Devoir Ecrit

Le Vendredi 23 mai 2014

Durée : 2h

Sans documents, sans calculatrice

GAUTIER
Arthur

14,

L1CPI1
2013

Avant de débuter le DE :

- Prenez le temps de bien lire les énoncés des exercices. Vous répondrez dans les espaces laissés libres à cet effet dans le sujet. Aucune autre copie ne sera prise en compte.
- N'oubliez pas de reporter vos nom, prénom, groupe et école dans les espaces ci-dessous
- **La qualité de la rédaction de vos réponses entrera en compte pour la notation des exercices**

NOM GAUTIER

PRENOM Arthur

GROUPE A

Promotion L1/PL1 [X] CPI1 [ ]

Thème 1 : Structures

a) En langage C, que signifie **typedef** ?

typedef signifie que le programmeur va définir un nouveau type

b) Définissez une structure permettant de définir une personne dans le cadre d'une application bancaire.

```
Structure personne
{
    caractere nom[50];
    caractere prenom[50];
    adresse p_adresse;
    entier Solde_compte;
    entier NumCompte;
    caractere adresse_mail[50]
};
```

```
Structure adresse
{
    entier numero;
    enumeration t_voie;
    entier code postal
    caractere ville[70];
};
```

c) Soit le programme suivant, à gauche duquel on a porté les numéros des lignes :

```
1     structure t_complexe
2     {
3           reel re,im;
4     };
5     fonction principale()
6     {
7           t_complexe z1, z2;
8
9           z1 ← 1+3i;
10          z2.re ← 5.2;
11          t_complexe.im ← 4;
12          z1.re ← z2.re;
13          z1.im ← 1;
14
15          z2 ← z1+z2;
16
17          afficher(z2);
18          retourner;
19    }
```

Indiquez les lignes incorrectes de ce programme en expliquant pourquoi elles sont incorrectes (réponse page suivante)

Ligne 9: car i n'est pas défini, l'expression 1+3i est n'est donc pas valide. De plus, on ne sait pas ci quel champ on essaye d'accéder.

Ligne 11: t_complexe n'est pas une variable, on ne peut donc pas accéder à t_complexe.im

Ligne 15: Z est de type complexe, il faut donc savoir auquel de ses champs on accède et quels sont les champs que l'utilisateur peut

Ligne 17: Il faut donner le champ auquel on veut accéder

4,5

d) Soit la structure suivante décrivant une heure

```
structure t_heure
{
      entier hh, mm, ss;
};
```

e) Ecrivez les fonctions suivantes :

- Affichage d'une heure
- Saisie d'une heure
- Comparaison de deux heures
- Addition de deux heures

Affichage : entrée

```
fonction affichage(t_heure heure)
{
  afficher("Il est ", heure.hh);
  afficher("h ", heure.mm);
  afficher("min et ", heure.ss);
  afficher("s");
  retourner;
}
```

1

Saisie :

```
fonction saisie ( ) -> saisie : t_heure)
{
  t_heure heure
  afficher (" Quelle heure est-il ? ");
  Saisir (heure.hh);
  Saisir (heure.mm);
  Saisir (heure.ss);
  retourner heure;
}
```

(1)

Comparaison :

```
                                                    entrée
fonction comparaison (t_heure heure1 t_heure heure2)
{
  si (heure1.hh < heure2.hh)
  { afficher ("heure 2 est plus tôt");
  }
  sinon si (heure1.hh > heure2.hh)
  { afficher ("heure 1 est plus tôt");
  }
  sinon
  {
    si (heure1.mm < heure2.mm)
    { afficher ("heure 2 est plus tôt");
    }
    sinon si (heure1.mm > heure2.mm)
    { afficher ("heure 1 est plus tôt");
    }
    sinon
    {
      si (heure1.ss < heure2.ss)
      {
      afficher ("heure 2 est plus tôt");
      }
      sinon si (heure1.ss > heure2.ss)
      { afficher ("heure 1 est plus tôt");
      }
      sinon
      { afficher ("Ce sont les mêmes heures");
      }
    }
  }
  retourner;
}
```

(1)

k) Ecrire une fonction récursive qui résout le problème des tours de Hanoi pour N disques (vu en cours) – vous pouvez faire des schémas si vous le souhaitez pour vous aider à écrire cette fonction

Thème 3 : VRAI/FAUX

j) VRAI/FAUX : répondez par *vrai* ou par *faux* aux affirmations suivantes. Une bonne réponse rapporte 1 point.

Une fonction récursive sans condition d'arrêt peut générer un plantage du programme.	V
Une structure contient des champs.	V
Un champ est identifié par son type et son nom.	F
On utilise la notation '.' pour accéder au champ d'une structure si ce champ est un pointeur.	F
Une variable de type structure n'a pas d'adresse.	F

1,5

Addition (attention aux retenues en additionnant les secondes et les minutes)

```
fonction addition(entree t_heure heure1, t_heure heure2 -> sortie : t_heure)
{
  t_heure somme;
  entier retenue=0;
  somme.ss <- heure1.ss + heure2.ss;
  si (somme.ss >= 60)
  {
    somme.ss <- somme.ss - 60;
    retenue <- 1;
  }
  somme.mm <- heure1.mm + heure2.mm + retenue;
  si (somme.mm >= 60)
  {
    somme.mm <- somme.mm - 60;
    retenue <- 1;
  }
  somme.hh <- heure1.hh + heure2.hh + retenue;
  retourner somme;
}
```

2

f) Soit la structure suivante décrivant un événement :

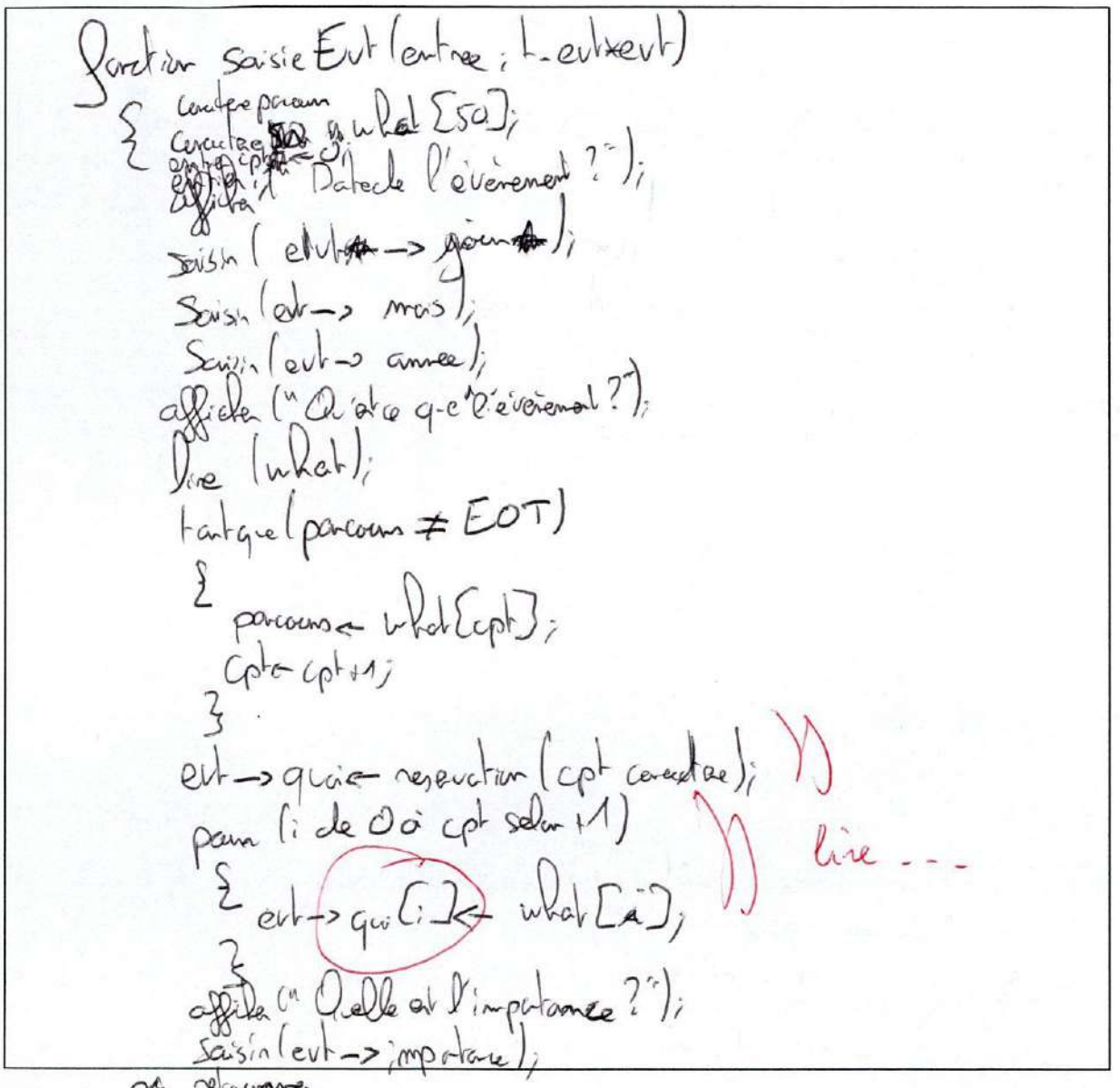
```
structure t_evt
{
        entier   jour,mois,annee;
        caractere       *quoi;
        entier   importance;
};
```

Ecrivez la fonction de saisie d'un événement pour laquelle on vous fournit l'entête suivante :

```
fonction saisieEvt(entree : t_evt *evt)
```

note : cette fonction est appelée dans la fonction principale de la manière suivante :

```
fonction principale()
{
      t_evt e;
      saisieEvt(&e);
      ...
}
```



g) Soit **tabag** un tableau (statique ou dynamique, peu importe) de **t_evt** dont la taille utile est connue. Ecrire une fonction qui trie un tableau de **t_evt** par ordre chronologique. L'entête de cette fonction est la suivante :

```
fonction triChrono(entree : t_evt *tab, entier util)
```

Note : cette fonction est appelée dans la fonction principale de la manière suivante :

```
fonction principale()
{
    t_evt      *tabag;
    entier     nbEvt;
    entier     cpt;

    afficher("entrez le nombre d'evenements:");
    saisir(nbEvt);

    tabag ← reservation(nbEvt t_evt);
```

```
        pour cpt de 0 à nbEvt-1
        {
             saisieEvt(&tabag[cpt]);
        }

        triChrono(tabag,nbEvt);
        ...
}
```

```
triChrono (entree: t_evt * tab, entier dim)
{
  entier cpt = 0;
  t_evt tmp;
  entier i=1;
  tant que (tab[cpt]
  tant que (tab[cpt].annee > tab[1].annee
  tant que (i ≠ dim)
  {pour (cpt de 0 à dim-2-saison+1)
    {
      si (tab[cpt].annee > tab[cpt+1].annee
        {
          tmp ← tab[cpt+1];
          tab[cpt+1] ← tab[cpt];
          tab[cpt] ← tmp;
        }
      Sinon si (tab[cpt].mois > tab[cpt+1].mois)
        {
          tmp ← tab[cpt+1];
          tab[cpt+1] ← tab[cpt];
          tab[cpt] ← tmp;
        }
      Sinon si (tab[cpt].jour > tab[cpt+1].jour)
        {
          tmp ← tab[cpt+1];
          tab[cpt+1] ← tab[cpt];
          tab[cpt] ← tmp;
        }
    }
  i = i+1;
  }retourner 0;}
```

1,5

Thème 2 : un peu de récursivité

h) Qu'est-ce qu'une fonction récursive ?

Une fonction récursive est une fonction qui s'appelle elle-même lors de son exécution. Cette fonction doit avoir une condition d'arrêt sous peine de faire planter le programme

1

i) Soit la fonction récursive suivante : que fait-elle si on l'appelle avec une valeur initiale de position égale à 0 ? (vous pouvez choisir un texte à afficher en exemple)

```
fonction devinette(entree : caractere *texte, entier position)
{
      si ( position < longueur_texte(texte))
      {
            afficher(texte[position]);
            devinette(texte, position+1);
      }

      retourner;
}
```

Si on appelle cette fonction avec une valeur initiale de position à 0, la fonction va afficher l'intégralité du message contenu dans texte.

Exemple avec texte : aRC

0 < 3
{ a
1 < 3
{ R
2 < 3
{ C
}
}
}

1