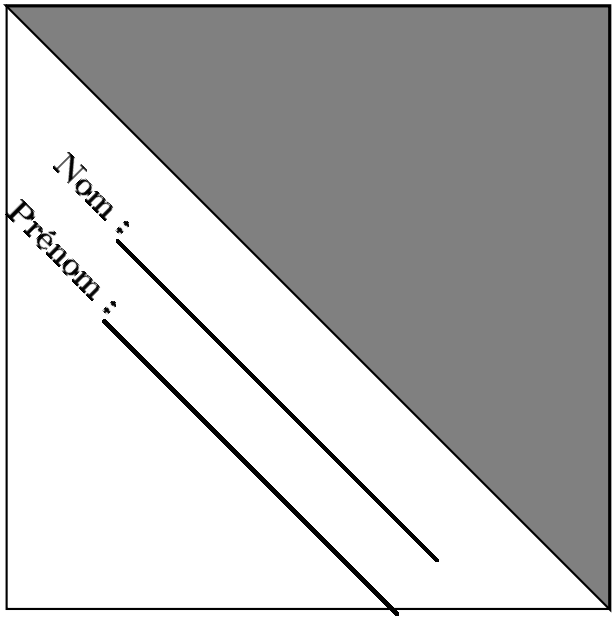
Université de Nice-Sophia Antipolis
Licence professionnelle
contrôle continu 2002–2003

Examen de Langage C

Durée : 1/2 heure
Aucun document autorisé

Note :



Exercice 1

1. Donnez la valeur de la variable `res` après l'exécution des instructions suivantes :

```
res=0;
switch(lettre) {
  case 'l' : res=1; break;
  case 'F' : res=2;
  case 'n' : res++; break;
  default : res=10; break;
  case '?' : res=11;
}
```

si lettre == 'F',

---

puis si lettre == 'l',

---

et enfin si lettre == 'N'.

Exercice 2

Voici un ensemble de définitions et de déclarations :

```
typedef char Couleur;

typedef struct pneu {
  float temperature, pression;
} Pneu;

struct voiture {
  short nb_portes, nb_passagers;
  long int vitesse;
  Pneu roue[4];
  Couleur c;
};

float val=12;
char car="a", lettre;
Couleur c=32;
int tab[5]={0,1,2,3,4,5};
int tab2[c];
char ch[20];
double tab3[];
const double vitesse_max=9e1;
int i=0, j=3+9;
short res;
```

2. Pour chacune des définitions de variables, donnez son numéro de ligne et indiquez si elle est valide, sinon expliquez pourquoi.

3. Donnez le prototype d'une fonction fct dont la valeur de retour est du *type structuré voiture* et prenant comme premier paramètre un tableau dont les éléments sont du *type structuré pneu* et un deuxième paramètre de type Couleur.

4. Déclarez une variable qui soit du *type structuré voiture* et dégonflez les 4 pneus de cette voiture (dégonfler un pneu revient à mettre sa pression à 0). Vous utiliserez pour cela une instruction de boucle.

5. Écrivez une instruction qui permet de lire un caractère sur l'entrée standard et de l'affecter au 5ème élément du tableau de caractères ch.

Exercice 3

nbc et c sont deux variables de type **int**. Soit le fragment de programme suivant :

```
nbc=0;
while( (c=fgetc(stdin)) != EOF ) {
   fputc(toupper(c), stdout);
   nbc++;
}
```

6. Réécrivez ce fragment en utilisant une boucle **for** à la place de la boucle **while**.

7. Réécrivez de nouveau le fragment en utilisant une boucle **do while** à la place de la boucle **while**.

Exercice Bonus (facultatif)

8. Pour chacune des instructions qui suivent vous mettrez des parenthèses pour indiquer l'ordre dans lequel sont évaluées les différentes sous expressions, puis vous donnerez les valeurs des variables x et y de type **int**.
   Rappel : les expressions logiques valent 1 si la relation est vraie, 0 sinon.
   Vous trouverez en annexe le tableau des priorités.

   y = (x = 0, x++) && x++ ;

   y = (x = 0, ++x) && x−− ;

Annexe

Tableau des opérateurs rangés par priorité décroissante.

Types	Symboles	Associativité
postfixé	`()`, `[]`, `.`, `->`, `++`, `--`	G → D
unaire	`&`, `*`, `+`, `-`, `~`, `!`, `++`, `--`, **sizeof**	D → G
casting	`(<type>)`	D → G
multiplicatif	`*`, `/`, `%`	G → D
additif	`+`, `-`	G → D
décalage	`<<`, `>>`	G → D
relationnel	`<`, `>`, `<=`, `>=`	G → D
(in)égalité	`==`, `!=`	G → D
et bit à bit	`&`	G → D
xor	`^`	G → D
ou bit à bit	`|`	G → D
et logique	`&&`	G → D
ou logique	`||`	G → D
condition	`? :`	D → G
affectation	`=`, `*=`, `/=`, `%=`, `+=`, `-=`, `<<=`, `>>=`, `&=`, `^=`, `|=`	D → G
virgule	`,`	G → D

Extraits du *man* d'UNIX :

int **fputc** (int c, FILE *stream) ;

`fputc() écrit le caractère c, transformé en unsigned char, dans le flux stream. fputc() renvoie le caractère écrit en tant qu'unsigned char, converti en int ou EOF en cas d'erreur.`

int **fgetc** (FILE *stream) ;

`fgetc() lit le caractère suivant depuis le flux stream et renvoie ce caractère, lu sous forme unsigned char, puis transformé en int, ou EOF en cas d'erreur ou de fin de fichier. fgetc() renvoie un caractère, lu comme un unsigned char, et transformé en int, ou EOF à la fin du fichier, ou en cas d'erreur.`

int **toupper** (int c) ;

`toupper() convertit la lettre c en majuscule si c'est possible. La valeur renvoyée est celle de la lettre convertie, ou bien c si la conversion n'etait pas possible. Si c n'est ni un caractère non-signé, ni EOF, le comportement de la fonction est imprévisible.`