Sup Galilée
Mathématiques Appliquées et Calcul Scientifique
Deuxième année
Année universitaire 2018–2019

Langage C++(G4SCCP)
Cours : X. Juvigny
TPs : S. Landier
samlandier@gmail.com

Travaux pratiques N°3

Les bases du langage C++

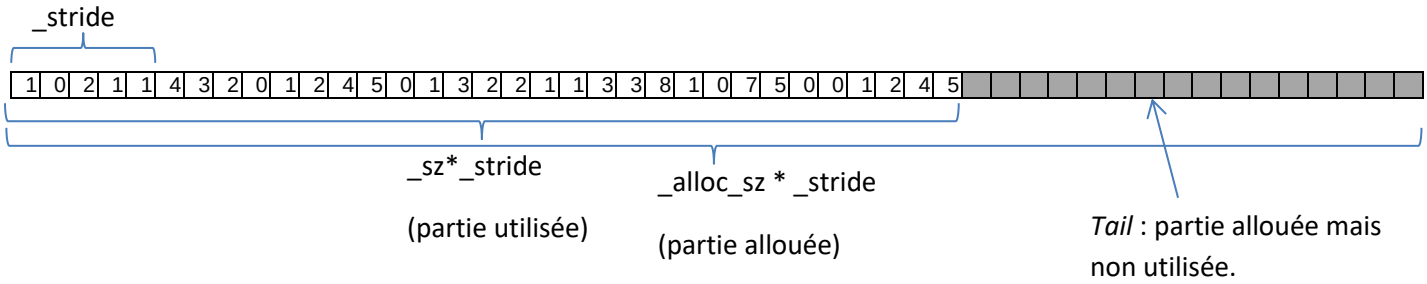
https://github.com/Macs1718/Promotion_2020
https://en.cppreference.com/w/
https://stackoverflow.com/

Exercice 11 (classe IntArray)

L'objectif est de créer une classe IntArray qui représente un type de tableau dynamique pouvant stocker des n-uplets d'entiers, *n* étant choisi lors de l'instanciation d'un objet :

IntArray t(3) ; -> tableau pouvant stocker par exemple des triangles

IntArray q(4) ; -> '' '' '' '' '' des quadrangles



11.1 Définir la classe IntArray dont l'interface est :

```
class IntArray
{
public:
  IntArray(int stride); // stride == taille des n-uplets
  ~IntArray();
  IntArray() = delete; // Que signifie cette instruction ?

  // accessors
  bool empty() const;       // retourne vrai si la taille utilisée est nulle
  int size() const;         // retourne la taille utilisée
  int stride() const;       //retourne la taille des n-uplets
  const T* get(int i) const;// renvoie un pointeur sur le ième n-upplet.
  int alloc_sz() const      // renvoie la taille allouée (en nb de n-uplets)

  // modifyers
  void clear();                 // mets la taille utilisée à 0.
```

```
  void reserve(int n);          // alloue (si besoin) la memoire pour n n-uplet,
                                // et preserve les donnees existantes
  void destroy();               // desalloue la memoire
  void resize(int n, int val);  // en plus de ce que fait reserve, initialisation avec
                                // val de la partie utilisée non encore renseignée
  void add(T* elt);             // ajoute un n-uplet

private:
  ?? _data;                     // tableau des donnees
  int _stride;                  // taille des n-uplets
  int _sz, _alloc_sz;           // nb de n-uplet, nb max de n-uplets
};
```

Quel type choisir pour _data ?

Quelques infos sur certaines fonctions à implémenter :

1. reserve
   Cette fonction gère l'allocation mémoire, elle change le _alloc_sz. Si la taille spécifiée est inférieure à la taille déjà allouée, il n'y a rien à faire. Si en revanche on demande plus de mémoire que déjà allouée, alors :
   - on effectue une nouvelle allocation,
   - on migre les données (si elles existent) vers le nouvel emplacement,
   - on désalloue l'ancien emplacement,
   - on met à jour les attributs pertinents de IntArray.

2. resize
   Cette fonction change la taille utilisée du tableau (_sz). Ce changement peut nécessiter une réallocation (et donc un appel à reserve) si la taille demandée est supérieur à la taille déjà allouée.
   La partie rendue utile (passée de non utilisée à utilisée) est initialisée avec la valeur donnée en argument.

3. add
   - Si la taille allouée n'est pas suffisante pour ajouter un élément à la fin, doubler la taille allouée
   - Assigner la valeur spécifiée
   - Mettre à jour les attributs pertinents de IntArray.

11.2 Comparaison des performances et de l'occupation mémoire

Ecrire une boucle qui ajoute N n-uplets dans un IntArray<n> (fonction add).

Afficher et comparer les temps d'exécution et l'occupation mémoire (alloc_sz) de 2 exécutions, l'une sans appel à reserve avant la boucle et l'autre avec. Conclusions ?

Exercice 12 (classe Array<T>)

Créer un fichier Array.h et y templatiser IntArray en : template<typename T> Array<T> pour gérer notamment les tableaux de coordonnées.
Le fichier Array.h contiendra également le corps des fonctions.

Déclaration de la classe :

```
template <typename T>
class Array
{…} ;
```

Pour l'implémentation des fonctions, vous avez le choix entre :

- Mettre le corps de la fonction (i.e. sa définition) dans le scope de la classe (pas forcément très lisible pour les grosses fonctions, mais parfaitement valide)

```
template <typename T>
class Array
{
public:
  …
 void foo(int a, double b){

   // corps de la fonction
 }
} ;
```

- Séparer les définitions des déclarations, mais dans le même fichier :

```
template <typename T>
class Array
{
public:
  …
 void foo(int a, double b) ;
} ;

template <typename T>
void Array<T>::foo(int a, double b){

   // corps de la fonction
}
```

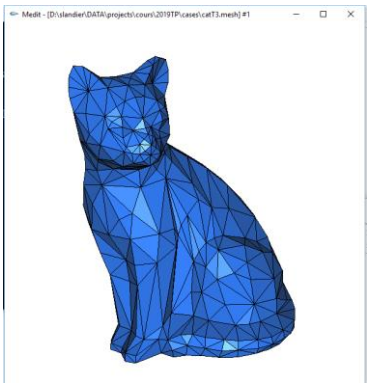
Exercice 13 (entrées/sorties maillage)

Aller sur le github est télécharger le fichier medit.hxx qui fournit 2 fonctions (read et write) permettant de lire/écrire des fichiers de maillage au format MEDIT. Télécharger également le fichier de test catT3.mesh (répertoire mesh).

1) Vérification de l'accès au logiciel medit.
   Dans un terminal, dans le répertoire contenant catT3.mesh, effectuer la commande :

   /LOCAL/medit-3.0/medit-linux catT3.mesh

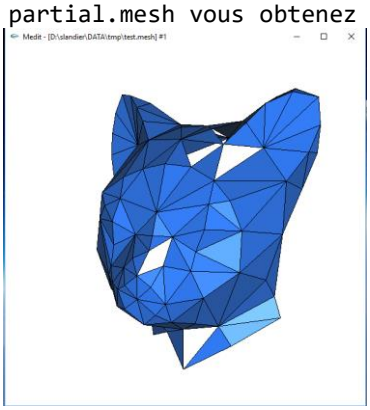
Vous devriez voir la fenêtre suivante apparaître :



2) Vérification de la bonne implémentation de la classe Array.
   Dans un fichier exo13.cpp, inclure votre Array.h, ce fichier medit.hxx et créer le main suivant qui permet de lire le fichier catT3.mesh (contient exclusivement des triangles) et le réécrire partiellement dans un autre fichier partial.mesh:

```
int main()
{
  Array<int> cntE2(2), cntT3(3), cntQ4(4), cntTH4(4), cntHX8(8);
  Array<double> crd(3);

  medith::read("../catT3.mesh", crd, cntE2, cntT3, cntQ4, cntTH4, cntHX8);

  // On ne prend que les 180 premiers triangles

  cntT3.resize(180, 0);

  medith::write("partial.mesh", crd, cntT3, "TRI");
  return 0;
}
```

Si votre code compile sans erreur, s'exécute sans erreurs et que lorsque vous ouvrez avec medit le fichier partial.mesh vous obtenez :



C'est gagné ! le TP est terminé.

Sinon, votre implémentation de Array coince quelque part…Pour debugger et trouver le ou les problèmes, il faut s'assurer que chaque fonction effectue bien le cahier des charges spécifiés (règles de gestion) et que les attributs sont cohérents avec le contenu, c'est-à-dire que _sz, alloc_sz sont mis à jour à chaque fois que la taille du tableau ou son contenu est modifié..

Si les messages obtenus lors de la compilation ou de l'exécution ne sont pas assez explicites ou si vous ne parvenez pas à pointer la fonction fautive, n'hésitez pas à ajouter des messages d'affichage dans chaque fonction pour traquer les erreurs.

Bon courage !