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 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

Listing 2: exo1.c

```c
#include <stdio.h>

void f(double *i, double j){
  *i=j+3./2;
  j=*i-1.;
}

void g(double x,double *y){
  *y=x+1.;
  x=*y-2.;
}

int main(){
  double s=1.,t=-1;
  f(&s,t);
  printf("s=%lf, t=%lf\n",s,t);
  g(s,&t);
  printf("s=%lf, t=%lf\n",s,t);
  return 0;
}
```

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 3

Listing 3: exo9.c

```c
#include <stdio.h>
#include <math.h>

void f1(int i, int *j, double *x, double y);

int main(){
  int i=1,j=2;
  double x=3.,y=M_PI;

  f1(i,&j,&x,y);
  printf("main: i=%6d, j=%6d, x=%.16lf, y=%.16lf\n",i,j,x,y);
  f1(j,&i,&y,x);
  printf("main: i=%6d, j=%6d, x=%.16lf, y=%.16lf\n",i,j,x,y);
  return 1;
}

void f1(int i, int *j, double *x, double y){
  (*j)++;
  i--;
  *x=y/2.;
  y*=2;
  printf("f1  : i=%6d, *j=%6d, *x=%.16lf, y=%.16lf\n",i,*j,*x,y);
}
```

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

```c
#include <stdio.h>

int main(void){
   int x;

   x=-3+4*5-6;
   printf("%d\n",x);
   x=3+4%5-6;
   printf("%d\n",x);
   x=-3*4%-6/5;
   printf("%d\n",x);
   x=(7+6)%5/2;
   printf("%d\n",x);
   return 1;
}
```

Q. 1 *Parenthéser et évaluer chacune des expressions du programme* `exo1.c`. ▪

Q. 2 *Donner l'affichage du code.* ▪

Exercice 7

Listing 5: exo2.c

```c
#include <stdio.h>

int main(void){
   int x=2,y,z;

   x*=3+2;
   printf("%d\n",x);
   x*=y=z=4;
   printf("%d\n",x);
   x=y==z;
   printf("%d\n",x);
   x==(y=z);
   printf("%d\n",x);
   return 0;
}
```

Q. 1 *Parenthéser et évaluer chacune des expressions du programme* `exo2.c`. ▪

Q. 2 *Donner l'affichage du code.* ▪

Exercice 8

Listing 6: exo3.c

```c
#include <stdio.h>

int main(void){
   int x,y,z;

   x=2,y=1,z=0;
   printf("%d\n",0 && 2);
   printf("%d\n",x);
   x=x&&y||z;
   printf("%d\n",x);
   printf("%d\n",x||!y&&z);
   return 1;
}
```

Q. 1 *Parenthéser et évaluer chacune des expressions du programme* `exo3.c`. ▪

Q. 2 *Donner l'affichage du code.* ▪

Exercice 9

Listing 7: exo4.c

```c
#include <stdio.h>

int main(void){
   int x,y,z;

   x=y=1;
   z=x++-1;
   printf("x=%d,y=%d,z=%d\n",x,y,z);
   z+=-x++ + ++y;
   printf("x=%d,y=%d,z=%d\n",x,y,z);
   z=x/++x;
   printf("x=%d,y=%d,z=%d\n",x,y,z);
   return 1;
}
```

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	Egalité	== !=	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 opérandes : ? :) et **binaire** (deux opérandes : tous les autres).

$\triangle$ et $\triangledown$ deux opérateurs binaires et $\odot$ et $\otimes$ deux opérateurs unaires.

L'opérateur $\triangle$ a une priorité **plus forte** que l'opérateur $\triangledown$:

$$a \triangle b \triangledown c \iff (a \triangle b) \triangledown c$$

$$a \triangledown b \triangle c \iff a \triangledown (b \triangle c)$$

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

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

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

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

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

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

Les opérateurs $\triangle$ et $\triangledown$, de même priorité, sont **associatifs de gauche à droite (G à D)** :

$$a \triangle b \triangledown c \iff (a \triangle b) \triangledown c$$

Les opérateurs $\triangle$ et $\triangledown$, de même priorité, sont **associatifs droite à gauche (D à G)** :

$$a \triangle b \triangledown c \iff a \triangle (b \triangledown c)$$

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

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

[1] appel de fonction