

TRAVAUX DIRIGÉS - 1

1 Les fonctions

EXERCICE 1

Listing 1: exo1.c

```
1 #include <stdio.h>
2
3 void g(int i, int *j){
4     i=*j+4;
5     *j=-1;
6 }
7
8 void h(double x, double *y){
9     *y=5.0;
10    x=0.;
11 }
12
13 int main(){
14     int i=1,j=3;
15     double x=1.;
16     g(j,&i);
17     h(x,&x);
18     printf("i=%d,j=%d,x=%lf\n",i,j,x);
19     return 1;
20 }
```

Q. 1 Donner la ou les commandes Linux permettant de créer le programme exécutable `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

Listing 2: exo1.c

```

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

```

Q. 1 Donner la ou les commandes Linux permettant de créer le programme exécutable `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 3

Listing 3: exo9.c

```

1 #include <stdio.h>
2 #include <math.h>
3
4 void f1(int i, int *j, double *x, double y);
5
6 int main(){
7     int i=1, j=2;
8     double x=3., y=M_PI;
9
10    f1(i, &j, &x, y);
11    printf("main: i=%6d, j=%6d, x=%.16lf, y=%.16lf\n", i, j, x, y);
12    f1(j, &i, &y, x);
13    printf("main: i=%6d, j=%6d, x=%.16lf, y=%.16lf\n", i, j, x, y);
14    return 1;
15 }
16
17 void f1(int i, int *j, double *x, double y){
18     (*j)++;
19     i--;
20     *x=y/2.;
21     y*=2;
22     printf("f1: i=%6d, j=%6d, x=%.16lf, y=%.16lf\n", i, *j, *x, y);
23 }

```

Q. 1 A l'aide de diagrammes de représentation mémoire, expliquer le déroulement du programme et donner les résultats affichés. ■

Q. 2 Donner les commandes Linux/Unix permettant d'exécuter le code précédent. ■

EXERCICE 4

Soit a , b et c trois réels donnés, on souhaite résoudre dans \mathbb{R} l'équation

$$ax^2 + bx + c = 0.$$

Q. 1 *Ecrire un programme, sans définir de nouvelles fonctions, permettant de*

- Entrer les données,
- Calculer, si possible, les solutions,
- Afficher les solutions. ■

Q. 2 *Même question en utilisant pour chacune des trois étapes une nouvelle fonction.* ■

EXERCICE 5

Soit $N \in \mathbb{N}^*$ et $p \in \mathbb{N}$. On souhaite calculer

$$S_N^p = \sum_{k=1}^N k^p.$$

Q. 1 *Ecrire la fonction S1 retournant la valeur S_N^p .* ■

Q. 2 *Ecrire la fonction S1BIS retournant la valeur 1 si les données sont correctes, 0 sinon et la valeur de S_N^p sera récupérée via les arguments de la fonction.* ■

Q. 3 *Ecrire un programme utilisant successivement ces deux fonctions* ■

2 Priorité des opérateurs

EXERCICE 6

Listing 4: exo1.c

```
1 #include <stdio.h>
2
3 int main(void){
4     int x;
5
6     x=-3+4*5-6;
7     printf("%d\n",x);
8     x=3+4%5-6;
9     printf("%d\n",x);
10    x=-3*4%-6/5;
11    printf("%d\n",x);
12    x=(7+6)%5/2;
13    printf("%d\n",x);
14    return 1;
15 }
```

Q. 1 *Paranthéser et évaluer chacune des expressions du programme exo1.c.* ■

Q. 2 *Donner l'affichage du code.* ■

EXERCICE 7

Listing 5: exo2.c

```
1 #include <stdio.h>
2
3 int main(void){
4     int x=2,y,z;
5
6     x*=3+2;
7     printf("%d\n",x);
8     x*=y=z=4;
9     printf("%d\n",x);
10    x=y==z;
11    printf("%d\n",x);
12    x==(y=z);
13    printf("%d\n",x);
14    return 0;
15 }
```

Q. 1 *Parthéser et évaluer chacune des expressions du programme exo2.c.* ■

Q. 2 *Donner l'affichage du code.* ■

EXERCICE 8

Listing 6: exo3.c

```
1 #include <stdio.h>
2
3 int main(void){
4     int x,y,z;
5
6     x=2,y=1,z=0;
7     printf("%d\n",0 && 2);
8     printf("%d\n",x);
9     x=x&&y || z;
10    printf("%d\n",x);
11    printf("%d\n",x || !y&&z);
12    return 1;
13 }
```

Q. 1 *Parthéser et évaluer chacune des expressions du programme exo3.c.* ■

Q. 2 *Donner l'affichage du code.* ■

EXERCICE 9

```
1 #include <stdio.h>
2
3 int main(void){
4     int x,y,z;
5
6     x=y=1;
7     z=x++-1;
8     printf("x=%d,y=%d,z=%d\n",x,y,z);
9     z+=-x++ + ++y;
10    printf("x=%d,y=%d,z=%d\n",x,y,z);
11    z=x/++x;
12    printf("x=%d,y=%d,z=%d\n",x,y,z);
13    return 1;
14 }
```

Q. 1 *Parenthéser et évaluer chacune des expressions du programme exo4.c.* ■

Q. 2 *Donner l'affichage du code.* ■

Priorité	Type	Opérateurs	Associativité
0	Primaire	() ¹ . ->	G à D
1	Unaire	! ~ ++ -- + - & * (type) sizeof	D à G
2	Multiplicatif	* / %	G à D
3	Additif	+ -	G à D
4	Décalage	<< >>	G à D
5	Relationnel	< > >= <=	G à D
6	Égalité	== !=	G à D
7	Bits	&	G à D
8	Bits	^	G à D
9	Bits		G à D
10	Logique	&&	G à D
11	Logique		G à D
12	Conditionnel	? :	D à G
13	Affectation	= += -= ...	D à G
14	Virgule	,	G à D

Plus le numéro de priorité est faible, plus la priorité est forte.

Les opérateurs peuvent être **unaire** (une opérande : ! ~ ++ -- + - & * (type) sizeof), **ternaire** (trois opérands : ? :) et **binaires** (deux opérands : tous les autres).

Δ et ∇ deux opérateurs binaires et \odot et \otimes deux opérateurs unaires.

L'opérateur Δ a une priorité **plus forte** que l'opérateur ∇ :

$$a \Delta b \nabla c \iff (a \Delta b) \nabla c$$

$$a \nabla b \Delta c \iff a \nabla (b \Delta c)$$

L'opérateur Δ a une priorité **plus forte** que l'opérateur \odot :

$$a \Delta b \odot c \iff (a \Delta b) \odot c$$

$$\odot a \Delta b \iff \odot (a \Delta b)$$

L'opérateur \otimes a une priorité **plus forte** que l'opérateur unaire ∇

$$\otimes b \nabla c \iff (\otimes b) \nabla c$$

$$b \nabla c \otimes \iff b \nabla (c \otimes)$$

Les opérateurs Δ et ∇ , de même priorité, sont **associatifs de gauche à droite (G à D)** :

$$a \Delta b \nabla c \iff (a \Delta b) \nabla c$$

Les opérateurs Δ et ∇ , de même priorité, sont **associatifs droite à gauche (D à G)** :

$$a \Delta b \nabla c \iff a \Delta (b \nabla c)$$

Les opérateurs \odot et \otimes sont **associatifs de droite à gauche** :

$$\odot b \otimes \iff \odot (b \otimes)$$

¹appel de fonction