AMICHI Lounès – TP B2
29 Septembre

IAP - TP N°1

Objectifs :

- Etre capable de coder en langage C une analyse en pseudo-code basée sur des structures de contrôle de base (conditionnelle et « tant que »)
- Spécification des tests
- Débogage d’une application

Partie 1 – Transcription d’une analyse en pseudo-code en code C

Chaque exercice contient le programme, ainsi qu’une capture d’écran de ses retours pour les cas limites. Le code est aussi disponible dans l’archive.

Exercice n°1

Calculer la somme des n premiers nombres impairs. Le programme demandera à l'utilisateur de saisir la valeur de n et affichera le résultat.

```c
#include <stdio.h>

/* Remarque: ce programme pourrait être largement simplifié, il suffit de calculer le carré de n.
int main(){
  int n;
  if (n > 0) {
    printf("La somme des %d premiers entiers impairs est %d.\n", n, n*n);
  } else {
    printf("\a Erreur: n n'est pas positif (Code 1)\n");
    return 1;
  }
  return 0;
}
*/
int main(){
  int i, n, somme = 0; // valeur d'itération de la boucle, valeur max de la fonction, somme
  printf("Entrez une valeur entière positive:\n> ");
  scanf("%d", &n);
  if (n > 0) {
    for (i = 1; i < 2*n; i += 2) { // 2*n car i est incrémenté de 2 en 2.
      somme += i;
    }
    printf("La somme des %d premiers entiers impairs est %d.\n", n, somme);
  } else {
    printf("\a Erreur: n n'est pas positif (Code 1)\n");
    return 1;
  }
  return 0;
}
```

```
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice1
Entrez une valeur entière positive:
> 500
La somme des 500 premiers entiers impairs est 250000.
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice1
Entrez une valeur entière positive:
> 10
La somme des 10 premiers entiers impairs est 100.
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice1
Entrez une valeur entière positive:
> 0
 Erreur: n n'est pas positif (Code 1)
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice1
Entrez une valeur entière positive:
> -503
 Erreur: n n'est pas positif (Code 1)
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice1
Entrez une valeur entière positive:
> 4.34
La somme des 4 premiers entiers impairs est 16.
```

Exercice n°2

Recherche des nombres parfaits inférieure à une valeur limite saisie par l'utilisateur.

```c
#include <stdio.h>

int sommeDiv(int n) {
  int i, somme = 0;
  if(n > 0) {
    for (i = 1; i < n; i++) {
      if (n % i == 0) somme += i;
    }
  } else return 0;
  return somme;
}
```

```c
int main(){
  int i, n;
  printf("Entrez une valeur entière positive:\n> ");
  scanf("%d", &n);
  if (n <= 0) {
    printf("\a Erreur: n n'est pas positif (Code 1)\n");
    return 1;
    }
  else {
    if (n < 6) {
      printf("Il n'existe aucun nombre parfait inférieur ou égal à %d\n", n);
    } else {
      printf("Les nombres parfaits inférieurs ou égaux à %d sont:\n", n);
      for (i = 1; i < n+1; i++) {
        if (sommeDiv(i) == i) printf("%d\n", i);
      }
    }
  }
  return 0;
}
```

```
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice2
Entrez une valeur entière positive:
> 500
Les nombres parfaits inférieurs ou égaux à 500 sont:
6
28
496
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice2
Entrez une valeur entière positive:
> 5
Il n'existe aucun nombre parfait inférieur ou égal à 5
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice2
Entrez une valeur entière positive:
> 0
 Erreur: n n'est pas positif (Code 1)
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice2
Entrez une valeur entière positive:
> -500
 Erreur: n n'est pas positif (Code 1)
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice2
Entrez une valeur entière positive:
> 4.34
Il n'existe aucun nombre parfait inférieur ou égal à 4
```

Exercice n°3

Ecrire un programme permettant de résoudre une équation du second degré. Les paramètres de ce programme seront les coefficients a, b et c de l'équation ax2+bx+c=0. Vous envisagerez tous les cas de figure, y compris les cas ou certains coefficients sont nuls.

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

int main(){
    int a, b, c, delta;
    float roots[] = {0.0, 0.0}; // Dans R, P(X) peut avoir 0, 1 ou 2 racines.
    // scan des coefficients
    printf("Veuillez entrer les coefficients du polynôme:\n");
    printf("\na:");
    scanf("%d", &a);
    printf("\nb:");
    scanf("%d", &b);
    printf("\nc:");
    scanf("%d", &c);

    // calcul du discriminant
    delta = b * b - 4 * a * c;

    // calcul des racines
    if (delta < 0) {
        printf("Le polynôme n'a pas de racines réelles\n");
    } else if (delta == 0) {
        roots[0] = (float)-b / (2 * a); // (float) force la division par au moins un réel, pour obtenir le bon résultat
        printf("Le polynôme a une racine double x1 = x2 = %1.3f\n", roots[0]); // %1.3f arrondit à 3 chiffres après la virgule
    } else {
        roots[0] = ((float)-b - sqrt(delta)) / (2 * a);
        roots[1] = ((float)-b + sqrt(delta)) / (2 * a);
        printf("Les racines du polynôme sont:\nx1=%1.3f\nx2=%1.3f\n", roots[0], roots[1]);
    }
    return 0;
}
```

```
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice3
Veuillez entrer les coefficients du polynôme:

a:2

b:-5

c:2
Les racines du polynôme sont:
x1=0.500
x2=2.000
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice3
Veuillez entrer les coefficients du polynôme:

a:4

b:-12

c:9
Le polynôme a une racine double x1 = x2 = 1.500
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog. structurée$ ./Exercice3
Veuillez entrer les coefficients du polynôme:

a:1

b:0

c:1
Le polynôme n'a pas de racines réelles
```

Partie 2 - Débogage

Exercice 4

Quelques erreurs simples dans le code, notamment dans les fonctions printf et scanf : mauvais format (%d vs %f) ou inversion de pointeur et variable (var et &var). Il y a aussi quelques problèmes de structure (mauvaise définition des blocs, crochets manquants).

```c
// resolution d'une equation du 2e degre
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int main()
{
//déclaration des variables
int a,b,c;
float d;
//acquisition des variables
printf("\n Ax²+Bx+C=0");
printf("\n entrez A: ");
scanf("%d", &a);
printf("\n entrez B: ");
scanf("%d", &b);
printf("\n entrez C: ");
scanf("%d", &c);
//Condition invalidant la suite du prgm
if(a==0)
{
        printf("\n ce n'est plus du 2nd degré");
        return 0;
}
else
{
        d=(b*b)-(4.0*a*c);
        if(d<0)
                printf("\n 2 solutions complexes conjuguées: (-%d+i racine(%f)) /%d et (-%d-i racine(%f)) /%d",b,-d,2*a,b,-d,2*a);
        else
                if(d>0)
                        printf("\n 2 solutions réelles: (-%d+racine(%f)) /%d et (-%d-racine(%f)) /%d",b,d,2*a,b,d,2*a);
                else
                        printf("\n 1 solution double :-%d /%d",b,2*a);

}
printf("\n");
return 1;
```

Exemple du programme eq2degre.c

Note : le code de chaque programme est disponible dans l'archive.