Introduction à l'informatique et aux algorithmes

Découvrez les fondements de l'informatique et le rôle essentiel des algorithmes dans notre monde numérique.

Université Abou Bakr Belkaïd - Tlemcen
Faculté des Sciences - Département de Mathématiques
abourachanez@gmail.com

Qu'est-ce que l'informatique ?

Plus que de simples ordinateurs

L'informatique est la science du traitement automatique de l'information. Elle dépasse largement la simple utilisation d'outils numériques (ordinateur, smartphone) ou un logiciel (Word, Excel, réseaux sociaux…) .

Ces usages ne sont que la partie visible et pratique de l'informatique, celle que tout le monde peut manipuler au quotidien.

L'informatique, comme science, va plus loin:

- elle conçoit les outils numériques (programmes, applications, systèmes) ;
- elle élabore des algorithmes pour résoudre des problèmes complexes qui seraient insurmontables manuellement;
- elle organise et traite l'information (bases de données, réseaux, sécurité) ;
- elle s'appuie sur des concepts théoriques (logique, mathématiques, intelligence artificielle, etc.).

L'informatique façonne notre quotidien, des applications mobiles aux systèmes de navigation spatiale, en passant par la recherche scientifique.

Définition d'un algorithme

> "Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème ou d'obtenir un résultat."

Méthode précise et systématique

Il s'agit d'une série d'étapes claires et ordonnées, décrivant la démarche à suivre pour résoudre une famille de problèmes équivalents.

☞ Le mot "algorithme" vient du mathématicien et astronome Muhammad Ibn Al-Khawarizmi, le père de l'algèbre, qui formalisa au IXe siècle la notion d'algorithme.

☞ L'algorithmique est la science qui étudie les règles et les techniques utilisées pour définir et concevoir des algorithmes.

Algorithmes hors-informatique

- Recette de cuisine : Étapes pour préparer un plat.
- Itinéraire GPS : Instructions pour aller d'un point A à un point B.
- Addition posée : La méthode enseignée à l'école pour additionner des nombres.

Exemple d'algorithme: Une recette de Spaghetti

- Porter de l’eau à ébullition.
- Ajouter les pâtes et les laisser cuire 8 à 10 minutes.
- Égoutter les pâtes.
- Servir avec la sauce et ajouter la garniture.

Chaque étape doit être simple et mener au résultat attendu : des spaghettis parfaitement cuits !

Pourquoi les algorithmes sont-ils essentiels en informatique ?

Les algorithmes ne sont pas de simples concepts théoriques; ils sont le moteur de l'innovation et de l'efficacité dans le monde numérique.

Efficacité

Permettre aux ordinateurs d'effectuer des tâches complexes à une vitesse optimale et l'utilisation efficace des resources.

Automatisation

Automatiser la résolution de problèmes complexes, souvent impossibles ou irréalisables manuellement.

L’intérêt de la pensée algorithmique

Cœur des applications modernes

Ils sont omniprésents, des moteurs de recherche aux applications mobiles, en passant par l'intelligence artificielle.

Développer la rigueur

Apprendre les algorithmes, c'est acquérir une méthode de pensée structurée et rigoureuse pour résoudre tout type de problème.

Compétence clé

C'est une compétence fondamentale et très recherchée dans tous les métiers du numérique et pour la recherche scientifique.

Propriétés d’un bon algorithme

Clair

Chaque étape doit être définie avec précision, sans laisser de place à l'ambiguïté.

Correct

Il doit toujours produire le bon résultat pour toutes les entrées valides.

Fini

Il doit s’arrêter après un nombre fini d'étapes, quelle que soit la situation.

Efficace

Il utilise le moins de ressources possible (temps de calcul, mémoire).

Compréhensible

Il doit être facile à comprendre, à implémenter et à maintenir.

Indépendant du langage

Il peut être écrit et compris sans dépendre d’un langage de programmation particulier.

Les composants d’un algorithme

Entrées

Les données initiales que l'algorithme reçoit pour commencer son traitement.

Instructions

La suite d’opérations précises (calculs, décisions, manipulations) à effectuer, réalisée pas à pas après avoir décomposé le problème en sous-problèmes si nécessaire.

Sorties

Le ou les résultats produits par l'algorithme après l'exécution de toutes les instructions.

Structure générale d'un algorithme

La conception d'un algorithme repose sur une structure logique fondamentale, essentielle pour sa clarté, sa cohérence et son exécution.

1. L'En-tête

Identifie l'algorithme par un nom et peut inclure un résumé de son objectif.

2. Les Déclarations

Section où sont définis tous les éléments nécessaires à l'algorithme : variables, constantes, et types de données.

3. Le Corps de l'Algorithme

Il contient la séquence des instructions à exécuter, délimitée par les mots-clés **Début** et **Fin**, et définit les outils nécessaires pour exprimer l'algorithme en précisant l'enchaînement chronologique des actions.

- **Début:** Marque le point de départ de l'exécution.
- **Instructions**: Les opérations logiques et calculs nécessaires à la résolution du problème.
- **Fin:** Indique la terminaison de l'algorithme.

Exemple: Addition de deux nombres

1. **En-tête**: Nom de l'algorithme.
2. **Déclarations:** déclarer les éléments utilisés dans l'algorithme.
3. Indiquer le **début** de l'exécution de l'algorithme.
4. Lire les deux valeurs d'entrées nécessaires.
5. Calculer la somme.
6. Présenter à l'utilisateur la somme en tant que résultat.
7. Marquer la **fin** de l'algorithme.

Algorithme Addition

Variables a, b, S : réels

Début

Lire (a, b)

S← a + b

Afficher (S)

Fin.

Structures de contrôle fondamentales

Ces trois structures sont les piliers de la construction de tout algorithme, permettant de gérer la logique et le déroulement des opérations.

1

Séquence

Les instructions sont exécutées l'une après l'autre, dans l'ordre où elles sont écrites.

2

Conditionnelle

Exécution d'un bloc d'instructions seulement si une condition est vraie (Si... Alors... Sinon).

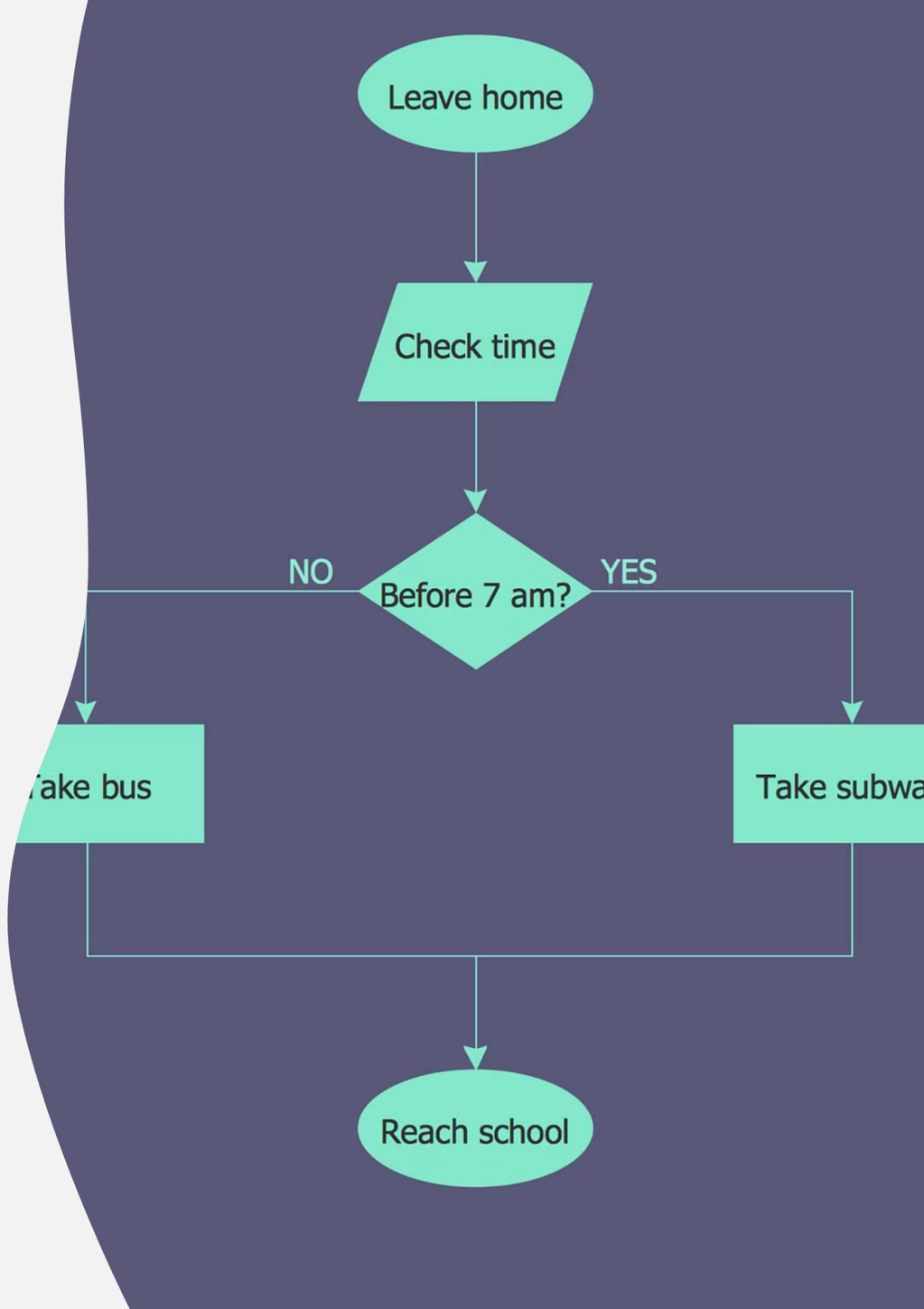
3

Boucle

Répétition d'un bloc d'instructions tant qu'une condition est vraie ou pour un nombre défini de fois (Pour, Tant que).

Qu'est-ce qu'un organigramme ?

Un **organigramme** (ou **diagramme de flux**) est une représentation graphique standardisée d'un algorithme.Il utilise des **symboles normalisés** (ovales, rectangles, parallélogrammes, losanges, flèches, etc.) pour montrer, de façon claire et structurée, l'enchaînement des étapes à suivre afin d'atteindre un objectif.



Pourquoi les organigrammes sont-ils indispensables ?

Clarté et Compréhension

Un organigramme simplifie la compréhension d'algorithmes complexes, même pour les débutants, en décomposant le problème en étapes visuelles.

Détection des Erreurs

Il aide à identifier les erreurs logiques, les oublis ou les inefficacités avant même de commencer à coder, économisant ainsi du temps et des efforts.

Support Pédagogique

Excellent outil d'enseignement et de collaboration, il facilite l'explication et la discussion autour de la logique de programmation.

L' importance de la conception des organigrammes

Pour les débutants, les organigrammes sont une aide précieuse pour :

- Apprendre la logique de programmation.
- Visualiser le déroulement d'un programme.
- Éviter les erreurs courantes.

Pour les professionnels, ils sont utiles pour :

- Documenter des systèmes complexes.
- Collaborer sur des projets.
- Réviser et optimiser des algorithmes existants.

Les principaux symboles d’un organigramme

Les organigrammes utilisent un ensemble de symboles standardisés pour représenter différents types d'opérations et de flux logiques. Voici les plus courants :

Début/Fin (Terminator)

Représente le point de départ et d'arrivée de l'algorithme.

Processus (Process)

Indique une étape de traitement ou une action spécifique (calcul, affectation, etc.).

Entrée/Sortie (Input/Output)

Utilisé pour les opérations de lecture de données (entrée) ou d'affichage de résultats (sortie).

Décision (Decision)

Symbolise une **Décision** ou un **Test** (condition Oui / Non), avec des chemins différents selon la réponse.

Flux (Flowline)

Les flèches montrent le sens dans lequel les étapes de l’organigramme s’enchaînent.

Connecteur (Connector)

Utilisé pour relier des parties d'un organigramme, souvent lorsque le diagramme s'étend sur plusieurs pages.

Exemple d'organigramme: Addition de deux nombres

Algorithme Addition

Variables a, b, S : réels

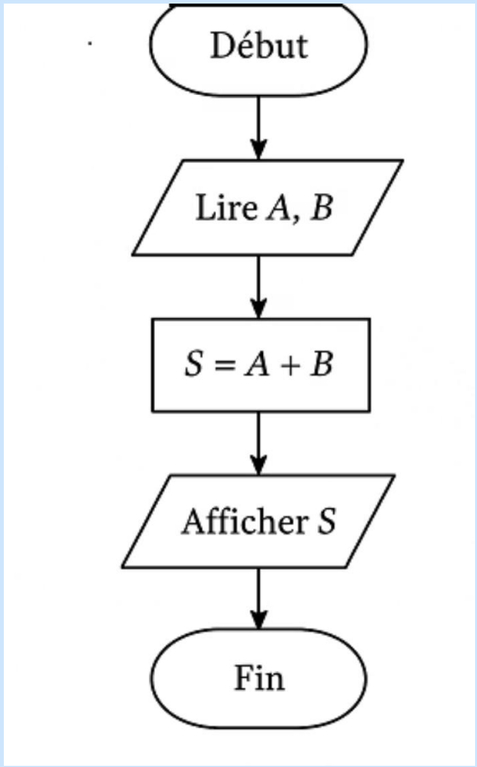
Début

Lire (a, b)

S← a + b

Afficher (S)

Fin.



Programme et langages de programmation

Un ordinateur ne possède aucune intelligence propre ou capacité d'initiative. Il est une machine qui exécute fidèlement les instructions qu'on lui donne. Sans algorithme, l’ordinateur reste inactif.

Un programme informatique est la traduction précise d’un algorithme dans un langage de programmation. Ce langage est ensuite traduit en langage machine (binaire : 0 et 1) afin d’être compris et exécuté par l’ordinateur.

De quoi a-t-on besoin pour programmer ?

Les programmeurs ont besoin de trois outils :

Éditeur de texte

Écrire le code source.

Compilateur

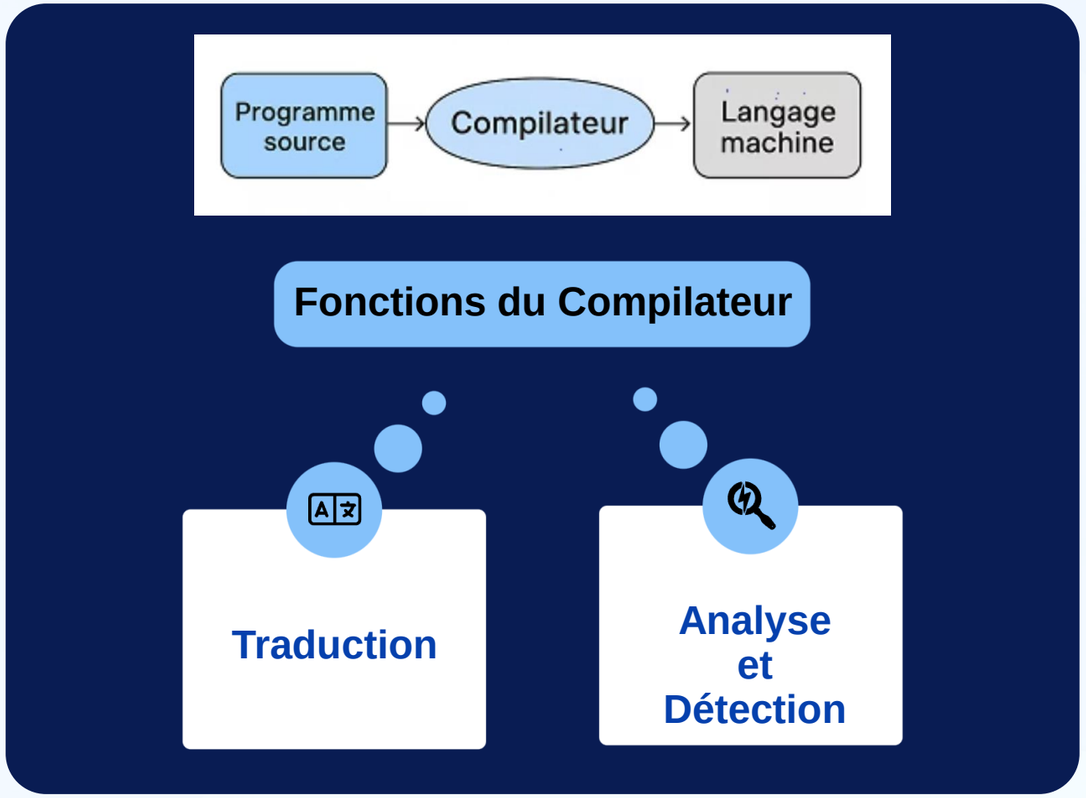
Traduire le code source en langage machine exécutable.

Débogueur

Détecter et analyser les erreurs d'execution et logiques, afin de les corriger.

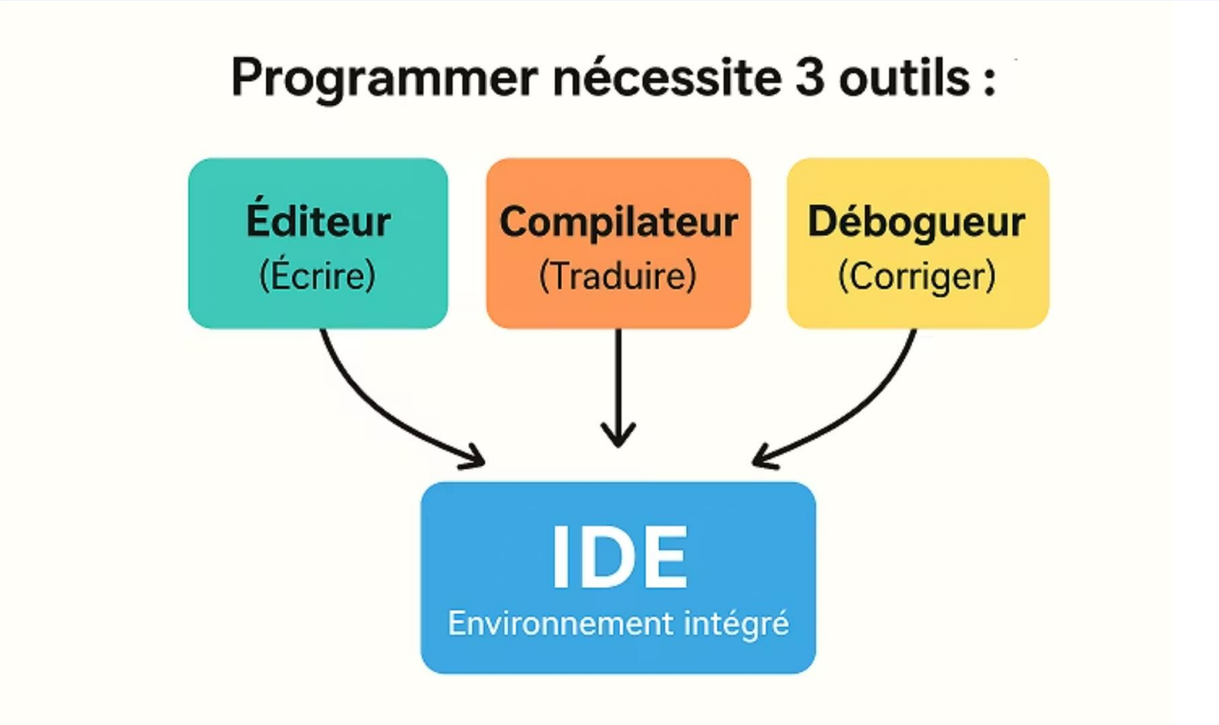
Tout langage possède un compilateur ou au moins un interpréteur. Il a pour rôle de traduire le programme écrit en langage humain, appelé *programme source*, en *langage machine* (ou *code objet*) compréhensible par l'ordinateur.

Le compilateur assure également une analyse du programme source afin de détecter les éventuelles erreurs de syntaxe commises par le programmeur.



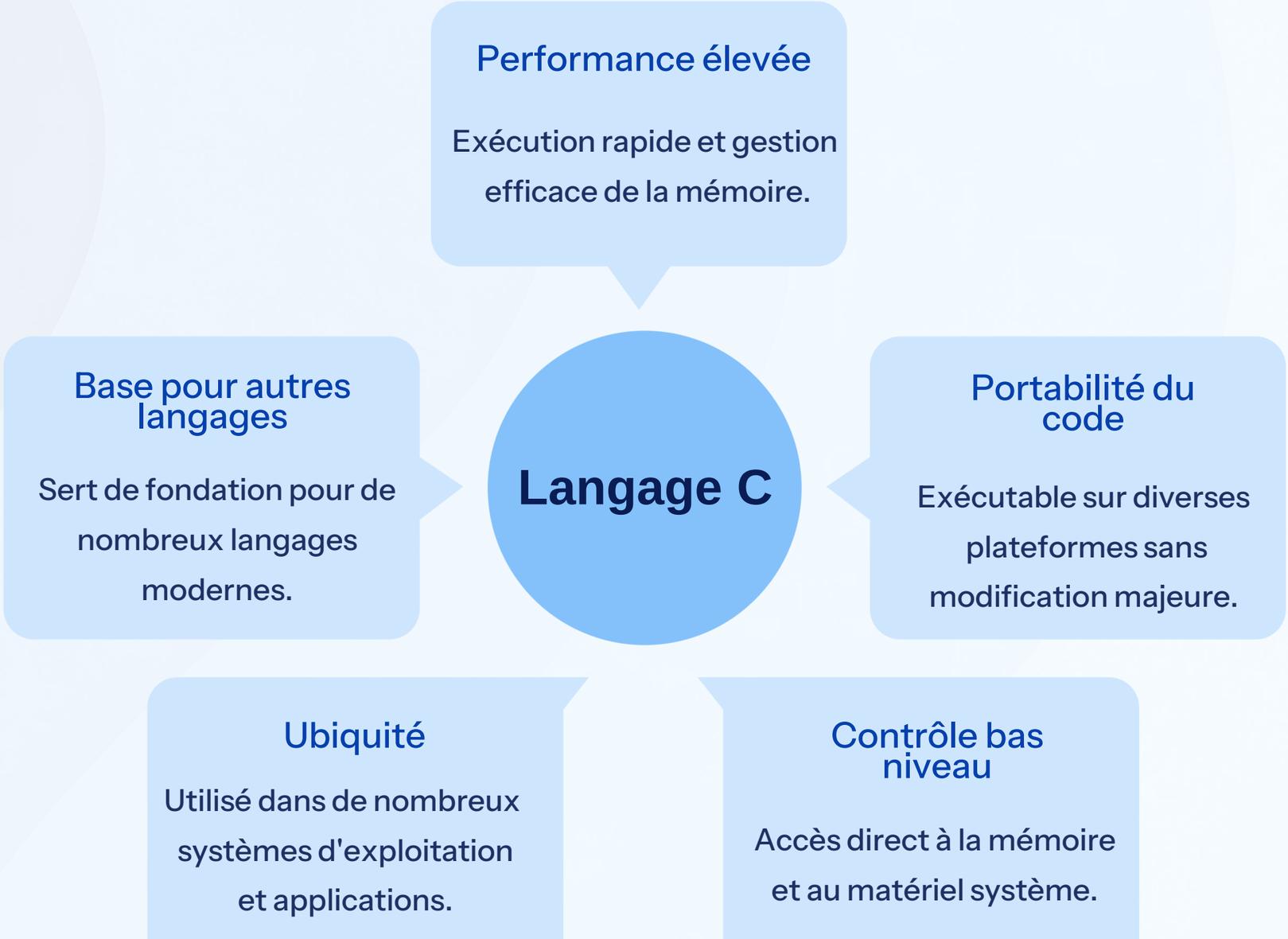
☞ Il est possible d'installer ces outils séparément, mais aujourd'hui on les retrouve souvent regroupés dans un **IDE** (*Environnement de Développement Intégré*), qui offre un package complet et pratique pour le développement.

☞ Les **IDE** les plus célèbres pour programmer en C sont :
Code::Blocks, Dev-C++, Eclipse CDT, Visual Studio, CLion et Xcode.

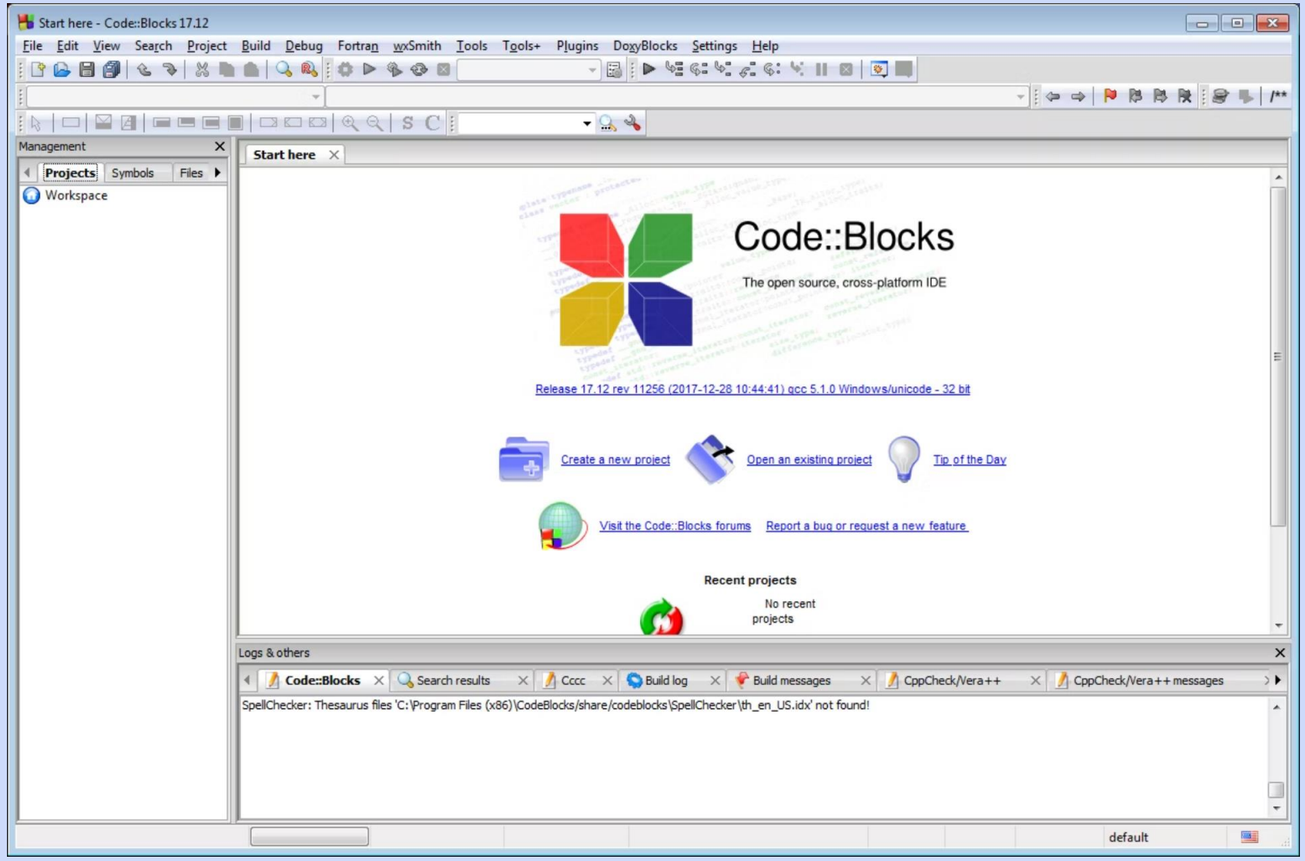


Pourquoi programmer en C ?

Il existe de nombreux langages de plus ou moins haut niveau en informatique tels que le C, C++, Java, Visual Basic, Delphi, etc.



L'IDE le plus célèbre pour programmer en C



Exemple de programme en C

```c
#include <stdio.h>

#include <stdlib.h>

int main ( )

{

    printf("Hello world! \n");

    return 0;

}
```

Phases de résolution de problème par ordinateur

La résolution d'un problème à l'aide d'un ordinateur est un parcours structuré qui transforme une idée initiale en une solution concrète.

Problème

Le point de départ : un besoin, une question à résoudre ou une tâche à automatiser.

Analyse

Décortiquer l'énoncé, identifier clairement les entrées, les sorties attendues et les contraintes. Ensuite proposer une méthode ou une stratégie de résolution.

Algorithme

L'élaboration de la séquence logique des étapes nécessaires pour traiter le problème.

Programme

La traduction de l'algorithme en un langage de programmation compréhensible par la machine (par ex. Python, Java, C++...).

Exécution

L'ordinateur interprète et exécute le programme, effectuant les calculs et les opérations.

Résultats

L'obtention des données traitées, des solutions ou des actions produites par le programme.

Conclusion:
Le coeur de l'informatique

Les algorithmes sont essentiels à toutes les applications en informatique. Qu'il s'agisse d'une simple calculatrice ou d'un système d'intelligence artificielle très avancé, il y a toujours un algorithme qui dirige les opérations en arrière-plan.

Apprendre les algorithmes permet de mieux comprendre comment fonctionne la technologie et aide aussi à trouver des solutions plus efficaces à des problèmes complexes, quel que soit le domaine.

Avec l'évolution rapide de la technologie, les algorithmes resteront au centre et continueront à façonner l'avenir de nos mondes numérique et réel.