À ficher pour la rentrée janvier 2022 : Langage C

Éléments des expressions

→ Types de base : `int, float, bool, char, char*`
Citer par exemple: `1.5` avec un point, `true, false` sans majuscules, `'a'` mais `"bjr"`

→ Autres types pour les nombres
Utiliser le tableau récap fait en TP, avec le nombre d'octets utilisés

→ Opérateurs arithmétiques sur les `int` et `float` : `+, *, -, / , %`
préciser leur arité, type int ou float du résultat, différence de division selon le type des opérandes

→ Opérateurs de comparaison : `==, !=, <, <=, >, >=`
Attention `==` n'est pas magique !

→ Opérateurs booléens `!, &&, ||`
expliquer l'évaluation paresseuse

Affichage, saisie, lecture, écriture

→ spécificateurs de format `%d, %f, %c, %s` *et autres ...*

→ caractères spéciaux `\n, \t ,\b, \0`

→ affichage : syntaxe `printf`,
noter que les arguments sont évalués avant l'affichage, rappeler de mettre `\n` *pour débugger*

→ saisie : syntaxe pour `scanf`
et expliquer pourquoi on passe une adresse en argument, + éviter de mettre `%s`

→ lecture/écriture dans un fichier avec `cat`, `|` et `getchar`, opérateurs de redirection `>`

Chaînes de caractères

→ Particularité des `char*` le caractère de fin `\0`

→ Gestion particulière des chaînes explicites (constantes, non modifiables)

→ Fonctions classiques (longueur, test d'égalité..) : à la main, avec `string.h`

Les instructions et structures de contrôle

→ déclaration, initialisation et modification de variables.

→ Fonction : déclaration, définition et appel, `return` et `void`
+ expliquer la passage par valeur (copie locale des arguments) et les variables locales

→ Alternative (`if, else, else if`)
préciser quand on peut omettre les accolades

→ Boucle `while`

Organiser et utiliser son code

→ Structure d'un programme C : `#include`/ définitions de fonction/ `main`/ .../ `return 0;`

→ Structure d'un "gros" code C : organisation du code en `xxx.h`, `xxx.c`, `test_xxx.c`

→ Compilation simple, compilation séparée *Revoir énoncé projet*

→ *(facultatif)* `Makefile`: utilisation *cf. énoncé projet*, création *Cf. code du projet*

→ Passage de paramètres à un programme C (`argc,argv`)

→ Exécution d'un programme C en ligne de commande, y compris avec paramètres

Pointeurs

→ Organisation de la mémoire, bits, octets, adresse mémoire, différence entre pile et tas

→ Pointeurs : définition, type, codage sur 8 octets, comment en afficher, opérateurs `*`, `&`, `[i]`

→ Passage par référence vs passage par valeur à une fonction

→ Allocation dynamique d'espace mémoire : `malloc`, `sizeof`, `free`
+ explications de à quel moment on libère la mémoire...

Tableaux statiques et structures

→ Tableaux statiques : déclaration avec constante littérale, initialisation explicite dès la déclaration, accès aux valeurs

→ Allocation dynamique d'espace mémoire : `malloc`, `sizeof`, `free`

→ Structures : déclaration d'un "type", déclaration d'une instance, initialisation explicite dès la déclaration, accès aux champs avec `.` ou avec `->` si on a un pointeur

→ Renommage de types avec `typedef`

Divers

→ Commentaires, sur une ligne `//` ou plusieurs `/*...*/`

→ Jeu de tests avec `assert`
+ exemple pour une fonction booléenne, pas de `==true`...

→ Quelle librairie inclure pour quoi
citer les usages fréquents, pas besoin d'être exhaustif

→ Erreurs classiques, à la compilation ou à l'exécution, *par exemple les "segmentation fault", "undefined reference to"... entre autre cf projet pour les problèmes de compilation*