TD1

September 4, 2022

1 TD1

1.1 Exécutions ligne à ligne

Exécution ligne à ligne de mysteryFunction(5):

[1]:
```
int syracuse (int n)
{
    int cpt = 0 ;
    while (n != 1)
    {
        if (n % 2 == 0)
            n = n/2 ;
        else
            n = 3*n+1 ;
        cpt = cpt + 1 ;
    }
    return cpt ;
}
```

[2]:
```
int mysteryFunction (int n)
{
    int result ;
    int mysteryVar = -1 ;
    int cpt ;
    for (int i = 1; i <= n; i++)
    {
        cpt = syracuse(n);
        if (cpt > mysteryVar)
        {
            result = i ;
            mysteryVar = cpt ;
        }
    }
    return result ;
}
```

Exercice: exécuter “min(12,42)”, “max(12,42)” et “sum(4)” ligne à ligne.

[4]:
```
int min (int x, int y)
{
    if (x < y)
        return y ;
    else
        return x ;
}
```

[5]:
```
int max (int y, int x)
{
    if (x > y)
        return y ;
    else
        return x ;
}
```

[6]:
```
int sum (int i)
{
    int result = 0 ;
    for (int n = 0; n <= i; n++)
    {
        result = result + n ;
    }
    return result ;
}
```

Exercice: écrivez le programme ci dessous en ajoutant les accolades optionnelles. Que retourne ce programme sur les entrées (0,0), (1,0), (0,1), (1,1) ?

[2]:
```
void danglingElse (int a, int b)
{
    if (a == 1)
        if (b == 1)
            printf("Passe dans le IF \n") ;
    else
        printf("Passe dans le ELSE \n") ;
}
```

```
input_line_8:6:5: warning: add explicit braces to avoid dangling else
[-Wdangling-else]
    else
    ^
```

Exercice: exécuter ligne à ligne les programmes suivants:

[4]:
```
int x,y,z;
x = 2;
y = 3;
```

```
z = x + y;
x = 2 * x;
```

[5]:
```
int x,y,z;
scanf("%d", &x); // L'utilisateur tape 3
x = x + 1;
y = x - 3;
z = 2;
y = x + y + z;
```

[8]:
```
// Exécuter f(1,2,4)
// et f(4,1,6)
void f(int x,int y,int z) {
    if (x + y < z){
        if(x > 3){
            x = 1;
            y = 3;
        }
        else{
            z = 1;
        }
    }
}
```

Que s'affiche sur la "sortie standard" lorsque l'on exécute g(5,7) ?

[10]:
```
void g(int y,int z) {
    for(int i = y; i < 2 * z; i = i + 1){
      printf("%d ", i + 3);
    }
}
```

Même question avec h(1,9)

[12]:
```
void h(int y, int z) {
    for(int i = y; i < 8; i = i + 1){
      printf("%d ", i + z);
    }
}
```

1.2 Exécutions de programmes récursifs

Exécuter la fonction suivante sur quelques entrées puis inférer ce qu'elle calcule:

[14]:
```
int whoamI(int n)
{
    if (n == 0)
    {
```

```
        return 0 ;
    }
    else
    {
        return (whoamI(n-1)+n) ;
     }
}
```

Exercice: exécuter la fonction suivante sur quelques valeurs. Que calcule cette fonction ?

Aide: n % 2 vaut 0 lorsque n est pair, 1 sinon.

[18]:
```
int fctMystere (int n)
{
    if (n == 0 )
        return 1 ;
    else if (n % 2 == 0)
        return (fctMystere(n/2) * fctMystere(n/2)) ;
    else
        return (2*fctMystere((n-1)/2) * fctMystere((n-1)/2)) ;
}
```

Exécuter debutFin(3)

[17]:
```
void debutFin(int nbAffichages)
{
   printf("début %d\n", nbAffichages);
   if (nbAffichages > 1)
      debutFin(nbAffichages - 1);
   printf("fin %d\n", nbAffichages);
}
```

1.3 Portée des variables

On suppose que l'on dispose de deux fonctions f et g de la forme suivante:

```
int f ( int n )
{
    [corps de f]
    return z ;
}

int g ( int m )
{
  [corps 1]
  int y = f(x) ;
  [corps 2]
}
```

On définit la fonction h comme:

```
int h (int m)
{
    [corps 1]
    int n = x ;
    [corps f]
    int y = z ;
    [corps 2]
}
```

Les fonctions g et h sont-elles “équivalentes” ?

[ ]: