**MASTER 2 MPM** **Université Paris XIII**
**Année 2011-2012** **Institut Galilée**

# Langage C avancé pour la simulation numérique

## Examen du 19 janvier 2012

Durée : 3h

**Aucun document autorisé**
**Aucun appareil electronique autorisé**

## Exercice 1 : (3 points)

Listing 1: exo1.c

```
1  #include <stdio.h>
2
3  void g(int *i, int j){
4    *i=j+2;
5    j=*i-3;
6  }
7
8  void h(double x,double *y){
9    *y=x+1.;
10   x=*y-2.;
11 }
12
13 int main(){
14   int i=-1,j=-3;
15   double s=1.,t=-1;
16   g(&j,i);
17   h(s,&t);
18   printf("i=%d,j=%d,s=%lf,␣t=%lf\n",i,j,s,t);
19   return 0;
20 }
```

**Q. 1** *Donner la ou les commandes Linux permettant de créer le programme executable* `exo1` *associé au fichier* `exo1.c`. ▪

**Q. 2** *A l'aide de diagrammes de représentation mémoire, expliquer le déroulement de ce programme et donner l'affichage.* ▪

## Exercice 2 : (7 points)

Soit $z_1$ et $z_2$ deux nombres complexes. On a, en posant $z_1 = x_1 + \imath y_1$ et $z_2 = x_2 + \imath y_2$, avec $x_1$, $x_2$, $y_1$ et $y_2$ des réels :

$$\begin{aligned} z_1 + z_2 &= (x_1 + x_2) + \imath(y_1 + y_2), \\ z_1 z_2 &= (x_1 x_2 - y_1 y_2) + \imath(y_1 x_2 + x_1 y_2), \\ |z_1| &= \sqrt{x_1^2 + y_1^2} \end{aligned}$$

**Q. 1** *Définir le type* `Complex` *correspondant à une structure de champs deux* `double` *nommés* `Re` *et* `Im`. ▪

**Q. 2** *1. Ecrire une fonction* `InitComplex` *permettant, à partir de deux* `double` *a et b, d'initialiser un «nombre complexe» à* $a + ib$.

2. *Ecrire une fonction* `PrintComplex` *permettant d'afficher un «nombre complexe» $a+ib$ sous la forme $(a, b)$.*

3. *Ecrire une fonction* `AddComplex` *permettant de calculer la somme de deux nombres complexes $z \leftarrow z_1 + z_2$.*

4. *Ecrire une fonction* `ProdComplex` *permettant de calculer le produit de deux nombres complexes $z \leftarrow z_1 * z_2$.*

5. *Ecrire une fonction* `AbsComplex` *permettant de calculer le module d'un nombre complexe $|z| \leftarrow z$*

6. *Ecrire une fonction* `aXpbYComplex` *permettant de calculer le nombre complexe $z \leftarrow \alpha z_1 + \beta z_2$, où $(\alpha, \beta) \in \mathbb{R}^2$ et $z_1$, $z_2$ deux nombres complexes.* ▪

On suppose, par la suite, ces cinq fonctions écrites dans le fichier `Complex.c`.

**Q. 3** 1. *Ecrire le fichier header* `Complex.h` *associé au fichier* `Complex.c`.

2. *En utilisant, dès que possible, les fonctions précédentes, écrire le fichier* `PPcomplex.c` *contenant le programme principal effectuant les opérations suivantes*

   - *Initilaliser les nombres complexes $z_1 \leftarrow 1 + i$ et $z_2 \leftarrow 1 - i$*
   - *Initialiser les nombres réels $A \leftarrow 2$ et $B \leftarrow -1$.*
   - *Calculer $z_1 \leftarrow z_1 + z_2$*
   - *Calculer $z_2 \leftarrow z_1 * z_2$*
   - *Calculer $z_3 \leftarrow A * z_1 - B * z_2$*
   - *Afficher $z_3$ et $|z_1 + z_2|$.* ▪

**Q. 4** *On suppose les fichiers* `PPcomplex.c`, `Complex.c` *et* `Complex.h` *présents dans le répertoire courant. Ecrire un* `Makefile` *permettant de créer l'exécutable* `PPcomplex`. ▪

## Exercice 3 : (10 points)

Dans cet exercice, nous allons implémenter et utiliser quelques outils d'algèbre linéaire.

**Q. 1** *Définir le type* `vector` *correspondant à une structure ayant pour champs :*

- `dim` *: dimension du vecteur (type* `int`*),*
- `pval` *: pointeur sur les composantes du vecteur (type* `double *`*).* ▪

**Q. 2** *Nous voulons écrire une fonction* `AllocVector` *permettant d'allouer un objet de type* `vector` *à la dimension $N$.*

1. *Donner deux prototypes possibles (non trivialement similaire) pour la fonction* `AllocVector`.

2. *Choisir un prototype et écrire la fonction associée.* ▪

**Q. 3** 1. *Ecrire une fonction* `freeVector` *permettant de désallouer le champ* `val` *d'un objet de type* `vector`.

2. *Justifier le choix de ses arguments!* ▪

**Q. 4** 1. *Ecrire une fonction* `FREEVector` *permettant de désallouer le champ* `val` *d'un objet de type* `vector`, *de mettre à zéro le champ* `dim` *et à* `NULL` *le champ* `val`.

2. *Justifier le choix de ses arguments!* ▪

**Q. 5** *Nous voulons écrire une fonction* `RandVector` *permettant d'initaliser chacune des composantes d'un vecteur par un nombre aléatoire (utiliser la fonction drand48() du langage C qui retourne un* `double` *entre 0 et 1 :* `double drand48 (void);`*). Le vecteur est supposé déjà alloué.*

1. *Donner deux prototypes possibles (non trivialement similaire) pour la fonction* `RandVector`.

2. *Choisir un prototype et écrire la fonction associée.* ▪

**Q. 6** *Soient* $\boldsymbol{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \in \mathbb{R}^n$, $\boldsymbol{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \in \mathbb{R}^n$ *et* $\alpha$, $\beta$ *deux réels. Les fonctions* `ScalarProd`, `aUpbV` *et* `Norm` *correspondent à :*

`ScalarProd` : $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{n} u_i v_i$,

`aUpbV` : $\alpha \boldsymbol{u} + \beta \boldsymbol{v}$,

`Norm` : $\|\boldsymbol{u}\| = \left( \sum_{i=1}^{n} u_i^2 \right)^{1/2}$.

1. *Pour chacune des trois fonctions, donner deux prototypes possibles (non trivialement similaire).*
2. *Pour chacune des trois fonctions, choisir un prototype et écrire la fonction associée.* ▪

**Q. 7** *Ecrire un* `main` *simple et fonctionnel utilisant au moins les fonctions* `RandVector`, `ScalarProd`, `aUpbV` *et* `Norm`. ▪