

# INFO0947 : Compléments de Programmation

## Les Chiffres Romains

B. Donnet, S. Liénardy  
Université de Liège

### 1 Problème

Les chiffres romains étaient un système de numération utilisé par les Romains de l'Antiquité pour, à partir de seulement sept lettres, écrire des nombres entiers (mais pas le zéro qu'ils ne considéraient pas comme un nombre).<sup>1</sup>



(a) XIII (©J. Van Hamme, W. Vance. Ed. Dargaud (1984))



(b) Astérix (©R. Goscinny, A. Uderzo. Ed. Dargaud (1966))

FIGURE 1 – Exemples d'utilisation de chiffres romains dans la BD franco-belge.

La numérotation romaine repose sur quatre principes :

1. Toute lettre placée à la droite d'une autre figurant une valeur supérieure ou égale à la sienne s'ajoute à celle-ci ;
2. Toute lettre d'unité (i.e. *I*, *X* et *C*) placée immédiatement à la gauche d'une lettre plus forte qu'elle (de maximum deux rangs – voir la Table 1) indique que le nombre qui lui correspond doit être retranché au nombre qui suit ;
3. Les valeurs sont groupées en ordre décroissant, sauf pour les valeurs à retrancher selon la règle 2 ;
4. La même lettre ne peut pas être employée quatre fois consécutivement, sauf *M*.

Le tableau 1 donne la conversion (en chiffres arabes) des chiffres romains de base :

Chiffres romains	<i>I</i>	<i>V</i>	<i>X</i>	<i>L</i>	<i>C</i>	<i>D</i>	<i>M</i>
Chiffres arabes	1	5	10	50	100	500	1000

TABLE 1 – Correspondance chiffres romains – chiffres arabes. La force des lettres est à lire de gauche à droite. Par exemple, *L* et *C* sont les lettres qui sont supérieures à *X* de, respectivement, 1 et 2 rangs.

1. Voir [https://fr.wikipedia.org/wiki/Numération\\_romaine](https://fr.wikipedia.org/wiki/Numération_romaine) pour plus de détails

Pour connaître la valeur d'un nombre écrit en chiffres romains, il faut le lire de droite à gauche. Il suffit d'ajouter la valeur du chiffre (cfr. tableau 1), sauf s'il est inférieur au précédent, dans ce cas, on le soustrait. Ainsi :

- $XVI = 1 + 5 + 10 = 16$  (application de la règle 1,3)
- $XIV = 5 - 1 + 10 = 14$  (application des règles  $1 \rightarrow 3$ )
- $DIX = 10 - 1 + 500 = 509$  (application des règles  $1 \rightarrow 3$ )
- $MMMMCMXCIX = 10 - 1 + 100 - 10 + 1000 - 100 + 1000 + 1000 + 1000 + 1000 = 4999$  (application des règles  $1 \rightarrow 4$ )
- $MMMMDCCCLXXXVIII = 4888$  (application des règles  $1 \rightarrow 4$ )

Dans ce travail, nous vous demandons, entre autre, de construire la fonction suivante :

```
unsigned int romain_rec(char *chiffres , int n);
```

où **chiffres** est un tableau de caractères encodant un nombre romain et **n** donne le nombre de chiffres composant le nombre romain. Ainsi, pour l'invocation suivante :

```
unsigned int x = romain_rec("MMMMDCCCLXXXVIII" , 16);
```

la variable **x** contiendra la valeur 4888 après retour de la fonction **romain\_rec()**.

Dans la suite de l'énoncé, vous trouverez à la Sec. 2 une énumération qui vous liste, avec précision le travail que vous devez réaliser pour mener à bien ce projet. La Sec. 3 détaille l'archive à partir de laquelle vous devez réaliser ce travail, se concentrant en particulier sur les aspects liés à la compilation (code et rapport). La Sec. 4 détaille ce qu'on attend de vous pour la partie codage, notamment en décrivant le code qui vous est déjà fourni. La Sec. 5 décrit ce qu'on attend de vous dans la rédaction du rapport. Enfin, la Sec. 6 vous précise la deadline et les conditions de soumission du projet.

## 2 Travail à Réaliser

On vous demande d'effectuer le cheminement suivant :

1. Fournir une formulation récursive du problème ;
2. Spécifier le plus précisément et formellement possible la fonction **romain\_rec()** ;
3. Construire la fonction **romain\_rec()** en suivant *l'approche constructive* appliquée aux fonctions/-procédures récursive ;
4. Rédiger la fonction finale dans le fichier **romain.c** ;
5. Donner les traces d'exécution de la fonction **romain\_rec()** sur les cinq exemples donnés à la Sec. 1 ;
6. Démontrer de manière rigoureuse et formelle<sup>2</sup> la complexité théorique, dans le pire des cas (i.e., en utilisant la notation de Landau), de votre fonction **romain\_rec()** ;
7. Dérécursiver la fonction **romain\_rec()** en utilisant le pseudo-langage vu au cours théorique. Attention, il n'est pas demandé de fournir une solution itérative, mais bien d'éliminer la récursivité comme cela a été vu au cours et illustré dans les « Exercices dont vous êtes le Héros · l'Héroïne ».

Lors de la rédaction du code de la fonction, veillez à respecter les principes de la **programmation défensive**.

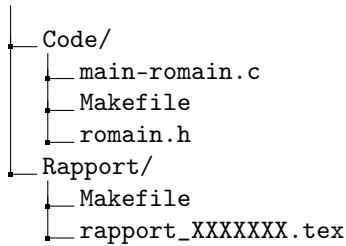
## 3 Archive

Pour réaliser le projet, vous devez télécharger l'archive **Projet.tar.gz** disponible sur eCampus. Une fois désarchivée dans le répertoire courant, vous obtiendrez l'architecture suivante :

```
./
├─ Makefile
└─ Makefile.compilation
```

---

2. Toute justification impliquant un texte argumentatif en lieu et place d'équations ne sera pas lue. Expliquez tout de même votre raisonnement.



Pour compiler votre projet (code C et rapport  $\text{\LaTeX}$ ), il vous suffit de modifier légèrement le fichier `Makefile.compilation`, uniquement les variables `TARNAME` et `RAPPORTSOURCE`. Il vous suffit de remplacer les `x` par votre matricule ULiège (e.g., `s191234`). Pensez aussi, dans le répertoire `Rapport/` à modifier le nom du fichier  $\text{\LaTeX}$  (remplacez les `X` par votre matricule ULiège – e.g., `s191234`).

Pour compiler le code source et le rapport à partir du répertoire courant, il vous suffit de faire :

```
1 $>make all
```

Pour nettoyer les différents répertoires à partir du répertoire courant, il vous suffit de faire :

```
1 $>make clean
```

Pour produire l'archive `tar.gz` à soumettre sur la plateforme de soumission, il vous suffit de faire :

```
1 $>make archive
```

## 4 Code

Un squelette de programme a été créé pour vous (la Fig. 2 vous donne une vision de haut niveau de l'architecture de votre code ainsi que les relations entre les différents fichiers sources) et est présent dans le répertoire `Code/` dans l'archive `Projet.tar.gz`. Un de vos objectifs, dans ce projet, est de compléter le code existant.

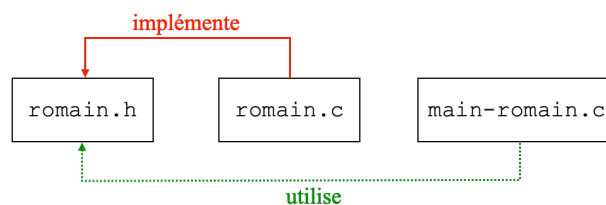


FIGURE 2 – Architecture du code.

Le programme est constitué de trois fichiers sources :

1. `romain.h` C'est le fichier de header contenant la déclaration de la fonction `romain_rec()` que vous devez compléter. **Vous ne devez pas modifier ce fichier.**
2. `romain.c` Il s'agit de l'implémentation du header `romain.h`. Il contient deux fonctions (inutile de rajouter autre chose) :
  - la fonction statique `mapping()` permet de faire la correspondance entre un chiffre romain et un chiffre arabe. Cette fonction implémente donc le tableau 1. Vous devez utiliser cette fonction.
  - la fonction `romain_rec()` que vous devez compléter avec votre code.
3. `main-romain.c` C'est le programme principal. Il contient la fonction `main()` qui implémente déjà quelques tests (i.e., les exemples donnés à la Sec. 1). Si vous le désirez, vous pouvez modifier la fonction `main()`. Dans tous les cas, lors de la correction, nous compilerons votre code avec notre propre programme principal.

## 5 Rapport

Un squelette de rapport, rédigé en  $\text{\LaTeX}$ , a été créé pour vous et est présent dans le répertoire `Rapport/` dans l'archive `Projet.tar.gz`. Un de vos objectifs, dans ce projet, est de compléter le rapport existant.

Le rapport est composé de six sections (que vous devez compléter), inspirées du format des documents « Un exercice dont vous êtes le Héros · l'Héroïne » :

**Formulation Récursive** Dans cette section, vous fournissez et expliquez votre formulation récursive du problème (voir Sec. 4.2.2. du calcul d'une exponentielle pour un exemple).

**Spécifications** Dans cette section, vous fournissez et discutez la spécification formelle de la fonction `romain_rec()` (voir Sec. 4.3.3. du calcul d'une exponentielle pour un exemple).

**Construction Récursive** Dans cette section, vous fournissez et discutez la construction formelle (avec les assertions intermédiaires) de la fonction `romain_rec()` (voir Sec. 4.4.2 et Sec. 4.5 du calcul d'une exponentielle pour un exemple).

**Traces d'Exécution** Dans cette section, vous fournissez et discutez les traces d'exécution de la fonction `romain_rec()` pour les exemples donnés à la Sec. 1. En particulier, le fichier  $\text{\LaTeX}$  contient, en commentaires, une trace d'exécution pour un exercice sur les Piles (voir « Un exercice dont vous êtes le Héros · l'Héroïne »). Inspirez-vous du code  $\text{\LaTeX}$  pour produire vos traces d'exécution.

**Complexité** Dans cette section, vous fournissez et justifiez la complexité théorique de la fonction `romain_rec()` (voir Sec. 4.6.2 du calcul d'une exponentielle pour un exemple).

**Dérécursification** Dans cette section, vous fournissez et discutez la dérécursification de la fonction `romain_rec()`. Attention, il n'est pas question ici de fournir un algorithme itératif mais bien d'éliminer la récursivité comme cela a été vu au cours. La solution doit être proposée en utilisant le pseudo-code vu au cours (voir Sec. 9.6 et 9.7 du calcul du nombre d'occurrences d'un caractère).

Pensez à modifier l'en-tête du fichier  $\text{\LaTeX}$ , en particulier les déclarations des commandes `\intitule` (titre du travail), `\Prenom` (votre prénom), `\Nom` (votre nom de famille) et `\matricule` (votre matricule ULiège, au format `syyxxxx`, e.g., `s191234`).

## 6 Agenda

La participation à ce projet est obligatoire et remplace, vu les circonstances sanitaires, l'examen oral. Aucun étudiant ne peut s'y soustraire, sous peine de se voir attribuer une note d'absence pour le cours.

Puisque le projet a pour vocation de remplacer l'examen, il doit être réalisé **seul**.

L'archive `tar.gz` générée par le Makefile fourni (elle contiendra votre code, le fichier source du rapport et le PDF correspondant) doit être postée sur la plateforme de soumission (<https://submit.montefiore.ulg.ac.be>) avant le **28/05/2020, 18h00**. Toute soumission postérieure à cette date ne sera pas prise en compte. Comme l'heure de l'horloge du serveur de soumission et celle de votre ordinateur peuvent être légèrement différentes, il est **fortement déconseillé** de soumettre votre projet à 17h59 et 59 secondes : ne prenez pas de risque inutile.