
public:
    static void call() {
        count++;}
    static int getCount() {
        return count;}
};
int Test::count = 0;

int main() {
    Test::call();
    Test::call();
    Test::call();
    Test::call();
    cout <<  "le nombre de call est : "<<Test::getCount() << endl;

    return 0;
}
```