```
/*
 Correction du TP2 de POO
 6 fevrier 2024
 (c) Wilfrid Lefer - UPPA
 The code est a usage des etudiants de Licence Informatique de l'UPPA annee 2023-2024 et
 ne doit en aucun cas etre diffuse de quelque maniere que ce soit !
*/

#include <unistd.h>
#include <stdio.h>
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

/// @def L'aide a l'utilisateur avec le synopsis de la commande en ligne
#define USAGE() { \
 printf("Usage: %s [-c] [-v] [-n <size>]\n", argv[0]); \
 exit(1); \
}

/// Taille du tableau dans le cas d'une allocation statique
#define MAX 10

/// Version C : lit un entier au clavier
/// Version avec purge de l'entree standard si chaine non convertible
/// @return l'entier lu
int
C_LitEntier()
{
 int n;

 while (1) {
  switch (scanf("%d", &n)) {
  case 0:
   getchar();
   break;
  case EOF:
   fprintf(stderr, "Erreur : rupture de flux avant lecture de l'ensemble des valeurs\n");
   exit(1);
  default:
   return n;
  }
 }
}
```

```
/// Version C++ : lit un entier au clavier
/// @return l'entier lu
int
CPP_LitEntier()
{
 int n;
 char c;

 // Tant qu'on n'est pas parvenu a lire un entier valide
 while (1) {
  cin >> n;
  /*
  cout << "lu : " << n << endl;
  if (cin.fail())
   cout << "cin.fail()" << endl;
  if (cin.eof())
   cout << "cin.eof()" << endl;
  */
  // Si la lecture d'un entier valide n'a pas ete possible (ex : "pt45")
  if (cin.fail()) {
   // Si on a atteint la fin du flux d'entree (^D entre au clavier) on ne peut plus lire et on renvoie donc -1 (valeur non valide)
   if (cin.eof())
        return -1;
   // Purger le buffer d'entree de tous les caracteres qui sont indesirables
   cin.clear(); // reinitialise l'etat du stream sinon si cin.fail() les prochaines utilisations vont echouer
   do {
        c = cin.get();
        //cout << "j'ai lu '" << c << "'" << endl;
   } while (c != EOF && !isdigit(static_cast<unsigned char>(c)));
   // S'il n'y a plus rien a lire on renvoie une valeur non valide qui sera traitee selon situation
   if (c == EOF)
        // Cette fois on ne genere pas une erreur et terminaison de programme afin de pouvoir faire le 3e cas (nombre de valeurs
inconnu au depart)
        return EOF;
   // Sinon on a lu un caractere a priori valide (separateur ou chiffre) et il faut le remettre dans le flux
   cin.unget();
  } else break;
 }
 return n;
}

// Compare 2 valeurs entieres et renvoie -1, 0 ou 1 selon
int
int_compare(const void *v1, const void *v2)
{
 int i1=*((int *)v1);
 int i2=*((int *)v2);
 return i1<i2 ? -1 : i1>i2 ? 1 : 0;
}

int
main(int argc, char *argv[])
{
 char opt;
 int c=0;
 int n=0, *t;
 vector<int> tt;

 while ((opt = getopt(argc, argv, "cvn:")) != -1) {
  switch (opt) {
  case 'n':
   n = atoi(optarg);
   if (n > 0)
    break;
   USAGE();
  case 'c':
   c = 1;
   break;
  case 'v':
   c = 2;
   break;
  default:
   USAGE();
  }
 }
 if (optind != argc)
  USAGE();
```

```
 // Saisie des valeurs du tableau
 switch (c) {
 case 1:
  if (n == 0) {
   printf("Nombre de valeurs a saisir (maximum : %d) : ", MAX);
   n = C_LitEntier();
  }
  // Allocation dynamique du tableau
  if ((t = (int *)malloc(n*sizeof(int))) == NULL) {
   fprintf(stderr, "Memory allocation failed\n");
   exit(1);
  }
  printf("Entrez les valeurs du tableau : ");
  for (int i=0 ; i<n ; i++)
   do
        t[i] = C_LitEntier();
   while (t[i]<1 || t[i]>9);
  // Tri du tableau
  qsort(t, n, sizeof(int), int_compare);
  // Impression du contenu du tableau
  printf("Contenu du tableau :");
  for (int i=0 ; i<n ; i++)
   printf(" %d", t[i]);
  putchar('\n');
  free(t);
  break;
 case 0:
  if (n == 0) {
   do {
        cout << "Nombre de valeurs a saisir (maximum : " << MAX << ") : ";
        n = CPP_LitEntier();
   } while (n<1 || n>MAX);
  }
  // Allocation dynamique du tableau
  t = new int[n];
  cout << "Entrez les valeurs du tableau : ";
  for (int i=0 ; i<n ; i++)
   while (1) {
        t[i] = CPP_LitEntier();
        if (t[i]<1 || t[i]>9) {
         if (t[i] == EOF) {
          fprintf(stderr, "Erreur : rupture de flux avant lecture de l'ensemble des valeurs\n");
          exit(1);
         }
        } else break;
   }
  // Tri du tableau
  qsort(t, n, sizeof(int), int_compare);
  // Impression du contenu du tableau
  cout << "Contenu du tableau : ";
  for (int i=0 ; i<n ; i++)
   cout << t[i] << " ";
  cout << endl;
  delete[] t;
  break;
 case 2:
  cout << "Entrez les valeurs du tableau : ";
  while (1) {
   int val = CPP_LitEntier();
   if (val>0 && val<10)
        tt.push_back(val);
   if (cin.eof())
        break;
  }
  // Tri du tableau
  sort(tt.begin(), tt.end());
  // Impression du contenu du tableau
  cout << "Contenu du tableau : ";
  for (int i : tt)
   cout << i << " ";
  cout << endl;
 }
 return 0;
}
```