

Université Mohammed V  
Ecole Nationale Supérieure de l'Informatique et de l'Analyse des Systèmes  
Département Génie Logiciel

# Correction du TP1 – Programmation en langage C

Travaux pratiques

Langage C (M1.1.2)  
Semestre 1 – 2026–2027  
Correction

1

## Correction du TP1

Types, controle, boucles et suites

Pr. Youness BOUTYOUR

youness.boutyour@um5.ac.ma

## Table des matières

|                                                               |          |
|---------------------------------------------------------------|----------|
| <b>Méthode de correction</b>                                  | <b>2</b> |
| Exercice 1 – Rectangle . . . . .                              | 2        |
| Exercice 2 – Conditions et équation du second degré . . . . . | 2        |
| Exercice 3 – Boucles, maximum et nombres premiers . . . . .   | 3        |
| Exercice 4 – Nombres parfaits . . . . .                       | 4        |
| Exercice 5 – Nombres amicaux . . . . .                        | 5        |
| Exercice 6 – Suite $U_n$ . . . . .                            | 5        |
| Exercice 7 – Suite $T_n$ . . . . .                            | 6        |
| Exercice 8 – Factorielle et exponentielle . . . . .           | 7        |
| Exercice 9 – Approximation de $\pi$ . . . . .                 | 8        |

## Méthode de correction

Chaque programme est indépendant et vérifie les données essentielles avant de les utiliser. Compiler avec `gcc -std=c11 -Wall -Wextra -Wpedantic fichier.c -lm` lorsque la bibliothèque mathématique est nécessaire. Les solutions privilégient des fonctions courtes, des bornes explicites et des tests sur les cas limites.



### Exercice 1 – Rectangle

Calcul du périmètre et de la surface à partir de deux longueurs positives.



### Correction complète

```

1 #include <stdio.h>
2
3 int main(void) {
4     double longueur, largeur;
5
6     if (scanf("%lf %lf", &longueur, &largeur) != 2 || longueur < 0 ||
7         largeur < 0) {
8         return 1;
9     }
10    printf("perimetre = %.2f\n", 2.0 * (longueur + largeur));
11    printf("surface = %.2f\n", longueur * largeur);
12    return 0;
13 }
```



### Exercice 2 – Conditions et équation du second degré

Les quatre questions sont regroupées : comparaison avec  $a/A$ , reconnaissance de la casse, conversion d'un jour et résolution robuste de  $ax^2 + bx + c = 0$ , y compris les cas dégénérés.



### Correction complète

```

1 #include <math.h>
2 #include <stdio.h>
3
4 static void equation(void) {
5     double a, b, c, delta;
6     if (scanf("%lf %lf %lf", &a, &b, &c) != 3) return;
7     if (a == 0.0) {
```

```

8         if (b == 0.0) puts(c == 0.0 ? "une infinite de solutions" : "aucune
           solution");
9         else printf("x = %.6f\n", -c / b);
10        return;
11    }
12    delta = b * b - 4.0 * a * c;
13    if (delta < 0.0) puts("aucune solution reelle");
14    else if (delta == 0.0) printf("x = %.6f\n", -b / (2.0 * a));
15    else printf("x1 = %.6f, x2 = %.6f\n", (-b - sqrt(delta)) / (2.0 * a), (-
           b + sqrt(delta)) / (2.0 * a));
16 }
17
18 int main(void) {
19     char lettre;
20     int jour;
21
22     if (scanf(" %c", &lettre) != 1) return 1;
23     printf("devinette : %s\n", lettre == 'a' || lettre == 'A' ? "correct" :
           "incorrect");
24     printf("casse : %s\n", lettre >= 'A' && lettre <= 'Z' ? "majuscule" :
           (lettre >= 'a' && lettre <= 'z' ? "minuscule" : "autre"));
25
26     if (scanf("%d", &jour) != 1) return 1;
27     if (jour >= 1 && jour <= 7) {
28         const char *jours[] = {"lundi", "mardi", "mercredi", "jeudi", "
           vendredi", "samedi", "dimanche"};
29         puts(jours[jour - 1]);
30     } else puts("jour invalide");
31     equation();
32     return 0;
33 }
    
```



### Exercice 3 – Boucles, maximum et nombres premiers

Lecture jusqu'à l'étoile, comptage des lettres b et des espaces, recherche du maximum avec sa position, puis test de primalité jusqu'à la racine carrée.



### Correction complète

```

1 #include <stdio.h>
2
3 static int est_premier(int n) {
4     if (n < 2) return 0;
5     for (int d = 2; d <= n / d; ++d) if (n % d == 0) return 0;
6     return 1;
7 }
    
```

```

8
9  int main(void) {
10     char c;
11     int compte_b = 0, espaces = 0;
12     int valeurs[5], maximum, position = 0;
13     int n;
14
15     while (scanf("%c", &c) == 1 && c != '*') {
16         putchar(c);
17         if (c == 'b') ++compte_b;
18         if (c == ' ') ++espaces;
19     }
20     printf("\nb = %d, espaces = %d\n", compte_b, espaces);
21     for (int i = 0; i < 5; ++i) scanf("%d", &valeurs[i]);
22     maximum = valeurs[0];
23     for (int i = 1; i < 5; ++i) if (valeurs[i] > maximum) maximum = valeurs[i];
24     position = i;
25     printf("maximum = %d, position = %d\n", maximum, position + 1);
26     if (scanf("%d", &n) == 1) printf("%d est %s\n", n, est_premier(n) ? "premier" : "compose");
27     return 0;
28 }
    
```



#### Exercice 4 – Nombres parfaits

La somme des diviseurs propres est calculée sans tester tous les diviseurs deux fois ; elle sert ensuite à tester  $N$  et à énumérer les nombres parfaits précédents.



#### Correction complète

```

1  #include <stdio.h>
2
3  static int somme_diviseurs_propres(int n) {
4      int somme = n > 1 ? 1 : 0;
5      for (int d = 2; d <= n / d; ++d) {
6          if (n % d == 0) {
7              somme += d;
8              if (d != n / d) somme += n / d;
9          }
10     }
11     return somme;
12 }
13
14 int main(void) {
15     int n;
    
```

```

16     if (scanf("%d", &n) != 1 || n < 1) return 1;
17     printf("%d est %s\n", n, somme_diviseurs_propres(n) == n ? "parfait" : "
        non parfait");
18     for (int x = 2; x < n; ++x) if (somme_diviseurs_propres(x) == x) printf(
        "%d ", x);
19     putchar('\n');
20     return 0;
21 }
    
```



### Exercice 5 – Nombres amicaux

Deux entiers sont comparés par les sommes croisées de leurs diviseurs propres.



### Correction complète

```

1  #include <stdio.h>
2
3  static int somme_diviseurs_propres(int n) {
4      int somme = n > 1 ? 1 : 0;
5      for (int d = 2; d <= n / d; ++d) if (n % d == 0) {
6          somme += d;
7          if (d != n / d) somme += n / d;
8      }
9      return somme;
10 }
11
12 int main(void) {
13     int n, m;
14     if (scanf("%d %d", &n, &m) != 2 || n <= 0 || m <= 0) return 1;
15     puts(somme_diviseurs_propres(n) == m && somme_diviseurs_propres(m) == n
        ? "amicaux" : "non amicaux");
16     return 0;
17 }
    
```



### Exercice 6 – Suite $U_n$

La récurrence est évaluée itérativement avec seulement les deux termes précédents, puis réutilisée pour afficher les cent premiers termes.

### Correction complète

```

1  #include <stdio.h>
2
3  static long long terme(unsigned n) {
4      long long u0 = 4, u1 = 4;
5      if (n < 2) return 4;
6      for (unsigned i = 2; i <= n; ++i) {
7          long long suivant = 3 * u1 + 4 * u0 + 6;
8          u0 = u1;
9          u1 = suivant;
10     }
11     return u1;
12 }
13
14 int main(void) {
15     unsigned n;
16     if (scanf("%u", &n) != 1) return 1;
17     printf("U_%u = %lld\n", n, terme(n));
18     for (unsigned i = 0; i < 100; ++i) printf("U_%u = %lld\n", i, terme(i));
19     return 0;
20 }
    
```



### Exercice 7 – Suite $T_n$

La relation de récurrence est calculée de manière itérative pour éviter une explosion du nombre d'appels récursifs.

### Correction complète

```

1  #include <stdio.h>
2
3  static double terme(unsigned n, double x) {
4      double t0 = 1.0, t1 = x;
5      if (n == 0) return t0;
6      if (n == 1) return t1;
7      for (unsigned k = 2; k <= n; ++k) {
8          double suivant = 2.0 * x * t1 - t0;
9          t0 = t1;
10         t1 = suivant;
11     }
12     return t1;
13 }
14
15 int main(void) {
    
```

```

16     unsigned n;
17     double x;
18     if (scanf("%u %lf", &n, &x) != 2 || x < -1.0 || x > 1.0) return 1;
19     printf("T_%u(%.6f) = %.12f\n", n, x, terme(n, x));
20     return 0;
21 }
    
```



### Exercice 8 – Factorielle et exponentielle

La factorielle, la somme partielle de  $e^x$  et l'arrêt à une précision de  $10^{-6}$  sont regroupés dans des fonctions distinctes.



### Correction complète

```

1  #include <math.h>
2  #include <stdio.h>
3
4  static double factorielle(unsigned k) {
5      double resultat = 1.0;
6      for (unsigned i = 2; i <= k; ++i) resultat *= i;
7      return resultat;
8  }
9
10 static double serie_exp(double x, unsigned n) {
11     double terme = 1.0, somme = 1.0;
12     for (unsigned k = 1; k <= n; ++k) {
13         terme *= x / k;
14         somme += terme;
15     }
16     return somme;
17 }
18
19 int main(void) {
20     unsigned k, n;
21     double x;
22     if (scanf("%u %lf %u", &k, &x, &n) != 3 || x <= 0.0) return 1;
23     printf("K! = %.0f\n", factorielle(k));
24     printf("somme = %.12f\n", serie_exp(x, n));
25     {
26         unsigned i = 0;
27         double approximation = 1.0, terme = 1.0;
28         do {
29             ++i;
30             terme *= x / i;
31             approximation += terme;
        
```

```

32     } while (fabs(terme) > 1e-6 * fabs(approximation));
33     printf("exp(%.6f) = %.12f (%u termes, erreur %.3e)\n", x,
           approximation, i, fabs(approximation - exp(x)));
34 }
35 return 0;
36 }
    
```



### Exercice 9 – Approximation de $\pi$

On utilise  $\pi = \sqrt{6 \sum_{k \geq 1} 1/k^2}$  et on arrête la somme lorsque le terme ajouté devient négligeable.



### Correction complète

```

1  #include <math.h>
2  #include <stdio.h>
3
4  int main(void) {
5      double somme = 0.0, terme;
6      unsigned k = 0;
7      do {
8          ++k;
9          terme = 1.0 / ((double) k * k);
10         somme += terme;
11     } while (terme > 1e-8);
12     printf("pi = %.12f\n", sqrt(6.0 * somme));
13     printf("termes = %u\n", k);
14     return 0;
15 }
    
```