

Algorithme et programmation en C

Séance 1/2

Contrat pédagogique



Début de la première partie

Objectifs

- Définir les attentes des deux parties
- Définir le contenu de l'enseignement
- Etablir le cadre de travail pour un apprentissage efficace

Contenu

1. Attentes
2. Présentation de l'UE
3. Organisation de l'enseignement
4. Ressources et outils

1. Attentes

1.1. Attentes de l'enseignant

→ **Attentes directes**

- ◆ Comprendre les bases de l'algorithmique et de la programmation.
- ◆ Résoudre des problèmes de manière algorithmique.
- ◆ Mettre en œuvre des algorithmes grâce aux compétences de programmation.

→ **Attentes indirectes**

- ◆ Adopter les bonnes pratiques de programmation.
- ◆ Travail en équipe.

1.2. Attentes des apprenants



2. Présentation de l'UE

2.1. Objectif général

Développer chez l'apprenant, la capacité à concevoir, analyser et mettre en œuvre des algorithmes pour résoudre des problèmes de manière efficace.

2.2. Objectifs spécifiques

L'apprenant sera capable de :

- Comprendre les bases de l'algorithmique et de la programmation.
- Résoudre des problèmes de manière algorithmique.
- Mettre en œuvre des algorithmes grâce aux compétences de programmation.

2.3. Evaluation

Selon les cas :

- Devoir sur table : 25 %
- Projets et/ou interrogations et/ou devoirs de maison : 25 %
- Examen : 50 %.

3. Organisation de l'enseignement

3.1. Organisation

→ Séparation en groupes :

- Etudiants en IA BIGDATA et en GC => M. ANAKPA (en distanciel) ;
- Etudiants en IS et en GM => M. HOETOWOU (en distanciel) ;
- Etudiants en LT et GE => Mme AMOUZOU (en présentiel).

→ Cf Emploi du temps pour les horaires.

→ Début : Semaine prochaine.

3.2. Enseignants

- ANAKPA Manawa +228 90 95 92 98 anakpa@yahoo.fr
- HOETOWOU Yaovi +288 90 55 74 81
yaovihoetowou@gmail.com
- AMOUZOU Grâce Dorcas A. +228 92 11 31 05
grace.dorcas.amouzou@gmail.com

3.3. Éléments de discipline

- Être en salle avant l'heure.
- Pénalités de bruit.
- Être en tenue.
- Participer au cours.
- Faire les exercices, les travaux pratiques et les travaux de recherche et les exercices de maison.

3.4. Evaluation par enseignant

Selon les cas :

- Devoir sur table : 25 % (tous)
- Projets et/ou interrogations et/ou devoirs de maison : 25 % (particulier)
- Examen : 50 % (tous).

3.5. Pourquoi cette UE ?

- ➔ Résolution de problèmes : L'algorithmique permet de développer des compétences avancées en résolution de problèmes. Elle enseigne à décomposer des tâches complexes en étapes simples et à concevoir des solutions efficaces et reproductibles, ce qui est utile dans une grande variété de domaines.

3.5. Pourquoi cette UE ?

- Pensée logique et structurée : Étudier les algorithmes favorise une pensée plus logique et rigoureuse. En suivant des processus bien définis pour résoudre des problèmes, on apprend à organiser ses idées de manière systématique, à anticiper les erreurs, et à valider des solutions.

3.5. Pourquoi cette UE ?

- Optimisation des ressources : L'algorithmique permet de concevoir des solutions optimisées pour utiliser les ressources (temps, mémoire, énergie) de façon efficiente. Dans un contexte de plus en plus orienté vers la gestion durable et l'efficacité des systèmes, savoir concevoir des algorithmes optimisés est un atout considérable.

3.5. Pourquoi cette UE ?

- Polyvalence et adaptabilité : Les algorithmes sont appliqués dans une multitude de secteurs (finance, santé, ingénierie, intelligence artificielle, etc.). Maîtriser l'algorithmique rend plus polyvalent et permet d'aborder divers métiers où la manipulation des données et l'automatisation sont cruciales.

3.5. Pourquoi cette UE ?

- Fondement de la programmation : L'algorithmique est la base de la programmation. Un programme est essentiellement une implémentation d'algorithmes en code. En maîtrisant l'algorithmique, on devient un meilleur programmeur, capable de créer des logiciels plus performants et adaptés aux besoins des utilisateurs.

3.6. GE

- Systèmes embarqués
- Robotique et automatisation industrielle
- Internet des Objets (IoT)
- Intelligence artificielle et apprentissage automatique
- Énergie renouvelable et réseaux intelligents
- Télécommunications
- Cybersécurité des systèmes critiques
- Conception et simulation de circuits électroniques

3.7. GC

- Conception assistée par ordinateur (CAO)
- Bâtiments intelligents et durables
- Modélisation des informations du bâtiment
- Gestion des infrastructures et maintenance prédictive
- Simulation de structures et de matériaux
- Systèmes de gestion de la construction
- Smart cities et urbanisme
- Gestion des ressources en eau

3.8. GM

- Simulation numérique et analyse par éléments finis
- Robotique industrielle
- Conception de systèmes thermiques et fluidiques
- Impression 3D et fabrication additive
- Systèmes de propulsion et d'énergie
- Maintenance prédictive et monitoring des équipements
- Automatisation des processus de fabrication

3.9. IS

- Développement d'applications et de logiciels
- Génie des systèmes embarqués
- Ingénierie des bases de données
- Sécurité logicielle et cybersécurité
- Cloud computing et architectures distribuées
- Développement mobile et web
- Tests et validation de logiciels

3.10. IA & BIGDATA

- Analyse des données massives
- Apprentissage automatique et apprentissage profond
- Traitement du langage naturel (NLP)
- Vision par ordinateur
- Systèmes de recommandation
- Analyse prédictive et prescriptive
- Gestion et gouvernance des données

3.11. LT

- Optimisation des chaînes d'approvisionnement
- Gestion des stocks et entrepôts automatisés
- Systèmes de gestion du transport
- Véhicules autonomes et connectés
- Modélisation des flux de transport
- Systèmes de transport intelligents
- Logistique durable et gestion des ressources
- Planification et optimisation des itinéraires

Fin de la première partie



Début de la seconde partie

Séance 2/2

Introduction aux algorithmes

Objectifs

- Découvrir les éléments de base d'un algorithme
- Résoudre un problème simple avec un algorithme

Contenu

1. Concepts
2. Description d'un algorithme
3. Exemple d'algorithme

1. Concepts

Préalables :

- Ecrire la recette d'une sauce gombo ;
- Ecrire l'itinéraire entre le lieu actuel et l'aéroport international Gnassingbé Eyadema ;
- Ecrire un exemple de processus de raisonnement de Light Yagami pour une "situation" lors de l'écriture dans le Death Note ;
- Décrire le processus de recrutement des membres de l'équipage du Capitaine Luffy ;
- Imaginer un processus de recherche et de collecte des Dragon Balls.

Algorithme :

- ➔ Suite finie d'instructions à appliquer dans un ordre déterminé, à un ensemble des données pour résoudre un problème donné.
- ➔ Décomposition d'une tâche quelconque en un ensemble de tâches élémentaires exécutées en séquence suivant un enchaînement strict.
- ➔ Moyens de communication par lesquels le programmeur communique avec l'ordinateur.
- ➔ **Instruction** : opération élémentaire.
- ➔ **Exemples** :
 - ◆ Résolution d'une équation du second degré,
 - ◆ Recette de préparation de sauce gombo.

- Un algorithme doit être lisible.
- Un algorithme comporte trois phases :
 - ◆ Entrée des données,
 - ◆ Traitement des données,
 - ◆ Sortie des résultats.
- Un algorithme doit être écrit suivant la syntaxe d'un langage proche du langage naturel appelé **langage algorithmique** ou **pseudo-code**.

Exemple d'un algorithme en pseudo-code :

Algorithme SommeEntiers

Début

Variables :

a, b, somme : Entier

// Lire les deux nombres

Écrire("Entrez le premier nombre : ")

Lire(a)

Écrire("Entrez le deuxième nombre : ")

Lire(b)

// Calculer la somme

somme \leftarrow a + b

// Afficher le résultat

Écrire("La somme de ", a, " et ", b, " est : ", somme)

Fin

- « *L'algorithmique est la science des algorithmes* » - Dictionnaire l'internaute.
- L'algorithmique s'intéresse à l'art de construire des algorithmes ainsi qu'à :
 - ◆ leur validité (produit un résultat correct en un nombre fini d'étapes),
 - ◆ leur robustesse (satisfait des hypothèses de départ),
 - ◆ leur réutilisabilité,
 - ◆ leur complexité,
 - ◆ leur efficacité (respecte un temps optimal de résolution d'un problème).

- Un **programme** informatique est un ensemble d'instructions traduites dans un langage et destinées à être exécutées par un ordinateur.
- C'est le résultat de la traduction d'un algorithme dans un langage de programmation.

Exemple d'un programme en C :

```
int main() {  
    // Déclaration des variables  
    int a, b, somme;  
    // Lecture des deux nombres  
    printf("Entrez le premier nombre : ");  
    scanf("%d", &a);  
    printf("Entrez le deuxième nombre : ");  
    scanf("%d", &b);  
    // Calcul de la somme  
    somme = a + b;  
    // Affichage du résultat  
    printf("La somme de %d et %d est : %d\n", a, b, somme);  
    return 0;  
}
```


Exemple d'un programme en Java :

```
public static void main(String[] args) {  
    // Déclaration des variables  
    int a, b, somme;  
    // Création d'un objet Scanner pour lire les entrées utilisateur  
    Scanner scanner = new Scanner(System.in);  
    // Lecture des deux nombres  
    System.out.print("Entrez le premier nombre : ");  
    a = scanner.nextInt();  
    System.out.print("Entrez le deuxième nombre : ");  
    b = scanner.nextInt();  
    // Calcul de la somme  
    somme = a + b;  
    // Affichage du résultat  
    System.out.println("La somme de " + a + " et " + b + " est : " +  
    somme);  
}
```

Exemple d'un programme en Python :

Déclaration des variables et lecture des entrées

```
a = int(input("Entrez le premier nombre : "))
```

```
b = int(input("Entrez le deuxième nombre : "))
```


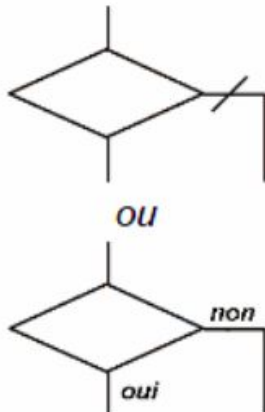
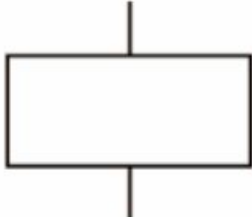
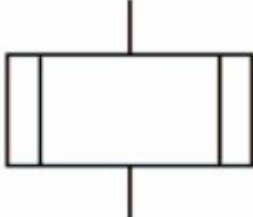
Calcul de la somme

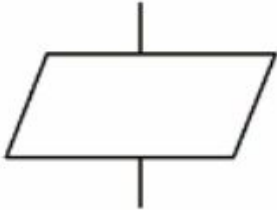


```
somme = a + b
```

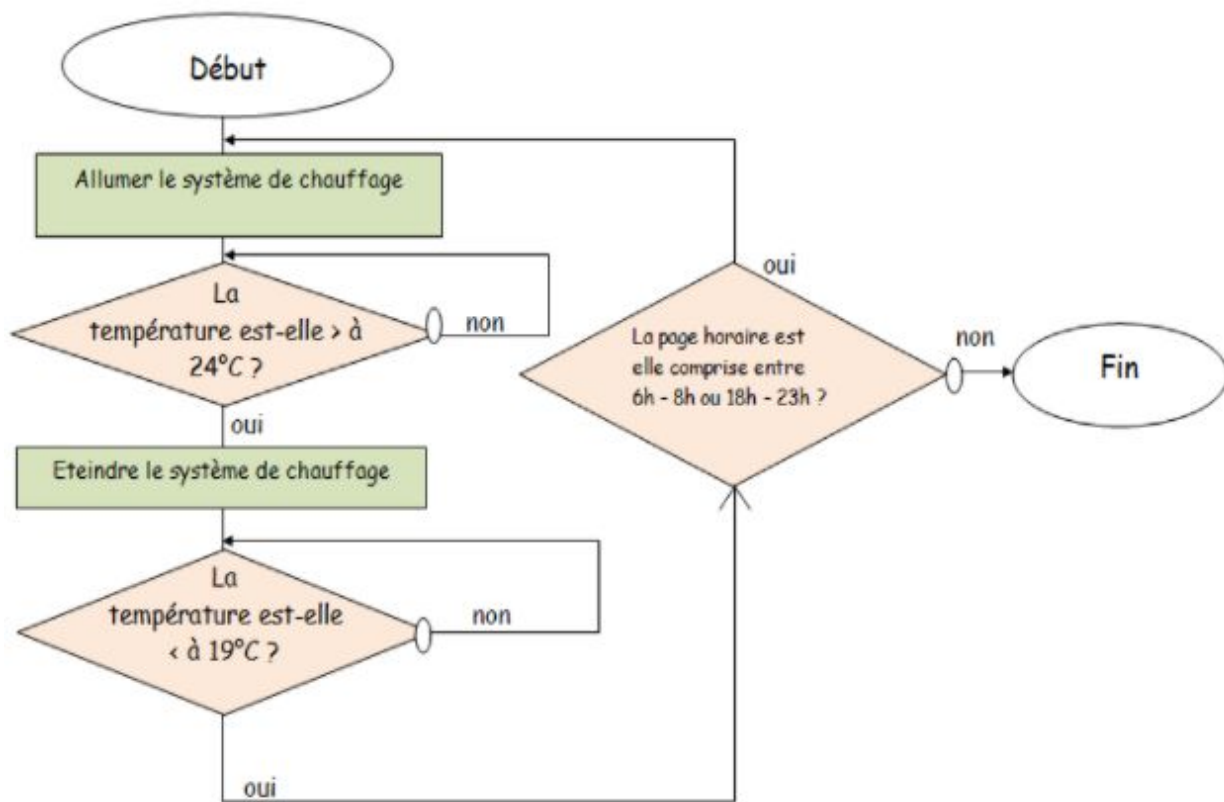
Affichage du résultat

```
print(f"La somme de {a} et {b} est : {somme}")
```

- Un algorithme est un diagramme qui représente un algorithme.
- Les différentes instructions et structures algorithmiques sont représentées par des symboles graphiques.
- L'algorithme se lit généralement :
 - ◆ du haut vers le bas,
 - ◆ de gauche à droite.
- Lorsque le sens ainsi défini n'est pas respecté, des pointes de flèches indiquent le sens utilisé.

SYMBOLE	DÉSIGNATION	SYMBOLE	DÉSIGNATION
	début ou fin d'un algorithme		<p>Test ou Branchement conditionnel</p> <p>décision d'un choix parmi d'autres en fonction des conditions</p>
	<p>symbole général de « traitement »</p> <p>opération sur des données, instructions, ... ou opération pour laquelle il n'existe aucun symbole normalisé</p>		<p>sous-programme</p> <p>appel d'un sous-programme</p>

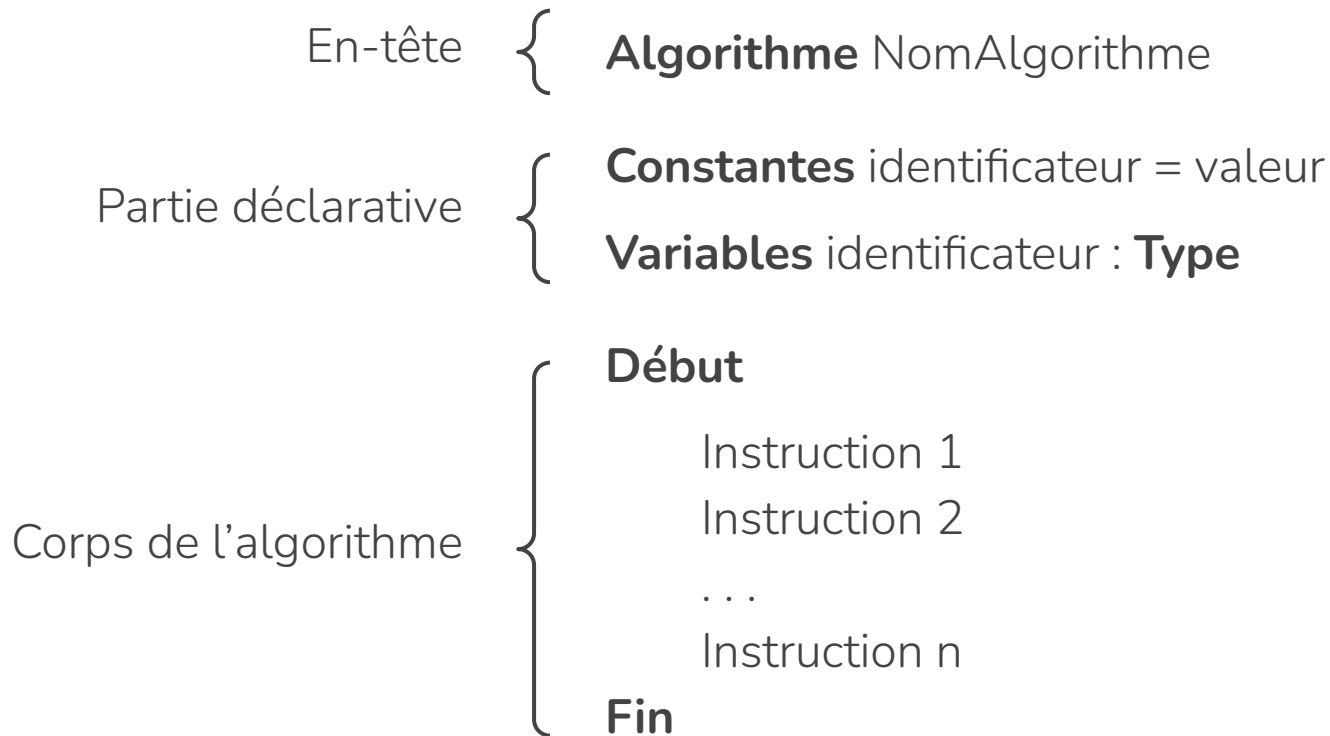
SYMBOLE	DÉSIGNATION	SYMBOLE	DÉSIGNATION
	entrée / sortie		Liaison Les différents symboles sont reliés entre eux par des lignes de liaison. Le cheminement va de haut en bas et de gauche à droite. Un cheminement différent est indiqué à l'aide d'une flèche
	commentaire		



Exercice :

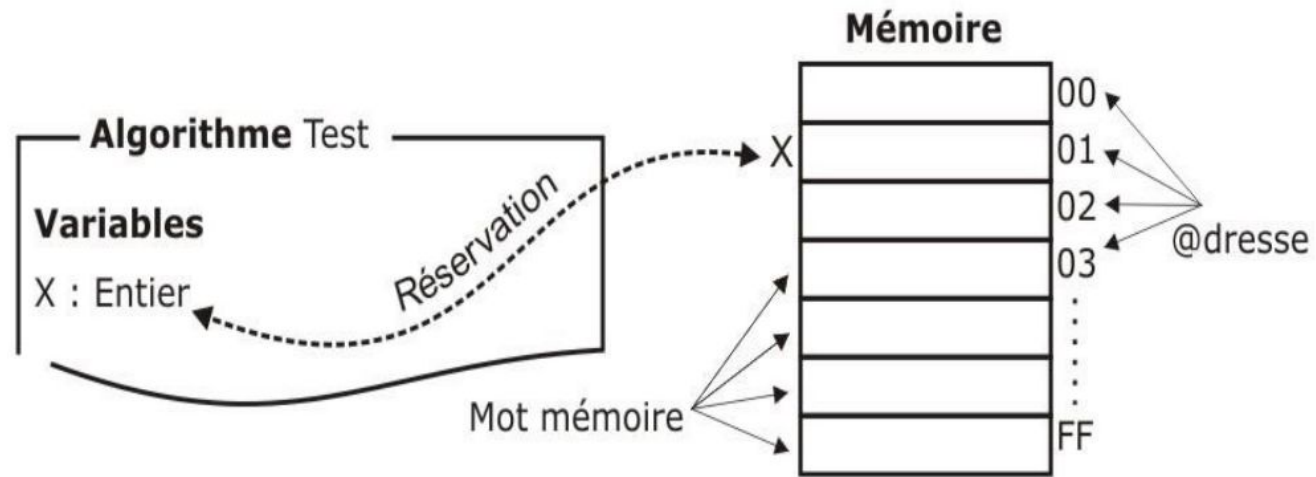
Ecrire l'algorithme de calcul de la somme (Cf page 37).

2. Description d'un algorithme



- C'est un élément faisant partie intégrante de l'exécution d'un algorithme.
- Un objet doit être décrit par sa nature, un nom (identificateur), un type et/ou une valeur.
- **Nature** : constante, variables, ...
- **Type** : l'intervalle ou l'ensemble des valeurs permises.
- **Valeur** : un nombre, un caractère, une chaîne de caractères, ...

→ Une variable est une « case » de la mémoire de l'ordinateur. On peut lui affecter différentes valeurs : des nombres, des lettres, des chaînes de caractères... Sa valeur qui peut varier au cours du temps de l'exécution d'un algorithme.



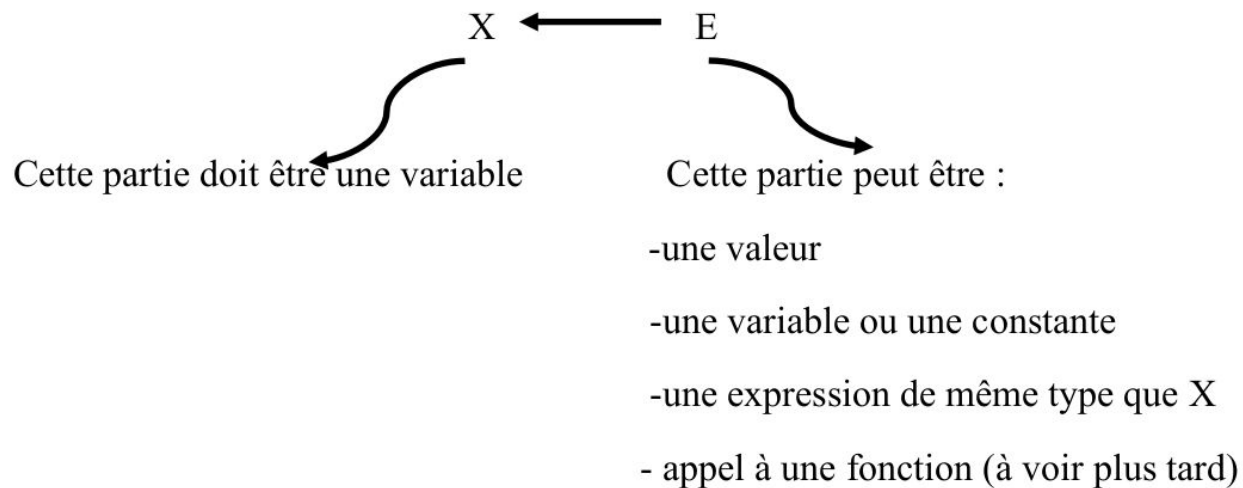
Représentation d'une variable dans la mémoire

Types de données en Algorithmique :

- ➔ Booléen : Représente un état binaire. Vrai (1) ou Faux(0)
- ➔ Reel : Représente un nombre quelconque
 - Entier : Représente un nombre entier. 12345
 - Flottant : Représente un nombre à virgule. 3,14
- ➔ Caractère : Représente un caractère unique (comme une touche du clavier). 'c'
- ➔ Chaîne de caractères : Représente un texte. "contenu de la chaîne"

- Une constante représente un nombre, un caractère, une chaîne de caractères, ... dont la valeur ne peut pas être modifiée au cours de l'exécution de l'algorithme.
- La déclaration se fait avec un identificateur et une valeur.
- Exemple : pi

- L'affectation est l'action par laquelle nous pouvons attribuer une valeur à une variable.



- La lecture permet d'introduire des valeurs par l'utilisateur via un périphérique d'entrée dans la mémoire de la machine.
- Elle est représentée comme suit : **Saisir(x1 , x2 , ... , xn)**

- L'écriture permet d'afficher un résultat ou un message à l'utilisateur.
- Elle est représentée comme suit : **Afficher(x1 , x2 , ... , xn)**
- **Exemple :**
 - Afficher('Bonjour')
 - Afficher ('Résultat = ' , x1)

3. Exemple d'algorithme

On souhaite écrire un programme pour calculer la circonférence d'un cercle. Le programme demande à l'utilisateur de saisir le diamètre du cercle. Il calcule ensuite la circonférence et affiche la valeur à l'écran.

1. Ecrire un algorithme pour résoudre ce problème.
2. Transcrire l'algorithme en un programme écrit en langage C.

Algorithme Circonference

Constantes

$\pi = 3.14$

Variables

d : réel

c : réel

Début

Afficher("Entrer le diamètre du cercle")

Saisir(d)

$c \leftarrow d * \pi$

Afficher("La circonférence est ", c)

Fin

```
5  #include <stdio.h>
6
7  int main() {
8      // Constante
9      const double pi = 3.14;
10
11     // Variables
12     double d, c;
13
14     // Début
15     printf("Entrer le diamètre du cercle : ");
16     scanf("%f", d);
17
18     c = d * pi;
19
20     printf("La circonférence est %f", c);
21
22     // Fin
23     return 0;
24 }
```

Exercices :

- Ecrire un algorithme qui calcule la moyenne de 5 notes d'un étudiant ;
- Ecrire un algorithme qui résout une équation du second degré ;
- Ecrire un algorithme qui vérifie qu'un nombre est pair ou impair.

- [1] Y. A.. GBEDEVI, « Initiation à l'algorithmique », Université de Lomé, Support de cours, 2022-2023.
- [2] R. Christophe, « Bases d'algorithmique. Support de Cours au Lycée Vincent d'Indy ». 2015-2016.
- [3] L, Baba Ahmed et S, Hocine, algorithmique et structure de données statistiques. OPU, 2016.
- [4] E. Thiel, « Support de cours Algorithmes et programmation en Pascal ». Faculté des Sciences de Luminy, Université d'Aix-Marseille AMU, 1997.
- [5] A. Rabia, F. Rachid, A. O. Mohand, B. Moufida, et Y. Smain, « Algorithmique :Cours et Exercices en Programmation Pascal ». Cours, Exercices et Programmation Pascal Première Année Universitaire, USTHB, 2018-2017