

TP1 : LES BASES

Compte rendu – Leclerc Valentin & Le Breton Clément



Table des matières

TP1 : Les bases.....	2
Exercice 1 : Le jeu des 7 erreurs	2
Exercice 2 : Année bissextile ou non ?	3
Exercice 3 : Jeu des allumettes.....	5
Exercice 4 : Diviseur d'un nombre.....	7

TP1 : Les bases

Exercice 1 : Le jeu des 7 erreurs

Le programme ci-dessous comporte 7 erreurs, qui posent un problème soit à la compilation soit à l'exécution.

L'écrire ou le recopier dans Code::Blocks. Essayer de le compiler et indiquer quelles sont ces erreurs et leur nature (erreur à la compilation ou erreur à l'exécution).

Corriger ces erreurs pour que le programme fonctionne.

Nous avons surligné en jaune les erreurs dans le code suivant :

```
#include<stdio.h>
main()
{
    int a, somme;
    while(1)
    {
        printf("saisissez une valeur de a entre 0 et 10 : ");
        scanf("%d", a);
        printf("vous avez saisi la valeur : %f\n", a);
        if(a<0 OR a>10);
            continue;
        for(i=1 , i<=a , i++)
            somme = somme + i;
        printf("La somme des %d premiers nombres entiers est %d\n", a, somme);
    }
}
```

1. Il manque le type de la fonction main : void
2. Il manque le & devant le nom de la variable. Cela est nécessaire pour écrire la donnée dans la case mémoire de la variable a.
3. « %d » doit être utilisé puisqu'on attend un entier
4. Dans l'instruction « if », le OR doit être remplacé par le symbole || correspondant au OU logique. Des accolades doivent être utilisées pour encadrer le code conditionné.
5. Dans la boucle « for », le type de la variable i doit être déclarée et les séparateurs ne sont pas des virgules mais des points virgules.

Le code corrigé est disponible ci-dessous.

```
#include<stdio.h>
void main()
{
    int a, somme;
    while(1)
    {
        printf("saisissez une valeur de a entre 0 et 10 : ");
        scanf("%d", &a);
        printf("vous avez saisi la valeur : %d\n", a);
        if(a<0 || a>10){
            continue;
        }
        for(int i=1; i<=a; i++) {
            somme = somme + i;
        }
        printf("La somme des %d premiers nombres entiers est %d\n", a, somme);
    }
}
```

```
}  
  
saisissez une valeur de a entre 0 et 10 : 1  
vous avez saisi la valeur : 1  
La somme des 1 premiers nombres entiers est 1  
saisissez une valeur de a entre 0 et 10 : 5  
vous avez saisi la valeur : 5  
La somme des 5 premiers nombres entiers est 16  
saisissez une valeur de a entre 0 et 10 : 10  
vous avez saisi la valeur : 10  
La somme des 10 premiers nombres entiers est 71  
saisissez une valeur de a entre 0 et 10 : -10  
vous avez saisi la valeur : -10  
saisissez une valeur de a entre 0 et 10 : 20  
vous avez saisi la valeur : 20
```

Figure 1 : résultats générés par le programme ci-dessus

Exercice 2 : Année bissextile ou non ?

Ecrire un algorithme, et le programme C correspondant, permettant de savoir si une année est bissextile ou non. L'année sera entrée par l'utilisateur. Vérifier les résultats avec quelques exemples.

Remarque : Le programme devra comporter une boucle pour gérer le cas des saisies d'un nombre négatif (en redemandant la saisie dans ce cas), et tourner indéfiniment (en demandant une nouvelle saisie après chaque affichage de résultat) tant qu'on ne le quitte pas par Ctrl-c (ou fermeture de la fenêtre console).

Nous avons réalisé l'algorithme suivant pour réaliser notre code :

ALGORITHME IsAnneeBissextile

ENVIRONNEMENT

ENTREES

Annee : entier

SORTIE

IsBissextile : booléen

VARIABLES INTERNES

i : entier

TRAITEMENT

DEBUT

TANT QUE 1

SAISIR UNE ANNEE

SI L'ANNEE EST POSITIVE

SI L'ANNEE EST DIVISIBLE PAR 4 MAIS PAS PAR 100 OU DIVISIBLE PAR 400

ECRIRE IsBissextile = true

SINON

ECRIRE IsBissextile = false

SINON

RESAISIR UNE ANNEE

AFFICHER IsBissextile

FIN TANT QUE

FIN

FIN ALGORITHME

Le code ci-dessous découle de l'algorithme présenté précédemment. Le code nous permet donc de vérifier si les saisies sont positives sinon l'utilisateur doit ressaisir une année. Ensuite, nous testons si l'année est bissextile. Pour ce faire nous devons vérifier les conditions suivantes :

1. L'année doit être divisible par 4,
2. L'année ne doit pas être divisible par 100
3. Enfin, l'année doit être divisible par 400.

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

void main()
{
    int annee;
    bool isAnneeBissextile;

    while(1){
        printf("Saisir une annee : ");
        scanf("%d", &annee);

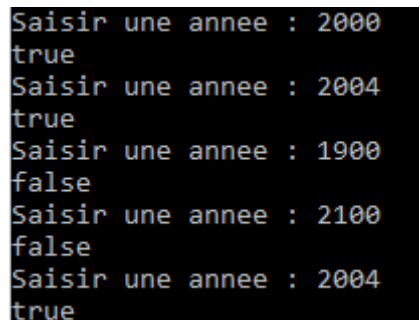
        if(annee >= 0) {
            if ((annee % 4 == 0 && annee % 100 != 0) || annee % 400 == 0) {
                isAnneeBissextile = true;
            } else {
                isAnneeBissextile = false;
            }
        } else {
            continue;
        }
        printf("%s\n", isAnneeBissextile ? "true" : "false");
    }
}
```

Dans la figure 2, nous avons saisi des années bissextiles et d'autre non. Le but du test est de vérifier que le code nous retourne bien « true » si l'année en question est bissextile et « false » si l'année ne l'est pas.

D'après l'énoncé, 2000 est une année bissextile et 1900 ou encore 2100 ne le sont pas. De plus, une année bissextile se produit tous les 4 ans.

Nous avons donc testé les années suivantes : 1900, 2000, 2004 et 2100.

Les résultats concordent avec la réalité : 2000 et 2004 sont bissextiles alors que 1900 et 2100 ne le sont pas.



```
Saisir une annee : 2000
true
Saisir une annee : 2004
true
Saisir une annee : 1900
false
Saisir une annee : 2100
false
Saisir une annee : 2004
true
```

Figure 2 : Résultats du test des années bissextiles

Exercice 3 : Jeu des allumettes

Le jeu des allumettes est une déclinaison du jeu de Nim (ou jeu des bâtonnets dans l'émission Fort Boyard). Il se joue à deux. Il y a au départ N allumettes. Chacun à son tour, les 2 joueurs retirent entre n=1 et 3 allumettes. Celui qui retire la dernière allumette a perdu.

- a) Ecrire un algorithme permettant de faire jouer alternativement l'ordinateur et le joueur. L'ordinateur jouera de façon à gagner (cf règle ci-dessus), et le programme devra demander à l'utilisateur de saisir le nombre d'allumettes qu'il veut retirer. Le programme pourra être basé sur une boucle "tant que", à chaque itération de laquelle il faudra distinguer le cas Ordinateur ou Joueur.

Nous avons réalisé l'algorithme suivant pour réaliser notre code :

ALGORITHME JeuDesAllumettes

ENVIRONNEMENT

CONSTANTES

Ordinateur : 0

Humain : 1

ENTREES

choixAllumettes : entier

nbAllumettes : entier

SORTIE

NbrAllumettesRestantes : entier

VARIABLE INTERNE

tourJoueur : entier

TRAITEMENT

DEBUT

SAISIR le nombre d'allumettes

FAIRE

SAISIE du nombre d'allumettes à retirer (NbrAllumettesRetirees)

TANT QUE le nombre d'allumettes à retirer]1 ; 3[

SAISIR un nombre d'allumettes à retirer

FIN TANT QUE

NbrAllumettes = NbrAllumettes – NbrAllumettesRetirees

SI NbrAllumettes != 0

AFFICHER au tour du joueur 2 ; saisir un nombre d'allumettes à retirer entre 1 & 3.

SAISIE du nombre d'allumettes à retirer (NbrAllumettesRetirees)

TANT QUE le nombre d'allumettes à retirer]1 ; 3[

SAISIR un nombre d'allumettes à retirer

FIN TANT QUE

NbrAllumettes=NbrAllumettes–

NbrAllumettesRetirees

AFFICHER le nombre d'allumettes restantes

SINON

AFFICHER vous avez perdu la partie

FIN SI

TANT QUE NbrAllumettes > 0

b) Ecrire le programme en langage C correspondant.

Ci-dessous, le code que nous avons réalisé pour nous permettre de jouer au jeu des allumettes.

```
#include<stdio.h>
#include<stdbool.h>
#define Ordinateur = 0;
#define Humain = 1;
int NbAllumettes = 0;
void main()
{
    int nombre_Allumettes, nombre_Utilisateur;
    int a;
    printf("\tLe jeu des allumettes\n");
    printf("Saisir le nombre d'allumettes : ");
    scanf("%d", &nombre_Allumettes);

    do
    {
        printf("\nAu tour de l'ordinateur !\nSaisir un nombre d'allumettes
entre 1 et 3 : ");
        scanf("%d", &nombre_Utilisateur);
        while (nombre_Utilisateur < 1 || nombre_Utilisateur > 3)
        {
            printf("Le chiffre saisi n'est pas compris entre 1 et
3.\nVeuillez reessayer : ");
            scanf("%d", &nombre_Utilisateur);
        }
        nombre_Allumettes = nombre_Allumettes - nombre_Utilisateur;

        printf("\nle nombre d'allumettes restante est %d\n",
nombre_Allumettes);

        if (nombre_Allumettes > 0)
        {
            printf("\nAu tour du joueur ! Saisir un nombre d'allumettes
entre 1 et 3 : ");
            scanf("%d", &nombre_Utilisateur);
            while (nombre_Utilisateur < 1 || nombre_Utilisateur > 3)
            {
                printf("Le chiffre saisi n'est pas compris entre 1 et
3.\nVeuillez reessayer : ");
                scanf("%d", &nombre_Utilisateur);
            }

            nombre_Allumettes = nombre_Allumettes - nombre_Utilisateur;

            printf("\nle nombre d'allumettes restante est %d\n",
nombre_Allumettes);
        }
        else
            printf("\nVous avez perdu la partie \n");
    } while (nombre_Allumettes > 0);
}
```

Exercice 4 : Diviseur d'un nombre

- a) Ecrire un programme C pour déterminer et afficher tous les diviseurs d'un nombre positif (qu'on pourra fixer en "dur" dans le programme).

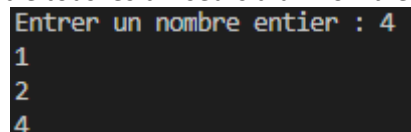
Nous avons réalisé le code suivant pour nous permettre de savoir tous les diviseurs d'un nombre positif :

```
#include<stdio.h>
main()
{
    int nombre;

    while(1){
        printf("Entrer un nombre entier : ");
        scanf("%d", &nombre);

        for (int i = 1; i<= nombre; i++) {
            if (nombre % i == 0)
                printf("%d\n", i);
        }
    }
}
```

Ce code nous a permis de connaître tous les diviseurs d'un nombre. Prenons l'exemple du chiffre 4 :



```
Entrer un nombre entier : 4
1
2
4
```

Figure 3 : Test des diviseurs du nombre 4

- b) Sachant qu'un nombre premier est un nombre qui ne peut être divisé que par 1 et par lui-même (par exemple 7), compléter ce programme pour déterminer (et afficher) si le nombre testé est premier ou non.

Nous avons modifié le code ci-dessous pour nous permettre de vérifier qu'un nombre est premier ou non. Pour ce faire, il a fallu rajouter une variable qui compte les lignes correspondantes au nombre de diviseurs de l'entier choisi. Si notre compteur est égal à 2 cela veut dire que notre nombre est premier (divisible par 1 et par lui-même).

```
#include<stdio.h>

main()
{
    int nombre, nbrPremier;
    nbrPremier = 0 ;

    while(1){
        printf("Entrer un nombre entier : ");
        scanf("%d", &nombre);

        for (int i = 1; i<= nombre; i++) {
            if (nombre % i == 0){
                printf("%d\n", i);
                nbrPremier++;
            }
        }
        if(nbrPremier == 2)
            printf("Ce nombre est premier.\n");
    }
}
```


- c) Sachant qu'un nombre parfait est égal à la somme de ses diviseurs (stricts, c'est-à-dire à l'exclusion de lui-même), compléter ce programme pour déterminer (et afficher) si ce nombre est parfait ou non.

Nous avons réalisé le code ci-dessous pour afficher un nombre parfait.

```
#include<stdio.h>

main()
{
    int nombre, nbrPremier, somme;
    nbrPremier = 0;
    somme = 0;

    while(1){
        printf("Entrer un nombre entier : ");
        scanf("%d", &nombre);

        for (int i = 1; i<= nombre; i++) {
            if (nombre % i == 0){
                printf("%d\n", i);
                nbrPremier++;
            }
            if(nombre != i) {
                somme = somme + i;
            }
        }
        if(nbrPremier == 2){
            printf("Ce nombre est premier.\n");
        }
        if(somme == nombre) {
            printf("Ce nombre est parfait.\n");
        }
    }
}
```

- d) Modifier ce programme pour déterminer quel est le 40000^e nombre premier.

```
#include<stdio.h>

main()
{
    int nombre, nbrPremier, nbrPremierTotal, somme;
    nbrPremier = 0;
    nbrPremierTotal = 0;
    nombre = 1;
    somme = 0;

    while(nbrPremier < 40000){
        for (int i = 1; i<= nombre; i++) {
            if (nombre % i == 0){
                nbrPremier++;
            }
        }
        if(nbrPremier<=2){
            nbrPremierTotal++;
            printf("%d\n",nombre);
            printf("%d\n",nbrPremierTotal);
        }
    }
    printf("%d\n",nombre);
}
```

Exercice 5 (bonus) : Lecture à partir de la console et accumulation de valeurs

Ecrire un programme permettant de saisir une valeur entière n positive, puis n valeurs positives également, au clavier (avec vérification, puis en redemandant à l'utilisateur de saisir le nombre tant que celui-ci est négatif), et calculant et affichant (à chaque saisie) :

Les valeurs minimale et maximale de ces n valeurs

Leur moyenne

Et ceci sans utiliser un tableau.

On utilisera la formule récursive de la moyenne :

$$m_i = \frac{(i-1).m_{i-1} + x_i}{i}$$

Où x_i est la i^e valeur et m_i la moyenne de i valeurs.

Aurait-on pu utiliser une méthode plus simple (toujours sans utiliser de tableau) ?

```
#include<stdio.h>

main()
{
    int nombre1, nombre2, minimum, maximum;
    float moyenne;
    maximum = 0;
    do{
        printf("Combien prenons-nous de valeurs ?\n");
        scanf("%d",&nombre1);
    }while(nombre1<0);

    for(int i=0; i<nombre1; i++){
        do{
            printf("Quelle valeur prenons-nous pour ce %d-ieme nombre ?\n",
i+1);
            scanf("%d",&nombre2);
        }while(nombre2<0);
        if(i==0){
            minimum = nombre2;
            moyenne = nombre2;
        }
        if(nombre2>maximum)
            maximum = nombre2;
        if(nombre2<minimum)
            minimum = nombre2;
        moyenne = ((nombre1-1)*moyenne+nombre2)/nombre1;
        printf("La valeur maximale pour l'instant est %d\n",maximum);
        printf("La valeur minimumimal pour l'instant est %d\n",minimum);
        printf("La valeur moyenne pour l'instant est %f\n",moyenne);
    }
}
```

Nous avons utilisé une méthode de calcul de la moyenne qui aurait pu être optimisée en faisant la somme des valeurs divisée par le nombre de valeurs. Dans ce cas nous aurions utilisé i pour récupérer le nombre de valeurs.