INFO0947 : Compléments de Programmation
Récursivité et Élimination de la Récursivité

B. Donnet
Université de Liège

1 Système de Numération

Un *système de numération* d'un nombre naturel N est constitué :

1. d'une *base*, $b \in \mathbb{N}$ $(b > 1)$
2. d'un ensemble de b symboles, appelés *chiffres*, compris entre 0 et $b-1$.

La représentation d'un naturel N via le système de numération repose sur la structure suivante :

$$\begin{aligned} N &= a_{n-1} \times b^{n-1} + a_{n-2} \times b^{n-2} + \ \ldots \ + a_1 \times b^1 + a_0 \times b^0 \\ &= \sum_{i=0}^{n-1} a_i \times b^i \\ &= (a_{n-1}a_{n-2} \ \ldots \ a_1 a_0)_b \end{aligned}$$

avec $a_i \in \{0, 1, \ \ldots \ , b-1\}$.

Les systèmes de numération les plus classiques sont les suivants :

décimal : $b = 10$ et l'ensemble des symboles est $\{0, 1, \ \ldots, 9\}$. Par exemple, $(77)_{10}$.

binaire : $b = 2$ et l'ensemble des symboles est $\{0, 1\}$. Par exemple, $(77)_{10}$ est représenté, en binaire, par $(1001101)_2$. La base binaire est utilisée pour le codage de l'information dans un ordinateur.

octal : $b = 8$ et l'ensemble des symboles est $\{0, 1, \ \ldots, 7\}$. Par exemple, $(77)_{10}$ est représenté, en octal, par $(115)_2$. La base octale a longtemps été utilisée aux débuts des ordinateurs. Elle a, petit à petit, laissé sa place à la base hexadécimale.

hexadécimal : $b = 16$ et l'ensemble des symboles est $\{0, 1, \ \ldots, 9, A, B, \ \ldots, F\}$. Par exemple, $(77)_{10}$ est représenté, en octal, par $(4D)_{16}$. C'est le système utilisé actuellement pour la représentation des adresses mémoires.

2 Problème

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

```
1 unsigned int hexa_dec_rec(char *hexa, int n);
```

où `hexa` est un tableau de caractères encodant un nombre en base hexadécimale et `n` donne le nombre de symboles composant le nombre hexadémical. Le tableau 1 donne la conversion (en symboles décimaux) des symboles hexadécimaux de base.

Ainsi, pour le code suivant :

```
1 #include <stdio.h>
2
3 int main(){
4   char hexa1[] = "27";
```

Symboles hexadécimaux	0	1	2	3	4	5	6	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*
Symboles décimaux	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

TABLE 1 – Correspondance symboles hexadécimaux – symboles décimaux.

```
  char hexa2[] = "A23";
  char hexa3[] = "A78E";

  printf("%s: %u\n", hexa1, hexa_to_dec_rec(hexa1, 2));
  printf("%s: %u\n", hexa2, hexa_to_dec_rec(hexa2, 3));
  printf("%s: %u\n", hexa3, hexa_to_dec_rec(hexa3, 4));

  return 0;
}//fin programme
```

le résultat affiché sur la sortie standard sera :

```
27: 39
A23: 2595
A78E: 42894
```

Dans la suite de l'énoncé, vous trouverez à la Sec. 3 une énumération qui vous liste, avec précision le travail que vous devez réaliser pour mener à bien ce projet. La Sec. 4 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. 5 détaille ce qu'on attend de vous pour la partie codage, notamment en décrivant le code qui vous est déjà fourni. La Sec. 6 décrit ce qu'on attend de vous dans la rédaction du rapport. Enfin, la Sec. 7 vous précise la deadline et les conditions de soumission du projet.

3 Travail à Réaliser

On vous demande d'effectuer le cheminement suivant :

1. Fournir une formulation récursive du problème ;
2. Spécifier le plus précisement et formellement possible la fonction `hexa_dec_rec()` ;
3. Construire la fonction `hexa_dec_rec()` en suivant *l'approche constructive* appliquée aux fonctions/procédures récursives ;
4. Rédiger la fonction finale dans le fichier `hexadecimal.c` ;
5. Donner les traces d'exécution de la fonction `hexa_dec_rec()` sur les exemples donnés à la Sec. 2 ;
6. Démontrer de manière rigoureuse et formelle [1] la complexité théorique, dans le pire des cas (i.e., en utilisant la notation de Landau), de votre fonction `hexa_dec_rec()` ;
7. Dérécursiver la fonction `hexa_dec_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 GAMECODES.

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

4 Archive

Pour réaliser le projet, vous devez télécharger l'archive `Recursivite_Hexa.tar.gz` disponible sur eCampus (Sec. Projets). Une fois désarchivée dans le répertoire courant, vous obtiendrez l'architecture suivante :

```
./
```

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

```
Makefile
Makefile.compilation
Code/
    hexadecimal.c
    Makefile
    hexadecimal.h
Rapport/
    Makefile
    rapport_XXXXXXX.tex
```

Pour compiler votre projet (code C et rapport 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., s201234). Pensez aussi, dans le répertoire `Rapport/`, à modifier le nom du fichier LaTeX(remplacez les `X` par votre matricule ULiège – e.g., s201234).

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

```
$>make all
```

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

```
$>make clean
```

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

```
$>make archive
```

5 Code

Un squelette de programme a été créé pour vous (la Fig. 1 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 `Recursivite_Hexa.tar.gz`. Un de vos objectifs, dans ce projet, est de compléter le code existant.

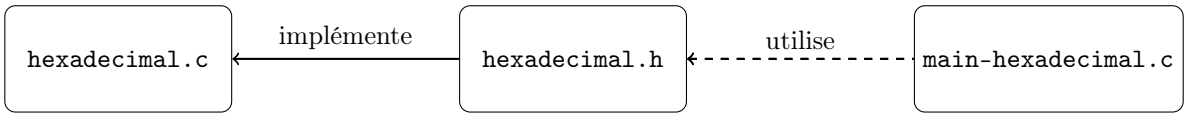

FIGURE 1 – Architecture du code.

Le programme est constitué de trois fichiers sources :

1. `hexadecimal.h` C'est le fichier de header contenant la déclaration de la fonction `hexa_dec_rec()` que vous devez compléter. **Vous ne devez pas modifier ce fichier**.
2. `hexadecimal.c` Il s'agit de l'implémentation du header `hexadecimal.h`. Il contient deux fonctions :
   — la fonction statique `convert()` permet de faire la correspondance entre un chiffre en hexadécimal et un chiffre décimal. Cette fonction implémente donc le tableau 1. Vous devez utiliser cette fonction.
   — la fonction `hexa_dec_rec()` que vous devez compléter avec votre code.
3. `main-hexadecimal.c` C'est le programme principal. Il contient la fonction `main()` qui implémente déjà quelques tests (i.e., les exemples donnés à la Sec. 2). 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.

6 Rapport

Un squelette de rapport, rédigé en LaTeX, a été créé pour vous et est présent dans le répertoire `Rapport/` dans l'archive `Recursivite_Hexa.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 GameCodes :

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

Spécification Dans cette section, vous fournissez et discutez la spécification formelle de la fonction `hexa_dec_rec()` (voir Sec. 4.3.3. du GameCode sur le 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 `hexa_dec_rec()` (voir Sec. 4.4.2 et Sec. 4.5 du GameCode sur le calcul d'une exponentielle pour un exemple).

Traces d'Exécution Dans cette section, vous fournissez et discutez les traces d'exécution de la fonction `hexa_dec_rec()` pour les exemples donnés à la Sec. 2. En particulier, le fichier LaTeX contient, en commentaires, une trace d'exécution pour un exercice sur les Piles (voir le GameCode associé). Inspirez-vous du code LaTeX pour produire vos traces d'exécution.

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

Dérécursification Dans cette section, vous fournissez et discutez la dérécursification de la fonction`hexa_dec_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 les sections associées dans les GameCodes du Chapitre 9).

Pensez à modifier l'en-tête du fichier 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., `s201234`).

7 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 avant le **25/05/2021, 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.