

Projet C : Nombres Binaires

Avant de commencer : la qualité des commentaires, avec notamment la présence des antécédents, des conséquents, des invariants de boucle, les rôles de chacune des fonctions, ainsi que les noms donnés aux variables, l'emploi à bon escient des majuscules et la bonne indentation rentreront pour une part importante dans l'appréciation du travail. Ce projet doit permettre de montrer votre autonomie et votre compréhension tant dans la conception du programme que dans sa réalisation. Enfin, si les codes de plusieurs projets se trouvent être identiques, ou être copiés depuis le web, tous les projets concernés seront immédiatement sanctionnés par un zéro.

1 Type binaire

On souhaite représenter un entier long non signé (**unsigned long int**) sous forme **binaire** à l'aide d'un **tableau** d'entiers courts (**short**) contenant des 0 et des 1. Le bit de poids faible (2^0) est à l'indice 0, et le bit de poids fort ($2^{\text{NB_BITS}-1}$) est à l'indice **NB_BITS**-1. Par exemple, si **NB_BITS** est égal à 8, le tableau qui contient {0, 1, 1, 0, 1, 0, 0, 0} représente l'entier décimal non signé 22. On définira le type **binaire**, comme suit :

```
typedef short binaire[NB_BITS];
```

1) À l'aide d'un **define**, votre programme commencera par définir la constante **NB_BITS** égale au nombre d'éléments du tableau nécessaires à la représentation d'un entier long non signé (**unsigned long int**).

2) Vous définirez ensuite les procédures et fonctions de manipulation des nombres binaires suivantes :

- **longIntToBinaire** qui initialise un **binaire** à partir d'un **unsigned long int** passé en paramètre ;
- **binaireToLongInt** qui renvoie la conversion d'un **binaire** en **unsigned long int** ;
- **printBinaire** qui écrit un **binaire** sur la sortie standard ;
- **printlnBinaire** qui écrit un **binaire** sur la sortie standard suivi d'un passage à la ligne ;
- **LireBinaire** qui lit un **binaire** sur l'entrée standard ;
- **affecter** qui affecte un nombre **binaire** à un autre ;
- **et** , **ou**, et **xou** qui calculent respectivement la conjonction, la disjonction et la disjonction exclusive de deux **binaires**. Le résultat est dans un 3ème paramètre **binaire** ;
- **plus** , **moins**, **mult**, **div** et **rest** qui calculent respectivement la somme, la différence, le produit, la division euclidienne et le reste de la division euclidienne de deux **binaires**. Le résultat est dans un 3ème paramètre **binaire** ;
- **decalageGauche** qui produit un décalage de **n** bits par la gauche d'un **binaire**.
- **decalageDroit** qui produit un décalage de **n** bits par la droite d'un **binaire**.

3) Vous écrirez une fonction **main** qui teste toutes vos fonctions et procédures précédentes. Par exemple, le code suivant :

```
binaire b1, b2;

longIntToBinaire(64, b1);
plus(b1, b1, b2);
printf("%lu\n", binaireToLongInt(b2));
decalageGauche(b2, 4);
printf("%lu\n", binaireToLongInt(b2));
```

écrit sur la sortie standard :

```
128
2048
```

2 Calculette binaire

À l'aide des routines précédentes, écrivez un programme qui réalise *une calculette binaire*. Vous proposerez à l'utilisateur une interface textuelle permettant d'accéder à toutes les fonctionnalités précédentes. Vous pouvez bien sûr en ajouter de nouvelles.

L'interface textuelle pourra être également développée à l'aide de la bibliothèque **ncurse** (<https://invisible-island.net/ncurses/>). Ce n'est pas une obligation, mais ce sera un plus pour votre projet.

3 Remise du projet

Votre projet est à faire en *binôme*. Il est à rendre au plus tard :

Le 16 janvier 2022, 23h59

Vous déposerez votre projet dans l'espace *Projet n. 1* du cours de *Langage C* sur le moodle de l'université sous forme une **archive** (tar) **comprimée** (gz) : **binaire-nom1-nom2.tgz** avec **nom1** et **nom2** les noms de chaque étudiant du binôme.

3.1 L'archive devra contenir :

- votre fichier source **.c** correctement documenté (chaque fonction doit avoir un commentaire, les invariants de boucle doivent être marqués), indenté, et codé (les noms de variables explicites, éviter les trop longues fonctions) ;
- un fichier **Documentation** au **uniquement** format **pdf** et décrivant le fonctionnement général du programme, les algorithmes, ainsi que les choix de programmation ;
- la compilation avec les options **-Wall** ne doit pas donner de *warning*.

Bon travail et bon courage