Polytech-Sophia
Xidian University

Novembre 2020
Durée : 1 mois

Projet C : Ensemble de nombres premiers

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 Ensemble

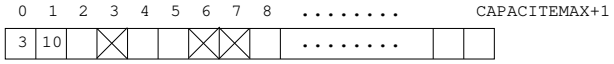
On souhaite représenter des ensembles d'entiers compris entre 0 et $n-1$. Les éléments d'un ensemble seront contenus dans un tableau de taille maximale `CAPACITEMAX+2`. Pour un ensemble e, n devra donc toujours être inférieur ou égal à `CAPACITEMAX`. On mémorisera la capacité effective d'un ensemble (*i.e.* n) à l'indice 1 du tableau. La cardinalité effective de l'ensemble sera mémorisée à l'indice 0.

La présence ou l'absence d'un entier i dans un ensemble sera donnée par la valeur mémorisée à l'indice $i+2$ dans le tableau.

Pour votre projet, vous utiliserez les déclarations suivantes :

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

Par exemple, l'ensemble $e = \{1, 4, 5\}$ d'une capacité égale à 10 sera représenté par une variable de type `ensemble`, tableau de la forme :



Les éléments de l'ensemble sont aux emplacements des croix. Choisissez la valeur la plus adéquate pour les représenter.

Pour manipuler le type `ensemble`, vous définirez les routines suivantes :

- `initEns` (2 paramètres), initialise un ensemble e à vide de capacité n ;
- `vide` (1 paramètre), vide un ensemble e ;
- `plein` (1 paramètre), remplit entièrement un ensemble e ;
- `estVide` (1 paramètre), teste si un ensemble e est vide ou pas ;
- `egal` (2 paramètres), teste si 2 ensembles $e1$ et $e2$ sont égaux ou pas ;
- `affecter` (2 paramètres), affecte un ensembles $e2$ à un ensemble $e1$;
- `appartient` (2 paramètres), teste si entier n appartient à un ensemble e ;
- `ajouter` (2 paramètres), ajoute un entier n à un ensemble e ;
- `enlever` (2 paramètres), enlève un entier n d'un ensemble e ;
- `printEns` (1 paramètre), écrit un ensemble e sur la sortie standard ;
- `printlnEns` (1 paramètre), écrit un ensemble e sur la sortie standard suivi d'un passage à la ligne ;
- `intersection` (3 paramètres), calcule $e3 = e1 \cap e2$;
- `Union` (3 paramètres), calcule $e3 = e1 \cup e2$;
- `difference` (3 paramètres), calcule $e3 = e1 - e2$;

- `complementaire` (2 paramètres), calcule $e2$ complémentaire de $e1$ dans l'ensemble plein ;
- `diffSym` (3 paramètres), calcule la différence symétrique $e3 = e1 \ \Delta \ e2$.

Dans la définition de vous routines, vous ferez bien attention à respecter le nombre de leurs paramètres et leur validité (antécédents).

2 Test

Une fois ces routines écrites, vous écrirez un programme qui les teste. Ce programme doit :

1. déclarer et initialiser 2 ensembles $e1$ et $e2$ à une capacité de 50 ;
2. ajouter $2, 19, 31$ dans $e1$ et $10, 19, 34$ dans $e2$;
3. écrire sur la s/s les ensembles $e1$ et $e2$;
4. tester si 10 et 19 appartiennent à $e1$.
5. ajouter 33 à $e1$ et $e2$;
6. écrire sur la s/s les ensembles $e1$ et $e2$;
7. écrire sur la s/s, l'intersection de $e1$ et $e2$;
8. écrire sur la s/s, l'union de $e1$ et $e2$;
9. écrire sur la s/s, la différence de $e1$ et $e2$;
10. écrire sur la s/s, le complémentaire de $e1$;
11. écrire sur la s/s, la différence symétrique de $e1$ et $e2$;
12. vérifier que le complémentaire de intersection de $e1$ et $e2$ est la réunion de leurs complémentaires (loi de De Morgan) ;
13. affecter $e2$ à $e1$ et vérifier que $e1 = e2$;

```
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
```

3 Érastosthène

On désire trouver tous les nombres premiers compris entre 2 et une valeur maximale $n-1$. Pour cela, vous utiliserez l'algorithme du *crible d'Ératosthène*.

Rappel : un nombre premier est un nombre entier positif qui n'est divisible que par 1 et par lui-même (0 et 1 ne sont pas premiers).

On commence par placer dans un ensemble, appelé *crible*, tous les nombres entiers de 2 à $n-1$. Les nombres premiers seront placés dans un ensemble *résultat*, au départ initialisé à vide. Puis, pour trouver tous les nombres premiers, on suit l'algorithme ci-dessous :

a) *le crible contient tous les nombres de 2 à n* ;
b) *le plus petit nombre contenu dans le crible est premier : l'ajouter à l'ensemble résultat* ;
c) *on l'enlève du crible ainsi que tous ses multiples* ;
d) *si le crible n'est pas vide, on recommence à partir de b)* ;

Écrivez en C la procédure `nbPremiers` qui met en œuvre cet algorithme. Cette procédure a comme donnée la borne supérieure n et comme résultat l'ensemble de tous les nombres premiers compris entre 2 et $n - 1$.

4 Remise du projet

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

Le 3 janvier 2021, minuit

1. vous enverrez à `dmei2006@xidian.edu.cn`. une archive `ensemble-n1-n2.tar.gz` avec `n1` et `n2` sont les deux derniers chiffres de numéro d'étudiant
2. vous enverrez à `vg@unice.fr` un rapport en français de 2 pages (pas moins, pas plus), *UNIQUEMENT* au format pdf, qui décrit votre projet.

4.1 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 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