Phase 1 : étude préliminaire

Groupe 4 : Bourgeois Manech & Rousseau Colin

Sommaire :

Planning

Code couleur : Bleu → Manech ; Rouge → Colin ; **Noir** → Simultanément

Jeudi :

- Compréhension du sujet
- Organisation du Planning
- Organisation des supports à utiliser
- Réaliser la démonstration mathématique
- Répartition des tâches

Vendredi :

- Rédaction de l'étude préliminaire
- Evaluation des taches (difficultés / temps)
- Publier l'étude préliminaire avant 12h00
- Création des structures
- Réaliser fonctions de lecture du fichier CSV
- Ecrire algorithmes de recherche de données : ancêtres sur deux générations + fratrie d'une personne (à partir d'un ID)

Lundi

- Ecrire fonction d'exportation des données dans un fichier HTML
- Mise en place du menu
- Créer de la feuille de style CSS
- Créer de la feuille de script JS

Mardi

- Ecrire l'algorithme de recherche qui est optionnel : plus lointain ancêtre / ancêtre commun

- Explorer la bibliothèque ncurses, et l'intégrer si nous avons assez de temps
- Elaborer le Powerpoint pour la présentation

Mercredi

- Elaborer le Powerpoint pour la présentation
- Rendre le projet entre 11h00 et 12h00 (deadline 12h00)
- Présenter le projet devant le jury

Implémentation

A chaque début de cours , nous ferons une petite réunion pour se mettre au clair sur les missions à réaliser de la journée (à rattraper de la veille si retard).

A chaque fin de cours, nous ferons une petite réunion pour mettre en commun nos travaux , faire le point de ce qui est réussi ou non.

A chaque tâche réalisée, on se doit de tester la fonction plusieurs fois, si besoin on se fait corriger/aider par notre partenaire. Quand la fonction est opérationnelle , on la met en ligne sur le répertoire Github.

Nos structures :

- Personne :

```
12  typedef struct person{
13          int id;
14          int father_id;
15          int mother_id;
16          char lastname[20];
17          char firstname[20];
18          int birthday; int birthmonth; int birthyear;
19          char birthzipcode[30];
20          struct person * p_father;
21          struct person * p_mother;
22      }Person;
```

- Population :

```
13      typedef struct pop{
14          Person* tab_personne;
15          int nb_personne;
16      }Population;
```

Nos menus :

- Rechercher les ancêtres sur 2 générations d'une personne
- Rechercher la fratrie d'une personne
- Rechercher le plus lointain ancêtre d'une personne
- Recherche d'un ancêtre commun entre 2 personnes

Ensuite :

- Exportation d'un fichier HTML (création d'un dossier avec la page HTML, fichier CSS, et fichier JS + ouverture automatique de la page HTML)
- Retour au menu principal

Démonstration

Montrons que pour tout n > 1, nous avons :

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1$$

I – Initialisation

Pour n = 2, nous avons : $\sum_{i=0}^{1} 2^i = 2^0 + 2^1 = 1 + 2 = 3$

et $2^2 - 1 = 4 - 1 = 3$

La formule est donc vraie pour n = 2.

II – Récurrence

Supposons que la formule est vraie pour un certain entier k ≥ 2. C'est-à-dire :

$$\sum_{i=0}^{k-1} 2^i = 2^k - 1$$

Montrons que la formule est alors vraie pour k + 1. C'est-à-dire, nous voulons montrer que :

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

En utilisant l'hypothèse de récurrence, nous avons :

$$\sum_{i=0}^{k} 2^i = \left(\sum_{i=0}^{k-1} 2^i\right) + 2^k$$

Donc :

$$\sum_{i=0}^{k} 2^i = (2^k - 1) + 2^k$$

$$\sum_{i=0}^{k} 2^i = 2^k - 1 + 2^k$$

$$\sum_{i=0}^{k} 2^i = 2^k + 2^k - 1$$

$$\sum_{i=0}^{k} 2^i = 2 \cdot 2^k - 1$$

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

Nous avons donc montrée que si la formule est vraie pour k, alors elle est aussi vraie pour k + 1.

III – Conclusion

Par le principe de récurrence, la formule est donc vraie pour tout n > 1.

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1$$