Allocation Dynamique

int \*variable = NULL;

variable = new int; | tableau = new int[taille];

delete variable; | delete[] tableau;

Fonction valeur par défaut

int nombreDeSecondes(int heures, int minutes = 0, int secondes = 0);

Seulement dans le PROTOTYPE !!

Les valeurs par défaut doivent se trouver à la fin de la liste des paramètres ("à droite").

Objet en pointeur

Arme \*m\_arme; comme attribut de Personnage

Personnage::Personnage() : m\_arme(0), m\_vie(100), m\_mana(100){

m\_arme = new Arme();

}

Modifier le destructeur

Personnage::~Personnage(){

delete m\_arme;

}

Surcharge d’opérateur

Prototype en dehors de la classe sauf cas +=

Duree operator+(Duree const& a, Duree const& b);

std::ostream& operator<<(std::ostream& out, Duree const& duree);

bool operator==(Duree const& a, Duree const& b);

Tableau statique

int tab[var const];

void fonction(double tableau[]){..}

Tableau dynamique

#include <vector>

vector<int> tableau(5, 3); //Crée un tableau de 5 entiers valant tous 3

tableau.push\_back(8);

tableau.pop\_back();

tableau.size();

ajouté virtual seulement dans le prototype, le .h

Il faut utilisé des reference ou des pointeur par la suite.

virtual pure

class Vehicule{

virtual int nbrRoues() const = 0;

};

On écrit friend suivi du prototype de la fonction et on place le tout à l'intérieur de la classe.

Template

Pour les fonctions templates, c'est différent. TOUT doit obligatoirement se trouver dans le fichier .h, sinon votre programme ne pourra pas compiler.

template <typename T> ou <class T>

T maximum(const T& a, const T& b){ifa<b return a...}

Il faut juste indiquer entre des chevrons quelle « version » de la fonction on souhaite utiliser : maximum<double>(pi,e);

Plusieurs para :

template<typename T, typename S>

S moyenne(T tableau[], int taille){...}

Specialisation :

template <>

string maximum<string>(const string& a, const string& b){…}

Il faut respecter un ordre particulier la fonction générique puis les fonctions spécialisées.

Class template :

template <typename T> ou <class T>

class Rectangle{

//…

};

Si on veut séparer la declaration du corps, on doit quand meme reste dans le .h

template <typename T>

class Rectangle{

public:

T hauteur() const;

};

template<typename T>

T Rectangle<T>::hauteur() const{

return m\_haut-m\_bas;

}

Instanciation d'une classe template

Rectangle<double> monRectangle(1.0, 4.5, 3.1, 5.2);