Maths collège

Programme de calcul Scratch corrigé : méthode simple et erreurs

Programme de calcul Scratch corrigé : traduction en blocs, méthode pas à pas et erreurs fréquentes pour réussir en 4e, 3e et au Brevet.

Cours de mathématiques niveau

Mis à jour le 24 avril 2026

Télécharger la fiche PDF du cours

Version imprimable · 4981 mots

Télécharger

Un programme de calcul Scratch corrigé montre comment traduire un énoncé en blocs, exécuter les opérations dans le bon ordre et vérifier le résultat. Pour le réussir, il faut identifier la variable de départ, suivre chaque étape sans inversion et comparer avec l'expression littérale correspondante.

Tu lis "choisir un nombre, ajouter 3, multiplier par 2" et tout semble clair… jusqu'au moment de le coder sur Scratch. Faut-il créer une variable ? Demander une valeur ? Multiplier avant ou après ? C'est exactement là que beaucoup d'élèves de 4e et 3e se trompent, alors que la logique est simple quand on suit une vraie méthode. Avec une bonne traduction entre l'énoncé, les blocs Scratch et le calcul littéral, on comprend mieux, on corrige plus vite et on évite les erreurs classiques avant un contrôle ou le Brevet.

En bref : les réponses rapides

Comment vérifier rapidement qu'un programme Scratch correspond bien à l'énoncé ? — Le plus simple est de tester le programme avec un nombre facile, comme 2, puis de refaire le calcul à la main. Si le résultat diffère, l'erreur vient souvent de l'ordre des blocs ou d'une variable mal utilisée.

Faut-il utiliser une ou deux variables dans un programme de calcul Scratch ? — Une seule variable peut suffire pour des programmes simples, mais

deux variables sont plus sûres quand il faut conserver le nombre de départ. Cela évite d'écraser une valeur utile pour la suite du calcul.

Quelle différence entre un programme de calcul et une expression littérale ? — Le programme de calcul décrit des actions successives, tandis que l'expression littérale résume ces actions sous forme algébrique. Les deux représentent pourtant le même calcul si la traduction est correcte.

Comment réussir un exercice Scratch sans connaître parfaitement le logiciel ? — Il faut d'abord comprendre l'algorithme sur papier avant d'ouvrir Scratch. Une fois les étapes claires, il ne reste qu'à associer chaque action au bon bloc.

Comprendre un programme de calcul sur Scratch : de l'énoncé aux blocs, sans se tromper

Un **programme de calcul sur Scratch** consiste à choisir un nombre de départ, appliquer des opérations dans l'ordre, puis afficher le résultat. Pour réussir, il faut repérer la **variable Scratch** d'entrée, traduire chaque action en blocs, et vérifier que la suite d'instructions respecte exactement l'énoncé mathématique.

Scratch est un logiciel d'**algorithmique** où l'on programme avec des blocs. Au collège, il sert à transformer un énoncé en suite d'actions claires. Un **programme de calcul** demande de partir d'un nombre, noté souvent x, puis d'enchaîner des opérations pour obtenir un résultat final. En **4e** et en **3e**, cela relie directement Scratch, **calcul littéral** et préparation au **Brevet**. Quand on cherche *comment expliquer Scratch* ou *comment se servir de Scratch*, l'idée simple est celle-ci : l'ordinateur suit des consignes, une par une, sans deviner. Un **programme de calcul scratch corrigé** n'est utile que si l'on comprend cette logique, pas si l'on recopie des blocs vus en PDF sur un site de collège.

Dans un **algorithme maths collège**, on retrouve presque toujours les mêmes éléments : une **entrée utilisateur**, une variable qui stocke la valeur, des calculs intermédiaires, puis un affichage. La variable garde le nombre choisi. La séquence d'instructions impose l'ordre. Si l'énoncé dit "multiplier par 3 puis ajouter 5", on obtient $3x+5$, alors que "ajouter 5 puis multiplier par 3" donne $3(x+5)$. Ce n'est pas pareil. Pour **comment faire un programme de calcul avec Scratch**, il faut donc lire les verbes avec précision.

Verbe de l’énoncé	Traduction Scratch	Écriture littérale
choisir un nombre	demander puis mettre *nombre* à réponse	x
ajouter a	mettre *nombre* à *nombre* $+a$	$x+a$
multiplier par a	mettre *nombre* à *nombre* $\times a$	ax
soustraire a	mettre *nombre* à *nombre* $-a$	$x-a$
prendre le carré	mettre *nombre* à *nombre* $\times$ *nombre*	x^2
diviser par a	mettre *nombre* à *nombre* $:a$	$\frac{x}{a}$

Exemple 1. Énoncé : choisir un nombre, ajouter 4, puis multiplier par 2. Si le nombre de départ vaut 3, on fait $3+4=7$, puis $7\times 2=14$. L’expression est $2(x+4)$. **Exemple 2.** Énoncé : choisir un nombre, multiplier par 2, puis ajouter 4. Avec 3, on obtient $3\times 2=6$, puis $6+4=10$. L’expression est $2x+4$. La différence entre **programme de calcul 4e** et **programme de calcul 3e** tient souvent à cette traduction fine entre texte, blocs et écriture littérale.

Exercice 1. Choisir 5, soustraire 2, prendre le carré. Résultat : $(5-2)^2=3^2=9$. **Exercice 2.** Choisir 8, diviser par 4, ajouter 7. Résultat : $\frac{8}{4}+7=2+7=9$. **Exercice 3.** Choisir x, ajouter 3, puis soustraire 10. Expression : $x+3-10=x-7$. **Exercice 4.** Choisir x, prendre le carré, puis diviser par 2. Expression : $ \frac{x^{2}}{2}$. Ces petits corrigés montrent la méthode : lire, traduire, calculer, vérifier l’ordre.

À retenir

À retenir : sur **Scratch**, un programme de calcul est une suite d'actions exécutées sans erreur d'interprétation. Repérez la variable d'entrée, traduisez chaque verbe en bloc, puis comparez avec l'expression littérale. Si l'ordre change, le résultat change aussi. C'est la base pour comprendre, corriger et déboguer efficacement.

Grille de traduction unique : énoncé → blocs Scratch → expression littérale

Pour corriger un **programme de calcul Scratch corrigé**, utilise toujours la même passerelle : *phrase* → *bloc* → *écriture mathématique*. "Choisir un nombre" devient demander un nombre puis stocker la réponse dans la variable n, donc l'expression est n. "Ajouter 5" devient mettre résultat à $n+5$. "Multiplier par 3" donne mettre résultat à $(n+5)\times 3$, soit $3(n+5)$. "Enlever 2" donne $3(n+5)-2$. "Prendre le carré" devient résultat $\times$ résultat, donc $(3(n+5)-2)^2$. Cette lecture évite de mélanger consigne, code et calcul littéral.

La force de cette grille est simple : elle relie **Scratch** et **algèbre** sans changer de logique. Si l'énoncé dit "multiplier la somme par 3", le bloc correct porte sur *tout le résultat précédent*, pas seulement sur 5 ; on obtient donc $3(n+5)$ et non $n+15$. Même vigilance avec "prendre le carré" : on élève l'expression entière au carré, donc $(3(n+5)-2)^2$, pas $3(n+5)-2^2$. Avec ce réflexe, un **programme de calcul Scratch corrigé** devient plus lisible, et la vérification du code se fait presque comme une traduction mot à mot.

Programmes de calcul dans Scratch - Questions flash — Maths et Jeux (Juliette Hernando)

Exemple complet corrigé : créer un programme de calcul Scratch et retrouver l'expression littérale

Pour **créer un programme de calcul avec Scratch**, on demande un **nombre de départ**, on le stocke dans une variable, on applique les opérations dans l'ordre, puis on affiche le résultat. La correction vérifie toujours trois points : *la bonne variable*, *le bon enchaînement des blocs* et la bonne **expression littérale** associée.

Prenons un **exemple de programme Scratch** classique au **collège** : « Choisis un nombre, ajoute 4, multiplie le résultat par 5, puis retire le double du nombre de départ. » En mathématiques, si le nombre choisi vaut x, on traduit chaque étape sans rien mélanger : on obtient d'abord $x+4$, puis $5(x+4)$, enfin $5(x+4)-2x$. Cette écriture est l'**expression littérale Scratch** du programme. Si on réduit, on trouve $5x+20-2x=3x+20$. Les deux formes sont justes, mais la forme $5(x+4)-2x$ montre mieux la logique de l'algorithme, tandis que $3x+20$ permet un calcul plus rapide.

La cohérence entre **algorithme**, blocs **Scratch** et calcul littéral repose sur une règle simple : chaque bloc doit correspondre à une opération écrite, dans le même ordre. Ici, le pseudo-code est net : demander un nombre ; mettre *nombre de départ* à la réponse ; mettre **variable résultat** à *nombre de départ* ; ajouter 4 à *résultat* ; mettre *résultat* à $5 \times \text{résultat}$; mettre *résultat* à $\text{résultat} - 2 \times \text{nombre de départ}$; afficher le résultat. Pour *comment faire un programme Scratch* sans erreur, il faut conserver deux variables si une opération réutilise la valeur initiale. Sinon, on perd le nombre de départ et le programme devient faux.

Version corrigée en trois lectures. Raisonnement mathématique : partir de x, puis écrire $x+4$, ensuite $5(x+4)$, enfin $5(x+4)-2x$. Lecture algorithmique : l'ordinateur mémorise le nombre de départ, calcule étape après étape, puis affiche. Lecture Scratch : bloc *demander*, variable **nombre de départ**, variable **résultat**, puis blocs *mettre résultat à*. On peut finir avec *dire résultat* pendant 2 secondes, ou afficher la variable à l'écran. Testons avec $x=3$: $3+4=7$, $7\times5=35$, $35-2\times3=29$. Avec l'expression, on vérifie aussi : $5(3+4)-2\times3=35-6=29$. Même sortie, donc programme correct. C'est exactement l'attendu d'un **programme de calcul Scratch brevet**.

Autre vérification utile, souvent demandée en **exercice Scratch 4ème corrigé** : repérer une erreur de code. Si on remplace la dernière ligne par « retirer le double de *résultat* », on calcule alors $5(x+4)-2\times5(x+4)$, ce qui n'a plus rien à voir avec l'énoncé. Si on oublie la variable initiale et qu'on modifie directement tout dans une seule case, on ne peut plus soustraire $2x$ correctement. Enfin, si on multiplie par 5 avant d'ajouter 4, on obtient $5x+4-2x=3x+4$, donc un autre

programme. Le débogage consiste donc à comparer mot à mot l'énoncé, les blocs et l'expression littérale.

Mini-corrigé d'application. Pour $x=0$, le programme donne $5(0+4)-2\times 0=20$. Pour $x=1$, il donne $5(1+4)-2=23$. Pour $x=-2$, il donne $5(2)+4=10+4=14$? Non : erreur fréquente. Il faut écrire $5((-2)+4)-2\times(-2)=5\times 2+4=14$. Ces tests montrent qu'un calcul juste confirme le code, mais révèle aussi les fautes de signe. Quand un résultat paraît étrange, on relit l'énoncé et on refait la traduction complète.

À retenir

À retenir : un bon programme de calcul utilise la bonne variable, respecte l'ordre des opérations et correspond à une **expression littérale** exacte. Ici, l'écriture correcte est $5(x+4)-2x$, réductible en $3x+20$. Si le résultat affiché par **Scratch** ne coïncide pas avec le calcul manuel, le bug vient presque toujours d'une variable écrasée ou d'un bloc mal placé.

Méthode de vérification en 3 passes avant de valider le corrigé

Avant de valider un corrigé, applique une **méthode en 3 passes** : vérifie d'abord l'**ordre des opérations**, puis teste le programme avec un nombre simple comme 2 ou 10, enfin compare le résultat obtenu avec l'*expression littérale* écrite sur feuille. Cette triple vérification repère vite une inversion, un bloc mal placé ou une parenthèse oubliée.

La **première passe** consiste à relire chaque bloc Scratch comme une phrase mathématique : ajouter, puis multiplier, n'est pas équivalent à multiplier, puis ajouter. Si l'énoncé dit « choisir un nombre, ajouter 3, puis multiplier par 5 », le programme doit traduire $(x+3)\times 5$ et non $x+3\times 5$. La **deuxième passe** sert à tester avec une valeur facile, car $x=2$ ou $x=10$ permet un calcul mental rapide ; si Scratch donne un résultat inattendu, l'erreur devient visible sans attendre le corrigé. La **troisième passe**, souvent négligée, consiste à comparer le résultat du programme avec celui de l'expression littérale obtenue sur feuille. Si les deux coïncident pour la même valeur de x, ton corrigé est cohérent ; sinon, il faut déboguer.

Les erreurs les plus fréquentes en Scratch : tableau avant/après pour corriger un même exercice

Les **erreurs Scratch collège** viennent souvent d'un **ordre de blocs** incorrect, d'une variable écrasée trop tôt ou d'un oubli du nombre de départ. Pour **corriger un programme Scratch**, comparez le code faux et le code juste sur le *même exercice*, puis repérez l'instant précis où le calcul ne suit plus l'énoncé.

Sur un programme de calcul du type « choisir un nombre, ajouter 3, prendre le carré, soustraire 5 », le bon réflexe est de traduire chaque phrase en une action unique sur une variable stable. En Scratch, on peut garder **nombre** pour l'entrée et **résultat** pour les transformations successives : on part de x, puis on obtient $(x+3)^2-5$. En **débogage**, une erreur n'est pas seulement un bloc faux : c'est un écart entre l'énoncé, le script Scratch et l'expression littérale attendue en **mathématiques** ou en **technologie**.

Au **Brevet**, on attend surtout trois vérifications concrètes : la variable de départ est bien initialisée, l'ordre des opérations respecte l'énoncé, et la valeur affichée correspond à la bonne variable. Si l'exercice demande de tester plusieurs nombres, un script juste doit donner pour $x=2$ la valeur $(2+3)^2-5=20$. Ce test simple suffit souvent à repérer une confusion entre *mettre* et *ajouter*, une soustraction inversée comme $5-résultat$ au lieu de $résultat-5$, ou la traduction fausse de « prendre le carré » en « multiplier par 2 ».

Erreur	Avant	Après	Symptôme, cause, correctif
Variable non initialisée	`ajouter 3 à résultat`	`mettre résultat à nombre` puis `ajouter 3 à résultat`	Le résultat change selon l'essai précédent. Cause : **résultat** garde une ancienne valeur. Correctif : toujours repartir de **nombre**.
Écraser au lieu d'ajouter	`mettre résultat à 3`	`ajouter 3 à résultat`	On perd x et on obtient ensuite $3^2-5=4$. Cause : confusion entre *mettre* et *ajouter*.
Carré mal traduit	`mettre résultat à`		Le script calcule $2(x+3)-5$ au lieu de $(x+3)^2-5$. Erreur

Erreur	Avant	Après	Symptôme, cause, correctif
	`résultat * 2`	`mettre résultat à résultat * résultat`	classique en **exercice scratch 3eme corrigé**.
Soustraction inversée	`mettre résultat à 5 - résultat`	`mettre résultat à résultat - 5`	Les signes deviennent faux. Pour $x=2$, on obtient -20 au lieu de 20.
Mauvaise variable affichée	`dire nombre`	`dire résultat`	Le calcul est juste mais l'écran affiche l'entrée. Fréquent en **débogage Scratch maths**.

Exemple résolu 1 : avec $x=2$, un script faux qui fait `mettre résultat à nombre`, `mettre résultat à 3`, `mettre résultat à résultat * résultat`, `mettre résultat à résultat - 5` donne 4. On voit où ça casse : la ligne `mettre résultat à 3` supprime la trace du nombre de départ. Le correctif donne $2 \to 5 \to 25 \to 20$. C'est la méthode la plus efficace pour **corriger un programme Scratch** sans deviner.

Exemple résolu 2 : avec $x=-1$, un script qui remplace le carré par * 2 donne $(-1+3)\times 2-5=-1$, alors que le bon calcul vaut $(-1+3)^2-5=-1$? Mauvais test, car ici les deux coïncident. Prenez aussi $x=2$: le faux script donne 5, le bon donne 20. En **3e**, choisir plusieurs valeurs évite de valider un script faux par hasard, surtout en **exercice scratch technologie 3ème corrigé**.

Mini-corrigé pratique : pour $x=4$, le bon programme produit $(4+3)^2-5=44$. Si votre script affiche 4, vous montrez la mauvaise variable. S'il affiche 9, vous avez écrasé la variable avant le carré. S'il affiche 2, vous avez fait $7-5$ après un doublement raté. S'il affiche -44, la soustraction est inversée. En contrôle de **mathématiques** ou de **technologie**, je conseille aux élèves de **3e** d'écrire à côté du script la chaîne $x \to x+3 \to (x+3)^2 \to (x+3)^2-5$: c'est court, et cela sécurise tout le **débogage**.

À retenir

À retenir : un bon script Scratch garde la trace du nombre de départ, transforme une seule variable à la fois et affiche la bonne sortie. Pour le **Brevet**, testez au moins deux valeurs de x et comparez toujours le script à l'expression littérale attendue.

Savoir remonter au nombre de départ et réussir les questions type collège ou Brevet

Pour **trouver le nombre de départ** d'un programme de calcul, on remonte soit les opérations en *sens inverse*, soit on écrit une **équation** à partir du résultat final. Le bon choix dépend du programme : si les étapes sont simples, l'inverse suffit ; si le calcul se complique, l'**expression littérale** devient plus sûre.

Le **nombre de départ** est la valeur inconnue entrée au début du programme. Pour répondre à *comment trouver le nombre de départ d'un programme de calcul*, on note souvent cette valeur x, puis on traduit chaque étape en calcul. En **Scratch**, cela revient à suivre la variable depuis l'entrée jusqu'à l'affichage final. En maths, c'est exactement le lien entre **algorithme**, calcul littéral et recherche d'inconnue.

Si le programme enchaîne des opérations réversibles, on peut remonter en appliquant les opérations contraires dans l'ordre inverse : ajouter $\leftrightarrow$ soustraire, multiplier par a $\leftrightarrow$ diviser par a si $a \neq 0$. En revanche, dès qu'apparaît un carré, une double utilisation du nombre de départ ou une écriture du type $3x+5$, il est souvent plus efficace d'écrire directement une **équation**. Cette méthode est fréquente en **4e**, en **3e** et dans les sujets de **Brevet**, où l'on demande aussi de comparer deux programmes.

Exemple 1. On choisit un nombre, on ajoute 7, puis on multiplie par 2, et on obtient 26. On remonte : avant le dernier calcul, on avait $26 : 2 = 13$; avant l'addition, on avait $13 - 7 = 6$. Le nombre de départ est donc 6. Cette méthode répond bien à *comment faire un algorithme en maths* : on lit les étapes dans un sens, puis on les inverse proprement dans l'autre.

Sur un contrôle, les formulations classiques sont : “Quel nombre faut-il choisir pour obtenir 26 ?” ou “Retrouver le nombre initial”.

Exemple 2. On choisit un nombre, on le met au carré, puis on ajoute 3, et on obtient 19. En **Scratch**, pour *comment mettre un nombre au carré sur Scratch*, on écrit la variable multipliée par elle-même, soit x^2. Ici, on remonte d’abord : $19-3=16$. Ensuite, il faut résoudre $x^2=16$, donc $x=4$ ou $x=-4$. Le carré change la difficulté, car remonter ne donne plus toujours une seule réponse. C’est précisément le type de piège vu dans un **programme de calcul scratch corrigé brevet** ou un **dm scratch 4ème corrigé**.

Exercice 1. Ajouter 5, puis multiplier par 3, résultat 24. On remonte : $24\div 3=8$, puis $8-5=3$. **Réponse :** 3. **Exercice 2.** Multiplier par 4, puis soustraire 1, résultat 15. On remonte : $15+1=16$, puis $16\div 4=4$. **Réponse :** 4. **Exercice 3.** Choisir x, calculer $2x+7$, obtenir 31. On écrit $2x+7=31$, donc $2x=24$, puis $x=12$. **Exercice 4.** Choisir x, calculer x^2-9, obtenir 27. On écrit $x^2-9=27$, donc $x^2=36$, d’où $x=6$ ou $x=-6$. Cette progression est la bonne : d’abord les inverses simples, ensuite l’**expression littérale**, enfin l’équation avec carré.

À retenir

À retenir : pour remonter un programme, identifiez toujours la variable de départ, puis choisissez la méthode la plus stable. Si chaque étape s’inverse facilement, remontez le calcul. Si le programme contient x^2, plusieurs occurrences de x ou une comparaison de programmes, passez à l’**équation**. L’entraînement le plus utile au collège consiste à transformer un énoncé en algorithme, puis en code **Scratch**, puis en écriture littérale : c’est cette chaîne qui fait réussir en **4e**, en **3e** et au **Brevet**.

Mini feuille de route pour s’entraîner efficacement en algorithmique et Scratch au collège

Pour progresser en **Scratch** au **collège**, suivez toujours le même ordre : comprendre l’énoncé, écrire l’algorithme, coder, tester avec une valeur simple, puis relier le programme à l’expression littérale. Cette méthode évite la copie mécanique, structure l’apprentissage de **l’algorithmique et Scratch** et prépare mieux aux contrôles comme aux **annales** du Brevet.

Un bon entraînement ne consiste pas à accumuler des *activités Scratch avec correction*, mais à refaire souvent la même chaîne mentale : phrase → actions → blocs → calcul. Si l’énoncé dit “choisir un nombre, ajouter 3, puis multiplier par 2”, l’élève doit pouvoir dire, sans écran, que le résultat vaut $2(x+3)$ ou $2x+6$. Le code n’est qu’une traduction visible. Cette logique relie **algorithmique et programmation**, calcul littéral et vérification.

La progression la plus efficace va du plus simple au plus piégeux. On commence par des programmes additifs : partir de x, calculer $x+5$ ou $x-7$. Puis on enchaîne deux opérations : $(x+4)\times 3$, $5x-2$, ou le carré et le double, par exemple x^2+2x. Ensuite seulement viennent les programmes avec deux variables, puis la recherche du nombre de départ, où l’on remonte un calcul à l’envers. À chaque étape, un test avec $x=2$ ou $x=5$ suffit pour repérer beaucoup d’erreurs.

Exemple 1 : “Choisir un nombre, le doubler, ajouter 7.” Algorithme : prendre x, calculer $2x$, puis $2x+7$. Dans **Scratch**, on stocke le nombre de départ dans une variable, puis on modifie cette variable. Test : avec $x=3$, on obtient 13. Exemple 2 : “Choisir un nombre, ajouter 1, prendre le carré.” Ici, l’ordre change tout : on calcule $(x+1)^2$, pas x^2+1. Avec $x=2$, le bon résultat est 9 ; ce test simple révèle immédiatement un mauvais enchaînement de blocs.

Feuille de route courte et actionnable :

- Travaillez 3 ou 4 programmes simples, puis réécrivez chacun en phrase, en blocs et en expression comme $3(x-2)$.
- Passez ensuite à des **exercices scratch pdf**, mais corrigez d'abord au brouillon avant d'ouvrir le **PDF** de solution.
- Ajoutez des exercices de recherche du nombre de départ, car ce sont eux qui vérifient vraiment la compréhension.
- Terminez par quelques **annales** et des **fiches de révision Scratch collège** pour automatiser les réflexes.

À retenir

Pour les parents et enseignants, le bon test est simple : demander à l'élève d'expliquer oralement ce que fait le programme, puis de prévoir le résultat pour $x=4$ avant d'exécuter le code. S'il sait passer de l'énoncé à $x+3$, puis à $(x+3)\times 2$, et repérer seul une erreur d'ordre, la compréhension est là. Le corrigé vient *après*, comme outil de vérification, pas comme point de départ.

Comment expliquer scratch ?

Scratch est un langage de programmation visuel conçu pour apprendre à coder simplement. Au lieu d'écrire des lignes de code, on assemble des blocs colorés qui représentent des actions, des conditions ou des calculs. Je le trouve idéal pour comprendre la logique, créer des animations, des jeux et des programmes de calcul sans se perdre dans la syntaxe.

Comment se servir de Scratch ?

Pour se servir de Scratch, il faut choisir un sprite, puis glisser-déposer les blocs dans la zone de script. On utilise les catégories comme Mouvement, Apparence, Événements, Contrôle et Opérateurs. Je conseille de commencer par un bloc "quand drapeau vert cliqué", puis d'ajouter des actions simples, des variables et des tests pour construire un programme progressivement.

Comment faire un programme de calcul avec Scratch ?

Pour créer un programme de calcul Scratch corrigé, on demande d'abord un nombre avec "demander". Ensuite, on stocke la réponse dans une variable, puis on enchaîne les opérations avec les blocs Opérateurs : addition, multiplication, carré ou division. Enfin, on

affiche le résultat avec “dire”. Je recommande de tester plusieurs nombres pour vérifier que le programme fonctionne correctement.

Comment trouver le nombre de départ d'un programme de calcul ?

Pour retrouver le nombre de départ, il faut remonter les étapes du programme en utilisant les opérations inverses. Si le programme ajoute 5 puis multiplie par 2, on commence par diviser le résultat par 2, puis on retire 5. En maths comme dans un programme de calcul Scratch corrigé, cette méthode permet de retrouver la valeur initiale avec précision.

Comment faire un programme de calcul ?

Un programme de calcul suit des étapes simples : choisir un nombre de départ, appliquer une ou plusieurs opérations, puis annoncer le résultat. Je conseille d’écrire d’abord les consignes en français, par exemple “choisir un nombre, ajouter 3, multiplier par 4”. Ensuite, on peut le traduire en Scratch, en algorithme ou en expression mathématique pour le corriger facilement.

Comment faire un algorithme en maths ?

Un algorithme en maths est une suite d’instructions ordonnées. Pour le rédiger, on note clairement le début, la donnée de départ, les calculs à effectuer et le résultat final. Je recommande d’utiliser des verbes simples : demander, calculer, stocker, afficher. Cette méthode aide à passer facilement d’un exercice de maths à un programme Scratch clair et corrigé.

Comment mettre un nombre au carré sur Scratch ?

Pour mettre un nombre au carré sur Scratch, il suffit d’utiliser le bloc multiplication et de multiplier le nombre par lui-même. Par exemple, si la variable s’appelle “nombre”, on calcule “nombre * nombre”. Je conseille de stocker le résultat dans une autre variable pour mieux suivre les étapes du programme de calcul et faciliter la correction.

comment faire un programme scratch

Pour faire un programme Scratch, on commence par définir ce que l’on veut obtenir : animation, jeu ou calcul. Ensuite, on choisit un événement de départ, on ajoute des blocs d’actions, puis on teste. Je recommande de créer le programme par petites étapes, en vérifiant à chaque fois le résultat, afin d’éviter les erreurs et de corriger plus vite.

Pour réussir un programme de calcul sur Scratch, retiens une règle simple : une étape de l’énoncé = une action claire dans les blocs, dans le bon ordre. Si un résultat semble faux, compare toujours le code, le calcul écrit et l’expression littérale. Cette vérification croisée aide à repérer presque toutes les erreurs. Garde aussi quelques exemples corrigés sous la main pour t’entraîner : en algorithmique, la régularité fait souvent toute la différence.

Continue sur maths-college.fr

Maths collège - Document pédagogique