

TE 5

Exercices d'introduction en fortran et C

Dans votre répertoire `mni`, créer un répertoire `f90+c`, puis un sous-répertoire `te5`. Ne pas oublier de nettoyer votre répertoire de travail à la fin de la session, en effaçant les fichiers binaires objets et exécutables.

Options des compilateurs Afin de profiter pleinement de l'analyse effectuée sur le fichier source par le compilateur, il est fortement conseillé de préciser le standard choisi et de lui demander d'avertir l'utilisateur dès qu'une instruction présente un risque d'erreur. Ce choix se traduira notamment par l'usage des options de compilation suivantes :

en C89 : `gcc -std=c89 -W -Wall`

en fortran95 avec `g95` : `g95 -std=f95 -Wall`

en C99 : `gcc -std=c99 -W -Wall`

ou, avec `gfortran` : `gfortran -std=f95 -Wall`

Ces options seront automatiquement appliquées en utilisant les alias fournis par les fichiers de configuration de l'UE :

- `gcc-mni-c89` ou `gcc-mni-c99` pour le C
- `g95-mni` (ou `g2003-mni`) avec `g95` et `gfortran-mni` (ou `gfortran03-mni`) avec `gfortran` pour le fortran 95 (ou le fortran 2003),

A) Notions fondamentales

Exercice 1 : Premier programme (édition, compilation, exécution) A

Cet exercice a pour but de vous présenter, sur un exemple simple, les étapes de base allant de l'écriture d'un programme, à sa compilation, son édition de lien et son exécution. Elles s'appliqueront dans toute la suite des TEs à des programmes de plus en plus compliqués.

1. Édition du fichier source

- Indiquez ce qu'est un code source.
- Avec un éditeur de texte (`vi`, `emacs`, ...), recopiez le code source suivant dans des fichiers nommés `bonjour` et dont l'extension permet de déterminer si le programme est écrit en C ou fortran. On lancera la commande d'édition depuis une console pour préciser dès le départ le nom du fichier : son extension est reconnue par l'éditeur qui active alors la colorisation syntaxique.

```

      _____  bonjour.c  _____
1  /* Premier programme en C89 */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  int main(void)
6  {
7      printf("Bonjour ! \n");
8
9      exit(0);
10 }
```

```

      _____  bonjour.f90  _____
1  ! Mon premier programme en fortran
2
3  program essai
4
5  implicit none
6
7  write(*,*) "Bonjour !"
8  stop
9
10 end program essai
```

- Comptez le nombre de lignes, de mots, de caractères et d'octets des fichiers de code source.

2. Compilation

- (a) Compilez le code source à l'aide du compilateur `gcc / g95` :
`gcc-mni-c89 bonjour.c, g95-mni bonjour.f90` ou `gfortran-mni bonjour.f90`
 - (b) Quel est le rôle du compilateur ?
 - (c) Quel est le nom du fichier produit par la compilation dans chaque langage ?
 - (d) Quelle est la taille en octets de l'exécutable (comparer `g95` et `gfortran`) ? Quels sont ses droits d'accès ? Vérifier que ce fichier binaire contient le texte `Bonjour`.
3. Exécution
- (a) Exécutez le fichier produit (à l'invite du système, entrez : `./nom_de_fichier`). Commentez le résultat obtenu en sortie standard.
 - (b) Si vous n'obtenez pas le résultat attendu, que vous recevez des messages d'avertissement ou d'erreur, repartez de l'étape édition, identifiez les erreurs et corrigez-les, puis recompilez le programme. Ne relancez d'exécution que si la compilation réussit (comparez les dates du source et de l'exécutable). Reprenez ce cycle jusqu'à une exécution correcte.
4. Compilation seule, puis édition de liens
- Trouvez l'option vous permettant de conserver le fichier objet produit à la compilation (fichier d'extension `.o`, e.g. `bonjour.o`). Quelle est la différence entre le fichier objet et l'exécutable ? Générez un fichier exécutable à partir du fichier objet. Exécutez-le et vérifiez que le résultat est le même qu'en 3a.
5. Trouvez l'option vous permettant de produire un fichier exécutable de nom `bonjour1` à partir du fichier objet `bonjour.o`. Vérifiez que le résultat de l'exécution est le même qu'en 3a et 4.
6. À partir du fichier source `bonjour.c` ou `bonjour.f90`, produire un exécutable de nom `bonjour2` en une seule ligne de commande. Après exécution, vérifiez que le résultat est le même qu'en 3a, 4 et 5.
7. Si vous voulez donner à quelqu'un un programme que vous avez écrit, quel fichier devez-vous lui fournir sachant que vous ne savez pas sur quelle machine il doit l'utiliser ? Connectez-vous sur `sappli1` et essayez de lancer l'exécutable C compilé sur la machine virtuelle. Expliquez en utilisant la commande `file`. Recompilez sur `sappli1` et vérifiez.
- B** Essayez d'exécuter sur la machine virtuelle le fichier compilé sur `sappli1`. Expliquez. Reprenez la question avec l'option `--static` des compilateurs.

Exercice 2 : Découpage des lignes affichées en sortie **A**

Sans compiler les programmes `lignes.c` et `lignes.f90` suivants, prédire combien de lignes ils doivent afficher, ainsi que le contenu de ces lignes.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void){
5     printf("Combien de lignes\n\n sont affichees"
6           " par ce programme et ");
7     printf("\nquel est le contenu des ces lignes ?\n");
8     printf("Observer ... la fin !");
9     exit(EXIT_SUCCESS);
10 }

```

```

1 program lignes
2
3     write(*,*) "Combien de lignes"
4     write(*,*)
5     write(*,*) " sont affichees&
6           & par ce programme et "
7     write(*,*) "quel est le contenu des ces lignes ?"
8     write(*,'(a)',advance="no" ) "Observer ... "
9     write(*,'(a)',advance="no" ) "la fin !"
10 end program lignes

```

Compiler et exécuter pour vérifier. Modifier le programme pour que l'invite unix apparaisse en début de ligne.

Rappel des syntaxes élémentaires d'entrées-sorties standard

En fortran, en format libre :

```

write(*,*) "entrer un entier (i) et un réel (r)"
read(*,*) i, r
write(*,*) "valeurs lues i = ", i, " r = ", r

```

Chaque ordre `read` ou `write` provoque un changement de ligne (sauf avec `advance=no`).

En C :

```

printf("entrer un entier (i) et un float (r)\n") ;
scanf("%d %g", &i, &r) ;
printf("valeurs lues i = %d , r = %g \n", i, r) ;

```

Ne pas oublier le `&` en entrée. Insérer `\n` dans le format d'affichage pour passer à la ligne.

Exercice 3 : Saisie, calcul, affichage **A**

1. Saisissez avec un éditeur de texte et sauvegardez dans un fichier de suffixe approprié le code source suivant :

```

1 // Titre de mon programme en C99
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int main(void)
7 {
8     int x, y, z, t, u;
9     double a, b;
10    /* Mon premier commentaire */
11    printf("Entrez un nombre entier \n");
12    scanf("%d", &x);
13    /* Mon
14     deuxieme
15     commentaire */
16    printf("Entrez un deuxieme nombre entier \n");
17    scanf("%d", &y);
18
19    z = x * y;
20    t = x / y  u = x % y; // 3ieme commentaire
21    printf("%d * %d = %d  t = %d u = %d\n",
22           x, y, z, t, u);
23    // decrire la suite
24    printf("Saisir deux doubles \n");
25    scanf("%lg %lg", &a, &b);
26    c = a / b;
27    printf("a=%g  b=%g  c=%g\n", a, b, c);
28    printf("fin\n");
29
30    exit(0);
31 }

```

```

1 ! Titre de mon programme en fortran
2
3 program calcul
4
5 implicit none
6
7 integer :: x, y, z, t, u
8 real :: a, b
9
10 ! Mon premier commentaire
11 write(*, *) "Entrez un nombre entier:"
12 read(*, *) x
13
14 ! Mon deuxieme
15 ! commentaire
16 write(*, *) "Entrez un deuxieme entier"
17 read(*, *) y
18
19 z = x * y ! 3ieme commentaire
20 t = x / y  u = mod(x,y)
21 write(*, *) x, " * ", y, " = ", z, &
22    " t = ", t, " u = ", u
23 ! decrire la suite
24 write(*,*) "saisir deux reels "
25 read (*,*) a, b
26 c = a / b
27 write(*,*) "a=", a, "  b=", b, "  c=", c
28 write(*,*) "fin"
29
30 end program calcul

```

Compiler ce code source en examinant attentivement les messages du compilateur. Corriger pas à pas les erreurs dans l'ordre des lignes du code¹. Que fait le programme ?

2. – Lister les variables et leur type.
 - Modifier les commentaires de manière à décrire les actions du programme.
 - Repérer les opérateurs utilisés.
 - **B** Repérer deux opérateurs présents dans le programme en C et pas dans celui en fortran; citer un opérateur qui existe en fortran mais pas en C.
 - **(C)** `#include <stdio.h>`
 Quel est le rôle de `#include` ? Qu'est ce que le fichier : `stdio.h` ? Pourquoi ce fichier permet le bon fonctionnement du programme ?
 - **B** **(f90)** Quel est le rôle de la déclaration `implicit none` ? Où peut-on la placer ?

Exercice 4 : Échange **A**

Sans connaître les valeurs de `a` et `b`, compléter le code des programmes ci-après de façon à placer la valeur de la variable `a` dans `b` et réciproquement.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void) {
5     double a, b;
6     a = 5.2;
7     b = 9.3;
8     printf("La variable 'a' contient la valeur %g\n", a);
9     printf("La variable 'b' contient la valeur %g\n", b);
10    // a completer
11    printf("La variable 'a' contient la valeur %g\n", a);
12    printf("La variable 'b' contient la valeur %g\n", b);
13    exit(0);
14 }

```

```

1 program exchange
2 implicit none
3 real :: a, b
4
5 a = 5.2
6 b = 9.3
7 write(*,*) "La variable 'a' contient la valeur ", a
8 write(*,*) "La variable 'b' contient la valeur ", b
9 ! a completer
10 write(*,*) "La variable 'a' contient la valeur ", a
11 write(*,*) "La variable 'b' contient la valeur ", b
12
13 end program exchange

```

Copier les fichiers source associés. Compléter le code et tester.

¹C'est le débogage du programme – vient de `bug` = anomalie en anglais, terme francisé en `bogue`.

B) Entrées sorties standard élémentaires

Exercice 5 : Format de sortie f en virgule fixe A

Compiler les programmes `format_f.c` et `format_f.f90`. Exécuter ces programmes et expliquer l'affichage. Modifier le programme pour obtenir un affichage plus robuste.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5      float r1, r2;
6      r1 = 1.234e-5f;
7      r2 = r1 / 100.f;
8      printf("r1 vaut %f et r2 vaut %f\n", r1, r2);
9      exit (EXIT_SUCCESS);
10 }

```

```

1  program format_f
2  implicit none
3
4      real :: r1, r2
5      r1 = 1.234e-5
6      r2 = r1 / 100.
7      write(*, "(a, f12.6, a, f12.6)") &
8          "r1 vaut ", r1, " et r2 vaut ", r2
9  end program format_f

```

Exercice 6 : Formats de sortie erronés A

Compiler les programmes `erreur_format.c` (avec `gcc` sans option ou avec l'alias qui implique `-Wformat`) et `erreur_format.f90`. Prendre connaissance des messages et s'assurer que l'on a bien créé un exécutable (comment faire?). Lancer l'exécution et expliquer. Modifier le programme pour obtenir un affichage correct.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5      float r;
6      int i;
7      r = 5.25f;
8      i = 12;
9      printf("r vaut %e et i vaut %e\n", r, i);
10     printf("r vaut %d et i vaut %d\n", r, i);
11     exit (EXIT_SUCCESS);
12 }

```

```

1  program err_format
2  implicit none
3  real :: r
4  integer :: i
5
6  r = 5.25
7  i = 12
8  write(*, "(a, i12, a, i12)") &
9      "r vaut ", r, " et i vaut ", i
10 write(*, "(a, e12.3, a, e12.3)") &
11     "r vaut ", r, " et i vaut ", i
12 end program err_format

```

C) Types et conversions

Exercice 7 : Conversions A

Sans éditer et avant de compiler les programmes suivants, indiquer tout d'abord les affichages que doivent produire leur exécution.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5      float q;
6      float a;
7      int g;
8
9      q = 5.25;
10     g = 5;
11
12     a = q / g;
13     printf("La valeur de a est %g\n", a);
14     a = a + g;
15     printf("La valeur de a est %g\n", a);
16     g = a - q;
17     a = g + q;
18     printf("La valeur de a est %g\n", a);
19     exit(0);
20 }

```

```

1  program real_int_conv
2  implicit none
3
4  real :: q, a
5  integer :: i
6
7  q = 5.25
8  i = 5
9
10 a = q / i
11 write(*,*) "valeur de a = ", a
12
13 a = a + i
14 write(*,*) "valeur de a = ", a
15
16 i = a - q
17 a = i + q
18 write(*,*) "valeur de a = ", a
19
20 end program real_int_conv

```

Copier les fichiers source associés. Compiler ce code source avec `gcc` (`g95` ou `gfortran`) sans option. Exécuter ce code et comparer les affichages avec vos prévisions. Afin de rechercher les points délicats sur le code fortran, recompiler avec les options conseillées des alias. Repérer les avertissements et expliquer.

Exercice 8 : Divisions AB

Compiler les programmes `div_int.c` et `div_int.f90` avec respectivement `gcc` et `g95` sans option. Les exécuter et expliquer les résultats affichés.

Recompiler avec les options des alias. Modifier le programme pour que cette compilation exigeante se fasse sans avertissement.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void)
5  {
6      int i, j, k;
7      float a, b, c, d;
8      i = 15;
9      j = 4;
10     a = i / j ;
11     b = (float ) (i / j) ;
12     printf("a = %g   b = %g \n", a, b);
13     c = (float) i / (float) j ;
14     d = (float) i / j ;
15     printf("c = %g   d = %g \n", c, d);
16     k = d;
17     printf("k = %d\n", k);
18     exit(0);
19 }

```

```

1  program divisions
2
3  implicit none
4
5  integer :: i, j, k
6  real :: a, b, c, d
7
8      i = 15
9      j = 4
10     a = i / j
11     b = real(i / j)
12     write(*,*) "a=", a, " b=", b
13     c = real(i) / real(j)
14     d = i / real(j)
15     write(*,*) "c=", c, " d=", d
16     k = d
17     write(*,*) "k=", k
18
19 end program divisions

```

Exercice 9 : Domaine des entiers AB

Copier et compiler les programmes ci-après. Expliquer le résultat des additions. Retrouver par le calcul les valeurs maximales des entiers par défaut (stockés sur 32 bits). Vérifier les calculs à l'aide de `kcalc` qui permet d'effectuer les conversions entre décimal et binaire.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <limits.h>
4
5  int main(void){
6  int i, j, k1, k2, k3;
7  int m1;
8  i = 1073741824 ; // 2**30
9  j = i - 1;
10 k1 = i + j;
11 k2 = k1 + 1;
12 k3 = k1 + 2;
13 printf(" i= %d, i-1 = %d\n", i, j);
14 printf("k1= %d k1+1= %d k1+2= %d\n" , k1,k2, k3);
15 m1 = INT_MAX;
16 printf("plus grand entier via INT_MAX %d\n", m1);
17 exit(EXIT_SUCCESS);
18 }

```

```

1  program int_range
2  implicit none
3  integer :: i, j, k1, k2, k3
4  integer :: m1
5  i = 2**30
6  j = i - 1
7  k1 = i + j
8  k2 = k1 + 1
9  k3 = k1 + 2
10 write(*, *) "i=", i, " i-1=", j
11 write(*, *) "k1=", k1, " k1+1=", k2, " k1+2=" , k3
12 ! utilisation de fonctions d'interrogation
13 m1 = huge(i)
14 write(*,*) "plus grand entier via huge", m1
15 end program int_range

```

D) Domaine et précision des flottants

Exercice 10 : Conversion et précision A

Compiler les programmes `precis_affect.c` et `precis_affect.f90` avec respectivement `gcc` et `gfortran` sans option. Les exécuter et expliquer les résultats affichés.

Recompiler avec les options des alias. Modifier le programme pour que cette compilation exigeante se fasse sans avertissement.

```

_____ precis_affect.c _____
#include <stdio.h>
#include <stdlib.h>
int main(void){
float a;
int i, j;

i = 1000000020;
printf("i = %d\n", i);
a = i;
j = a;
printf("j = %d\n", j);
exit(EXIT_SUCCESS);
}

```

```

_____ precis_affect.f90 _____
program int_real_conv
implicit none
real :: a
integer :: i, j

i = 1000000020
write(*,*) "i = ", i
a = i
j = a
write(*,*) "j = ", j

end program int_real_conv

```

Exercice 11 : Constante de Stefan B

Préliminaire

Compiler et exécuter les programmes `domaine.c` et `domaine.f90`, qui affichent les limites du domaine des valeurs positives représentables en flottant sur 32 bits.

```

_____ float_32_range.c _____
#include <stdio.h>
#include <stdlib.h>
#include <float.h>
/* domaine des float sur 32 bits */
int main(void){
printf(" valeur positive max %g\n", FLT_MAX);
printf(" valeur positive min %g\n", FLT_MIN);
exit(EXIT_SUCCESS);
}

```

```

_____ real_32_range.f90 _____
program real_32_range
! domaine des reels sur 32 bits

implicit none
write(*,*) " valeur positive max ", HUGE(1.)
write(*,*) " valeur positive min ", TINY(1.)

end program real_32_range

```

On souhaite calculer la constante de Stefan σ qui intervient dans l'expression de l'émittance totale du corps noir :

$$M_B = \pi B = \sigma T^4 \quad \text{où} \quad \sigma = \frac{2\pi^5}{15} \frac{k^4}{c^2 h^3}$$

h	constante de Planck	$h = 6,626 \times 10^{-34}$	J s
k	constante de Boltzmann	$k = 1,38 \times 10^{-23}$	J K^{-1}
c	vitesse de la lumière	$c = 3,00 \times 10^8$	m s^{-1}

On doit obtenir $\sigma \approx 5,67 \times 10^{-8} \text{ W m}^{-2} \text{ K}^{-4}$.

Écrire un programme `stefan.c` (respectivement `stefan.f90`) qui évalue naïvement la constante de Stefan en flottants sur 32 bits. On prendra pour simplifier $\pi = 3.141593$. En C, on développera les puissances pour éviter de faire appel à la fonction `pow`. Expliquer pourquoi on n'obtient pas la valeur attendue.

Reformuler l'expression de σ en cherchant à limiter le domaine des valeurs intermédiaires à calculer. Exploiter la nouvelle formulation.

Reprenre la première partie de l'exercice en travaillant avec des flottants sur 64 bits (`double` en C et `double precision` en fortran). Expliquer le comportement en affichant les limites du domaine des sous-types flottants sur 64 bits.