Partiel de Programmation Impérative

Licence 1^{ere} année

Mardi 26 Février

Exercice 1. Face d'appel
Qu'affiche le programme suivant ? Justifier en dessinant la pile d'appel lors de l'exécution du programme.

```
void inutile(int * debut,int * fin){
  while(debut<=fin){
    *debut=*debut%2;
    debut=debut+1;
  }
}

int main(){
  int i;
  int tab[6]={1,2,3,4,5,6};
  inutile(&tab[2],&tab[5]);
  for(i=0;i<6;i++)
    printf("%d ",tab[i]);
  printf("\n");
  return EXIT_SUCCESS;
}
```

Les exercices qui suivent sont dépendants les uns des autres. Il est notamment nécessaire d'avoir définit les structures pour pouvoir écrire les fonctions qui les manipulent. Cependant, beaucoup de fonctions sont indépendantes les unes des autres. En cas de problème il est donc possible de passer aux questions suivantes. Si vous ne savez pas définir une fonction, pensez au moins à la déclarer.

Exercice 2. Structures *Écrire la définition des structures suivantes :*
- *Un point est caractérisé par ses coordonnées entières x et y.*
- *Un segment est composé de deux points a et b.*

– *Une figure est composée de segments et d'un compteur nb qui permet connaître le nombre segments. Un figure contient au plus* 100 *segments.*

Dans quel fichiers écririez vous la définition de chacune de ces structures ?

Exercice 3. Les points

Pour chacune des fonctions suivantes, écrire d'abord toutes les déclarations, puis les définitions.

– *La fonction* `creer_point` *qui prend en entrée deux entiers* x *et* y *et qui renvoit un point d'abcisse* x *et d'ordonnée* y*.*
– *La fonction* `cmp_point` *qui prend deux points en entrée, renvoit* 1 *si les deux points sont égaux et* 0 *sinon.*

Exercice 4. Les segments

Pour chacune des fonctions suivantes, écrire d'abord toutes les déclarations, puis les définitions.

– *La fonction* `creer_segment` *qui prend en entrée deux points* a *et* a *et qui renvoit un segment entre* a *et* b*.*
– *La fonction* `cmp_segment` *qui prend deux segments en entrée, renvoit* 1 *si les deux points sont les égaux et* 0 *sinon. Attention, deux segments* $[A, B]$ *et* $[B, A]$ *sont égaux.*
– *La fonction* `recherche_segment` *qui prend en entrée un tableau de segments* `tab`*, un entier* n *représentant le nombre de segments dans* `tab`*, un segment* `s` *et qui renvoit* 1 *si le segment appartient au tableau et* 0 *sinon.*
– *Quelle est la complexité en temps de la fonction* `recherche_segment` *? En espace ? Justifiez vos réponses.*

Exercice 5. Les figures

Pour chacune des fonctions suivantes, écrire d'abord toutes les déclarations, puis les définitions.

– *La fonction* `creer_figure` *qui ne prend rien en entrée, initialise une figure ne contenant aucun segment (le compteur nb est donc à* 0*).*
– *La fonction* `ajouter_segment` *qui prend en entrée un pointeur sur une figure et un segment* `s` *et ajoute* `s` *à la liste des segments uniquement si celui-ci n'y apparaît pas déjà.*
– *Quelle est la complexité de la fonction précédente ? Justifiez votre réponse.*
– *La fonction* `cmp_figure` *qui prend en entrée une figure* `F1` *et une figure* `F2` *et renvoit* 1 *si les deux figures contiennent exactement les mêmes segments. Attention : les segments des deux figures n'apparaissent pas nécessairement dans le même ordre !*
– *On note* n *le nombre de segments dans la figure* `F1` *et* m *le nombre de segments dans la figure* `F2`*. Quelle est la complexité en temps de la fonction* `cmp_figure` *? En espace ? Justifiez vos réponses.*

– *Écrire une fonction* `composition_figure` *qui prend en entrée un pointeur sur une figure* `F1`*, une figure* `F2` *et qui ajoute tous les segments de* `F2` *dans* `F1`*.*

Exercice 6. Le programme

On souhaite écrire un programme utilisant les fonctions précédemment définies.

– *Dans quel fichier va t'on écrire la fonction* `main` *?*
– *Quel(s) fichier(s) doit-on inclure ?*
– *Écrire une fonction main qui initialise trois figures :*

1. *un carré dont les coins sont aux coordonnées* $(-1, 1), (1, 1), (1, -1), (-1, -1)$
2. *un triangle aux coordonnées* $(1, 1), (1, -1), (3, 0)$
3. *une figure qui est la composition des deux autres.*

Exercice 7. Le makefile

Écrire un `Makefile` *qui permet de compiler le programme de l'exercice précédent.*