TD1 : Algorithmique et Programmation C

But du TD : Rappels d'algorithmique et programmation simple en langage C. Séquences, conditions et boucles. Fonctions printf et scanf.

Exercice 1 : Avec le langage algorithmique vu en cours, écrivez un algorithme qui permet d'affecter deux valeurs à des entiers et d'afficher leurs somme, produit, soustraction, division entière et reste de la division entière. Transformez l'algorithme en programme C.

Exercice 2 : Ecrivez un algorithme qui permet d'échanger les valeurs de deux variables entières. On affichera les variables avant et après l'échange. Transformez l'algorithme en programme C.

Exercice 3 : Ecrivez un algorithme puis un programme C qui permet d'afficher la parité d'un nombre tapé au clavier. On s'attachera à bien vérifier que ce qui est tapé par l'utilisateur est bien un nombre.

Exercice 4 : Ecrivez un algorithme puis un programme C qui calcule le produit d'entiers tapés au clavier. Le programme demandera à chaque fois à l'utilisateur s'il a terminé de taper les nombres.

Exercice 5 : Ecrivez un algorithme puis un programme C qui calcule la somme pondérée de N entiers tapés au clavier (on lira aussi au clavier le poids affecté à chaque nombre). Ici on demandera d'abord le nombre d'entiers N puis on effectuera une boucle **for**.

Rappels :

* Programme minimaliste :

```
#include <stdio.h>
int main(void)    {
        printf("Bonjour tout le monde \n") ;
        return 0 ;
}
```

* Types de base :
Entiers : short (2 octets), int (4 octets), long (4 ou 8 octets) ; Réels : float (4 octets), double (8 octets) ; Caractères : char
* Déclaration de variable : <type> <nom_variable> [,<nom_variable>]* ;
Exemples : int entier ; float reel ; int i,j ; char car ;
* Constantes :
Entières : 1, 2, -3, 1000 ; Réelles : 3.14, 0.5, .5, 1000.0, 1000, 1e3, 1E-3 ; Caractères : 'a', 'Z', '\n', '\0', '1'.
* Affectation : <nom_variable> = <expression> ;
Exemples : eniter = 1 ; reel = 1E-3 ; car = 'a' ;
* Opérateurs :
Entiers : +,-,* (produit),/ (division entière),% (reste de la division entière)
Binaires : & (et bit à bit),| (ou bit à bit),~ (inversion, complément à 1), ^(ou exclusif)
Réels : +,-,*,/
Comparaisons : == (égalité – ne pas confondre avec l'affectation !),!= (inégalité), <, >, <= (inférieur ou égal), >=
Logiques : && (et), || (ou), ! (non)
* Ecriture écran : printf(<format>, <expression> [,<expression>]*)
Format : %d pour une expression entière, %f pour une expression réelle de type float, \n retour à la ligne…
Exemple : printf("un entier : %d ; un réel : %f \n", entier, reel) ;
* Lecture clavier : scanf(<format>, &<variable1> [,&<variable2>]*) ; Note : scanf renvoie le nombre de lectures effectuées.
Exemple : scanf("%d",&entier) ; scanf("%f",&reel) ;
* Condition : if (<expression>) instruction1 [else instruction2] ;
// si expression est différent de 0 on exécute l'instruction 1 sinon on fait l'instruction 2 (facultatif)
Exemple : if(x>1) x = x + 1 ;
* Boucles :
while (<expression>) instruction ; // tant que expression est différent de 0 on exécute instruction.
Exemple : while (x>0) x = x – 1 ;
for(<instruction1> ;<expression> ;<instruction2>) instruction3 ; // on exécute instruction1 une fois puis on teste expression, si elle est différente de 0, on exécute instruction3 puis instruction2 et on recommence à tester. Exemple : for(i = 0 ; i < 10 ; i = i + 1) printf("%d\n", i) ;
Note : on peut toujours remplacer une instruction par un ensemble d'instruction dans un bloc : { inst1 ; inst2 ; inst3 ; }

TD1 : Corrigé

Exercice 1 : Avec le langage algorithmique vu en cours, écrivez un algorithme qui permet d'affecter deux valeurs à des entiers et d'afficher leurs somme, produit, soustraction, division entière et reste de la division entière. Transformez l'algorithme en programme C.

Solution :

```
Algorithme Calculette
/* Rôle : Réaliser une calculette avec des opérateurs arithmétiques de base */
Variable
  nb1, nb2 : entier /* Opérandes */
  resultat :  entier /* Résultat de l'opération */
Début
  /* Affectation de valeurs aux variables */
  nb1 <- 1
  nb2 <- 2
  /* Affichage des résultats */

  resultat <- nb1+nb2
  Ecrire("Somme : ", resultat)
  resultat <- nb1*nb2
  Ecrire("Produit : ", resultat)
  resultat <- nb1-nb2
  Ecrire("Soustraction : ", resultat)
  resultat <- nb1/nb2
  Ecrire("Division entière : ", resultat)
  resultat <- nb1%nb2
  Ecrire("Reste de la division entière : ", resultat)
Fin
```

```
/* Programme en C */
/* Inclusion du fichier d'entête contenant les prototypes de fonctions d'E/S */
#include <stdio.h>

/*************************************************************/
/* Rôle : Réaliser une calculette avec des opérateurs arithmétiques de base  */
/* Renvoie le code 0 quand tout s'est bien déroulé                           */
/*************************************************************/
int main(void)    {
        /* Déclaration des variables contenant les données */
        int nb1, nb2 ; /* Opérandes */
        int resultat ; /Résultat de l'opération
        /* Affectation de valeurs aux variables */
        nb1 = 1 ;
        nb2 = 2 ;
        /* Affichage des résultats */
        resultat = x+y ;
        printf("somme : %d\n", resultat) ;
        resultat = nb1*nb2 ;
        printf("produit: %d\n", resultat) ;
        resultat = nb1-nb2 ;
        printf("soustraction : %d\n", resultat) ;
        resultat = nb1/nb2 ;
        printf("division entière : %d\n", resultat) ;
        resultat = nb1%nb2 ;
        printf("reste de la division entière : %d\n", resultat) ;
        return 0 ;
}
```

Exercice 2 : Ecrivez un algorithme qui permet d'échanger les valeurs de deux variables entières. On affichera les variables avant et après l'échange. Transformez l'algorithme en programme C.

Solution :
Remarque : on va utiliser un ' ; ' comme fin d'instruction (idem qu'en langage C)

Algorithme Permutation
```
/* Permute deux valeurs */
Variable
  x, y : entier; /* Variables à permuter */
  tempo : entier ; /* Variable temporaire utilisée pour la permutation */
Début
  /* Affectation des valeurs à permuter */
  x <- 1 ;
  y <- 2 ;
  Ecrire ("x = , x, "y = ", y) ;
  /* Permutation */
  tempo <- x ;
  x <- y ;
  y <- tempo ;
  Ecrire ("x = , x, "y = ", y) ;
Fin
```

/* Programme en C sans affichage des commentaires */
```
#include <stdio.h>
int main() {
  int x;
  int y;
  int tempo;

  x = 1 ;
  y = 2 ;
  printf("x = %d y = %d\n", x, y) ;
  tempo = x;
  x = y;
  y = tempo;
  printf("x = %d y = %d\n", x, y) ;
  return 0;
}
```

Exercice 3 : Ecrivez un algorithme puis un programme C qui permet d'afficher la parité d'un nombre tapé au clavier. On s'attachera à bien vérifier que ce qui est tapé par l'utilisateur est bien un nombre.

Solution :

```
Algorithme Parité
/* Détermine si un nombre saisi au clavier est pair */
Variable
  x : entier ; /* nombre à saisir */
Début
  Ecrire ("Saisir un entier : ") ;
  Lire (x) ;
  /* Déterminer la parité  et affichage */
  Si ((x%2) = 0) Alors
    Ecrire ("Le nombre", x, "est pair") ;
  Sinon
    Ecrire ("Le nombre", x, "est impair") ;
Fin
```

Remarque : Dans les programmes suivants et pour focaliser sur les instructions, les commentaires ne sont pas mis. Mais, Attention, vous devez toujours commenter vos programmes.

```
#include <stdio.h>
int main() {
  int x;
  int res;
  printf("Saisir un entier : ");
  res = scanf("%d", &x);
  if (res != 1) {
    printf("Erreur : le nombre saisi n'est pas un entier\n");
    return -1 ;
  else {
    if ((x % 2) == 0)
      printf("le nombre %d est pair\n", x);
    else
      printf("le nombre %d est impair\n", x);
    return 0;
  }
}
```

Exercice 4 : Écrivez un algorithme puis un programme C qui calcule le produit d'entiers tapés au clavier. Le programme demandera à chaque fois à l'utilisateur s'il a terminé de taper les nombres.

Solution1 :

```
#include <stdio.h>
int main() {
  int x;
  char car;
  int produit = 1;
  int res;
  do {
    printf("Saisir un entier : ");
    res=scanf("%d", &x);
    fflush(stdin); /* Fonction de vidage du buffer d'entrée */
    if (res==1) {
      produit = produit * x;
      printf("Voulez-vous arrêter (Taper Y pour arrêter ou N pour continuer) : ");
      scanf ("%c", &car);
      fflush(stdin);
    }
  }
  while (car!='Y');
  printf("produit = %d\n", produit);
  return 0;
}
```

Solution 2 :

```
/* Pas d'utilisation de fflush(stdin) car non portable sur tous les systèmes */
/* Ecriture de code de vidage du buffer associé au clavier */
#include <stdio.h>
int main() {
  int x;
  int produit;
  char fin; // Caractère d'arrêt de saisie
  int res; // Retour de la saisie par scanf
  char cbuf; // caractère restant dans le buffer
  produit=1;
  do {
    printf("Saisir un entier : ");
```

```
   res=scanf("%d", &x);
   while ((cbuf = getchar ()) != '\n' && cbuf != EOF); /* Vidage de buffer */
   if (res==1) {
   produit = produit * x;
   printf("Voulez-vous arrêter (Taper Y pour arrêter ou une autre touche pour continuer) : ");
   scanf ("%c", &fin);
   while ((cbuf = getchar ()) != '\n' && cbuf != EOF);
   }
  }
  while (fin!='Y');
  printf("produit = %d\n", produit);
  return 0;
}
```

Remarque :

```
/* Il vaut mieux réaliser une fonction de vidage de buffer d'entrée  et l'appeler après chaque scanf()*/
void vidage_stdin(void)
{
  int c ;
  do {
    c=getchar() ;
  } while ((c != '\n') && (c != EOF)) ;
}
```

Exercice 5 : Écrivez un algorithme puis un programme C qui calcule la somme pondérée de N entiers tapés au clavier (on lira aussi au clavier le poids affecté à chaque nombre). Ici on demandera d'abord le nombre d'entiers N puis on effectuera une boucle **for**.

Solution :

```
#include <stdio.h>

/*************************************/
/* Vidage du buffer d'entrée         */
/*************************************/
void vidage_stdin(void)
{
  int c ;
  do {
    c=getchar() ;
  } while ((c != '\n') && (c != EOF)) ;
}

/*************************************/
/* Calcul de la somme pondérée de N entiers   */
/* Retour : 0 (succès)                        */
/*************************************/

int main(void){
  int nb; // Nombre de nombres à saisir
  int x , poids ; // nombre et son poids
  int somme ;
  int i; // variable de boucle
  int res ; // résultat du scanf()

  // Saisie du nombre d'entiers
  do {
    printf("Saisir le nombre d'entiers : ");
    res = scanf("%d", &nb);
    vidage_stdin() ;
  } while (res!=1);
```

```
  // Saisie des entiers avec le poids et calcul de la somme
  somme = 0;
  for(i = 0; i < nb ; i = i + 1) {
    do {
      printf("Saisir un entier et son poids : ");
      res = scanf("%d%d", &x, &poids);
      vidage_stdin() ;
    } while (res!=2);
    somme = somme + (poids * x) ;
  }

  // Affichage de la somme pondérée
  printf("somme pondérée : %d\n", somme);
  return 0 ;
}
```