Travaux pratiques n°3 (2h): Les itératives

Objectifs

- Programmer des itératives en langage C

Partie 1: Introduction pratique au langage C

Les itératives en C:

Le cours présente comment utiliser en C les itératives. La syntaxe générale des trois types d'itérative est la suivante:

1) Le while

1. Cas général:

```
while (expression)
  {
  instruction;
  ...
  }
/* fin du while */
```

Tant que l'expression est vraie (!=0), on effectue les instructions du bloc. L'expression est évaluée avant d'effectuer les instructions du bloc (si elle est fausse dès le début, les instructions ne sont jamais effectuée).

2. Cas particulier

```
while (expression) instruction; /* une seule instruction à effectuer */
/* fin du while */
```

exemple : Un programme qui affiche toutes les puissances de 2, jusqu'à une valeur maximale donnée par l'utilisateur.

```
#include <stdio.h>
void main(void)
 {
  int puissance=1,max;
  printf("nombre maximal désiré (ne pas dépasser 16000) ?");
  scanf("%d",&max);
  while (puissance<max) printf("%d\n",puissance*=2);
/* fin du while */
}
```

2) Le do ...while:

1. Cas général

```
do
 {
  instruction;
 ...
 }
while (expression);     /* attention au ; final /
/* fin do ... while */
```

comme while, mais les instructions sont effectuées au moins une fois, avant la première évaluation de l'expression.

2. Cas particulier

```
do instruction; while (expression) ; /* une seule instruction à effectuer */
/* fin du do ... while */
```

exemple :

```
#include <stdio.h>
int main(void)
 {
  int a;
  do
   {
            printf("veuillez entrer le nombre 482");
            scanf("%d",&a);
   }
  while (a!=482);
  /* fin du do... while */
  printf("c'est gentil de m'avoir obéi");
 }
```

3) Le for:

1. Cas général

```
for ( expr_initiale; expr_condition; expr_incrémentation)
   {
    instruction;
    ...
   }
/* fin du for */
```

Cette boucle est surtout utilisée lorsque l'on connaît à l'avance le nombre d'itérations à effectuer. L'**expr_initiale** est effectuée une fois, en premier. Puis on teste la condition. On

effectue les instruction puis l'incrémentation tant que la condition est vraie. L'instruction et l'incrémentation peuvent ne jamais être effectuées si **expr_condition** est fausse dès le début. Cette itérative est équivalente à :

```
expr_initiale;
while (expr_condition)
 {
 instruction;
 ...
 expr_incrémentation;
 }
```

3. Cas particulier

   Une ou plusieurs des trois expressions peuvent être omises, le bloc d'instruction peut être vide.

exemple :

```
#include<stdio.h>
void main(void)
{
 int i,n;
 float note,somme=0,moyenne;
 printf("nombre de notes ? ");
 scanf("%d",&n);
 for(i=0;i<n;i++)
   {
    printf("entrez votre %dième note",i+1);
    scanf("%f",&note);
    somme+=note;
   }
/* fin du for */
  moyenne=somme/n;
  printf("moyenne calculée :%5.2f\n",moyenne);
 }
```

Précisions sur scanf:

Pour être honnête, **scanf** est difficile à utiliser correctement et pose de gros problèmes pour les programmeur débutant. Je vous incite à imprimer le man du **scanf** et à le lire en détail. Par exemple supposez que vous souhaitiez lire un entier positif avec filtrage, vous allez écrire une itérative du genre:

```
do
        {
        printf("Saisissez un nombre entier positif : ");
```

```
            scanf("%d", &valeur);
            }
        while(valeur < 0);
```

Mais ce **do ... while** boucle à l'infini dans le cas par exemple ou l'utilisateur tape un caractère au lieu d'un entier!!! La raison est le suivante: le **scanf** ne retire pas le “retour chariot” (touche entrée: code ASCII 32) du tampon de saisie dans le cas ou le format attendu n'est pas celui entré. Du coup, à la prochaine itération **scanf** lit le “retour chariot” du premier tour tout en le laissant dans le tampon etc...

Une solution est de tester le retour de **scanf** et de vider son tampon grâce à **getchar();** qui lit un caractère dans le tampon d'entrée (donc de fameux “retour chariot”).

```
#include <stdio.h>

int main(void)
{
        int valeur, count;
        do
                {
                printf("Saisissez un nombre entier positif : ");
                count = scanf("%d", &valeur);
                /* si count == 0, alors erreur, mais il reste \n (retour chariot) dans le
buffer... */
                if ( count == 0 ) getchar();
                }
        while((count < 1) || (valeur < 0));
        printf("Valeur saisie : %d\n", valeur);
        return (0);
}
```

Vous pouvez par ailleurs utiliser **getchar()** (plus sain dans son utilisation que **scanf**) quand vous voulez lire un caractère au clavier.

```
#include <stdlib.h>   /* pour utiliser le getchar */

int main(void)
{

            char c;
            c = getchar(); /* c est initialisé avec le caractère lu au clavier par getchar()
            mais attention getchar laisse aussi le retour chariot dans le buffer car il n'y lit
            qu'un caractère */

}
```

ceil et floor:

ceil et *floor* sont deux fonctions de la librairie mathématiques (il faut inclure *math.h*) qui permettent d'arrondir des valeurs réelles respectivement à l'entier supérieur et à l'entier inférieur.
Par exemple:
x=ceil(14.5); /* x reçoit 15 */
y=floor(14.5); /* x reçoit 14 */

Ainsi *ceil* était bien utile pour calculer le nombre de pots dans l'exercice sur la peinture de la pièce.
nombrePots = ceil(quantitePeinture / capacitePot) ; /* nombrePots reçoit la valeur entière immédiatement supérieure à quantitePeinture / capacitePot */

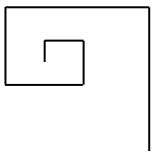
D'autres fonctions de la librairie mathématiques peuvent vous être utiles:

sin(x) donne le sinus de x
cos(x) donne le cosinus de x
tan(x) donne le tangente de x
sqrt(x) donne la racine carré de x
pow(x,y) donne x^y

Partie 2: Les itératives en langage C

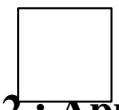
Exercice 1: La tortue ... le retour

- Au moyen de la tortue écrire un programme C traçant la spirale suivante sachant que :



1. le premier segment (au centre de la spirale) est de taille *longueurMin* (*longueurMin* > *0*)
2. la longueur de chaque segment est celle du segment précédent + *longueurMin.*
3. la longueur du dernier segment est inférieure ou égale à *longueurMax* (*longeurMax* ≥ *longueurMin*)

- Écrire un programme en C qui fera dessiner à la tortue un polygone dont chaque coté à une longueur de 10 et dont le nombre de cotés sera demandé à l'utilisateur. Le nombre de cotés devra être supérieur à 2 et inférieur à 10 (il faut filtrer les mauvaises valeurs). Ainsi si l'utilisateur donne 4 pour le nombre de coté la tortue affichera un carré de coté 10:



Exercice 2 : Application mathématiques

On se propose d étudier la suite de terme général $Un=\frac{1}{n^2}$ (n > 0) ainsi que la suite définie par $Vn=\sum_{k=1}^{n} Uk$. Pour ce faire, écrire un programme en C qui affiche, ligne par ligne, la valeur de *n*, celle de *Un*, et celle de *Vn*, tant que le terme *Un* est supérieur à un certain *seuil* (*seuil* ≤ *1*) dont la valeur sera demandée.

Exercice 3:

Écrire un programme en C qui permet d'afficher la puissance d'un circuit en fonction de *u* variant par pas de 0.5V et de *i* variant par pas de 0.05A sachant que les valeurs initiales et finales de *u* et *i* seront demandées à l'utilisateur.

Exemple d'affichage : (pour u variant de 0 à 2V et i variant de 0 à 0.15A)

```
u=0.000000 i=0.000000 p=0.000000
u=0.000000 i=0.050000 p=0.000000
u=0.000000 i=0.100000 p=0.000000
u=0.000000 i=0.150000 p=0.000000
**********************************
u=0.500000 i=0.000000 p=0.000000
u=0.500000 i=0.050000 p=0.025000
u=0.500000 i=0.100000 p=0.050000
u=0.500000 i=0.150000 p=0.075000
**********************************
…
```

Exercice 4 :

Écrire un programme en C permettant de calculer x^y avec x et y entiers positifs. Seule l'addition est autorisée.

Exercice 5 :

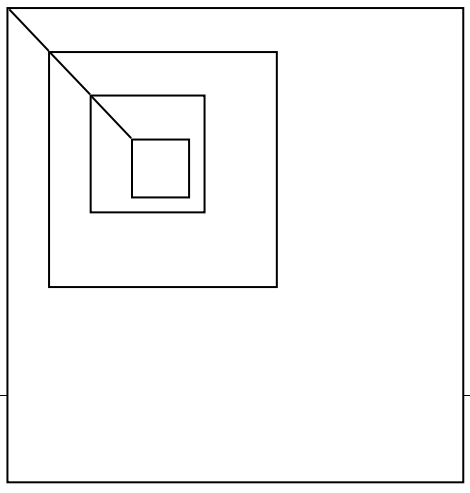
Donner un programme en C qui affiche la figure suivante en fonction de la hauteur qui sera demandée à l'utilisateur. On affiche des étoiles (*) et des traits(-).

exemple: (la hauteur est 4)

```
*---
**--
***-
****
```

Exercice 6:

Au moyen de la tortue écrire un programme C traçant la figure suivante :



Le nombre de carrés à tracer est demandé à l'utilisateur (4 dans l'exemple ci-dessus)
La longueur du coté du carré le plus petit (à l'intérieur) est demandé à l'utilisateur.
Ensuite, chaque carré à un coté double du coté du carré juste en dessous.
Enfin la diagonale qui relie chaque carré à une longueur fixe et égale au coté du plus petit carré et est une diagonale de chaque carré (donc fait un angle de 45°)