Université Mohamed Premier **Filière IGE - Semestre** 2
École Supérieure de Technologie **Année universitaire :** $2017 - 2018$
Oujda **Module : Algorithmique** & **Langage** C

Solutions des TD/TP N° 1

Exercice 1 (TD)

1. Les quatre types de base du langage C :
   - **char** : 1 octet.
   - **int** : 4 octets.
   - **float** : 4 octets.
   - **double** : 8 octets.
2. Un **identificateur** est un nom qu'on donne à une variable pour l'identifier dans un programme.
3. Les règles pour nommer les identificateurs :
   - Un identificateur doit commencer par une lettre (minuscule ou majuscule) ou par un caractère souligné '_'.
   - Un identificateur ne doit contenir que des lettres (minuscules ou majuscules), des chiffres et/ou des caractères soulignés '_'.
   - Un identificateur ne doit pas être un mot clé du langage C.

   Les lettres majuscules ne sont pas équivalentes aux lettres minuscules, car le langage C est sensible à la casse.
   Oui, il est tout à fait légal qu'un identificateur C contienne des chiffres.
4. – Une constante décimale commence par un chiffre non nul.
   - Une constante octale commence par le chiffre 0.
   - Une constante hexadécimale commence par $0x$ ou $0X$.

Exercice 2 (TD)

- **record**1 : identificateur valide.
- $**tax** : identificateur non valide, à cause du symbole $.
- **name_and_address** : identificateur valide.
- 2**record** : identificateur non valide, car il commence par un chiffre.
- **name** : identificateur valide.
- **name-and-address** : identificateur non valide, à cause du symbole $-$.
- **file**-3 : identificateur non valide, à cause du symbole $-$.
- **name and address** : identificateur non valide, car il contient des espaces.
- **return** : identificateur non valide, car c'est un mot-clé du langage C.
- **char** : identificateur non valide, car c'est un mot-clé du langage C.

– 123_45_6789 : identificateur non valide, car il commence par un chiffre.
– **_name_and_address** : identificateur valide.

Exercice 3 (TD)

(a) Déclaration des variables entières : index (initialisée à 0), p, q, compteur :

```
int index = 0, p, q, compteur;
```

(b) Déclaration des variables flottantes simple précision : racine_1, racine_2, x, y, z (initialisée à -5.78) :

```
float racine_1, racine_2, x, y, z = -5.78;
```

(c) Déclaration des variables caractères : a, b, c, courrant, dernier (initialisée à la lettre u) :

```
char a, b, c, courrant, dernier = 'u';
```

(d) Déclaration des variables double précision : tax, net, erreur (initialisée à $12.345E-13$) :

```
double tax, net, erreur = 12.345E-13;
```

Exercice 4 (TD)

(a) $2 * ((i/5) + (4 * (j-3))\%(i+j-2))$: type **int**, valeur = 18.

```
(i / 5) = (8 / 5) = 1
(4 * (j - 3)) = (4 * (5 - 3)) = (4 * 2) = 8
(i + j - 2) = (8 + 5 - 2) = (13 - 2) = 11
(4 * (j - 3)) % (i + j - 2) = 8 % 11 = 8
((i / 5) + (4 * (j - 3)) % (i + j - 2)) = (1 + 8) = 9
2 * ((i / 5) + (4 * (j - 3)) % (i + j - 2)) = 2 * 9 = 18
```

(b) $2 * x + (y == 0)$: type **float**, valeur = 0.01.

```
(y == 0) = 0 => vers float = 0.0
2 * x = 2 * 0.005 = 0.01
```

```
2 * x + (y == 0) = 0.01 + 0.0 = 0.01
```

(c) $5 * (i + j) >$ 'c' : type **int**, valeur $= 0$.

```
(i + j) = (8 + 5) = 13
5 * (i + j) = 5 * 13 = 65
5 * (i + j) > 'c' = 65 > 99 = 0
```

(d) $(i - 3 * j)\%(c + 2 * d)/(x - y)$: type **float**, valeur $= -466.666667$.

```
(i - 3 * j ) = (8 - 3 * 5) = (8 - 15) = - 7
(c + 2 * d) = (99 + 2 * 100) = 299
(x - y) = (0.005 - -0.01) = 0.015
(i - 3 * j ) % (c + 2 * d) = -7 % 299 = -7 => vers float = -7.0
(i - 3 * j ) % (c + 2 * d) / (x - y) = -7.0 / 0.015 = -466.666667
```

(e) $2 * x + y == 0$: type **int**, valeur $= 1$.

```
2 * x + y = 2 * 0.005 + -0.01 = 0.01 + -0.01 = 0.0
2 * x + y == 0 = 1
```

(f) $!(c == 99)$: type **int**, valeur $= 0$.

```
(c == 99) = 1
!(c == 99) = !1 = 0
```

(g) $(x > y)\&\&(i > 0)\&\&(j < 5)$: type **int**, valeur $= 0$.

```
(x > y) = (0.005 > -0.01) = 1
(i > 0) = (8 > 0) = 1
(j < 5) = (5 < 5) = 0
(x > y) && (i > 0) && (j < 5) = 1 && 1 && 0 = 1 && 0 = 0
```

(h) $(0 < i)\&\&(i < j)$: type **int**, valeur $= 0$.

```
(0 < i) = (0 < 8) = 1
(i < j) = (8 < 5) = 0
```

```
(0 < i) && (i < j) = 1 && 0 = 0
```

(i) $0 < i < j$: type **int**, valeur $= 1$.

```
0 < i < j = 0 < 8 < 5 = 1 < 5 = 1
```

Exercice 5 (TD)

```
k = (i + j) = (8 + 5) = 13
y -= x <==> y = y - x = -0.01 - 0.005 = -0.015
z = (x + y) = (0.005 + -0.015) = -0.01
x *= 2 <==> x = x * 2 = 0.005 * 2 = 0.01
i = j = 5
i /= j <==> i = i / j = 5 / 5 = 1
k = (x + y) = (0.01 + -0.015) = -0.005 => vers int = 0
i %= j <==> i = i % j = 1 % 5 = 1
k = c = 'c' = 99
i += (j - 2) <==> i = i + (j - 2) = i + (5 - 2) = 1 + 3 = 4
z = i / j = 4 / 5 = 0 => vers float z = 0.000000
a = b = d :
  - on affecte d à b : b = 'd' = 100
  - on affecte b à a : a = 100
i = j = 1.1 :
  - on affecte 1.1 à j => vers int j = 1
  - on affecte j à i : i = 1
z = k = x :
  - on affecte x = 0.01 à k => vers int k = 0
  - on affecte k = 0 à z => vers float z = 0.000000
i -= (j > 0) ? j : 0 <==> i = i - (j > 0) ? j : 0
   (j > 0) = (1 > 0) = 1
   (j > 0) ? j : 0 = j = 1
   i = i - (j > 0) ? j : 0 = 1 - 1 = 0
```

Exercice 6 (TP)

```
#include <stdio.h>

main()
{
   int a, b, g, p;
```

```
   int d, q, r;

   printf("Tapez deux entiers S.V.P : ");
   scanf("%d%d", &a, &b);
   printf("leur somme : %d\n", a + b);
   printf("leur produit : %d\n", a * b);
   g = (a > b) ? a : b;
   printf("le plus grand est : %d\n", g);
   p = (a < b) ? a : b;
   printf("le plus petit est : %d\n", p);
   d = g - p;
   q = g / p;
   r = g % p;
   printf("leur difference : %d\n", d);
   printf("leur quotient : %d\n", q);
   printf("leur reste de division : %d\n", r);
   system("pause");
}
```

Exercice 7 (TP)

```
#include <stdio.h>

main()
{
   char car;

   printf("Saisir un caractere : ");
   scanf("%c", &car);
   /* ou bien : car = getchar(); */
   printf("Vous avez saisi le caractere : %c\n", car);
   printf("Son code ASCII est : %d\n", car);
   system("pause");
}
```

Exercice 8 (TP)

```
#include <stdio.h>

main()
{
```

```
   float x, y, z;

   printf("Saisir deux nombres reels : ");
   scanf("%f%f", &x, &y);
   printf("x = %f, y = %f\n", x, y);
   z = x;
   x = y;
   y = z;
   printf("Apres echange : x = %f, y = %f\n", x, y);
   system("pause");
}
```

Exercice 9 (TP)

1. Algorithme :

```
Algorithme Facture de chaises.
CONST
   TVA : Réel = 0.2
VAR
   quantite : Entier
   prix_unit : Réel
   montant_brut : Réel
   montant_net : Réel
   montant_tva : Réel
   taux_remise : Réel
   montant_remise : Réel
DEBUT
   LIRE(quantite);
   LIRE(prix_unit);
   LIRE(taux_remise);
   montant_brut = quantite * prix_unit;
   montant_remise = montant_brut * taux_remise;
   montant_tva = (montant_brut - montant_remise) * TVA;
   montant_net = montant_brut - montant_remise + montant_tva;
   AFFICHER(montant_brut);
   AFFICHER(montant_net);
FIN
```

2. Programme C :

```
#include <stdio.h>

#define  TVA  0.2
```

```
main()
{
   int qte;
   float pu, brut, net, tva, taux_rem, remise;

   printf("quantite : ");
   scanf("%d", &qte);
   printf("prix unitaire : ");
   scanf("%f", &pu);
   printf("taux de remise : ");
   scanf("%f", &taux_rem);
   brut = qte * pu;
   remise = brut * taux_rem;
   tva = (brut - remise) * TVA;
   net = brut - remise + tva;
   printf("montant brut : %f\n", brut);
   printf("montant net : %f\n", net);
   system("pause");
}
```