COURS PI

L'école sur-mesure

de la Maternelle au Bac, Établissement d'enseignement privé à distance, déclaré auprès du Rectorat de Paris

Première - Module 4 - L'algorithmique et les projets numériques

Sciences de l'Ingénieur

v.5.1

- ***Guide de méthodologie*** *pour appréhender notre pédagogie*
- ***Leçons détaillées*** *pour apprendre les notions en jeu*
- ***Exemples et illustrations*** *pour comprendre par soi-même*
- ***Prolongement numérique*** *pour être acteur et aller + loin*
- ***Exercices d'application*** *pour s'entraîner encore et encore*
- ***Corrigés des exercices*** *pour vérifier ses acquis*

www.cours-pi.com

Paris & Montpellier

EN ROUTE VERS LE BACCALAURÉAT

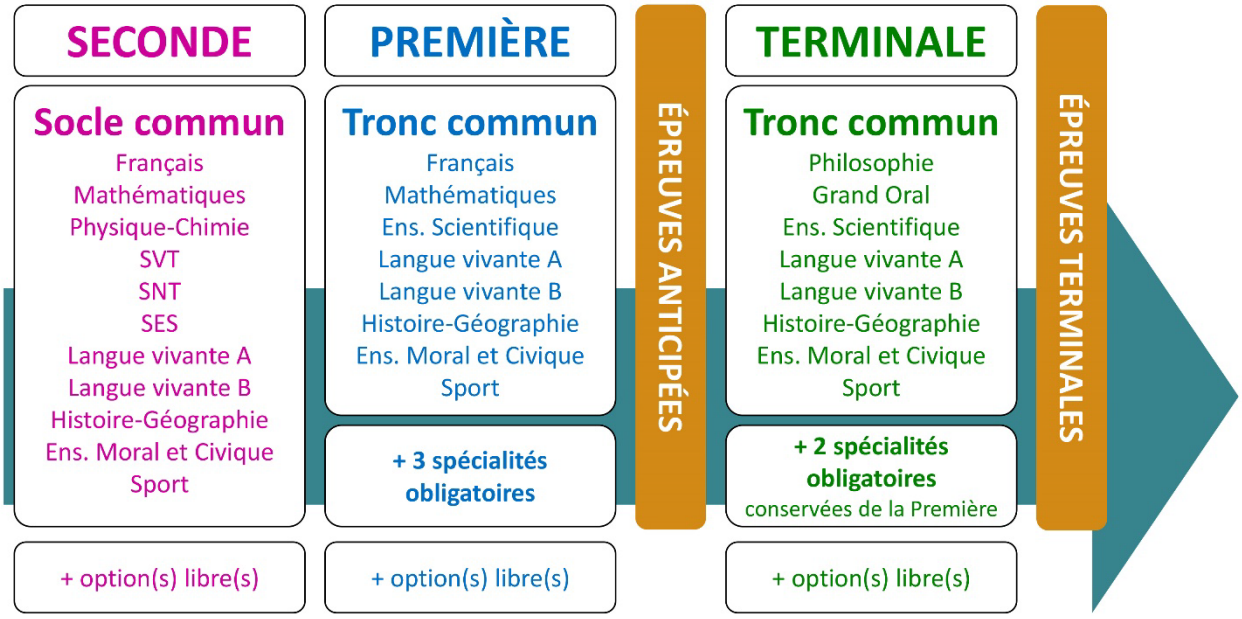
Comme vous le savez, **la réforme du Baccalauréat** est entrée en vigueur progressivement jusqu'à l'année 2021, date de délivrance des premiers diplômes de la nouvelle formule.

Dans le cadre de ce nouveau Baccalauréat, **notre Etablissement**, toujours attentif aux conséquences des réformes pour les élèves, s'est emparé de la question avec force **énergie** et **conviction** pendant plusieurs mois, animé par le souci constant de la réussite de nos lycéens dans leurs apprentissages d'une part, et par la **pérennité** de leur parcours d'autre part. Notre Etablissement a questionné la réforme, mobilisé l'ensemble de son atelier pédagogique, et déployé tout **son savoir-faire** afin de vous proposer un enseignement tourné continuellement vers l'**excellence**, ainsi qu'une scolarité tournée vers la **réussite**.

- Les **Cours Pi** s'engagent pour faire du parcours de chacun de ses élèves un **tremplin vers l'avenir**.
- Les **Cours Pi** s'engagent pour ne pas faire de ce nouveau Bac un diplôme au rabais.
- Les **Cours Pi** vous offrent **écoute** et **conseil** pour coconstruire une **scolarité sur-mesure**.

LE BAC DANS LES GRANDES LIGNES

Ce nouveau Lycée, c'est un enseignement à la carte organisé à partir d'un large tronc commun en classe de Seconde et évoluant vers un parcours des plus spécialisés année après année.



CE QUI A CHANGÉ

- Il n'y a plus de séries à proprement parler.
- Les élèves choisissent des spécialités : trois disciplines en classe de Première ; puis n'en conservent que deux en Terminale.
- Une nouvelle épreuve en fin de Terminale : le Grand Oral.
- Pour les lycéens en présentiel l'examen est un mix de contrôle continu et d'examen final laissant envisager un diplôme à plusieurs vitesses.
- Pour nos élèves, qui passeront les épreuves sur table, le Baccalauréat conserve sa valeur.

CE QUI N'A PAS CHANGÉ

- Le Bac reste un examen accessible aux candidats libres avec examen final.
- Le système actuel de mentions est maintenu.
- Les épreuves anticipées de français, écrit et oral, tout comme celle de spécialité abandonnée se dérouleront comme aujourd'hui en fin de Première.

A l'occasion de la réforme du Lycée, nos manuels ont été retravaillés dans notre atelier pédagogique pour un accompagnement optimal à la compréhension. Sur la base des programmes officiels, nous avons choisi de créer de nombreuses rubriques :

- **Suggestions de lecture** pour s'ouvrir à la découverte de livres de choix sur la matière ou le sujet
- **A vous de jouer** pour mettre en application
- **L'essentiel** et **Le temps du bilan** pour souligner les points de cours à mémoriser au cours de l'année
- **Pour aller plus loin** pour visionner des sites ou des documentaires ludiques de qualité
- Et enfin... la rubrique **Les Clés du Bac by Cours Pi** qui vise à vous donner, et ce dès la seconde, toutes les cartes pour réussir votre examen : notions essentielles, méthodologie pas à pas, exercices types et fiches étape de résolution !

SCIENCES DE L'INGÉNIEUR

Module 4 – L'algorithmique et les projets numériques

L'AUTEUR

Dorian JACQUOT

« C'est l'exigence et la bienveillance qui m'ont toujours animé pour concevoir ce manuel par lequel nous partons ensemble à la découverte de ce qui fait les sciences de l'ingénieur : la connaissance scientifique d'une part, mais aussi la démarche, la réflexion, la conception et la communication ». Ingénieur en systèmes mécaniques et professeur agrégé en sciences industrielles de l'ingénieur, Dorian enseigne cette discipline à la croisée des sciences et de l'innovation. Passionné par les langues, en plus de parler couramment Python, Java, HTML, CSS, PHP, C++... il maîtrise également l'anglais et l'espéranto.

PRÉSENTATION

Aujourd'hui, tout scientifique est confronté à la communication. Sa recherche n'est utile pour la société que si elle est communiquée, vulgarisée et expliquée. Savoir commenter des données, argumenter un point de vue scientifique et développer un raisonnement sont des qualités indéniables d'un chercheur ou d'un ingénieur dont les fondamentaux s'apprennent depuis le plus jeune âge.
La discipline « enseignement scientifique » va non seulement permettre aux élèves de constituer leur socle de connaissances culturelles et notionnelles scientifiques, mais aussi de les préparer à analyser, commenter, communiquer et argumenter ses raisonnements, qualités utiles à tout citoyen, à une époque où les grandes questions scientifiques deviennent la responsabilité de chacun.
Ce sont ces compétences qui seront évaluées au baccalauréat et c'est à cela que va vous préparer par étapes, de façon très guidée, ce module d'enseignement scientifique.

CONSEILS À L'ÉLÈVE

Vous disposez d'un support de Cours complet : **prenez le temps** de bien le lire, de le comprendre mais surtout de l'**assimiler**. Vous disposez pour cela d'exemples donnés dans le cours et d'exercices types corrigés.
Vous pouvez rester un peu plus longtemps sur une unité mais travaillez régulièrement.

LES FOURNITURES

Vous devez posséder :

- une **calculatrice graphique comportant un mode examen (requis pour l'épreuve du baccalauréat)** ;
- un **tableur** comme Excel de Microsoft (payant) ou Calc d'Open Office (gratuit et à télécharger sur http://fr.openoffice.org/). En effet, certains exercices seront faits de préférence en utilisant un de ces logiciels, mais vous pourrez également utiliser la calculatrice).

LES DEVOIRS

Les devoirs constituent le moyen d'évaluer l'acquisition de vos *savoirs* (« Ai-je assimilé les notions correspondantes ? ») et de vos *savoir-faire* (« Est-ce que je sais expliquer, justifier, conclure ? »). Placés à des endroits clés des apprentissages, ils permettent la vérification de la bonne assimilation des enseignements.

Aux *Cours Pi*, vous serez accompagnés par un professeur selon chaque matière tout au long de votre année d'étude. Référez-vous à votre « Carnet de Route » pour l'identifier et découvrir son parcours.
Avant de vous lancer dans un devoir, assurez-vous d'avoir **bien compris les consignes**.
Si vous repérez des difficultés lors de sa réalisation, n'hésitez pas à le mettre de côté et à revenir sur les leçons posant problème. **Le devoir n'est pas un examen**, il a pour objectif de s'assurer que, même quelques jours ou semaines après son étude, une notion est toujours comprise.

Aux *Cours Pi*, chaque élève travaille à son rythme, parce que chaque élève est différent et que ce mode d'enseignement permet le « sur-mesure ».

Nous vous engageons à respecter le moment indiqué pour faire les devoirs. Vous les identifierez par le bandeau suivant :

Vous pouvez maintenant faire et envoyer le devoir n°1

Il est **important de tenir compte des remarques, appréciations et conseils du professeur-correcteur**. Pour cela, il est **très important d'envoyer les devoirs au fur et à mesure** et non groupés. **C'est ainsi que vous progresserez !**

Donc, dès qu'un devoir est rédigé, envoyez-le aux *Cours Pi* par le biais que vous avez choisi :

1) Par **soumission en ligne** via votre espace personnel sur **PoulPi**, pour un envoi **gratuit**, **sécurisé** et plus **rapide**.
2) Par **voie postale** à *Cours Pi*, 9 rue Rebuffy, 34 000 Montpellier
   *Vous prendrez alors soin de joindre une **grande enveloppe** **libellée à vos nom et adresse**, et **affranchie** **au tarif en vigueur** pour qu'il vous soit retourné par votre professeur.*

N.B. : *quel que soit le mode d'envoi choisi, vous veillerez à **toujours joindre l'énoncé du devoir ;** plusieurs énoncés étant disponibles pour le même devoir.*
N.B. : *si vous avez opté pour un envoi par voie postale et que vous avez à disposition un scanner, nous vous engageons à conserver une copie numérique du devoir envoyé. Les pertes de courrier par la Poste française sont très rares, mais sont toujours source de grand mécontentement pour l'élève voulant constater les fruits de son travail.*

SOUTIEN ET DISPONIBILITÉ

VOTRE RESPONSABLE PÉDAGOGIQUE

Professeur des écoles, professeur de français, professeur de maths, professeur de langues : notre Direction Pédagogique est constituée de spécialistes capables de dissiper toute incompréhension.

Au-delà de cet accompagnement ponctuel, notre Etablissement a positionné ses Responsables pédagogiques comme des « super profs » capables de co-construire avec vous une scolarité sur-mesure.
En somme, le Responsable pédagogique est votre premier point de contact identifié, à même de vous guider et de répondre à vos différents questionnements.

Votre Responsable pédagogique est la personne en charge du suivi de la scolarité des élèves.
Il est tout naturellement votre premier référent : une question, un doute, une incompréhension ? Votre Responsable pédagogique est là pour vous écouter et vous orienter. Autant que nécessaire et sans aucun surcoût.

QUAND PUIS-JE LE JOINDRE ?

Du lundi au vendredi : horaires disponibles sur votre carnet de route et sur PoulPi.

QUEL EST SON RÔLE ?

Orienter les parents et les élèves.
Proposer la mise en place d'un accompagnement individualisé de l'élève.
Faire évoluer les outils pédagogiques.
Encadrer et coordonner les différents professeurs.

VOS PROFESSEURS CORRECTEURS

Notre Etablissement a choisi de s'entourer de professeurs diplômés et expérimentés, parce qu'eux seuls ont une parfaite connaissance de ce qu'est un élève et parce qu'eux seuls maîtrisent les attendus de leur discipline. En lien direct avec votre Responsable pédagogique, ils prendront en compte les spécificités de l'élève dans leur correction. Volontairement bienveillants, leur correction sera néanmoins juste, pour mieux progresser.

QUAND PUIS-JE LE JOINDRE ?

Une question sur sa correction ?

- faites un mail ou téléphonez à votre correcteur et demandez-lui d'être recontacté en lui laissant un message avec votre nom, celui de votre enfant et votre numéro.
- autrement pour une réponse en temps réel, appelez votre Responsable pédagogique.

LE BUREAU DE LA SCOLARITÉ

Placé sous la direction d'Elena COZZANI, le Bureau de la Scolarité vous orientera et vous guidera dans vos démarches administratives. En connaissance parfaite du fonctionnement de l'Etablissement, ces référents administratifs sauront solutionner vos problématiques et, au besoin, vous rediriger vers le bon interlocuteur.

QUAND PUIS-JE LE JOINDRE ?

Du lundi au vendredi : horaires disponibles sur votre carnet de route et sur PoulPi.
04.67.34.03.00
scolarite@cours-pi.com

LE SOMMAIRE

COMPÉTENCES VISÉES

- Traduire le comportement attendu ou observé d'un objet.
- Comportement séquentiel.
- Structures algorithmiques.
- Analyser le traitement de l'information.
- Analyser le comportement d'un objet à partir d'une description à événements discrets.
- Associer un modèle à un système asservi.

COMPÉTENCES VISÉES

- Imaginer une solution originale, appropriée et esthétique.
- Matérialiser une solution virtuelle.
- Analyser le besoin, l'organisation matérielle et fonctionnelle par une démarche d'ingénierie système.
- Diagramme d'états-transitions.
- Rendre compte de résultats.
- Quantifier les écarts de performances entre les valeurs attendues, les valeurs mesurées et les valeurs obtenues par simulation.

SUGGESTIONS CULTURELLES

QUESTIONS D'ORIENTATION PROFESSIONNELLE

- **Webinar : devenir ingénieur du numérique** *vidéo YouTube de la chaîne ESIEA*
- **Ingénieur.e.s d'un numérique utile : quels métiers ?** *vidéo YouTube de la chaîne ESIEA*

ESSAIS

- **Histoire de la révolution numérique** *Clive Gifford*
- **La belle histoire des révolutions numériques** *Henri Lilen*
- **L'intelligence artificielle n'existe pas** *Luc Julia*
- **Steve Jobs** *Walter Isaacson*
- **Ada ou la beauté des nombres** *Catherine Dufour*

BANDES DESSINÉES

- **Intelligences artificielles : miroirs de nos vies** *FibreTigre*
- **Les Rêveurs lunaires : 4 génies qui ont changé l'Histoire** *Edmond Baudoin & Cédric Villani*
- **La petite bédéthèque des savoirs : l'intelligence artificielle** *Marion Montaigne*

DOCUMENTAIRES AUDIOVISUELS

- **AlphaGo** *Greg Kohs*
- **Une contre-histoire de l'Internet** *Sylvain Bergère*

PODCASTS

- **La méthode scientifique** *France Culture*
- **Informatique et sciences numériques** *Podcasts du Collège de France*


Le Cours

INTRODUCTION

Les produits technologiques sont conçus et développés à la suite d'un processus complexe, en plusieurs étapes, qu'on appelle un **projet**. L'ingénieur doit connaître ce processus, et être capable de mener une démarche de projet, du début à la fin, pour créer un produit innovant, adapté au monde d'aujourd'hui et à ses futurs utilisateurs.

Dans ce module, vous allez apprendre à structurer et piloter un projet, dont l'objectif sera de créer un produit domotique (domotique = automatisation de la maison). En terminale, quand vous aurez acquis encore plus de connaissances, nous pourrons appliquer cette démarche pour créer un projet pluri-technologique, mélangeant tous les domaines technologiques des sciences de l'ingénieur.

Avant de se lancer dans les projets, il va falloir développer des compétences en **algorithmique**. L'algorithmique est un domaine des sciences et technologies dans lequel on apprend à concevoir les **programmes informatiques**. Les programmes que nous apprendrons à concevoir conviendront, à la fois à des ordinateurs, et à des produits intelligents, comme des robots, des drones ou des systèmes domotiques.

Pack de composants

Pour réaliser les prototypes électroniques du chapitre 2, nous travaillerons avec un outil de simulation : Tinkercad. Ainsi, vous pouvez faire ce module sans avoir besoin des composants réels.

Néanmoins, si vous souhaitez prototyper des produits, avec des composants réels, vous aurez besoin d'acheter un pack de composants électroniques. Voici un conseil d'achat : **ELEGOO UNO R3 Project Kit de démarrage (63 articles)**, qui est disponible sur **Amazon**. Ce pack, personnellement testé par l'auteur du module, contient tous les composants que nous utiliserons (sauf le commutateur), ainsi que tout le « petit matériel » (fils, breadboard, cable USB, etc.) nécessaire pour prototyper.

Algorigramme : un algorigramme est un diagramme décrivant graphiquement comment un produit intelligent doit fonctionner.

Algorithme : un algorithme est une série de consignes décrivant textuellement, ou graphiquement, comment un produit intelligent doit fonctionner.

Besoin : au début d'un projet, on identifie ce dont les personnes ont besoin. Combler ce besoin, grâce à un nouveau produit, devient l'objectif du projet.

Binaire / Information binaire : une information binaire est une information qui ne peut prendre que deux valeurs : 0 ou 1. Un code binaire, ou des données binaires, sont ainsi composées uniquement de 0 et de 1.

Cahier des charges : le cahier des charges d'un produit est un ensemble de documents. Ces documents décrivent, de manière technique et précise, tout ce que le produit doit être capable de faire.

Codeblocks : codeblocks est un langage de programmation graphique, dans lequel on crée le programme en assemblant des blocs de programmation.

Conception : la conception d'un produit, qui n'existe pas encore, est un processus lors duquel on va imaginer à quoi le produit ressemblera. On va ensuite dessiner des plans et des diagrammes pour décrire le produit.

Électronique : l'électronique est le domaine scientifique et technique qui étudie comment utiliser l'électricité pour donner un comportement intelligent à un produit.

Fonction : une fonction est une capacité que possède un produit. Par exemple, si un produit possède la fonction « acquérir de l'information », il pourra récupérer des informations de son environnement.

Produit : un produit est un objet dont la création a été rendue possible par la technologie humaine. La plupart des objets du quotidien sont des produits, que nous pouvons étudier en SI.

Prototype : un prototype est une première version d'un produit, fonctionnelle, qui pourrait être commercialisée. Le prototype est généralement ce qu'on obtient à la fin d'un projet.

Pseudo-code : le pseudo-code est un langage, qui est un français simplifié et structuré, permettant de décrire textuellement comment un produit intelligent doit fonctionner.

SysML / Diagramme SysML : le SysML, System Modeling Langage, est un langage de modélisation qui permet de décrire les produits grâce à des diagrammes.

BIENVENUE DANS LE MONDE DES MÉTIERS DE L'INGÉNIERIE !

Nous vivons actuellement dans un monde technologique et connecté qui a un impact majeur sur nos vies, que ce soit pour améliorer notre quotidien, le sécuriser ou tout simplement permettre l'établissement et le bon fonctionnement des infrastructures humaines que sont par exemple les usines, les réseaux de transport, les hôpitaux et l'ensemble des objets informatiques et connectés qui font notre quotidien. L'ingénierie est également au cœur de la réflexion globale pour la création du monde de demain, notamment en rapport avec les problématiques environnementales et économiques auxquelles font face les sociétés modernes.

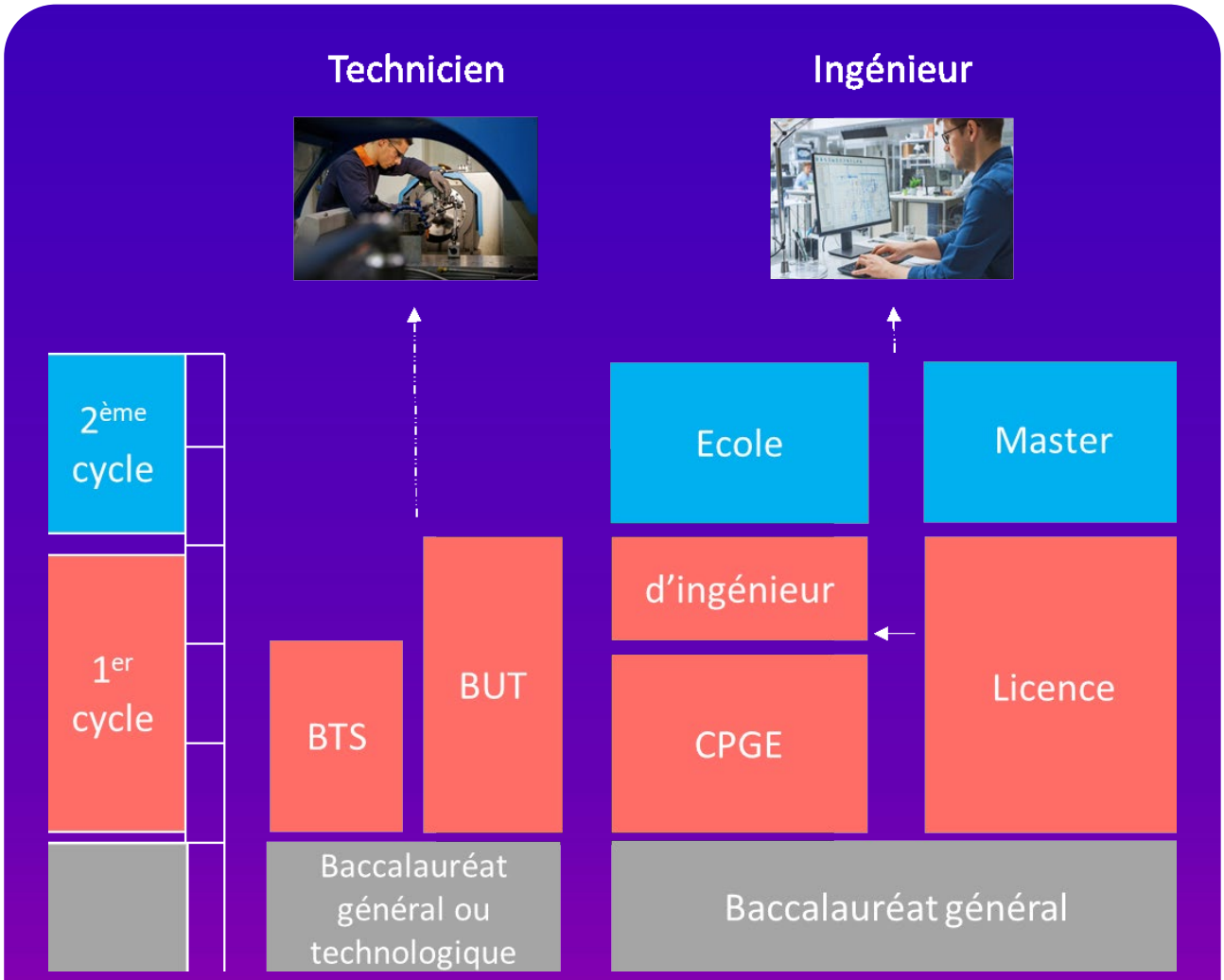
Comme vous l'aurez compris, les métiers de l'ingénierie sont donc essentiels et omniprésents dans nos vies personnelles et professionnelles. Suivant les appétences de chacun, les domaines d'étude et d'application sont extrêmement variés comme les biotechnologies, la domotique, l'informatique, le génie civil, les systèmes et réseaux, le génie civil...etc.

Deux grands types de profils se trouvent impliqués dans ces grands défis. D'un côté les techniciens, qui seront au cœur des réalisations avec des taches précises qui pourront prendre par exemple la forme d'application de protocoles, de la gestion et la maintenance des appareils ou bien de la réalisation de test et d'essais. A leurs côtés, les ingénieurs auront eux un rôle plus large et décisionnel, que ce soit pour créer, piloter ou bien assurer la gestion humaine des différents projets. Les ingénieurs pourront également évoluer au cours de leur carrière vers des fonctions de direction et de développement.

COMMENT ACCÉDER À CES MÉTIERS ?

Pour devenir **technicien**, il vous faudra tout d'abord obtenir un baccalauréat général à tonalité scientifique (enseignements de spécialité à choisir suivant les domaines d'études visés) ou bien un baccalauréat technologique. Une fois le diplôme obtenu, différentes formations préparent au métier de technicien. Le Brevet de Technicien Supérieur (BTS) mène à un diplôme de niveau Bac +2 et il vous faudra choisir votre formation en fonction du domaine d'étude visé. Vous pourrez également accéder au titre de technicien par un Bachelor Universitaire de Technologie (BUT, de niveau Bac +3) en choisissant tout comme pour le BTS une formation correspondant au domaine d'activité qui vous passionne.

Pour devenir **ingénieur**, nous vous conseillons un baccalauréat à tonalité scientifique en privilégiant, au choix et suivant votre projet professionnel, les enseignements de spécialités en science fondamentale que sont la Physique-Chimie et les Mathématiques, et bien sur les Sciences de l'Ingénieur. Les enseignements de Mathématiques Experte et d'Informatique seront également importants pour posséder les connaissances et compétences transverses nécessaires à la bonne réalisation de ce métier. Suite à l'obtention de votre diplôme, la voie royale pour devenir ingénieur passera par les Ecoles d'ingénieurs généralistes ou spécialisés dans le domaine souhaité. Ces formations sont accessibles par concours après deux années de classe préparatoire aux grandes écoles (CPGE, formation sélective à privilégier pour accéder aux meilleurs établissements). De plus en plus d'écoles proposent également des classes préparatoires intégrées pour une admission postbac, souvent moins sélectives que la CPGE mais il est à noter que ces formations sont de plus en plus reconnues professionnellement et constitue une voie d'avenir pour accéder aux métiers de l'ingénierie. Il est également possible d'accéder à ce métier par la filière universitaire, en suivant un Master en ingénierie. Ces deux formations vous amèneront à des diplômes de niveau Bac + 5.



Dans chacun des modules de notre discipline, nous vous présenterons différentes spécialités d'ingénierie, en lien avec le domaine étudié, au travers de la présentation de différents métiers d'ingénieur. Nous vous donnerons des pistes pour vous y plonger au travers de recommandations de visites, de vidéos, d'interviews, voire même de MOOC accessibles et gratuits.

Ainsi dans ce module vous découvrirez les métiers suivants

- Ingénieur IoT
- Ingénieur IA
- Ingénieur satellite
- Ingénieur réseau

CHAPITRE 1
ALGORITHMIQUE

Pour rendre un produit intelligent, il faut le doter de deux choses :

- des **composants** capables de produire ou manipuler de l'information, comme des capteurs, un microcontrôleur ou un microprocesseur, des mémoires, etc.
- un « guide » indiquant au produit ce qu'il doit faire en fonction des informations à sa disposition. Quelles décisions doit-t-il prendre ? Quelles actions doit-il faire ?

Dans le premier module de SI, nous avons déjà vu quels composants sont nécessaires à un produit intelligent : ce sont les composants formant la **chaîne d'information** du produit. Dans ce chapitre, nous allons voir comment créer le « guide », qu'on appelle un **algorithme**. Nous verrons ensuite comment le transmettre au produit, pour qu'il puisse l'utiliser, sachant que le produit ne comprend que le **binaire**.

Plusieurs méthodes existent pour créer des algorithmes, nous allons apprendre à utiliser les deux plus importantes :

- le **pseudo-code**, qui décrit textuellement les algorithmes ;
- les **algorigrammes**, qui décrivent graphiquement les algorithmes.

COMPÉTENCES VISÉES

- Traduire le comportement attendu ou observé d'un objet.
- Comportement séquentiel.
- Structures algorithmiques (variables, fonctions, structures séquentielles, itératives, répétitives, conditionnelles).
- Analyser le traitement de l'information.
- Analyser le comportement d'un objet à partir d'une description à événements discrets.
- Associer un modèle à un système asservi.

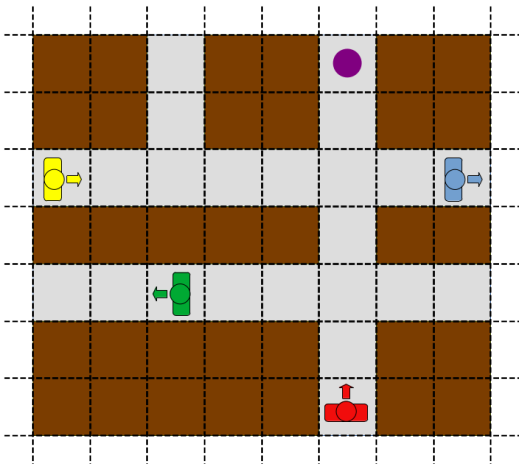
PRÉ-REQUIS

- Chaîne d'information.

Première approche
Les bases d'un algorithme

Voici le plan, en vue du dessus, d'une ville :

- Les parties grises sont des routes, sur lesquelles on peut se déplacer - les parties marron sont des bâtiments - chaque carreau représente un carré de 1m sur 1m.
- Les 4 symboles  représentent des personnages. La flèche indique le sens dans lequel ils regardent.
- Le point violet est un objectif, qu'on va faire atteindre aux personnages.



À VOUS DE JOUER 1

1. Le personnage rouge ne comprend que les phrases « avance de X mètres » et « recule de X mètres », avec X un nombre. Décrivez-lui comment atteindre le point violet.

2. Est-ce qu'avec ces deux phrases, uniquement, les autres personnages peuvent atteindre le point violet ?

3. Le personnage bleu comprend, en plus du personnage rouge, les phrases « Tourne de 90° à droite » et « Tourne de 90° à gauche ». Décrivez-lui comment atteindre le point violet, de deux manières différentes.

4. Le personnage vert comprend en plus, par rapport au personnage bleu, la phrase « Tourne de 180 ». Décrivez-lui comment atteindre le point violet en un minimum de phrases.

5. Le personnage jaune comprend uniquement les phrases « Avance de 1m », « Tourne de 90° à droite » et « Tourne de 90° à gauche ». Décrivez-lui comment atteindre le point violet.

Ce que vous avez décrit à chaque personnage s'appelle un **algorithme**, c'est-à-dire une suite de consignes. En suivant ces consignes, une à une, les personnages atteignent leur objectif : le point violet. Pour chaque personnage, les consignes possibles sont limitées et très précises, on les appelle des **instructions**. Certaines instructions peuvent comporter des parties **variables** (comme le X dans « avance de X mètres »). Si les instructions sont mal choisies (comme pour le premier cas), il est possible qu'un personnage ne puisse pas atteindre son objectif, à cause de sa position de départ.

À VOUS DE JOUER 2

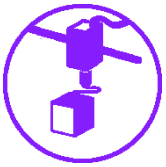
1. Essayez de trouver uniquement 2 instructions, qui suffiraient pour pouvoir décrire à chaque personnage comment atteindre le point violet. Ces deux instructions ne doivent pas comporter de partie variable.

01 ALGORITHMIQUE
Introduction à l'algorithmique

ALGORITHME

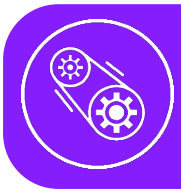
Un **algorithme** est une suite de consignes simples, qu'on appelle des **instructions**. En suivant les consignes, une par une et dans l'ordre, l'algorithme permet de réaliser une action complexe ou de résoudre un problème.

Le mot algorithme vient du nom d'un mathématicien perse du IX^{ème} siècle, Al-Khwârizmî (en arabe : الخوارزمي). Il décrivit des méthodes de résolution d'équations mathématiques (les mêmes que celles que vous utilisez en mathématiques). Ces méthodes sont composées de plusieurs consignes à suivre, comme nos algorithmes modernes.



EXEMPLES

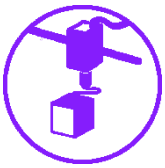
Pour construire un meuble ou assembler un vaisseau spatial en LEGO, on doit suivre l'algorithme présenté dans le guide de montage. Ce guide est composé de plusieurs étapes simples. La réalisation successive de ces étapes permet de réaliser l'action complète « construire un meuble » ou « monter un vaisseau spatial en LEGO »



L'ESSENTIEL

Un algorithme est une suite d'instructions, qui sont exécutées par un produit intelligent, pour réaliser une action complexe.

Un produit intelligent base son fonctionnement sur des algorithmes. Le produit peut utiliser plusieurs algorithmes différents.



EXEMPLES

Un aspirateur autonome utilise un algorithme pour nettoyer une pièce, en évitant les obstacles. Il utilise aussi un second algorithme pour retourner à sa base de chargement.

ALGORITHME ET PROGRAMMATION

Algorithmique

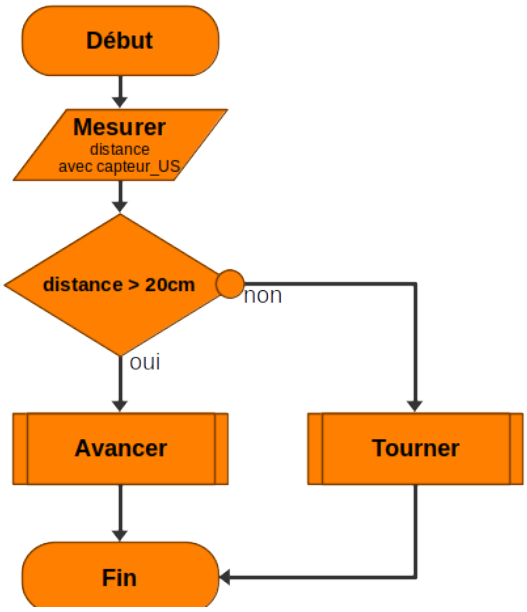
L'algorithmique est la discipline dans laquelle on apprend à concevoir et écrire des algorithmes afin de résoudre des problèmes complexes. Plusieurs représentations peuvent être utilisées pour écrire un algorithme. Nous allons voir les deux principales :

- le pseudo-code, qui propose d'écrire l'algorithme sous forme textuelle ;
- les algorigrammes, qui proposent de représenter l'algorithme sous forme graphique.

Algorithme écrit en pseudo-code

```
Début
   Écrire "Quel âge avez-vous?"
   Lire age
   Si age < 18
      Écrire "Vous êtes mineur"
   Sinon
      Écrire "Vous êtes majeur"
   FinSi
Fin
```

Algorithme représenté par un algorigramme

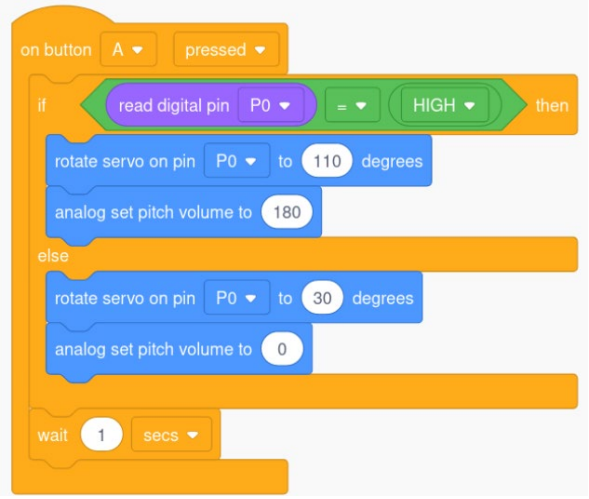


Programmation

L'algorithme, une fois terminé, peut être utilisé pour créer un programme informatique. Pour cela, il faut utiliser un langage de programmation. On trouve deux types de langages de programmation :

- les langages blocs, qui permettent de construire un programme en associant des blocs (Codeblocks que nous allons utiliser dans ce cours, Scratch, mBlock, etc.) ;
- des langages en lignes de code, qui permettent d'écrire un programme sous forme textuelle (Python, que nous apprendrons à utiliser en terminale, JavaScript, C++, Java, PHP, etc.).

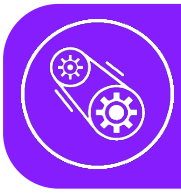
Programme réalisé en Codeblocks



Programme écrit en Python

```
if pins.button_pressed_a() == True :
    if pins.digital_read_pin(DigitalPin.P0) == True :
        pins.servo_write_pin(AnalogPin.P0, 110)
        pins.analog_set_pitch_volume(180)
    else :
        pins.servo_write_pin(AnalogPin.P0, 30)
        pins.analog_set_pitch_volume(0)
    basic.pause(1000)
input.on_button_pressed(Button.A, on_button_pressed_a)
```

PRINCIPE DU PSEUDO-CODE



L'ESSENTIEL

Le pseudo-code permet de concevoir et écrire un algorithme sous forme textuelle.

Les algorithmes écrits en pseudo-code possèdent la structure suivante :

```
Algorithme<nom>
Début
   <instructions>
Fin
```

La première ligne permet d'indiquer le nom de l'algorithme. Les éléments de structure Début et Fin permettent de mettre en évidence les limites de l'algorithme. <instructions> sera remplacé par autant d'instructions que l'on veut : nous allons découvrir les différentes instructions possibles par la suite.

Il n'y a pas de norme indiquant comment écrire un algorithme en pseudo-code. Dans ce cours, nous verrons une manière d'écrire en pseudo-code, mais il est possible que vous en rencontriez d'autres, légèrement différentes, au cours de vos études ou recherches.

Commentaires : il est possible d'ajouter des commentaires à son algorithme, en les faisant précéder des symboles //. Un commentaire est une indication aidant à comprendre un algorithme. Le commentaire ne fait pas vraiment partie de l'algorithme, et il sera ignoré par l'ordinateur ou le produit intelligent qui lira l'algorithme. Exemple :

```
Algorithme<nom>
Début
    <instructions>//Ceci est un commentaire
Fin
```

Pseudo-code et langage de programmation

Le pseudo-code permet d'écrire des algorithmes, indépendamment d'un langage de programmation en particulier. Les algorithmes écrits en pseudo-code sont donc universels, et pourront ensuite être traduits dans n'importe quel langage de programmation.

Pseudo-code et produit

Dans cette partie du chapitre, on va voir comment écrire des algorithmes en pseudo-code, destinés à des **ordinateurs**. Nous verrons ensuite comment les écrire pour des produits intelligents quelconques (robot, drone, voitures intelligentes, objets connectés, etc.).

ENTRÉES-SORTIES

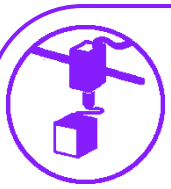
Un ordinateur est constitué de plusieurs composants :

- Certains de ces composants lui permettent de récupérer des informations, en provenance de son environnement (fonction **« acquérir »** de la chaîne d'information). C'est le cas du clavier, de la souris et du microphone. Ces composants sont appelés des composants **d'entrée**.
- D'autres composants lui permettent de restituer des informations, pour les transmettre à un utilisateur par exemple (fonction **« restituer »** de la chaîne d'information). C'est le cas de l'écran et des haut-parleurs. Ces composants sont appelés des composants de **sortie**.

Entrée-Sortie est parfois abrégé E/S. En anglais, on parle d'Input/Output, soit I/O.

INSTRUCTION DE SORTIE

En pseudo-code, quand nous créerons des algorithmes destinés à un ordinateur, nous utiliserons la syntaxe Écrire <information> pour noter l'instruction de sortie, qu'on appelle aussi **instruction d'écriture**. Cette instruction indique à l'ordinateur qu'il doit envoyer une information vers un de ses composants de sortie, habituellement l'écran. Le mot-clef Écrire vient du fait que, de son point de vue, l'ordinateur devra écrire sur l'écran. <information> sera remplacé par le texte qui doit être écrit sur l'écran. Enfin, le texte sera écrit entre guillemets.



EXEMPLES

L'algorithme suivant, destiné à un ordinateur, demande à l'ordinateur d'afficher « Bonjour » puis « Je suis un ordinateur » sur l'écran.

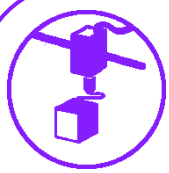
```
Algorithme bonjour_1
Début
    Écrire "Bonjour"
    Écrire "Je suis un ordinateur."
Fin
```

Sur l'écran de l'ordinateur, on obtiendra :

```
Bonjour
Je suis un ordinateur.
```

Avec ou sans saut de ligne

On considérera qu'à la fin de chaque instruction Écrire, l'ordinateur saute une ligne. Ainsi, dans l'exemple ci-dessus, l'ordinateur va écrire « Bonjour » sur une première ligne de l'écran, et « Je suis un ordinateur » sur une deuxième ligne. On va voir, avec un second exemple, comment écrire plusieurs chaînes de caractères sur la même ligne.



EXEMPLES

L'algorithme suivant, destiné à un ordinateur, demande à l'ordinateur d'afficher « Bonjour » puis « Je suis un ordinateur » sur la même ligne.

```
Algorithme bonjour_2
Début
    Écrire "Bonjour", "Je suis un ordinateur."
Fin
```

On a ajouté un espace avant le mot « Je » de la deuxième phrase, pour éviter qu'il soit collé à « Bonjour ».

Sur l'écran de l'ordinateur, on obtiendra :

```
Bonjour Je suis un ordinateur.
```

À VOUS DE JOUER ❸

1. Écrivez l'algorithme d'un programme pour ordinateur qui affiche à l'écran « Bonjour », puis « Comment vas-tu ? ». Chaque phrase est écrite sur une ligne différente de l'écran.

.....

.....

.....

.....

.....

2. Écrivez l'algorithme d'un programme pour ordinateur qui affiche à l'écran « Bonjour », puis « Comment vas-tu ? ». Les deux phrases sont écrites sur la même ligne de l'écran.

.....

.....

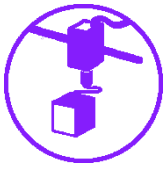
.....

.....

.....

MÉMOIRE

Tout produit intelligent, autant un robot qu'un ordinateur, qui utilise des algorithmes, a besoin de **mémoriser** des informations/données (fonction « stocker » de la chaîne d'information). Pour cela, il va posséder des **supports de stockage**. Ces supports de stockage constituent la **mémoire** du produit.



EXEMPLES

Un smartphone peut stocker des données grâce à une carte micro-SD.
Un drone peut stocker des données de vol grâce à une carte-SD.

Quand le produit « réfléchit », fait des calculs, prend des décisions, il a besoin de manipuler les informations se trouvant dans sa mémoire. Il va principalement faire les 4 opérations suivantes sur sa mémoire :

- chercher des informations dans sa mémoire ;
- modifier les informations déjà dans sa mémoire ;
- supprimer des informations de sa mémoire ;
- ajouter des informations dans sa mémoire.

Emplacement mémoire

Le support de stockage utilisé par le produit peut être représenté comme un tableau. Chaque case de ce tableau s'appelle un emplacement mémoire. Chaque **emplacement mémoire** peut contenir une information simple, comme un petit nombre ou une lettre par exemple. En combinant plusieurs emplacements mémoire, le produit peut mémoriser des grands nombres, des mots, voir des fichiers complets (image, vidéo, etc.).

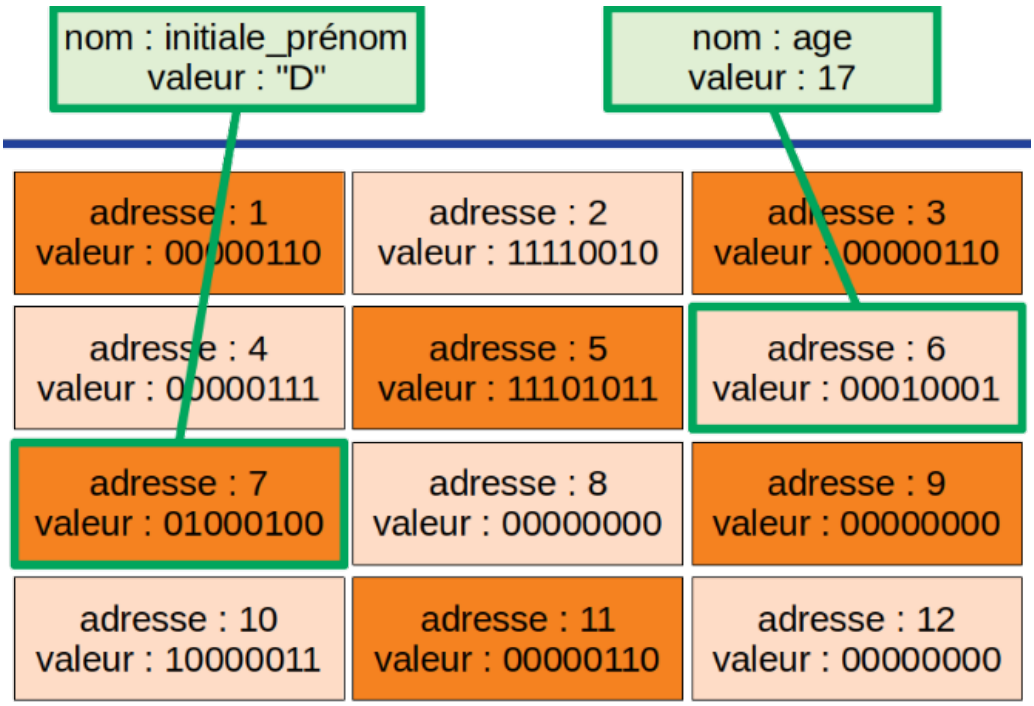
Les données stockées dans chaque emplacement sont sous forme binaire. Elles seront converties en nombres ou en caractères quand elles seront utilisées par le produit. Chaque emplacement mémoire possède une adresse. L'adresse est un numéro qui permet d'identifier et localiser un emplacement mémoire dans l'ensemble de la mémoire.

adresse : 1 valeur : 00000110	adresse : 2 valeur : 11110010	adresse : 3 valeur : 00000110
adresse : 4 valeur : 00000111	adresse : 5 valeur : 11101011	adresse : 6 valeur : 00010001
adresse : 7 valeur : 01000100	adresse : 8 valeur : 00000000	adresse : 9 valeur : 00000000
adresse : 10 valeur : 10000011	adresse : 11 valeur : 00000110	adresse : 12 valeur : 00000000

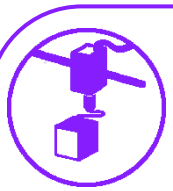
Fragment de la mémoire de l'ordinateur avec 12 emplacements mémoires

VARIABLE

Dans un algorithme, lorsqu'on veut faire manipuler la mémoire au produit, on va utiliser la notion de variable. Une **variable** est une **étiquette**, avec un **nom**, qui est liée à un emplacement mémoire : on dit qu'une variable fait **référence** à un emplacement mémoire. La variable possède une **valeur**, qui correspond à la valeur stockée dans l'emplacement mémoire auquel elle fait référence.



Deux variables, avec leurs noms et leurs valeurs, reliées à deux emplacements mémoires



EXEMPLES

L'algorithme suivant, destiné à un ordinateur, lui demande de :

- stocker la valeur numérique 5, dans un emplacement mémoire auquel fera référence la variable terme_1 ;
- stocker la valeur numérique 7, dans un emplacement mémoire auquel fera référence la variable terme_2 ;
- stocker la valeur numérique, qui sera la somme de terme_1 et terme_2, dans un emplacement mémoire auquel fera référence la variable somme.
- Afficher la valeur contenue dans l'emplacement mémoire auquel la variable somme fait référence. Ici, l'ordinateur affichera donc 12 sur l'écran.

```
Algorithme calcul_1
Variable
    terme_1, terme_2, somme : réel
Début
    terme_1 = 5
    terme_2 = 7
    somme ← terme_1 + terme_2
    Écrire "La somme vaut ", somme, "!"
Fin
```

Cet exemple montre qu'il est possible d'afficher la valeur d'une variable avec l'instruction de sortie Écrire
La flèche ← indique que le résultat du calcul terme_1 + terme_2 est donné comme valeur à la variable somme

Sur l'écran de l'ordinateur, on obtiendra :

```
La somme vaut 12 !
```

Choix du nom

Le nom ne doit pas comporter d'espace. Si le nom comporte plusieurs mots/nombres, on reliera chaque mot/nombre avec le symbole « _ », comme c'est demandé dans le langage Python qu'on verra en terminale.
Ex : quantité_pomme, moteur_roue_droite
Le nom choisi pour une variable doit être explicite, afin que quelqu'un qui lit l'algorithme comprenne immédiatement son rôle. Par exemple, dans l'exemple ci-dessus, on a choisi d'appeler les deux premières variables terme_1 et terme_2, au lieu de nombre_1 et nombre_2. terme est plus explicite que nombre, car il indique que les variables seront utilisées dans une addition. De même, la troisième variable est nommée somme, ce qui permet de savoir que ce sera le résultat d'une addition, même sans lire l'algorithme en entier.

Type

En plus de son nom, une variable possède un **type**, qui indique ce qu'elle contient. Les principaux types de variables sont les suivants :

- entier, pour les variables qui contiennent des nombres entiers. Ex : 1, -55, 200
- réel, pour les variables qui contiennent des nombres réels. Ex : 1, 15, -0,2784
- booléen, pour les variables qui contiennent des valeurs binaires, qui peuvent valoir vrai ou faux
- caractère, pour les variables qui contiennent un caractère alphanumérique ou un caractère spécial. Le caractère sera noté entre guillemets. Ex : "A", "8", "b", "@"
- chaîne, qui vient de « chaîne de caractères » pour les variables qui contiennent des chaînes de caractères alphanumériques/spéciaux. La chaîne sera notée entre guillemets. Ex : "ça va ?", "bonjour"

Les nombres entiers sont une sous-catégorie des nombres réels, et les caractères sont une sous-catégorie des chaînes de caractères.

À VOUS DE JOUER 4

1. Indiquez les types de variables possibles pour ces valeurs : 175, 4,23, -51, "au revoir", "ç", vrai, "0,18", faux

.....

.....

.....

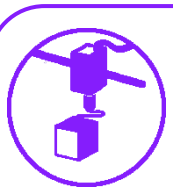
2. Écrivez l'algorithme d'un programme pour ordinateur qui demande à l'ordinateur de stocker les valeurs 10 et 3, puis qui les multiplie et qui affiche le résultat de la multiplication sur l'écran. Essayez d'utiliser les noms de variables les plus explicites possibles.

INSTRUCTION D'ENTRÉE

Nous avons déjà vu l'instruction de sortie, qui sert à demander à l'ordinateur de restituer une information à l'utilisateur. **L'instruction d'entrée** fait l'inverse : elle indique à l'ordinateur qu'il doit récupérer une information venant de l'utilisateur, habituellement via le clavier.

En pseudo-code, nous utiliserons la syntaxe Lire <variable> pour noter l'instruction d'entrée, qu'on appelle aussi **instruction de lecture**. Cette instruction indique à l'ordinateur qu'il doit attendre que l'utilisateur lui fournisse une information, habituellement du texte entré au clavier. L'instruction sera exécutée dès que l'utilisateur aura terminé d'écrire un texte au clavier, et appuyé sur la touche Entrée.

Le mot-clef Lire vient du fait que, de son point de vue, l'ordinateur devra lire ce que l'utilisateur a écrit avec le clavier. <variable> sera le nom d'une variable, qui servira à indiquer à l'ordinateur qu'il doit stocker ce qu'il a lu dans un emplacement mémoire.



EXEMPLES

L'algorithme suivant, destiné à un ordinateur, demande à l'ordinateur d'afficher « Bonjour », puis de demander à l'utilisateur d'entrer son nom. Ensuite l'ordinateur doit attendre que l'utilisateur ait entré et validé son nom, et ensuite l'ordinateur devra stocker le nom dans une variable nommée utilisateur_nom. Enfin, l'ordinateur dira à l'utilisateur que son nom est beau.

```
Algorithme bonjour_3
Variable
    utilisateur_nom : chaîne de caractères
Début
    Écrire "Bonjour"
    Écrire "Quel est ton nom ?"
    Lire utilisateur_nom
    Écrire utilisateur_nom, " est un beau nom."
Fin
```

Sur l'écran de l'ordinateur, on obtiendra :

```
Bonjour
Quel est ton nom ?
>>>Dorian
Dorian est un beau nom.
```

Nous utiliserons les symboles >>> pour indiquer que l'ordinateur attend que l'utilisateur écrive quelque chose. Ce qui est en bleu a été entré par l'utilisateur.

C'est le premier algorithme **interactif** et au comportement variable que nous écrivons : son résultat dépend de ce que fait l'utilisateur. Par exemple, si l'utilisateur indique que son nom est Hye-Yeong au lieu de Dorian, on obtiendra :

```
Bonjour
Quel est ton nom ?
>>>Hye-Yeong
Hye-Yeong est un beau nom.
```

À VOUS DE JOUER 5

1. Écrivez l'algorithme d'un programme pour ordinateur qui demande à l'utilisateur d'entrer deux nombres, qui soustrait le deuxième au premier, puis qui affiche le résultat de la soustraction à l'écran.

.....

.....

.....

.....

INSTRUCTION D'AFFECTATION

Nous avons déjà rencontré l'instruction d'affectation, qui se note avec une flèche <variable> ← <expression>, avec <variable> le nom d'une variable à laquelle on va donner la valeur de <expression>. <expression> est généralement un calcul, ou une valeur (nombre, chaîne de caractères, etc.). Les expressions peuvent être créées à l'aide des opérateurs mathématiques suivants :

- + qui est l'opérateur d'addition ;
- - qui est l'opérateur de soustraction ;
- * qui est l'opérateur de multiplication ;
- / qui est l'opérateur de division décimale ;
- // qui est l'opérateur de division euclidienne ;
- % qui est l'opérateur modulo (reste d'une division euclidienne).

On peut aussi utiliser des parenthèses (et), pour créer des expressions complexes.

EXEMPLES

L'algorithme suivant, destiné à un ordinateur, lui indique qu'il doit :

- demander à l'utilisateur la longueur d'un rectangle, et la stocker ;
- demander à l'utilisateur la largeur d'un rectangle, et la stocker ;
- calculer et stocker l'aire du rectangle, avec la variable aire ;
- calculer et stocker le périmètre du rectangle, avec la variable périmètre ;
- afficher l'aire et le périmètre du rectangle à l'écran.

```
Algorithme rectangle_1
Variable
    longueur, largeur, aire, périmètre : réel
Début
    Écrire "Donnez-moi la longueur d'un rectangle, en cm :"
    Lire longueur
    Écrire "Donnez-moi la largeur d'un rectangle, en cm :"
    Lire largeur
    aire ← longueur*largeur
    périmètre ← 2*longueur + 2*largeur
    Écrire "L'aire du rectangle vaut ", aire, "cm²."
    Écrire "Le périmètre du rectangle vaut ", périmètre, "cm."
Fin
```

À VOUS DE JOUER 6

1. Écrivez l'algorithme d'un programme pour ordinateur. L'ordinateur devra demander à l'utilisateur d'entrer une vitesse en km/h, puis la convertir en m/s et afficher le résultat de la conversion à l'écran.

2. Écrivez l'algorithme d'un programme pour ordinateur. L'ordinateur devra demander à l'utilisateur d'entrer combien d'argent il possède, puis d'entrer le prix d'un croissant. L'ordinateur devra ensuite déterminer le nombre de croissants que peut acheter l'utilisateur, et afficher le résultat à l'écran.

FONCTIONS

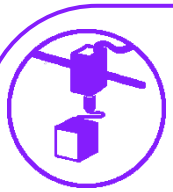
Enfin, en pseudo-code, il est possible d'intégrer à ses algorithmes ce qu'on appelle des fonctions. Une fonction permet de faire réaliser des tâches complexes à l'ordinateur, sans avoir à écrire toutes les lignes nécessaires. Par exemple, il existe une fonction max qui sert à trouver le plus grand nombre parmi une liste de nombres, ou la fonction racine_carrée qui calcule la racine carrée d'un nombre.

Une fonction reçoit des arguments, qu'on transmet à la fonction. Un argument est une valeur ou une variable. Ensuite, la fonction réalise une série d'opérations à partir de ces arguments : calculs, conversions, manipulations de chaîne, etc. Enfin, la fonction renvoie un résultat, qui est une valeur. Ce résultat peut être affecté à une variable, ou bien utilisé dans une instruction de lecture/écriture.

Voici quelques exemples de fonctions souvent utilisées :

- max(<nombre>, <nombre>) renvoie le plus grand nombre parmi les deux, ou plus, nombres passés en arguments.
- min(<nombre>, <nombre>) renvoie le plus petit nombre parmi les deux, ou plus, nombres passés en arguments.
- moyenne(<nombre>, <nombre>) renvoie la moyenne d'une série de nombres passés en arguments.
- aléatoire(<nombre>, <nombre>) donne un nombre entier aléatoire, compris entre les deux valeurs.
- racine_carrée(<nombre>) renvoie la racine carrée d'un nombre.
- puissance(<base>, <exposant>) renvoie la valeur du nombre <base>, élevé à l'exposant <exposant>.
- longueur(<chaîne>) calcule la longueur, en nombre de caractères, d'une chaîne de caractères.
- concaténer(<chaîne>, <chaîne>) met bout à bout deux, ou plus, chaînes de caractères.
- pause(<durée>) qui ne renvoie rien, mais fait faire une pause de <durée> secondes à l'ordinateur.

Ces fonctions existent dans tous les langages de programmation, c'est pourquoi on peut s'en servir en pseudo-code sans que le pseudo-code perde son universalité.



EXEMPLES

L'algorithme suivant, destiné à un ordinateur, lui indique qu'il doit demander à l'utilisateur ses trois dernières notes, puis lui afficher sa meilleure note et sa moins bonne.

```
Algorithme notes_1
Variable
    note_1, note_2, note_3, meilleure_note, pire_note : réel
Début
    Écrire "Quelles sont vos trois dernières notes ?"
    Lire note_1
    Lire note_2
    Lire note_3
    meilleure_note ← max(note_1, note_2, note_3)
    pire_note ← min(note_1, note_2, note_3)
    Écrire "Votre meilleur note est ", meilleure_note
    Écrire "Votre moins bonne note est ", pire_note
Fin
```

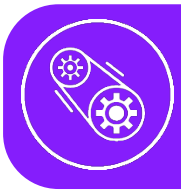
À VOUS DE JOUER 7

1. Écrivez l'algorithme d'un programme pour ordinateur. L'ordinateur devra demander à l'utilisateur ses deux dernières notes, et affichera la moyenne de l'utilisateur.

2. Écrivez l'algorithme d'un programme pour ordinateur, permettant à l'utilisateur de faire des calculs de puissance. L'ordinateur devra demander à l'utilisateur le nombre de base, ainsi que la puissance à laquelle ce nombre doit être élevé. L'ordinateur affichera ensuite le résultat.

03 ALGORITHMIQUE
Structures conditionnelles

Nous avons vu toutes les instructions de base. Pour aller plus loin, et écrire des algorithmes plus complexes, il nous faut maintenant apprendre à utiliser les **structures conditionnelles**, qui permettent à l'ordinateur de faire des choix, et de réaliser des opérations complètement différentes, en fonction du contexte.

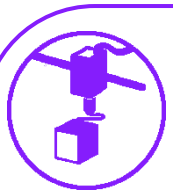


L'ESSENTIEL

Une structure conditionnelle est une structure de contrôle permettant de faire exécuter à l'ordinateur des instructions différentes en fonction du contexte.

STRUCTURE SI...

La structure conditionnelle Si... permet d'indiquer à l'ordinateur qu'il doit exécuter certaines instructions uniquement si une condition est respectée. On appelle cette structure la **structure conditionnelle simple**.



EXEMPLES

L'algorithme suivant, destiné à un ordinateur, lui indique qu'il doit demander à l'utilisateur son âge. Ensuite, si l'utilisateur a plus de 18 ans, l'ordinateur lui dira qu'il est majeur.

```
Algorithme demande_age_1
Variable
    age: entier
Début
    Écrire "Quel âge avez-vous ?"
    Lire age
    Si age ≥ 18
            Écrire "Vous êtes majeur"
            Écrire "Bienvenu"
    FinSi
            Écrire "Au revoir"
Fin
```

Voici ce qu'il se passe, si l'utilisateur dit qu'il a 22 ans :

```
Quel âge avez-vous ?
>>>22
Vous êtes majeur
Bienvenu
Au revoir
```

Voici ce qu'il se passe, si l'utilisateur dit qu'il a 15 ans :

```
Quel âge avez-vous ?
>>>15
Au revoir
```

Dans cet exemple, les deux instructions Écrire "Vous êtes majeur" et Écrire "Bienvenu" sont des instructions conditionnelles : elles ne sont exécutées que si la condition age ≥ 18 est validée. La structure conditionnelle représente l'ensemble des lignes allant de Si à FinSi.

Indentation

On peut noter une chose particulière dans l'écriture d'une structure Si, qu'on retrouvera dans toutes les autres structures algorithmiques : toutes les instructions conditionnelles sont écrites avec un décalage par rapport à la gauche. Cela s'appelle **l'indentation** et on dit que les lignes décalées sont indentées. L'indentation permet de mieux structurer son algorithme visuellement, et de le rendre plus lisible.

Dans certains langages de programmation l'indentation est conseillée, pour la lisibilité. En pseudo-code et en Python, l'indentation est obligatoire.

À VOUS DE JOUER 8

1. Avec le programme ci-dessous, qu'est-ce-qui sera affiché sur l'écran :
   - si l'utilisateur entre 5 ?
   - si l'utilisateur entre 9 ?
   - si l'utilisateur entre 0 ?

.....

.....

.....

.....

```
Algorithme algo_bizarre_1
Variable
    nombre: entier
Début
    Écrire "Entrez un nombre entre 0 et 10 : "
    Lire nombre
    Si nombre> 1
        Écrire"Réponse 1"
    FinSi
    Si nombre ≥ 7
        Écrire "Réponse 2"
    FinSi
Fin
```

CONDITION ET TESTS

Au cours de l'exécution d'un algorithme, lorsque l'ordinateur atteint une structure Si, il évalue sa **condition**. La condition est de la forme <élément><comparateur><élément>. Les deux éléments peuvent être des valeurs, des variables ou des expressions. L'ordinateur va comparer les deux éléments, en fonction du **comparateur**, aussi appelé **opérateur de comparaison**. Les principaux comparateurs sont :

- > qui est l'opérateur de stricte supériorité ;
- ≥ qui est l'opérateur de supériorité ou d'égalité ;
- < qui est l'opérateur de stricte infériorité ;
- ≤ qui est l'opérateur d'infériorité ou d'égalité ;
- = qui est l'opérateur d'égalité ;
- <> qui est l'opérateur de différence.

Le résultat de la comparaison peut être vrai, si la condition est validée, ou faux, si la condition n'est pas validée.

À VOUS DE JOUER 9

1. Écrivez l'algorithme d'un programme pour ordinateur, simulant un lancer de dés à 6 faces. L'ordinateur devra créer un nombre aléatoire entre 1 et 6 (avec la fonction aléatoire). Ensuite, le programme demandera à l'utilisateur de deviner ce nombre. Si le joueur a trouvé le bon nombre, l'ordinateur affichera « Gagné! »

2. Écrivez l'algorithme d'un programme pour ordinateur. L'ordinateur devra demander à l'utilisateur sa moyenne en SI, puis lui dire « Très bien » s'il a 16/20 ou plus.

STRUCTURE SI...SINON...

La structure conditionnelle Si...Sinon... permet d'indiquer à l'ordinateur qu'il doit exécuter certaines instructions, placées entre Si et Sinon uniquement si une condition est respectée. Si cette condition n'est pas respectée, d'autres instructions, placées entre Sinon et FinSi, seront exécutées à la place. On appelle cette structure la **structure conditionnelle avec alternative.**

EXEMPLES

Voici une variation du programme demande_age déjà vu précédemment.

```
Algorithme demande_age_2
Variable
    age: entier
Début
    Écrire "Quel âge avez-vous ?"
    Lire age
    Si age ≥ 18
            Écrire "Vous êtes majeur"
            Écrire "Bienvenu"
    Sinon
            Écrire "Vous êtes mineur"
            Écrire "Au revoir"
    FinSi
Fin
```

Voici ce qu'il se passe, si l'utilisateur dit qu'il a 22 ans :

```
Quel âge avez-vous ?
>>>22
Vous êtes majeur
Bienvenu
Au revoir
```

Voici ce qu'il se passe, si l'utilisateur dit qu'il a 15 ans :

```
Quel âge avez-vous ?
>>>15
Vous êtes mineur
Au revoir
```

À VOUS DE JOUER 10

1. Écrivez l'algorithme d'un programme pour ordinateur, simulant un lancer de dés à 6 faces. L'ordinateur devra créer un nombre aléatoire entre 1 et 6 (avec la fonction aléatoire). Ensuite, le programme demandera à l'utilisateur de deviner ce nombre. Si le joueur a trouvé le bon nombre, l'ordinateur affichera « Gagné! », sinon l'ordinateur affichera « Perdu! »

2. Écrivez l'algorithme d'un programme pour ordinateur. L'ordinateur devra demander à l'utilisateur sa moyenne en SI, puis lui dire « Bon travail » s'il a plus de 14/20. S'il a moins de 14/20, l'ordinateur lui dira « Il faut travailler davantage ».

STRUCTURE SI... SINONSI...

La structure conditionnelle Si...SinonSi... permet d'indiquer à l'ordinateur qu'il doit exécuter certaines instructions, placées entre Si et SinonSi uniquement si une condition est respectée. Si cette condition n'est pas respectée, on peut alors lui proposer une seconde condition : dans le cas où cette seconde condition est respectée, les instructions, placées entre SinonSi et FinSi, seront exécutées. On appelle cette structure la **structure conditionnelle avec alternative conditionnelle**.

EXEMPLES

Voici une variation du programme demande_age.

```
Algorithme demande_age_3
Variable
    age: entier
Début
    Écrire "Quel âge avez-vous ?"
    Lire age
    Si age ≥ 18
            Écrire "Vous êtes majeur"
    Sinon age < 12
            Écrire "Vous êtes un enfant"
            Écrire "Au revoir"
    FinSi
    Écrire « Au revoir »
Fin
```

Voici ce qu'il se passe, si l'utilisateur dit qu'il a 22 ans :

```
Quel âge avez-vous ?
>>>22
Vous êtes majeur
```

Voici ce qu'il se passe, si l'utilisateur dit qu'il a 15 ans :

```
Quel âge avez-vous ?
>>>15
Au revoir
```

Enfin, voici ce qu'il se passe, si l'utilisateur dit qu'il a 10 ans :

```
Quel âge avez-vous ?
>>>10
Vous êtes un enfant
```

Combinaisons de structures conditionnelles

Il est possible :

- d'ajouter autant de SinonSi que l'on souhaite ;
- d'ajouter un Sinon à la fin d'une structure conditionnelle avec alternative conditionnelle.

EXEMPLES

Voici encore une autre variation du programme demande_age, avec une structure conditionnelle complexe.

```
Algorithme demande_age_4
Variable
    age: entier
Début
    Écrire "Quel âge avez-vous ?"
    Lire age
    Si age ≥ 18
            Écrire "Vous êtes majeur"
    Sinon Si age ≥ 15
            Écrire "Vous êtes bientôt adulte"
    Sinon Si age ≤ 12
            Écrire "Vous êtes un adolescent"
```

```
    Sinon
        Écrire "Vous êtes un enfant"
    FinSi
    Écrire « Au revoir »
Fin
```

Dans le programme ci-dessus, on a une seule structure conditionnelle, qui va de Si à FinSi.
Il doit toujours y avoir une condition après un Si ou un SinonSi. En revanche, il n'y en a pas après un Sinon.
Il ne peut y avoir qu'un seul Sinon dans une structure conditionnelle, et il est toujours à la fin.

À VOUS DE JOUER 11

1. Écrivez l'algorithme d'un programme pour ordinateur, simulant un lancer de dés à 6 faces. L'ordinateur devra créer un nombre aléatoire entre 1 et 6 (avec la fonction aléatoire). Ensuite, le programme demandera à l'utilisateur de deviner ce nombre. Si le joueur a trouvé le bon nombre, l'ordinateur affichera « Gagné! ». Si le joueur s'est trompé, l'ordinateur affichera « Perdu, c'est trop petit! » ou « Perdu, c'est trop grand! » selon le cas. L'algorithme ne comportera qu'une seule structure conditionnelle.

2. Écrivez l'algorithme d'un programme pour ordinateur. L'ordinateur devra demander à l'utilisateur sa moyenne en SI, puis lui dire « Il faut travailler davantage » s'il a moins de 12, « C'est pas mal » s'il a entre 12 et 14, « Bon travail » s'il a entre 14 et 16, et « Très bon travail », s'il a plus de 16. L'algorithme ne comportera qu'une seule structure conditionnelle.

IMBRICATIONS

Il est possible de construire des structures conditionnelles plus élaborées en plaçant des structures conditionnelles à l'intérieur d'autres structures conditionnelles. On parle d'imbrication de structures conditionnelles.

EXEMPLES

Voici une dernière variation du programme demande_age, avec une imbrication de structures conditionnelles.

```
Algorithme demande_age_5
Variable
    age: entier

Écrire "Quel âge avez-vous ?"
    Lire age
    Si age ≥ 18
        Écrire "Vous êtes majeur"
    Sinon Si age ≥ 15
```

```
Début
    Écrire "Quel âge avez-vous ?"
    Lire age
    Si age ≥ 18
            Si age ≥ 60
                    Écrire "Vous êtes une personne âgée"
            Sinon age ≤ 12
                    Écrire "Vous êtes un adulte"
            Fin Si
    Sinon
            Si age ≥ 12
                    Écrire "Vous êtes un adolescent"
            Sinon
                    Écrire "Vous êtes un enfant"
              Fin Si
    Fin Si
    Écrire "Au revoir"
Fin
```

À VOUS DE JOUER 12

1. Refaites l'exercice 21 en utilisant une imbrication de structures conditionnelles.

2. Refaites l'exercice 22 en utilisant une imbrication de structures conditionnelles.

En plus des structures conditionnelles, un second type de structure existe : ce sont les **structures itératives**. Itérative vient du verbe « itérer », qui signifie « faire à plusieurs reprises ». Les structures itératives, qu'on appelle aussi des **boucles**, permettent d'indiquer à l'ordinateur qu'il doit exécuter des instructions à plusieurs reprises.

Comme avec les structures conditionnelles, il est tout à fait possible de faire des imbrications de structures itératives. On peut aussi imbriquer des structures conditionnelles dans des structures itératives, et inversement.

BOUCLE TANTQUE...

La boucle TantQue... permet d'indiquer à l'ordinateur qu'il doit exécuter une série d'instructions, tant qu'une condition est respectée. On appelle cette structure, **boucle avec condition de bouclage**.

EXEMPLES

Voici un algorithme, qui demande à un ordinateur d'afficher la liste des carrés des nombres entiers, en partant de 1, tant que l'utilisateur ne répond pas « non ».

```
Algorithme calcul_carré_1
Variable
    réponse: chaîne de caractères
    base, carré: entier
Début
réponse = "oui"
base = "1"
    TantQue réponse <> "non"
            carré ← base*base
            Écrire "Le carré de", base, "vaut", carré
            base ← base +1
            Écrire "Souhaitez-vous connaître le carré suivant ?"
            Lire réponse
    Fin TantQue
    Écrire "Au revoir"
Fin
```

Voici un exemple d'utilisation de ce programme :

```
Le carré de 1 vaut 1
Souhaitez-vous connaître le carré suivant ?
>>>oui
Le carré de 2 vaut 4
Souhaitez-vous connaître le carré suivant ?
>>>oui
```

```
Le carré de 3 vaut 9
Souhaitez-vous connaître le carré suivant ?
>>>non
Au revoir
```

Même si l'utilisateur répond autre chose que "oui", l'ordinateur fera le prochain calcul. Ce sera le cas, tant que l'utilisateur n'entrera pas "non". La condition après le TantQue est du même type que les conditions après un Si ou SinonSi.

À VOUS DE JOUER ⓭

1. Écrivez l'algorithme d'un programme pour ordinateur. Ce programme dit « Bonjour » à l'utilisateur. Ensuite, le programme tourne en boucle en disant « J'attends une réponse », tant que l'utilisateur n'a pas répondu « Bonjour » à son tour.

2. Écrivez l'algorithme d'un programme pour ordinateur. Au tout début du programme l'ordinateur choisira un nombre aléatoire entre 5 et 15, puis il calculera le carré de ce nombre. Ensuite, il demandera à l'utilisateur de trouver le carré, et de lui donner. Tant que l'utilisateur n'aura pas trouvé la bonne réponse, l'ordinateur lui dira qu'il se trompe. Quand l'utilisateur aura trouvé la bonne réponse, l'ordinateur lui dira qu'il a gagné.

3. Améliorez le programme précédent, en indiquant à chaque essai à l'utilisateur si le nombre à trouver est plus grand ou plus petit que sa proposition.

4. Améliorez le programme précédent, en ajoutant un compteur de nombres d'essais avant de trouver le bon nombre. À la fin, l'ordinateur indiquera à l'utilisateur combien d'essais ont été nécessaires pour trouver la bonne réponse.

BOUCLE INFINIE

Il faut choisir la condition après le TantQue avec soin, car sinon il est possible que l'ordinateur tombe dans une boucle infinie, dont il ne pourra jamais sortir. Parfois, on peut volontairement vouloir créer une boucle infinie, pour faire tourner un programme en boucle (par exemple une horloge) : on crée alors une boucle TantQue... avec pour condition vrai.

```
Algorithme algo_bizarre_2
Début
    TantQue vrai
            Écrire "Un tour de plus"
    Fin TantQue
Fin
```

La condition vrai est toujours validée, d'où la boucle infinie !

BOUCLE POUR…

La boucle Pour... permet d'indiquer à l'ordinateur qu'il doit exécuter une série d'instructions, un certain nombre de fois, qu'on lui précise au début de la boucle. On appelle cette structure, **boucle à itérations déterminées**.

EXEMPLES

Voici un algorithme qui calcule et affiche les carrés des nombres entiers 12 à 16.

```
Algorithme calcul_carré_1
Variable
    nombre, carré: entier
Début
    Pour nombre allant de 12 à 16 // 5 tours de boucle
            carré ← nombre*nombre
            Écrire "Le carré de", nombre, "vaut", carré
    Fin Pour
    Écrire "Au revoir"
Fin
```

Ici on a 5 tours de boucles, car nombre vaudra 12, puis 13, 14, 15 et 16. On dit que nombre est l'indice de boucle de la boucle Pour. On utilise souvent la lettre i, pour indice ou itération, comme nom de variable de l'indice de boucle.

Voici un exemple d'utilisation de ce programme :

```
Le carré de 12 vaut 144
Le carré de 13 vaut 169
Le carré de 14 vaut 196
Le carré de 15 vaut 225
Le carré de 16 vaut 256
Au revoir
```

Le nombre de tours peut être variable, et dépendre des circonstances ou de valeurs entrées par l'utilisateur.

EXEMPLES

Voici une variation du programme précédent, où l'utilisateur choisit les carrés qui seront calculés.

```
Algorithme calcul_carré_2
Variable
    Nombre_min, nombre_max, nombre, carré: entier
Début
    Écrire "Quel est le plus petit nombre ?"
        Lire nombre_min
    Écrire "Quel est le plus grand nombre ?"
        Lire nombre_max
    Pour nombre allant de nombre_min à nombre_max
        carré ← nombre*nombre
        Écrire "Le carré de", nombre, "vaut", carré
    Fin Pour
    Écrire "Au revoir"
Fin
```

Voici un exemple d'utilisation de ce programme :

```
Quel est le plus petit nombre ?
>>>3
Quel est le plus grand nombre ?
>>>5
Le carré de 3 vaut 9
Le carré de 4 vaut 16
Le carré de 5 vaut 25
Au revoir
```

À VOUS DE JOUER 14

1. Écrivez l'algorithme d'un programme pour ordinateur. Ce programme demande à l'utilisateur combien de fois l'utilisateur veut que le programme écrive « merci ». Ensuite, le programme écrit « merci », le nombre de fois choisi par l'utilisateur.

.....

.....

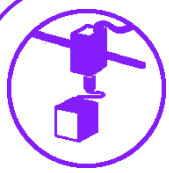
.....

.....

2. Écrivez l'algorithme d'un programme pour ordinateur. Ce programme demande 10 nombres à l'utilisateur, puis affiche la somme de ces 10 nombres. Le programme devra faire moins de 15 lignes et comporter au maximum 3 variables.

05 ALGORITHMIQUE
Pseudo-code avec un produit intelligent

Nous avons vu comment écrire des algorithmes pour des ordinateurs. Le pseudo-code permet aussi d'écrire des algorithmes destinés à des produits intelligents plus variés, comme des robots, ou n'importe quel type de produit autonome, par exemple. Les règles d'écriture sont les mêmes, mais on peut utiliser quelques éléments en plus, notamment en ce qui concerne les entrées et sorties. En effet, un produit intelligent va avoir des entrées et des sorties bien plus variées qu'un ordinateur.



EXEMPLES

Pour un drone, les capteurs de distance (obstacle, distance du sol), le module GPS et les caméras sont des composants d'entrées. Les voyants lumineux sont des composants de sortie. Les moteurs ne sont pas vraiment des composants de sortie, mais ils peuvent être commandés grâce à des signaux de commande : on les considérera comme des sorties lorsqu'on écrit un algorithme.

SORTIE : INSTRUCTION D'ÉCRITURE

Avec un produit intelligent, nous utiliserons des instructions de sortie plus spécifiques et variées que l'instruction Écrire <information>. Ces instructions seront liées aux composants contrôlés. Par exemple, on aura :

- Allumer LED
- Éteindre LED
- Faire_tourner Moteur avec sens=trigonométrique, vitesse=20tr/min

EXEMPLES

Voici une variation du programme précédent, où l'utilisateur choisit les carrés qui seront calculés.

```
Algorithme déplacement_robot_1
Début
    Allumer LED_A
    Allumer LED_B
    Faire tourner Moteur_A avec sens = horaire, vitesse = 4tr/min
Fin
```

La syntaxe générale sera <Action><composant> avec <paramètres> avec :

- <Action>, un verbe d'action lié à l'action réalisée par le composant ;
- <composant>, le composant qui recevra l'information ou l'ordre ;
- avec <paramètres> est une partie facultative, qui comporte les éventuelles informations complémentaires nécessaires pour paramétrer l'action.

Parfois, un produit intelligent possède un écran/afficheur. Dans ce cas, on pourra utiliser l'instruction Écrire <information> sur <composant>, qui fonctionne comme Écrire <information> en précisant sur quel composant l'information sera écrite.

À VOUS DE JOUER 15

1. Écrivez l'algorithme d'un programme pour un robot qui lui dit d'allumer sa première LED_A, puis de faire tourner Servomoteur_C à 90° et ensuite d'éteindre sa LED_A.

.....

.....

.....

.....

.....

.....

2. Écrivez l'algorithme d'un programme pour un robot qui lui dit d'écrire le message "Bonjour" sur son composant Afficheur_LCD, puis le numéro 34 sur son composant d'affichage Afficheur_7_Segments.

.....

.....

.....

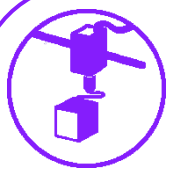
.....

.....

INSTRUCTION D'ENTRÉE

De nouveau, pour les produits intelligents, nous utiliserons des instructions d'entrée plus spécifiques et variées que Lire <variable>. Ces instructions seront liées aux composants d'acquisition. Par exemple, on aura :

- Mesurer température avec capteur_température
- Mesurer distance_obstacle avec capteur_distance
- Détecter présence_humain avec détecteur_infra-rouge



EXEMPLES

L'algorithme suivant, destiné à un thermomètre, lui indique de mesurer la température avec son capteur TMP36 (c'est un capteur de température), puis de faire une pause de 20s, et enfin d'afficher la température sur son afficheur-LCD.

```
Algorithme affiche_température
Variable
    température: réel
Début
    Mesurer température avec TMP36
    Pause 20s
    Afficher température avec Afficheur_LCD
Fin
```

Rappel : la fonction Pause <durée> permet, comme son nom l'indique, de demander au produit de faire une pause de la durée indiquée.

La syntaxe générale sera <Action><variable> avec <composant> avec :

- <Action>, un verbe d'action lié à l'action réalisée par le composant ;
- <variable>, la variable avec laquelle la valeur récupérée sera stockée ;
- <composant>, le composant qui réalisera l'acquisition.

FONCTIONS

On pourra utiliser tout un tas de fonctions, spécifiques à chaque produit, qui en général seront indiquées quand on a besoin de les utiliser.
Par exemple, on peut imaginer que :

- pour un robot, on aura des fonctions avancer(<vitesse>), reculer(<vitesse>), tourner(<angle>), s'arrêter(), etc.
- pour un drone on aura des fonctions avancer(<vitesse>), décoller(), atterrir(), etc.

```
Algorithme affiche_température
Début
    avancer(15)
    Pause 2s
    tourner(45)
    avancer(10)
    Pause 1s
    s'arrêter(15)
Fin
```

Pause met le programme en pause, mais pas les moteurs du robot. Cela signifie que durant les pauses de 2s et 1s du programme, le robot avance.

À VOUS DE JOUER 16

1. Écrivez l'algorithme d'un programme pour un robot qui lui dit de se déplacer en avançant pendant 1s, puis de tourner de 180°, et enfin de reculer pendant 3s avant de s'arrêter. Tous les mouvements se feront à 7km/h. Vous pouvez utiliser les fonctions avancer(<vitesse>), reculer(<vitesse>), tourner(<angle>), s'arrêter(), avec les vitesses en km/h et les angles en degrés.

EXERCICES DE SYNTHÈSE

Pour terminer, voici quelques algorithmes à écrire en vous servant des structures conditionnelles et des structures itératives. Pour ces programmes, vous pouvez utiliser les fonctions avancer(<vitesse>), reculer(<vitesse>), tourner(<angle>), s'arrêter(), avec les vitesses en km/h et les angles en degrés. Le robot utilisera un Capteur_ultrason (pour détecter un obstacle), qui lui donne, en cm, la distance avec le prochain obstacle en face de lui.

Tous ces programmes seront destinés à un robot. Ces programmes fonctionneront tant que le robot sera en marche, on va donc les mettre dans une boucle infinie, grâce à une boucle infinie « TantQue vrai ».

```
Algorithme déplacement_robot
Début
    TantQue vrai // début boucle infinie
            <instruction>
    FinTantQue // fin boucle infinie
Fin
```

On met donc volontairement le robot dans une boucle infinie. <instructions> sera remplacée par plusieurs instructions permettant d'expliquer au robot comment il doit se déplacer.

À VOUS DE JOUER 17

1. Écrivez l'algorithme qui demande au robot de reculer si un obstacle se trouve à moins de 20cm devant lui, ou d'avancer dans le cas contraire. Tous les mouvements se feront à 6 km/h.

2. Écrivez l'algorithme qui demande au robot d'avancer à 12 km/h s'il ne trouve aucun obstacle à moins de 50cm. Si un obstacle est plus proche, alors le robot l'évitera avec l'une des manœuvres suivantes : si l'obstacle est à moins de 20cm, le robot fera un tour à 180°, si l'obstacle est à plus de 20cm, le robot tournera de 90°.

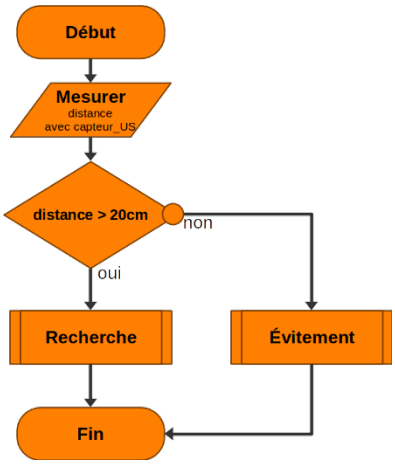
3. Écrivez l'algorithme qui demande au robot d'avancer à 10km/h pendant 4s, puis de réaliser 20 fois de suite cette séquence de déplacement : tourner de 90°, avancer à 3km/h pendant 4s. Une fois la séquence de mouvement terminée, le robot s'arrête durant 10s.

ALGORIGRAMME

L'ESSENTIEL

Un algorigramme est un diagramme qui représente graphiquement un algorithme.

Avec un **algorigramme**, les instructions et structures de contrôle sont représentées par des **symboles graphiques**, plutôt que par du texte comme en pseudo-code. Chaque type d'instruction ou de structure du pseudo-code possède son propre symbole graphique, et il est facile d'apprendre à dessiner des algorigrammes lorsqu'on connaît déjà le pseudo-code.



EXEMPLES

Voici l'algorigramme d'un robot, qui mesure la distance avec un éventuel obstacle, puis en fonction de la distance avec l'obstacle, se lance dans une opération de recherche d'un objet, ou dans une manœuvre d'évitement. Nous verrons la signification précise de chaque bloc juste après.

Les algorigrammes sont aussi appelés "organigrammes de programmation". On trouve aussi le nom de "diagrammes de flux", ou flow charts en anglais.

Comparaison avec le pseudo-code

Vous devez savoir lire des algorithmes écrits en pseudo-code ou dessinés sous forme d'algorigrammes, car les deux se retrouvent fréquemment dans de la documentation technique et dans des sujets d'examens. Il est aussi courant d'avoir des exercices dans lesquels on vous donne un algorithme en pseudo-code, qu'il faut traduire en algorigramme, ou inversement. Si vous avez le choix d'utiliser l'un ou l'autre lors d'un projet, voici quelques remarques pour orienter votre choix :

- Les algorigrammes permettent de présenter visuellement un algorithme, ils sont donc plus adaptés pour être utilisés dans une présentation orale ou un compte-rendu.
- Les algorigrammes sont souvent utilisés pour représenter le fonctionnement des produits intelligents, comme les robots, les drones, les systèmes industriels, etc.
- Le pseudo-code est préféré pour les algorithmes destinés à des ordinateurs, car le pseudo-code ressemble à un langage de programmation. Il sera assez facile de transformer du pseudo-code en code-source.
- Les algorigrammes deviennent difficiles à lire et à utiliser avec un algorithme complexe et long : dans ce cas, on préférera le pseudo-code.

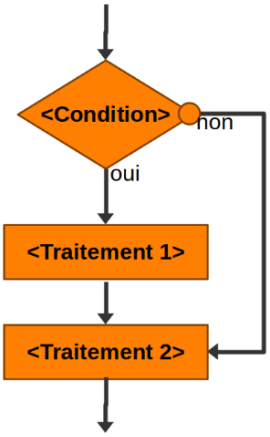
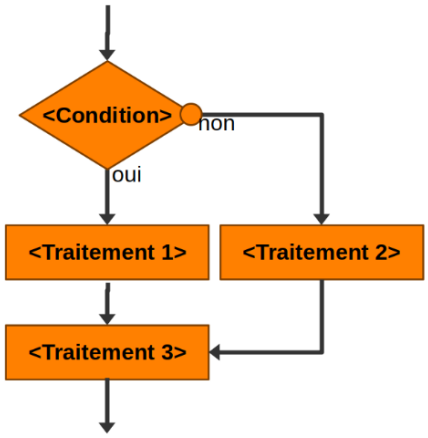
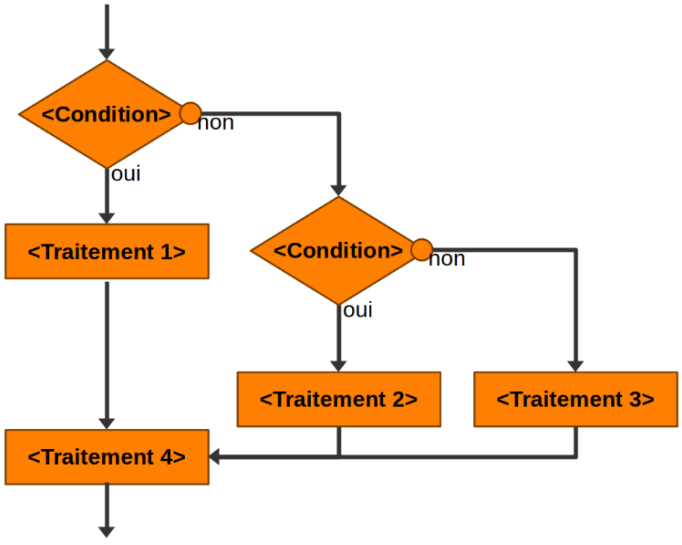
SYMBOLES

Le nombre de **symboles** pour écrire un algorigramme est limité : il n'y en a que 7. Néanmoins, ces 7 symboles sont suffisants pour représenter n'importe quel algorithme.

Nom	Symbole	Rôle	Équivalent en pseudo-code
Bloc de début	**Début** <nom>	Un unique bloc de début doit être placé au tout début de l'algorigramme. La lecture de l'algorigramme commence par ce bloc. Le nom de l'algorithme peut être ajouté (facultatif).	Début
Bloc de fin	**Fin** <nom>	Un unique bloc de fin doit être placé à la fin de l'algorigramme.Toutes les branches de l'algorigramme doivent rejoindre ce bloc. Le nom de l'algorithme peut être ajouté (facultatif).	Fin
Bloc d'entrée	**Lire** <variable> ou **<Action>** <variable> avec <composant>	Le bloc d'entrée indique qu'une information entre dans le composant de traitement, en provenance d'un composant d'acquisition. La première version (**Lire**) est destinée aux ordinateurs, la seconde aux produits intelligents.	Instruction d'entrée
Bloc de sortie	**Écrire** <information> ou **<Action>** <composant> avec <paramètres>	Le bloc de sortie indique qu'une information sort du composant de traitement, pour aller vers un composant de restitution. La première version (**Écrire**) est destinée aux ordinateurs, la seconde aux produits intelligents.	Instruction de sortie
Bloc de traitement	**<Traitement>**	Le bloc de traitement sert à représenter toutes les instructions qui ne sont pas des entrées ou des sorties (affectation, calcul, pause, fonction, etc.).	Autres instructions : affectation, calcul, pause, fonctions, etc.
Bloc de procédure	**<Procédure>**	Le bloc de procédure, aussi appelé bloc de sous-programme, sert à inclure un sous-algorigramme dans un algorigramme principal.	Nous ne l'avons pas vu, mais il est possible de créer des procédures en pseudo-code
Bloc de choix	**<Condition>** non oui	Le bloc de choix indique qu'une décision doit être prise, en fonction d'une condition, et qu'il y a deux options possibles (oui=vrai, non=faux). Le bloc de choix sépare l'algorigramme en deux branches qui devront se rejoindre.	Lignes Si…, Sinon… et/ou SiSinon… d'une structure conditionnelle
Annotation	annotation	L'annotation permet d'ajouter un commentaire sur un bloc, ou sur une partie de l'algorigramme, pour en faciliter sa compréhension.	Commentaire
Flèche		La flèche indique le passage d'un bloc vers un autre. Le passage se fait toujours dans le sens de la flèche	Retour à la ligne

STRUCTURES CONDITIONNELLES

Le bloc de choix ne suffit pas pour créer une **structure conditionnelle**. Pour créer une structure conditionnelle, il faut en plus placer les flèches et les autres blocs d'une certaine manière. Voici comment réaliser les trois structures de bases en algorigramme :

Structure	Algorigramme	Pseudo-code
Si...		**Si <Condition> <Traitement 1> FinSi <Traitement 2>**
Si...Sinon...		**Si <Condition> <Traitement 1> Sinon <Traitement 2> FinSi <Traitement 3>**
Si...SinonSi...Sinon...		**Si <Condition> <Traitement 1> SinonSi <Traitement 2> Sinon <Traitement 3> FinSi <Traitement 4>**

Comme avec le pseudo-code, on peut réaliser des combinaisons et des imbrications sans limite de taille.

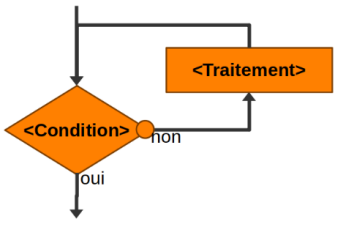
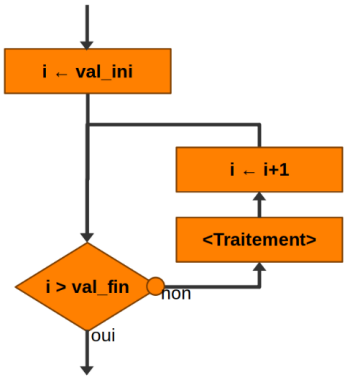
À VOUS DE JOUER 18

1. Dessiner l'algorigramme d'un programme pour ordinateur, simulant un lancer de dés à 6 faces. L'ordinateur devra créer un nombre aléatoire entre 1 et 6 (avec la fonction aléatoire, qui peut s'utiliser dans un bloc de traitement). Ensuite, le programme demandera à l'utilisateur de deviner ce nombre. Si le joueur a trouvé le bon nombre, l'ordinateur affichera « Gagné! ». Si le joueur s'est trompé, l'ordinateur affichera « Perdu, c'est trop petit! » ou « Perdu, c'est trop grand! » selon le cas.

2. Dessiner l'algorigramme de gestion d'obstacle d'un robot, qui lui demande de tourner à 90° si un obstacle se trouve à moins de 20cm devant lui, ou d'avancer dans le cas contraire. Vous pouvez utiliser les procédures Avancer et Virage_90. La mesure de distance se fera avec un Capteur_ultrason.

STRUCTURES ITÉRATIVES

De la même manière, pour réaliser les **structures itératives** TantQue… et Pour…, il faut placer les flèches et les autres blocs d'une certaine manière. Voici comment réaliser les deux structures itératives en algorigramme :

Structure	Algorigramme	Pseudo-code
TantQue...		**TantQue <Condition> <Traitement> FinTantQue**
Pour...		**Pour i allant de val_ini à val_fin <Traitement > FinPour**

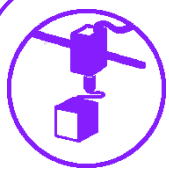
À VOUS DE JOUER 19

1. Dessiner l'algorigramme d'un programme pour ordinateur. Ce programme dit « Bonjour » à l'utilisateur. Ensuite, le programme tourne en boucle en disant « J'attends une réponse », tant que l'utilisateur n'a pas répondu « Bonjour » à son tour.

2. Dessiner l'algorigramme d'un programme pour ordinateur. Ce programme demande à l'utilisateur combien de fois l'utilisateur veut que le programme écrive « merci ». Ensuite, le programme écrit « merci », le nombre de fois choisi par l'utilisateur.

PROCÉDURE

Les blocs de **procédure** sont un moyen pratique de ne pas trop allonger un algorigramme. Un bloc de procédure représente en fait une partie de l'algorigramme, qu'on a retirée de l'algorigramme principal, et remplacée par un seul bloc.

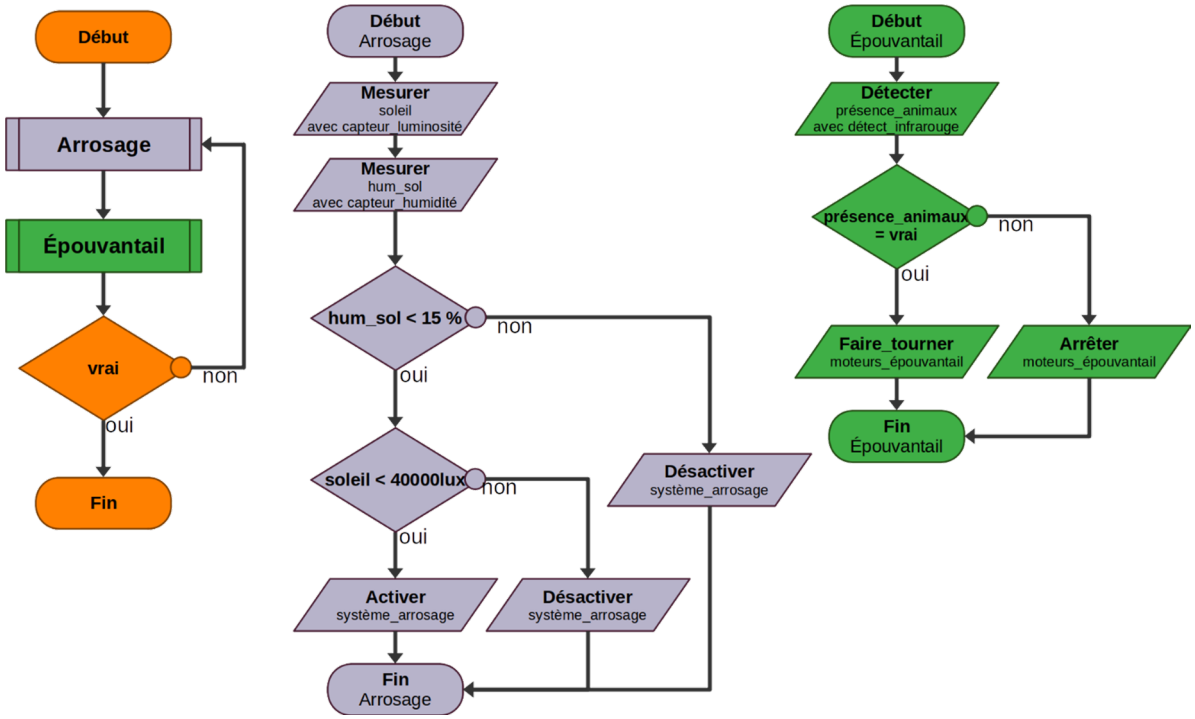


EXEMPLES

Soit un produit pour jardinier. Ce produit possède deux fonctions :

- il gère l'arrosage, en l'activant si le sol est sec et qu'il ne fait pas trop soleil ;
- il éloigne les animaux présents dans le jardin en faisant bouger un épouvantail.

Pour éviter d'avoir un algorigramme trop long, on a utilisé des procédures pour représenter l'algorigramme principal. Les deux procédures ont ensuite été détaillées, avec leurs propres algorigrammes :



Vous n'aurez pas à savoir dessiner vous-même un algorigramme avec des procédures, mais vous devez être capable d'en lire un. On peut faire la même chose en pseudo-code : il suffit d'écrire Procédure <nom> dans l'algorithme principal, puis de créer un algorithme à part pour décrire la procédure.

LE TEMPS DU BILAN

- Un algorithme est une suite d'instructions, qui sont exécutées par un produit intelligent, pour réaliser une action complexe.
- Un algorithme peut être écrit en pseudo-code ou sous la forme d'un algorigramme.
- L'algorithme est ensuite transformé en programme, à l'aide d'un langage de programmation graphique ou en ligne de code.

- Le pseudo-code permet de concevoir et écrire un algorithme sous forme textuelle.
- L'instruction d'écriture Écrire <information> indique à l'ordinateur qu'il doit envoyer une information vers un de ses composants de sortie, habituellement l'écran.
- Pour manipuler la mémoire d'un ordinateur, dans un algorithme, on se sert des variables. Une variable possède un nom, un type et une valeur.
- L'instruction d'entrée Lire <variable> indique à l'ordinateur qu'il doit récupérer une information venant de l'utilisateur, habituellement via le clavier.
- L'instruction d'affectation, <variable> ← <expression> permet de donner la valeur de <expression> à une variable.
- Les opérateurs disponibles pour écrire des expressions sont +, -, *, /, // et %. Il est aussi possible d'utiliser des parenthèses.
- Les fonctions permettent de faire réaliser des tâches complexes à l'ordinateur. Les fonctions courantes sont max, min, moyenne, aléatoire, racine_carrée, puissance, longueur et concaténer.

- Une structure conditionnelle est une structure de contrôle permettant de faire exécuter à l'ordinateur des instructions différentes en fonction du contexte.
- Les structures conditionnelles disponibles sont les structures Si..., Si...Sinon... et Si...SinonSi...
- La condition d'une structure conditionnelle peut être écrite avec les opérateurs de comparaisons >, ≥, <, ≤, = et <>.
- Il est possible de réaliser des combinaisons et des imbrications de structures conditionnelles.

- Les structures itératives, qu'on appelle aussi des boucles, permettent d'indiquer à l'ordinateur qu'il doit exécuter des instructions à plusieurs reprises.
- La boucle TantQue... permet d'indiquer à l'ordinateur qu'il doit exécuter une série d'instructions, tant qu'une condition est respectée. On appelle cette structure une boucle avec condition de bouclage.
- La boucle Pour... permet d'indiquer à l'ordinateur qu'il doit exécuter une série d'instructions, un certain nombre de fois, qu'on lui précise au début de la boucle. On appelle cette structure une boucle à itérations déterminées.

- Un algorigramme est un diagramme qui représente graphiquement un algorithme.

Vous pouvez maintenant faire et envoyer le devoir n°1