Projet C : Ensemble de nombres premiers

KANG Jiale; **ZHANG Liyun**
Département de Ingénierie électronique, L'université de Xidian, Xi'an, Shaanxi, Chine

19 décembre 2020

Résumé: Ce reportage inclus la solution de 2 problèmes : Type Ensemble et Ératosthène. Chaque problème inclus 3 parts : l'objet, les variables et fonctions, et le résultat. On explique chaque variable et chaque fonction en détail. Et on donne les résultats dans le dernier part.

Mot-clé: tableau, fonction, solution, résultat

1. Type Ensemble

1.1. Objet

On utilise les fonctions et le tableau pour réaliser de opérer les ensembles.

- Initialise un ensemble ;
- Vide un ensemble ;
- Remplit entièrement un ensemble ;
- Teste si un ensemble est vide ou pas ;
- Teste si 2 ensembles sont égaux ou pas ;
- Affecte un ensemble à un autre ensemble ;
- Test si entier n appartient à un ensemble ;
- Ajoute un entier n à un ensemble ;
- Enlève un entier n d'un ensemble ;
- Ecrit un ensemble sur la sortie standard ;
- Ecrit un ensemble sur la sortie standard avec une ligne ;
- Calcule l'interscetion, l'union, la difference et la complementaire.

1.2. Les variables et fonctions

On utilise les déclarations suivantes :

```
#define CAPACITEMAX 1000
typedef int ensemble [CAPACITEMAX+2];
ensemble e ;
```

Le sens de ensemble e :

- **e[0]** : numero de numbre dans cet ensemble.
- **e[1]** : capacite de cet ensemble.
- **e[2 ... CAPACITEMAX]** : si le numéro i est dans l'ensemble e ou pas.

Les fonctions sont le suivants.

```
int valid(ensemble e);
int initEns(ensemble e, int n);
int vide(ensemble e);
int plein(ensemble e);
int estVide(ensemble e);
int egal(ensemble e1, ensemble e2);
int affecter(ensemble e1, ensemble e2);
int appartient(ensemble e, int n);
int ajouter(ensemble e, int n);
int enlever(ensemble e, int n);
int printEns(ensemble e);
int printlnEns(ensemble e);
int intersection(ensemble e1, ensemble e2, ensemble e3);
int Union(ensemble e1, ensemble e2, ensemble e3);
int difference(ensemble e1, ensemble e2, ensemble e3);
int complementaire(ensemble e1, ensemble e2);
int diffSym(ensemble e1, ensemble e2, ensemble e3);
```

On va expliquer les fonctions.

- **valid** : Teste si l'ensemble e est valid ou pas. Il va return -1 si e n'est pas valid.
- **initEns** : Initialise un ensemble e à vide de capacite n pour $e_0 = 0$ et $e_1 = n$.
- **vide** : Vide un ensemble e pour $e_0 = e_1 = 0$ et $e_i = 0(2 \leq i < n + 2)$.
- **plein** : Remplit entierement un ensemble e pour $e_0 = e_1$ et $e_i = 1(2 \leq i < n + 2)$.
- **estVide** : Teste si un ensemble e est vide ou pas. Si $e_0 = 0$, il est vide et return 1; si non return 0.
- **egal** : Teste si 2 ensembles $e1$ et $e2$ sont egaux ou pas pour $e1_0 = e2_0$ et $e1_i = e2_i(2 \leq i < n + 2)$.

- **affecter** : Affecte un ensembles *e2* à un ensemble *e1* pour $e2_i = e1_i (0 \leq i < n+2)$.
- **appartient** : Teste si entier *n* appartient à un ensemble *e*. Si $e_{n+2} = 1$, *n* est dans l'ensemble *e*, et return 1; si non, return 0.
- **ajouter** : Ajoute un entier *n* à un ensemble *e* pour $e_0 + +$ et $e_{n+2} = 1$.
- **enlever** : Enleve un entier *n* à un ensemble *e* pour $e_0 - -$ et $e_{n+2} = 0$.
- **printEns** : Ecrit un ensemble *e* sur la sortie standard.
- **printlnEns** : Ecrit un ensemble *e* sur la sortie standard suivi d'un passage a ligne.
- **intersection** : Calcule $e3 = e1 \cap e2$ pour $e3_i = e1_i \& e2_i (2 \leq i < n+2)$.
- **Union** : Calcule $e3 = e1 \cup e2$ pour $e3_i = e1_i | e2_i (2 \leq i < n+2)$.
- **difference** : Calcule $e3 = e1 - e2$ pour $e3 = e1 \cap (e1 \cap e2)'$.
- **complementaire** : Calcule *e2* complementaire de *e1* dans l'ensemble plein.
- **diffSym** : Calcule la difference symetrique $e3 = e1 \Delta e2$ pour $e3 = (e1 \cup e2) - (e1 \cap e2)$.

1.3. Le résultat

```
gcc -Wall -o TypeEnsemble.out TypeEnsemble.c

./TypeEnsemble.out
e1 = [ 2 19 31 ]
e2 = [ 10 19 34 ]
0
1
e1 = [ 2 19 31 33 ]
e2 = [ 10 19 33 34 ]
e1 ∩ e2 = [ 19 33 ]
e1 ∪ e2 = [ 2 10 19 31 33 34 ]
e1 - e2 = [ 2 31 ]
~e1 = [ 0 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20 21 22 23 24 25 26 27 28
29 30 32 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 ]
e1 Δ e2 = [ 2 10 31 34 ]
De Morgan : 1
affectation : e1 = e2 => e1 == e2 = 1
```

Figura 1: Le résultat de TypeEnsemble

Environnement :
* Ubuntu 18.04.5 LTS
* gcc 7.5.0

2. Ératosthène

2.1. Objet

On désire trouver tous les nombres premiers compris entre 2 et une valeur maximale $n-1$.

2.2. Les variables et fonctions

On utilise les déclarations suivantes :

```
1 #define MAXNUM 10000 + 5
2 int crible[MAXNUM];
```

Le sens de crible :

- **crible** : juge si le numéro est premier ou pas.
- **crible[i] = 0** : *i* n'est pas premier.
- **crible[i] = 1** : *i* est premier.

Les fonctions sont le suivants.

```
1 void init_crible(int n,int ens[
    MAXNUM]);
2 void trouver_premiers(int n,int
    ens[MAXNUM]);
3 void print_premiers(int n,int ens[
    MAXNUM]);
```

On va expliquer les fonctions.

- **init_crible** : Initialise le ensemble pour $e_0 = e_1 = 1$ (parce que 0 et 1 ne sont pas premiers) et $e_i = 1 (2 \leq i \leq n)$.
- **trouver_premiers** : Trouve les premiers de 2 à $n-1$. Si i est premier, $k \cdot i (k = 2, 3, \dots)$ ne sont pas premiers. Donc on l'enlève du ensemble(crible) ainsi que tous ses multiples.
- **print_premiers** : Ecrit sur la sortie standard le tableau de premires.

2.3. Le résultat

On teste le cas $n = 100$.

```
gcc -Wall -o nbPremiers.out nbPremiers.c

./nbPremiers.out
100
   2    3    5    7   11
  13   17   19   23   29
  31   37   41   43   47
  53   59   61   67   71
  73   79   83   89   97
```

Figura 2: Le résultat de Ératosthène

Environnement :
* Ubuntu 18.04.5 LTS
* gcc 7.5.0

3. Codes

On upload les codes dans le site **github**:

1 https://github.com/kjle/Project-C-2020Fall
2 https://github.com/zestcode/2020-ProjetC

et le site **paste.ubuntu.com** :

Q1 https://paste.ubuntu.com/p/W9qshBx5yB/
Q2 https://paste.ubuntu.com/p/X3s4nD8nf6/