ESIAL 1ère année

27 mars 2010
9h00 - 11h00
Module C-Shell

Examen du module C-Shell : partie Langage C

Les exercices sont indépendants. La correction tiendra compte de la qualité de la rédaction et de la présentation. Tout document interdit sauf une feuille format A4 recto-verso à rendre avec votre copie.

⋆ Exercice 1. (3 pts)

On veut calculer une approximation du nombre d'Euler à ϵ près par la série tronquée : $e_N = \sum_{n=0}^{N} \frac{1}{n!}$.

▷ **Question 1.** Une façon simplifiée pour calculer cette approximation est d'exprimer une relation de récurrence entre deux termes successifs, (`sans utiliser le calcul de` $n!$) et d'additionner les termes de la somme jusqu'à ce que le terme suivant soit plus petit qu'une valeur ϵ.

Ecrire un programme qui calcule une approximation du nombre d'Euler et qui affiche N.

⋆ Exercice 2. (3 pts)

On souhaite écrire une fonction qui renvoie l'adresse de la dernière occurence d'un caractère (le paramètre `c`) dans une chaîne de caractères (le paramètre `str`), ou NULL si le caractère n'est pas présent. Le prototype de la fonction est le suivant : `char *dernier_car(char c, char *str)`

▷ **Question 1.** Ecrire cette fonction.

▷ **Question 2.** Ecrire la fonction ***main*** permettant de tester la fonction écrite précédemment. Un exemple d'exécution est le suivant :

```
$ ./occurence o Bonjour
la dernière occurence de 'o' dans 'Bonjour' est à la position 5.
$
```

⋆ Exercice 3. (2 pts)

Répertorier toutes les erreurs se trouvant dans le programme ci-dessous et indiquer la façon de les corriger.

```
1  #include <stdio.h>
2  int main(void) {
3    char *chaine, mot_passe = "Mot_2_Passe";
4
5    scanf("%s\n", &chaine);
6
7    if (chaine == mot_passe)
8      printf(" OK, c'est bon!\n");
9    else
10     printf("Ce n'est pas ça !\n");
11   return 1;
12 }
```

⋆ Exercice 4. (3 pts)

Qu'imprime le programme suivant ? Indiquer également les valeurs des variables citées dans chacune des 6 lignes de commentaire.

On rappelle que les caractères en C sont représentés par un `entier` (le code ASCII du caractère) et sont codés de manière consécutive selon l'ordre alphabétique. Par exemple, le caractère `'a'` est codé par l'entier 97, le caractère `'b'` est codé par 98, et `'b' + 2` vaut 100, soit le caractère `'d'`.

```
#include <stdio.h>

int main() {

  char chaine[] = "examen";
  char *ptr_ch;

  ptr_ch = chaine;
       // 1: valeur de chaine et de ptr_ch ?
  *ptr_ch = (*chaine) + 1;
  ptr_ch += 1;
  *ptr_ch = *(ptr_ch + 1);
      // 2: valeur de *ptr_ch ?
  *(ptr_ch + 1) += 2;
      // 3: valeur de *ptr_ch ?
  ptr_ch += 2;
      // 4: valeur de chaine ?
  *ptr_ch = 'i';
  ptr_ch += 1;
      // 5: valeur de ptr_ch  ?
  *(ptr_ch + 1) = *ptr_ch;
      // 6: valeur de chaine  ?
  *ptr_ch = (*ptr_ch) + 7;

  printf("%s ... comme est %s\n", chaine, ptr_ch);
  ptr_ch = "partiel d\'informatique : ";
  printf("%s %s  \n", ptr_ch, chaine);
}
```

★ Exercice 5. (3 pts)

▷ **Question 1.** A quoi sert le pré-processeur ? Donner (en les commentant) deux exemples de directives traitées par le pré-processeur.

▷ **Question 2.** Soit le ***Makefile*** suivant :

```
CC = gcc
CFLAGS = -g -ansi -Wall -o $<

OBJ = main.o arbre.o liste.o
REP = /users/outils/

appli : $(OBJ) $(REP)es.o
     $(CC)  $(OBJ) $(REP)es.o -o appli

main.o:  $(REP)const.h

arbre.o: global.h $(REP)const.h

liste.o: $(REP)const.h

$(REP)es.o: $(REP)const.h
```

Que fait ce ***Makefile*** ? Le commenter.
Représenter l'arborescence des fichiers.

⋆ Exercice 6. (4 pts)

Des relevés météorologiques sont effectués chaque jour dans différents sites (20 au total). Les données sont transmises à un opérateur qui dispose du programme nécessaire pour leur traitement. Ce programme permet la saisie de chaque relevé, détermine le maximum et le minimum des écarts de températures rencontrés dans les différents sites, puis affiche dans chaque cas, la liste des sites concernés. Un relevé comporte les informations suivantes : *site*, *température minimale*, *température maximale*.

Par exemple :

```
Toulouse   12   26
Nancy      07   22
...
```

▷ **Question 1.** Compléter les parties `(a)`, `(b)`, `(c)` et `(d)` du programme suivant afin que le traitement effectué soit conforme au traitement souhaité :

```
#include <stdio.h}
#define NB_RELEVE 200

/* Déclaration des constantes, types, variables globales, etc */
     ... (a)

/* Saisie des relevés et calcul des écarts de température */

   void saisir_releves(int *e_min, int *e_max)
     {   ... (b)    }


/* Fonction d'affichage des villes dont l'écart de température
   correspond à celui indiqué.
   Cette fonction renvoie le nombre de villes concernées.    */

  int lister_villes(int ecart)
     {   ... (c)    }


/* Programme  "principal" */

int main() {
  int ecart_min, ecart_max;
  saisir_releves ( ... (d) );

  printf(" Ecart minimum de la journée : %d\n", ecart_min);
  printf("%d villes concernées \n", lister_villes(ecart_min);
  printf(" Ecart maximum de la journée : %d\n", ecart_max);
  printf("%d villes concernées \n", lister_villes(ecart_max);
```

NOM : PRENOM :

★ **Exercice 7. QCM (3 pts)** Répondez sur la feuille fournie. Les réponses fausses seront pénalisées.

- Pour disposer dans une fonction `f` d'une variable `i` dont la valeur est préservée entre chaque appel et qui ne soit pas visible de l'extérieur de `f`, il faut la définir en tant que :
  - ☐ *static int i;* à l'intérieur de f
  - ☐ *int i;* à l'intérieur de f
  - ☐ *static int i;* à l'extérieur de f
  - ☐ *extern int i;* à l'extérieur de f
- Pour récupérer au clavier un entier et 2 caractères à placer dans une variable `i` et les 2 cases `t[0]` et `t[1]` d'un tableau de caractères, on peut écrire `scanf("%d%c%c ", ....)` avec en paramètres,
  - ☐ &i, t, t+1
  - ☐ &i, *t, *(t+1)
  - ☐ i, t[0], t[1]
  - ☐ &i, t
- Le corps de la boucle *for(i=1; i<10;) { ++i;* `corps`*}* s'exécute :
  - ☐ 0 fois
  - ☐ 9 fois
  - ☐ 10 fois
  - ☐ aucune des réponses précédentes
- Si `t` est défni par `int *t` alors `&(t[2] + 1)` est de type,
  - ☐ (int *)
  - ☐ (int **)
  - ☐ (int)
  - ☐ c'est une expression erronée
- Si `t` est défni par `int *t` alors `t[*t]` est de type :
  - ☐ (int)
  - ☐ (int *)
  - ☐ (int **)
  - ☐ c'est une expression erronée
- Si `t` est défni par `int *t` alors `&t` est de type :
  - ☐ (int)
  - ☐ (int *)
  - ☐ (int **)
  - ☐ c'est une expression erronée
- Si `t` est défni par `int *t` alors `&(t[2]+1)` est de type :
  - ☐ (int)
  - ☐ (int *)
  - ☐ (int **)
  - ☐ c'est une expression erronée
- Soient les deux déclarations suivantes : `char v[] = "abcd"; char *p = "abcd";`
  - ☐ ces 2 déclarations sont équivalentes
  - ☐ on peut écrire `*p='A'`
  - ☐ la taille de `v` est 5
  - ☐ le contenu de `v` n'est pas modifiable
- Si `p` est un pointeur sur un tableau de caractères, alors `p` peut se définir comme :
  - ☐ char *p
  - ☐ char *p[]
  - ☐ (char []) p *
  - ☐ char **p
- Pour inclure le fichier d'en-tête système ***glob.h*** on doit écrire :
  - ☐ #include <glob.h>
  - ☐ #include 'glob.h'
  - ☐ #include "glob.h "
  - ☐ #include glob.h
- Pour utiliser gdb, il faut compiler le source C avec l'option,
  - ☐ -Werror
  - ☐ -Wall
  - ☐ -g
  - ☐ -E
- Les chaînes de caractères en C se terminent, en mémoire, par
  - ☐ '\n'
  - ☐ '\0'
  - ☐ aucun caractère spécial
  - ☐ '\o'