Exercice 1

Écrire une fonction, sans argument ni valeur de retour, qui se contente d'afficher, à chaque appel, le nombre total de fois où elle a été appelée sous la forme : ***appel numéro 3***

```
#include <iostream>
using namespace std;
int main(){
 int tab[10],i;
 int min, max;
 for(i=1 ; i<=10 ; i++){
    cout<<"donnez le nombre "<<i<<":"<<endl;
    cin>>tab[i];
 }
 max=tab[1];
 min=tab[1];
     for(int i=1 ; i<=10 ; i++){
         if(tab[i] > max){
                max=tab[i];
         }
         if(tab[i] < min){
                min=tab[i];
         }
     }
     cout<<"le plus grand est : "<< max <<endl;
     cout<<"le plus petit est : "<< min <<endl;
     return 0;
}
```

Exercice 2

Écrire 2 fonctions à un argument **entier** et une valeur de retour **entière** permettant de préciser si l'argument reçu est multiple de **2** (pour la première fonction) ou multiple de **3** (pour la seconde fonction).

Utiliser ces deux fonctions dans un petit programme qui lit un nombre entier et qui précise s'il est pair, multiple de 3 et/ou multiple de 6, comme dans cet exemple (il y a deux exécutions) :

```
donnez un entier : 9
il est multiple de 3
--------------
donnez un entier : 12
il est pair
il est multiple de 3
il est divisible par 6
```

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

int multiplede2(int nombre) {
    return (nombre % 2 == 0);
}
```

```
int multiplede3(int nombre) {
    return (nombre % 3 == 0);
}
    int main() {
    int x;
    cout << "Donnez un entier : ";
    cin >> x;
    if (multiplede2(x)) {
        cout << "Il est pair" << endl;
    }
    if (multiplede3(x)) {
        cout << "Il est multiple de 3" << endl;
    }
    if (multiplede2(x) && multiplede3(x)) {
        cout << "Il est divisible par 6" << endl;
    }

    return 0;
}
```

Exercice 3

Écrire, de deux façons différentes, un programme qui lit **10 nombres entiers** dans un tableau avant d'en rechercher le plus grand et le plus petit :

a. en utilisant uniquement le « *formalisme tableau* » ;
b. en utilisant le « *formalisme pointeur* », à chaque fois que cela est possible.

a-

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

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

b-

```
#include <iostream>
using namespace std;
int main(){
 int tab[10],i;
 int min, max;
 int *p=tab;
 for(i=0 ; i<10 ; i++){
    cout<<"donnez le nombre "<<i+1<<":"<<endl;
    cin>>*(p+i);
 }
 max=*p;
 min=*p;
     for(int i=1 ; i<10 ; i++){
         if(*(p+i) > max){
                 max=*(p+i);
         }
         if(*(p+i) < min){
                 min=*(p+i);
         }
     }
     cout<<"le plus grand est : "<< max <<endl;
     cout<<"le plus petit est : "<< min <<endl;
     return 0;
}
```

Exercice 4

Écrire un programme **allouant dynamiquement** un emplacement pour un **tableau d'entiers**, dont la taille est fournie en donnée.

1. Utiliser ce tableau pour y placer des nombres entiers lus également en donnée.
2. Créer ensuite dynamiquement un nouveau tableau destiné à recevoir les carrés des nombres contenus dans le premier.
3. Supprimer le premier tableau, afficher les valeurs du second et supprimer le tout.

1-

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

int main() {
```

```
    int taille;

    cout << "Entrez la taille du tableau: ";
    cin >> taille;

    int* tableau = new int[taille];

    if (tableau == nullptr) {
        cout << "Erreur d'allocation de mémoire." << endl;
        return 1;
    }
    cout << "Entrez " << taille << " nombres entiers:" << endl;
    for (int i = 0; i < taille; i++) {
        cin >> tableau[i];
    }
    int* tableauCarres = new int[taille];
    if (tableauCarres == nullptr) {
        cout << "Erreur d'allocation de mémoire." << endl;
        delete[] tableau;
        return 1;
    }
    for (int i = 0; i < taille; i++) {
        tableauCarres[i] = tableau[i] * tableau[i];
    }
    delete[] tableau;
    cout << "Les carrés des nombres sont:" << endl;
    for (int i = 0; i < taille; i++) {
        cout << tableauCarres[i] << " ";
    }
    cout << endl;
    delete[] tableauCarres;

    return 0;
}
```

Exercice 5

Ecrire un programme C++ qui :

1. déclare un entier a;
2. déclare une référence vers cet entier ref_a;
3. déclare un pointeur vers cet entier p_a;
4. affiche les variables, leurs adresses, la valeur pointée.

```
#include <iostream>
 using namespace std;
 int main() {
 int a = 10;
```

```
 int &ref_a = a;
 int *p_a = &a;
 cout << "valeur de a : " << a << " address de a : " << &a << endl;
 cout << "valeur de ref_a : " << ref_a << " address de ref_a : " << &ref_a <<
endl;
 cout << "valeur de p_a : " << p_a << " point sur : " << *p_a << " address de
p_a : " << &p_a
 << endl;
 }
```

Exercice 6

Écrire une fonction nommée **incrementer**() permettant d'incrémenter la valeur d'une variable passée en paramètre et une fonction nommée **permuter**() permettant d'échanger les contenus de 2 variables de type int fournies en argument :

1. en transmettant l'adresse des variables concernées (seule méthode utilisable en C) ;
2. en utilisant la transmission par référence.

Dans les deux cas, écrire un programme (**main**) qui teste les deux fonctions.

1-

```
#include <iostream>
 using namespace std;
 void permuter(int *a, int *b) {
 int c = *a;
 *a = *b;
 *b = c;
 }
 void incrementer(int *a) {
 *a = *a + 1;
 }
 int main() {
 int a = 1;
 int b = 2;
 permuter(&a, &b);
 cout << "apres permutation a = " << a << " b = " << b << endl;
 incrementer(&a);
 cout <<" apres incrementation a = "<< a << endl;
 return 0;
 }
```

2-

```
#include <iostream>
 using namespace std;
      void permuter_per_ref(int &a, int &b) {
     int c = a;
     a = b;
```

```
    b = c; }
void incrementer_par_ref(int &a) {
    a++;
    }
    int main() {
    int a = 1;
    int b = 2;
permuter_per_ref(a, b);
    cout << "apres permutation a = " << a << " b = " << b << endl;
incrementer_par_ref(a);
    cout <<" apres incrementation a = "<< a << endl;
return 0;
}
```

Exercice 7

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit trier le tableau par ordre croissant et doit afficher le tableau.
Algorithme suggéré (tri bulle) :
On parcourt le tableau en comparant t[0] et t[1] et en échangeant ces éléments s'ils ne sont pas dans le bon ordre.

1. On recommence le processus en comparant t[1] et t[2],... et ainsi de suite jusqu'à t[8] et t[9].
2. On compte lors de ce parcours le nombre d'échanges effectués.
3. On fait autant de parcours que nécessaire jusqu'à ce que le nombre d'échanges soit nul : le tableau sera alors trié.

```
  #include <iostream>
  using namespace std;
int main() {
int tab[10];
      cout << "Entrer 10 entiers: " << endl;
   for (int i = 0; i < 10; i++) {
      cin >> tab[i];
   }
for (int i = 0; i < 9; i++) {
for (int j = 0; j < 9-i; j++) {
   if (tab[j] > tab[j + 1]) {
  int temp = tab[j];
  tab[j] = tab[j + 1];
  tab[j + 1] = temp;
  }
 }
}
       cout << "table triee: ";
for (int i = 0; i < 10; i++) {
       cout << tab[i] << " ";
}
       cout << endl;
return 0;
```

```
}
```