Commentaires synthétiques sur vos rendus

① De l'utilisation des bons formats d'affichage pour les flottants en C

- Relire le début du Chap. 10 du Overton : "Float and Double, Input and Output".
- b32 = 8 décimales
  b64 = 16 décimales
  ② au sens de égal ici
- éviter %f (1 arg entre 1 et 10 ...)
  préférer %e et spécifier le # de décimales
  %le (pour scanf)
  en cohérence avec le format.
- printf traite des double. Conversion implicite
  Un ni format par défaut est celui du float.

② De la conversion implicite de types numériques (par promotion).

- Ce point est toujours "piégeux" en C, en particulier lors de l'évaluation d'expressions (qui regroupent plusieurs opérations numériques).

- S'ajoute à ça les évolutions de la norme du C, les spécificités de certaines bibliothèques (C++) et les choix des "constructeurs" (des compilateurs dans ce cas).
- La règle générale en C est la conversion implicite au type "le plus précis" des opérandes, qui devient aussi celui du résultat.
- Les fonctions non suffixées sont à paramètres/valeurs double : log : double → double
- Il existe cependant 2 situations qui créent de la confusion :
  - logf : version suffixée pour un param/valeur float (math.h)
  - log peut être générique et ni pas d'ambiguïté, être similaire à logf. (<cmath>)

Voir exemple

- Conclusion : le cast (conversion explicite) est votre ami

		C	C++
float	a = 1.0f	a est promu en double par l'appel à sin qui calcule et renvoie un double	sin générique calcule avec a en float et renvoie un float qui est promu en double par l'affectation de b
double	b = sin(a)		

③ Contraction d'expression

C99 : "Une expression flottante peut être contractée en une expression atomique". Ainsi $r = a \times b + c$ peut être effectué avec un fma, c.à.d avec une précision intermédiaire supérieure de façon à garantir l'arrondi correct de r.

Les compilateurs proposent des options d'optimisation qui généralisent ce comportement. Exemple :
- -ffast-math (gcc) — à ne pas utiliser —

④ FLT_EVAL_METHOD

C99 clarifie le comportement par défaut de l'évaluation des expressions avec cette constante nommée.
Voir lien sur moodle.

Conseil : les variables intermédiaires sont vos amies

Conclusion

- Bien utiliser le chap. 10 du Overton
- Lire la doc de son compilo.
- Ne faire aucune hypothèse "par défaut"
- Si doute : cast + var. intermédiaires.