Travaux pratiques n°1:
Introduction au langage C

Objectifs

- Introduction au langage C
- Coder des algorithmes détaillés en langage C
- Découvrir les premières erreurs de compilation

Documents nécessaires

Sujet du td1 d'algorithmique.

Partie 1: Introduction pratique au langage C

Objectifs

- Introduction au langage C (premier programme, compilation, exécution, variable, constante, affectation, entrées-sorties)

Préparation

(Après votre connexion) créez un répertoire *bdp* dans lequel vous placerez les fichiers créés pendant les séances de travaux pratiques de bdp. Dans le répertoire bdp, créez un srépertoire *tp1* dans lequel vous placerez les fichiers créés pendant cette première séance. Si ce n'est déjà fait ouvrez un *terminal* pour vous placer dans le répertoire *tp1* qui devient votre répertoire courant.

Structure d'un programme C

Question 1:

En utilisant un éditeur de texte (*xemacs, nedit, ...)* , créer et éditer le programme *essai.c* suivant:

/ [en tête cf. cours] */*

#include <stdio.h>
int main(void)

```
{
  printf("bonjour \n");
  return (0) ;
}
```

Les parties délimitées par **/*** et ***/** sont des commentaires qui n'ont aucun effet sur l'exécution du programme mais sont très importants pour rendre ce dernier lisible.
int main(void) est obligatoire suivant la normalisation du langage C et sera expliqué ultérieurement. Les instructions sont placées dans le bloc délimité par **{** et **}** : on appelle ce bloc *le corps du programme*. L'unique instruction **return (0)** sert à signaler au système d'exploitation que le déroulement de l'exécution a été correct.

A l'exécution du programme, pour afficher un message vous devez utiliser l'instruction **printf**. La directive de compilation **#include <stdio.h>** est nécessaire pour pouvoir utiliser l'instruction **printf**. Il s'agit d'une instruction destinée au compilateur qui sera expliquée ultérieurement
\n est un code inséré dans le message indiquant un passage à la ligne suivante. Dans *essai.c*, il y aura passage à la ligne suivante après l'affichage de *bonjour* à l'exécution du programme.
NOTE : l'instruction **return (0)** sera toujours placée en dernier car elle met fin à l'exécution du programme.

Pour passer du programme *C* (ou *programme source*) à un programme exécutable opérez de la façon suivante depuis un terminal :
1 – compilation et édition des liens : *gcc essai.c –o essai* ↵
2 - exécution : *./essai* ↵

Note : par simplicité nous emploierons dans la suite de ce texte le terme *compiler*, pour désigner les 2 étapes de compilation et d'édition des liens.

Déclarer, affecter et afficher des variables

(les enseignants vous donneront les explications concernant les éléments de langage C présentés dans la suite de ce texte)

Question 2:

Modifier le fichier *essai.c* pour obtenir, compiler et exécuter le programme suivant (inclure les commentaires) :

```
#include <stdio.h>

int main(void)
```

```
{               /* DEBUT DES DECLARATIONS */

                /* variables */
        int    a  =  5   ;
        float b  = 6.3 ;


        /* DEBUT DES INSTRUCTIONS */

        printf("\n debut du programme" ) ;

                /* affichage des resultats*/
        printf("\n a vaut :  %d", a) ;  /* %d est le format pour une variable de type int */
        printf("\n b vaut : %f",  b)   ; /* %f est le format pour une variable de type float */

        printf("\n fin du programme" ) ;
        return (0) ;
}
```

Les variables *a* et *b* ont été initialisées dès leur déclaration.
Modifiez le programme de façon à déclarer ces variables sans les initialiser : **int a ; float b ;**
Compilez et exécutez. Que concluez-vous ?

Question 3:

Modifier le fichier *essai.c* pour obtenir, compiler et exécuter le programme suivant :

```
#include <stdio.h>

int main(void)
{               /* DEBUT DES DECLARATIONS */

                /* variables */
        int    a  =  5   ;
        float b  = 6.3 ;


        /* DEBUT DES INSTRUCTIONS */

        printf("\n debut du programme" ) ;
                        /* Traitement */
```

```
        a = 7 ;
        b = a + b + 1 ;

                /* affichage des resultats*/
        printf("\n a vaut :  %d", a) ;
        printf("\n b vaut : %f",  b)   ;

        printf("\n fin du programme" ) ;
        return (0) ;
}
```

Entrées-Sorties

On appelle les entrées-sorties les communications entre un programme et l'utilisateur de ce programme.

Les entrées

L'instruction qui gère les *entrées*, c'est à dire la saisie de valeurs fournies par l'utilisateur à l'intention du programme est **scanf** dont l'équivalent dans le langage algorithmique est **demander**. Chaque valeur saisie par **scanf** doit être stockée dans une variable du programme.
NOTE IMPORTANTE : Comme il le sera précisé en cours, dans un **scanf** il faut obligatoirement précéder le nom de la variable avec le caractère **&**. On vous expliquera à quoi correspond ce **&** ultérieurement.

Les sorties

Comme nous l'avons vu l'instruction qui gère les *sorties,* c'est à dire l'affichage de valeurs fournies par le programme vers l'utilisateur, est **printf** dont l'équivalent dans le langage algorithmique est **afficher**.

De façon générale une instruction **printf** peut afficher plusieurs valeurs, certaines de ces valeurs étant le résultat d'une expression (rappel : une expression est la description d'une opération qui rend un résultat comme, par exemple, une opération arithmétique).

Question 4:

Modifier le fichier *essai.c* pour obtenir, compiler et exécuter le programme suivant :

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

```
int main(void)
{
                    /* DEBUT DES DECLARATIONS */
                    /* variables */
        int   a  =  5   ;
        float b  = 6.3 ;
                    /* constantes */

                    /* DEBUT DES INSTRUCTIONS */
        printf("\n début du programme ") ;

        /* saisie des valeurs auprès de l'utilisateur */

        printf("\n entrez une valeur entiere : ") ;
        scanf("%d", &a) ;

        printf("\n entrez une valeur reelle : ") ;
        scanf("%f", &b) ;

                    /* Affichage des variables */

        printf("\n a vaut :  %d", a) ;
        printf("\n  b vaut : %f",  b)   ;
        printf("\n a vaut :  %d, b vaut %f, le double de b vaut : %f", a, b, 2*b) ;

        printf("\n fin du programme") ;
        return(0);
}
```

Vous constatez que les valeurs saisies ont écrasé les valeurs précédentes des variables *a* et *b* : une saisie est une affectation.

Partie 2: Traduire des corps d'algorithmes détaillés en langage C

Objectifs

- Créer son premier programme.
- Découvrir les premières erreurs de compilation

Préparation

Utilisez un éditeur de texte pour *éditer* et sauvegarder dans un fichier *modele.c* le programme suivant qui servira de modèle à tous vos futurs programmes *C* :

```
/* ….[en tête cf. cours] */

                /* DIRECTIVES D'INCLUSION */
#include <stdio.h>

int main(void)
{

/* DONNEES D'ENTREE */

        /* variables d'entrée initialisées */

        /* variables d'entrée non initialisées */

/* VARIABLES RESULTATS */

/* VARIABLES AUXILIAIRES */


/* DEBUT DES INSTRUCTIONS */




        return (0) ;
}
```

Exercice 1

A partir du modèle précédent créez (dans le répertoire *tp1*) compilez et exécutez :
un fichier *salaire.c* traduisant en langage *C* l'algorithme de la question b de l'exercice 3 du td1

Exercice 2 : La tortue existe!!!!

Écrire le programme correspondant à l'exercice 1 du td1.

NOTE pour l'utilisation de la tortue.

soit un programme *essaiTortue.c* utilisant la tortue,vous devez:

1. Ajouter la directive d'inclusion :
   #include <Tortue/Tortue.h>

3. Séparer les étapes de compilation et d'édition des liens en tapant:

   1) compilation :
   gcc –c essai.c –o essai.o
   2) édition des liens :
   gcc –o essai essai.o -lTortue

Exemple:

```
/* ….[en tête cf. cours] */

/* DIRECTIVES D'INCLUSION *
#include <stdio.h>
#include  <Tortue/Tortue.h>

int main(void)
{
                        /* DEBUT DES INSTRUCTIONS */
printf("\n debut du programme essaiTortue" ) ;
printf("\n Observez ce que sait faire la tortue");                    */
ouvre();        /* ouvre une fenêtre graphique, jamais de printf ou de scanf
                        après cette instruction */
avance(100);
tourne(90);
avance(100);
ferme();        /* ferme le fenêtre graphique, scanf et printf à nouveau autorisés */
printf("\n fin du programme essaiTortue" ) ;
return (0) ;
}
```