Développement logiciel (S4)

Initiation au langage C
1- Entrées/sorties et structures de contrôle

Objectifs : Être capable de développer des applications basiques de type console en C. Être capable d'utiliser les entrées/sorties et les structures de contrôle élémentaires.

Avertissement

Il n'est pas question de développer des applications en langage C de type console très complexes. La finalité est de vous permettre d'acquérir les bases de l'algorithmique et de la programmation en langage C pour développer ensuite sur des applications d'informatique embarquée sur des plate-formes comme Arduino, Mbed, PsoC, etc...

Tout au long de ce TP, vous répondrez au questions posées sur votre compte-rendu. Celui ci devra inclure les programmes réalisés. Ces programmes devront être commentés !

Analyse et modification du fichier Hello world

Comme nous l'avons vu dans le tutoriel Code::blocks, le fichier main.c permet d'afficher la phrase « Hello world ».

Des commentaires ont été rajoutés :

```
#include <stdio.h>      //  lignes destinées au compilateur (*)
#include <stdlib.h>     //  lignes destinées au compilateur

int main()             // En C, le programme principal est une fonction qui s'appelle toujours main
                      // quand on appelle main, cette fonction retourne un entier (int)
{                      // { début de fonction
    printf("Hello world!\n");   // Afficher hello world puis aller à la ligne \n
    return 0;                  // à la fin de la fonction on renvoie la valeur 0
}                              // } fin de fonction
```

(*) Les deux premières lignes qui commencent par # sont destinées au compilateur.
Stdio.h signifie Standard Input/Output Header, cette ligne permet d'utiliser les fonctions standard du langage C, comme la fonction printf.
Stdlib.h permet d'exécuter diverses opérations dont la conversion, la génération de nombres pseudo-aléatoires, etc.

Manipulation n°1

Q.1) Créer un projet appelé Bonjour qui affiche :
Hello world !
Bonjour monde !
Hola mundo !
Ciao mondo !

Faites vérifier votre travail par l'enseignant.

Q.2) Supprimer les \n. Que constatez vous ? Remplacer les \n par \t. Que constatez vous ?

Q.3) Dans votre répertoire de travail, listez tous les répertoires et fichiers présents avec leur attribut (comme .c ou .exe)*. Quels sont les répertoires et fichiers que vous avez directement créé et ceux créés par la compilation.

Remarque : en mode console, on se passera des accents sur les messages affichés !

Saisie de données

Programme exemple

On donne le programme d'exemple suivant :

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

int main()
{
    int age = 0; // On déclare et initialise la variable à 0
    printf("Quel age avez-vous ? ");
    scanf("%d", &age); // On récupère l'age avec scanf
    printf("Ah ! Vous avez donc %d ans !\n\n", age);
    return 0;
}
```

On remarque dans ce programme :
On définit une variable age qui est un entier int. Cette variable est initialisée à la valeur 0.
la lecture du clavier se fait avec l'instruction scanf
le format est le suivant : `scanf("%d", &age)`
ici `"%d"` spécifie le format de lecture %d indique la lecture d'un entier
`&age` représente l'adresse de age, le résultat sera stocké dans la variable age.
Voir l'annexe Affichage des variables

Manipulation n°2

Q.1) Créer le projet age, saisir le programme fourni, le tester et vérifier son fonctionnement.

Travaux dirigés

On donne l'algorithme suivant,

Q.1) Expliquer ce que fait cet algorithme et indiquer ce qui s'affiche à l'écran

```
---------------------------------------
Algorithme : Calcul d'age
---------------------------------------
Variables numériques : date, naissance, age (entiers)
     DEBUT
            Afficher "En quelle année sommes-nous ? "
            Saisir date
            Afficher "  En quelle année êtes vous né(e) ? "
            Saisir naissance
            age <- date - naissance
            Afficher "  Cette année, vous avez donc " , age, " ans !
      FIN
```

Manipulation n°3

Q.1) Modifier le programme age saisi précédemment conformément à l'algorithme que vous venez d'étudier pour obtenir à l'exécution un affichage conforme à l'écran suivant :

```
En quelle annee sommes nous ? 2014
En quelle annee etes-vous ne(e) ? 1995
Cette annee, vous avez donc 19 ans !
```

Mini-projet calcul de résistances

Nous allons maintenant travailler au développement d'un mini-projet logiciel.

Pour vous aider, vous disposez d'un document ressources intitulé « le langage C ». Ce document est accessible sur le serveur sous forme de fichier pdf.
Ce document très synthétique est un manuel du langage C, il vous sera utile pour développer en C.

Référence du document :
Le Langage C - Version 1.2 - (c) 2002 – Florence HENRY
Ce document est accessible sur internet à l'adresse suivante :
http://carnot.physique.uvsq.fr/~prd/polycoursc/cours.pdf

Nous allons maintenant et tout au long de cette séquence développer un logiciel qui calcule la résistance équivalente à plusieurs résistances associées en série ou en parallèle. Nous procéderons par étapes.

On rappelle lorsque deux résistances sont associées en série, la résistance équivalente est : $R_{EQSérie} = R1 + R2$, lorsque 2 résistances sont associées en parallèle, la résistance équivalente est : $R_{EQParallele} = \frac{1}{\frac{1}{R1} + \frac{1}{R2}}$

Manipulation n°4

Attention, les programmes commencent un peu à se compliquer. Cela peut être l'occasion de s'initier à la prise en main du debugger !

Q.1) Créer un projet nommé serie1 et générer une application qui calcule la résistance équivalente à 2 résistances associées en série.

Q.2) Créer un projet nommé para1 et générer une application qui calcule la résistance équivalente à 2 résistances associées en parallèle. Attention à la division avec des entiers !

Structures de contrôle

Le travail qui suit ne se fait pas derrière un clavier et un écran mais devant une feuille de papier avec un crayon (et une gomme) ! Une lecture attentive est demandée.

Instructions conditionnelles

Dans notre application de calcul de résistance, on souhaite n'avoir qu'un seul programme pour le calcul série et parallèle. A l'ouverture du programme, une première ligne demande si on veut un calcul d'association de résistances en série (taper 1) ou en parallèle (taper 2).

Pour cela, il est nécessaire d'employer une structure de contrôle. Deux solutions sont possibles en utilisant le branchement conditionnel (if) ou le branchement sur choix (switch).

Branchement conditionnel : if

L'instruction if est la structure de test la plus basique, on la retrouve dans tous les langages (avec une syntaxe différente...). Elle permet d'exécuter une série d'instructions si jamais une condition est réalisée.
Elle s'utilise ainsi: `if (expression) {instruction ;}`

que l'on peut traduire par « si (expression) est vrai, on exécute {instruction ;} ».

A noter : instruction peut représenter une instruction élémentaire mais aussi une suite d'instructions séparées par des ;

Cette structure est la plus simple, mais on utilise plus souvent la variante suivante :

```
if (expression){
     instruction1;
     } else {
     instruction2 ;
}
```

que l'on peut traduire par « si (expression) est vrai, on exécute {instruction1;} sinon on exécute {instruction2} » .

On peut aussi imbriquer les tests les uns dans les autres avec else if

```
if (expression 1) {
instruction 1 ;
} else if (expression 2){
instruction 2 ;
} else if (expression 3){
instruction 3 ;
} else {
instruction 4 ; }
```

Les conditions

Voici quelques exemples de conditions

```
if (a==b){…}     : si (a est égal à b)
if (a>b){…}      : si (a est supérieur à b)
if (a<b){…}      : si (a est inférieur à b)
if (a>=b){…}     : si (a est supérieur ou égal à b)
if (a<=b){…}     : si (a est inférieur ou égal à b)
if (a!=b){…}     : si (a est différent de b)
```

Attention : pour tester une égalité, on utilise le symbole « == », le symbole = étant réservé à l'affectation.
`If (a==2)` permet de tester si a est égal à 2 tandis que `a=2` met la valeur 2 dans a !

Pour faire des tests plus complexe on peut faire des et entre conditions ou des ou.
Structure du ET : if (condition1 && condition2)...
Structure du OU : if(condition1||condition2)...
Comme pour le == , on utilise && ou || pour le test.

Travaux dirigés

On donne le programme suivant (qui n'est pas commenté pour motifs pédagogiques!) :

```
#include <stdlib.h>
int main()
{
    int e = 0 ;
    int r = 0 ;
    printf("Saisir un nombre :");
    scanf("%d", &e);
    if (e<0)
    {
        r = -2*e ;
    }
    else
    {
        r = 2*e ;
    }
    printf("resultat = %d ", r);
    return 0;
}
```

Q.1) Indiquer ce qui s'inscrit à l'écran lorsque l'utilisateur lance le programme et saisi « 5 ».

Q.2) Indiquer ce qui s'inscrit à l'écran lorsque l'utilisateur lance le programme et saisi «- 3 ».

Q.3) Donner une écriture mathématique de la fonction r(e).

Q.4) Re-écrire le programme sans utiliser de else (astuce vous pouvez modifier la variable e).

Branchement sur choix : switch

On utilise cette instruction lorsque un entier ou un caractère prend un nombre fini de valeurs et que chaque valeur implique une instruction différente.

Exemple :

```
switch(i) {
case 1 : instruction 1 ; /* si i=1 on exécute l'instruction 1 */
break ; /* et on sort du switch */
case 2 : instruction 2 ; /* si i=2 ... */
break ;
case 10 : instruction 3 ; /* si i=10 ... */
break ;
default : instruction 4 ; /* pour les autres valeurs de i */
break ;
}
```

Attention : ne pas oublier le break à la fin de chaque case, sinon on exécute les instructions les unes après les autres !

Travaux dirigés

On donne le programme suivant :

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

int main()
{
    int e = 0 ;
    printf("Saisir un nombre entre 1 et 3 :");
    scanf("%d", &e);
    switch(e){
        case 1 :
            {
                printf("Hun");
            }
                break;
        case 2 :
            {
                printf("De");
            }
                break;
        case 3 :
            {
                printf("Troie");
            }
                break;
    default :
        {
            printf("Patate !");
        }
                break;
    }
    return 0;
}
```

Q.1) Indiquer ce qui s'inscrit à l'écran lorsque l'utilisateur lance le programme et saisi « 1», « 2 », « 3 », « 4 », « 123 ».

Vous allez maintenant mettre en application ce que nous venons de voir...

Manipulation n°5

Q.1) Créer un projet nommé seriepara1 et générer une application qui permet de calculer au choix, la résistance équivalente à 2 résistances associées en série ou en parallèle en utilisant un branchement conditionnel (if). Un algorithme vous est proposé, il est recommandé de s'en inspirer.

```
------------------------------------------------------------
Algorithme : Calcul 2 Résistances en série ou en parallèle
------------------------------------------------------------
Variables numériques : R1, R2, Req (réels), Reponse (entier)
     DEBUT
           Afficher "Résistance en série : taper 1 en parallèle : taper 2 "
           Saisir Reponse
           Afficher "Entrer la valeur de la résistance 1 en ohms "
           Saisir R1
           Afficher "Entrer la valeur de la résistance 2 en ohms "
           Saisir R2
           Si Reponse = 1 Alors
                 Req <- R1+R2
           Sinon
                 Req <- 1/((1/R1)+(1/R2))
           Fin Si
           Afficher "La resistance équivalente est", Req, " ohms "
      FIN
```

Manipulation n°6

Q.1) Créer un projet nommé seriepara2 et générer une application qui permet de calculer au choix, la résistance équivalente à 2 résistances associées en série ou en parallèle en utilisant un branchement sur choix (switch).

Les boucles

Nous allons maintenant généraliser le calcul à plus de 2 résistances en série ou en parallèle.
On rappelle :

$$R_{EQSérie} = R_1 + R_2 + .. + R_N \qquad R_{EQParallele} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \ldots \frac{1}{R_N}}$$

Pour cela, nous allons utiliser des **boucles**. Les boucles sont des structures qui permettent de réaliser plusieurs fois la même instruction ou la même suite d'instructions.

Boucle for

Dans la boucle for, le nombre de fois où l'on exécute la boucle est défini dès le départ.
La structure est la suivante :

```
for (i=0 ; i<N ; i++){
     instructions … ;
}
```

Cela peut se traduire ainsi : Pour i partant de 0 (i=0) et tant que i<N (i<N) on incrémente i (i++)!

i++ est une instruction revenant à faire i = i+1

Travaux dirigés

On s'intéresse au programme suivant :

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

int main()
{
    int i;
    for (i=0; i <4 ; i++)
    {
        printf("Coucou %d \n", i);
    }
    return 0;
}
```

Q.1) Déterminer ce qui apparaît à l'écran lorsque l'on exécute ce programme.
Q.2) Combien de fois est exécutée la boucle ?

On modifie la ligne avec le for et on écrit : `for (i=0; i <=4 ; i++)`

Q.3) Déterminer ce qui apparaît à l'écran lorsque l'on exécute le programme modifié. Combien de fois est exécutée la boucle ?

On modifie encore cette même ligne et on écrit : `for (i=1; i <=4 ; i++)`

Q.4) Déterminer ce qui apparaît à l'écran lorsque l'on exécute ce programme. Combien de fois est exécutée la boucle ?

Boucle while

Dans la boucle while, le nombre de fois où l'on exécute la boucle n'est pas connu à l'avance. La boucle s'exécute tant que la condition est vraie.
La structure est la suivante :

```
while (expression){
     instructions … ;
}
```

Attention en utilisant cette boucle : on peut faire des boucles infinies qui ne s'arrêtent jamais !

Travaux dirigés

On s'intéresse au programme suivant :

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

int main()
{
    int S = 0 ;
    while (S !=5){
          S = S+ 1 ;
          printf("S = %d \n", S);
    }
    return 0;
}
```

Q.1) Déterminer ce qui apparaît à l'écran lorsque l'on exécute ce programme.

Q.2) Quelle est la condition de sortie de la boucle ?

On modifie la ligne `int S = 0 ;` que l'on remplace par `int S=5 ;`

Q.3) Déterminer ce qui apparaît à l'écran lorsque l'on exécute ce programme. Combien de fois est exécutée la boucle ?

On revient à l'état initial : `int S=0 ;`
Mais, on modifie la ligne `S = S+1 ;` que l'on remplace par `S = S+2 ;`

Q.4) Déterminer ce qui apparaît à l'écran lorsque l'on exécute ce programme. Conclusion ?

Q.5) Ecrire avec une boucle while le programme étudié sur l'exercice de la boucle for (Coucou 1, ..Coucou 4)

Boucle do..while

A la différence d'une boucle while, les instructions sont exécutées au moins une fois. On exécute les instructions dans la boucle puis on évalue la condition.

```
do {
     instructions … ;
}while (expression)
```

Vous allez maintenant mettre en application ce que nous venons de voir...

On rappelle :

$$R_{EQSérie} = R_1 + R_2 + .. + R_N \qquad R_{EQParallele} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \dots \frac{1}{R_N}}$$

Manipulation n°7

Q.1) Créer un projet nommé seriepara3 et générer une application qui permet de calculer au choix, la résistance équivalente à N résistances associées en série ou en parallèle en utilisant **une boucle for**.

Conseil : pour les résistance en série, à chaque itération, on calcule la somme des résistance. Pour les résistances en parallèle, à chaque itération, on calcule la somme des **inverses** des résistances. La résistance équivalente est évaluée à la fin, une fois que toutes les résistances ont été prises en compte ! Un algorithme est proposé en annexe.

Bonus

Si vous avez terminé en avance, vous pouvez travailler sur les parties suivantes :

Manipulation n°8

Q.1) Créer un projet nommé seriepara4 et générer une application qui permet de calculer au choix, la résistance équivalente à N résistances associées en série ou en parallèle en utilisant une boucle while. On arrêtera de rentrer les valeurs des résistances dès que une valeur nulle sera saisie.

Si vous avez terminé en avance, améliorez la présentation de votre programme en rajoutant des graphiques style « ASCII Art » (voir ce nom sur Google) en rapport avec l'application calcul de résistances.

Annexes

Affichage des variables, principaux formats existant

Code format	
%c	Caractère pour une variable de type char
%d	Format décimal pour une variable de type int (entier) ou char
%x	Format hexadécimal pour une variable de type int (entier) ou char
%f	Format réel pour pour une variable de type float ou double
%s	Format chaîne de caractères pour une variable de type char *

Compléments de format :
Exemple 123
avec le format décimal %5d : décimal avec 5 caractères affiche « --123 » avec des espace à la place des -
avec le format décimal %05d , on affichera « 00123 »
avec le format %-5d, on affichera « 123-- » avec des espace à la place des -

Taille des variables (attention cela peut dépendre du compilateur)

Type	**Taille**	**Intervalle**
unsigned char	1 octet	$0 \leq X \leq 255$
char	1 octet	$-128 \leq X \leq 127$
short int	2 octets	$-32768 \leq X \leq 32767$
unsigned int	4 octets	$0 \leq X \leq 2^{32} - 1$
int	4 octets	$-2^{31} \leq X \leq 2^{31}-1$
float	4 octets	$3{,}4.10^{-38} \leq \lvert X \rvert \leq 1{,}7.10^{38}$
double	8 octets	$1{,}710^{-308} \leq \lvert X \rvert \leq 3{,}4{,}10^{308}$

Annexe 2 : algorithme pour TP n°7

```
--------------------------------------------------------------------------------
Algorithme : Calcul N Résistances en serie ou parallèle avec boucle for
--------------------------------------------------------------------------------
Variables numériques :  Reponse, Nres, i : entiers
                        R, Somme, Req : reels
     DEBUT
            Somme ← 0
            Afficher "Association de R en serie ou en parallèle "
            Afficher "Resistance en serie : taper 1 en parallèle : taper 2 "
            Saisir Reponse
            Afficher "Entrer le nombre de résistances :"
            Saisir Nres
            Pour i de 1 à Nres Faire
                  Afficher "Entrer la valeur de R" ;i ; " en ohms : "
                  Saisir R
                        Si Reponse = 1 Alors
                              Somme ← Somme + R
                        Sinon
                              Somme ← Somme + 1/R
                        Fin Si
            Fin Pour
            Si Reponse = 1 Alors
                  Req   ← Somme
                  Afficher  "R  equivalente  aux",Nres,"  en  serie  est",  Req,  "
ohms "
            Sinon
                  Req ← 1/ Somme
                  Afficher "R equivalente aux",Nres," en parallele est", Req, "
ohms "
            Fin Si
      FIN
```

Retrouvez d'autres cours et documents sur :
http://www.louisreynier.com