**Algorithmique**

**I – Qu’est-ce qu’un algorithme ?**

Un algorithme est un processus permettant d’aboutir de façon automatique au résultat d’une action ou à la résolution d’un problème en un nombre fini d’étapes.

* Voici quelques exemples :
* On souhaite préparer des biscottes beurrées avec de la confiture. On est obligé pour cela de procéder comme suit :

Prendre les biscottes et les poser sur la table

Etaler du beurre sur les biscottes

Etaler de la confiture sur les biscottes beurrées

Il est bien évident qu’on ne peut pas intervertir les étapes !

* Voici un algorithme qui décrit la construction d’un losange dont une diagonale est [AB] :

On se donne deux points A et B du plan

Tracer le cercle de centre A et passant par B

Tracer le cercle de centre B et passant par A

Nommer C et D les points d’intersection de ces cercles

Construire le polygone ADBC

* Voici un programme de calcul vu au collège :

**1.** Choisir un nombre

**2.** Le diviser par 3

**3.** Enlever 2 au résultat

**4.** Multiplier le résultat obtenu par 5

**5.** Donner le résultat obtenu

* Ecrire un algorithme consiste à donner une méthode détaillée décrivant toutes les étapes d’une tache à accomplir. Un algorithme est composé de trois phases :
* Les **entrées** : l’élément (ou les éléments) dont on part ;
* Le **traitement des données** : les actions à effectuer sur ces éléments ;
* Les **sorties** : le (ou les) résultat(s) obtenus.
* Exemples : déterminer dans les exemples précédents, les entrées, le traitement, les sorties.
* Ces trois phases, notamment la phase de traitement, se structurent à l’aide d’instructions :
* Les **instructions de base** : calculs à partir des données ;
* Les **instructions conditionnelles** : qui dépendent de la réponse à un test ;
* Les **boucles** et **itérateurs** : actions à effectuer un certain nombre de fois.

Ces instructions seront étudiées dans les paragraphes suivants.

* Les algorithmes utilisent des **variables**, qui sont des sortes de « boites » contenant un nombre ou une liste de nombres. Le contenu de la variable peut se modifier au cours de l’algorithme.
* Un algorithme se traduit enfin sous la forme d’un programme, afin d’être mis en œuvre à l’aide d’une calculatrice, d’un tableur ou d’un ordinateur. Il faut utiliser le « langage », ou la syntaxe, adapté à l’outil utilisé (TI 82…, Casio Graph 35…, logiciel Xcas, logiciel Algobox…)

**II – Les instructions de base**

On souhaite calculer le volume d’un cylindre de hauteur H et de rayon du disque de base R donnés.

Ce calcul sera effectué pour dix couples (H ; R). Présenter les résultats dans un tableau.

* **Calculs**

Définir le (ou les) calcul(s) à effectuer : chaque couple (H ; R) étant donné, on calcule le volume πR²H.

Le calcul à effectuer pour dix couples donnés incite à proposer de l’automatiser pour ne pas avoir à retaper, pour chaque couple donné, une séquence de calcul qui peut être longue.

On va donc écrire un algorithme permettant d’automatiser ces calculs, puis le programmer sur une calculatrice.

* **Algorithme**

Il faut faire apparaitre les trois types d’instructions :

* des instructions d’entrée des données : quelles valeurs donne-t-on, pour ce calcul, à H et R ?
* des instructions de calcul à effectuer : on calcule le volume.
* des instructions d’affichage de résultats.

Ainsi, on écrira l’algorithme sous la forme :

|  |  |
| --- | --- |
| Début  Demander la valeur de H  Demander la valeur de R  Calculer πR²H et placer le résultat dans la variable V  Afficher la valeur de V  Fin | Variables :  H hauteur, R rayon, V volume |

* **Programme**
* **Ecriture du programme**

Voici le programme calqué sur l’algorithme ci-dessus :

|  |  |
| --- | --- |
| **Calculatrices TI 82Stats à TI 84** | **Calculatrices Casio Graph 35 à 65** |
| NAME = CYLINDRE  Prompt H, R  π\*R^2\*H→V  Disp V (ou Disp "V=",V) | Program Name[CYLINDRE]  "H=" ? →H : "R=" ? →R  π\*R^2\*H→V  V (ou "V=" :V ) |

Sur une calculatrice, le programme peut être amélioré ou simplifié (Voir liste des touches)

* **Programmer la calculatrice**

Créer et donner un nom au programme :

|  |  |
| --- | --- |
| prgm ► ► (NOUV) entrer  C Y L I N D R E | Icône PRGM **F3** (New)  C Y L I N D R E |

Ecrire le programme (Voir liste des touches à utiliser au paragraphe **V-**).

* **Utiliser le programme**

Exécuter le programme :

|  |  |
| --- | --- |
| prgm EXEC entrer entrer | Icône PRGM **F1** (EXE) |

En demandant l’exécution du programme, la machine va :

Afficher H = ? et l’utilisateur donne une valeur à H qu’il valide par entrer ou EXE ;

Afficher R = ? et l’utilisateur donne une valeur à R qu’il valide par entrer ou EXE ;

La machine calcule alors la valeur de V, puis l’affiche à l’écran.

* Choisir une liste de dix couples (H ; R), faire fonctionner le programme, et donner les résultats dans le tableau suivant.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H |  |  |  |  |  |  |  |  |  |  |
| R |  |  |  |  |  |  |  |  |  |  |
| V |  |  |  |  |  |  |  |  |  |  |

**III – Les instructions conditionnelles : Si…, alors…, sinon…**

On souhaite tester si un triangle ABC est rectangle en A.

* **Calculs et tests**

Savoir si le triangle ABC est rectangle en A nécessite de calculer d’une part AB²+AC², d’autre part BC², puis de comparer les deux.

* **Algorithme**

On va utiliser les instructions de base, et ajouter un **test** permettant de décider si le triangle est rectangle ou non.

|  |  |
| --- | --- |
| Début  Demander la valeur de AB et la placer dans la variable c  Demander la valeur de AC et la placer dans la variable b  Demander la valeur de BC et la placer dans la variable a  Si c² = a²+b², écrire « le triangle est rectangle en A »  Sinon, écrire « le triangle n’est pas rectangle en A »  Fin du Si  Fin | Variables :  a, b, c longueurs des côtés |

* **Programme**

|  |  |
| --- | --- |
| **Calculatrices TI 82Stats à TI 84** | **Calculatrices Casio Graph 35 à 65** |
| Input "AB = ",C : Input "AC = ",B : Input "BC = ",A  If C²+B² =A²  Then  Disp "TRI RECTANGLE","EN A"  Else  Disp "TRI NON RECT","EN A"  End | "AB=" ? →C : "AC=" ? →B : "BC=" ? →A  If C²+B² =A²  Then "TRI RECTANGLE","EN A"  Else "TRI NON RECT","EN A"  IfEnd |

**Remarque :** dans un programme, les « retours à la ligne » sont imposés par les calculatrices car on ne doit pas rajouter d’espace manuellement après une instruction. Ainsi, la TI ne mettant pas d’espace après **Then** impose d’aller à la ligne pour écrire les instructions suivantes.

* Faire fonctionner le programme pour les triangles ABC tels que :
* AB = 5, AC = 6, BC = 7 ;
* AB = 9, AC = 12, BC = 15 ;
* AB = 7,1, AC = 2,5, BC = 9,4 ;
* AB = 11,7, AC = 15,6, BC = 19,5 ;
* …

**IV – Les boucles et itérateurs**

**1) Boucle « Pour »**

On souhaite trouver tous les triangles rectangles dont les côtés sont des entiers consécutifs, le plus petit étant compris entre 1 et 10.

* **Calculs et tests**

Il s’agit de répéter 10 fois le travail réaliser au paragraphe « tests » avec pour longueurs de côtés (1 ; 2 ; 3), (2 ; 3 ; 4), (3 ; 4 ; 5),…, (10 ; 11 ; 12).

* **Algorithme**

En plus des instructions de base et du test, on va utiliser une **boucle**.

On sait que le plus petit côté, appelé par exemple c, vaudra d’abord 1, puis 2, puis 3,…et enfin 10.

Si on écrivait toutes les instructions, cela donnerait :

|  |  |
| --- | --- |
| Début  Mettre 1 dans c  Mettre c+1 dans b  Mettre c+2 dans a  Si c²+b² = a², écrire « le triangle est rectangle »  Sinon, écrire « le triangle n’est pas rectangle »  Fin du Si  Mettre 2 dans c  Mettre c+1 dans b  Mettre c+2 dans a  Si c²+b² = a², écrire « le triangle est rectangle »  Sinon, écrire « le triangle n’est pas rectangle »  Fin du Si | Mettre 3 dans c  Mettre c+1 dans b  Mettre c+2 dans a  Si c²+b² = a², écrire « le triangle est rectangle »  Sinon, écrire « le triangle n’est pas rectangle »  Fin du Si  …………….  Mettre 10 dans c  Mettre c+1 dans b  Mettre c+2 dans a  Si c²+b² = a², écrire « le triangle est rectangle »  Sinon, écrire « le triangle n’est pas rectangle »  Fin du Si  Fin |

On répète ainsi 10 fois les mêmes instructions, la seule différence étant la valeur de c. Pour éviter cette répétition, on utilise alors une **boucle « Pour »** qui ajoute automatiquement 1 à la variable c à chaque fois et s’arrête lorsque la variable c vaut 10.

|  |  |
| --- | --- |
| Début  Pour c allant de 1 à 10 par pas de 1  Mettre c+1 dans b  Mettre c+2 dans a  Si c²+b² = a², écrire « le triangle est rectangle »  Sinon, écrire « le triangle n’est pas rectangle »  Fin du Si  Fin du pour  Fin | Variables :  a, b, c longueurs des côtés |

* **Programme**

|  |  |
| --- | --- |
| **Calculatrices TI 82Stats à TI 84** | **Calculatrices Casio Graph 35 à 65** |
| EffEcr  For(C,1,10,1)  C+1→B :B+1→A  Disp A,B,C  If C²+B² =A²  Then  Disp "TRI RECTANGLE"  Pause  Else  Disp "TRI NON RECT"  Pause  End  End | For 1→C To 10 Step 1  C+1→B :B+1→A  A  B  C  If C²+B² =A²  Then "TRI RECTANGLE"  Else "TRI NON RECT"  IfEnd  Next |

**Remarque :** le dernier « 1 » de l’instruction For sur TI comme le « Step » sur Casio ne sont pas obligatoire avec un pas de 1.

* Faire fonctionner le programme.

**2) Boucle « Tant que »**

On souhaite maintenant trouver 4 triangles rectangles dont les côtés sont des entiers, les deux plus petits étant consécutifs. Contrairement à l’exemple précédent, on ne sait pas pour quelle valeur de c on va s’arrêter : l’utilisation d’une boucle « Pour » est alors impossible. On va utiliser une **boucle « Tant que »** et afficher les triplets qui donnent des triangles rectangles.

* **Calculs**

c et b seront les deux plus petits côtés, et a sera l’hypoténuse. On obtient a grâce au théorème de Pythagore, puis on teste pour savoir si a est un entier.

Initialisation des variables : n sert de compteur (pour compter le nombre de triangles) et c est la plus petite longueur

* **Algorithme**

|  |  |
| --- | --- |
| Début  Initialiser c à 1  Initialiser n à 0  Tant que n < 4  Mettre c+1 dans b  Mettre racine(b²+c²) dans a  Si a est entier, afficher a, b et c et augmenter n de 1  Fin du Si  Contrairement au Pour, le Tant que n’incrémente pas automatiquement c  Ajouter 1 à c  Fin du Tant que  Fin | Variables :  a, b, c longueurs des côtés  « compteur » n allant de 1 à 4  On répète les calculs tant que l’on n’a pas trouvé 4 triangles |

* **Programme**

|  |  |
| --- | --- |
| **Calculatrices TI 82Stats à TI 84** | **Calculatrices Casio Graph 35 à 65** |
| 1→C : 0→N  While N < 4  C+1→B  √(B²+C²)→A  If ent(A) = A  Then  Disp A,B,C  N+1→N  End  C+1→C  End | 1→C : 0→N  While N < 4  C+1→B  √(B²+C²)→A  If Int(A) = A  Then A  B  C  N+1→N  IfEnd  C+1→C  WhileEnd |

**Remarque :** s’il n’y avait pas au moins 4 triangles « solutions », le programme ne s’arrêterait jamais. Il est donc important, dans une boucle **Tant que**, de bien prévoir les conditions d’arrêt. Ici, on pourrait, pour plus de sécurité, ajouter c < 1000 dans les conditions, afin de ne tester que 999 triangles au maximum ! A la place de « Tant que n < 4 », il suffit d’écrire « Tant que n < 4 et c < 1000 ».

* Faire fonctionner le programme. Peut-être faut-il rajouter un « Pause » au bon endroit…

**V – Trouver les instructions sur la calculatrice**

|  |  |
| --- | --- |
| **Calculatrices TI 82Stats à TI 84** | **Calculatrices Casio Graph 35 à 65** |
| Les instructions de base (Prompt, Input, Disp) se trouvent dans prgm ► E/S.  Les instructions de boucles et itérateurs (If, Then, Else, For, While, End) se trouvent dans prgm CTL.  **→** : STO→  **"** : alpha +  **=** : 2nde math (tests) 1  Il existe aussi un catalogue de toutes les instructions : 2nde 0 | **"**  : **F6** (SYBL) **F2**  **:**  : SHIFT VARS (PRGM) **F6** (›) **F5**  **?**  : SHIFT VARS (PRGM) **F4**  **→** : →  : SHIFT VARS (PRGM) **F5**  **=**  : SHIFT .  Les instructions de boucles et itérateurs se trouvent dans SHIFT VARS (PRGM) **F1** puis éventuellement **F6** plusieurs fois. |