

L'activité de programmation dans  
l'option informatique des lycées.

Des pistes pour le nouveau  
programme de mathématiques ?

François Heuzé

Lycée Salvador Allende - Hérouville Saint Clair

francois.heuze@ac-caen.fr

# L'option informatique des lycées

- Trois périodes
  - 1981 - 1994
  - 1995 - 1997 (APTIC)
  - 1998 - 2001
- Horaire hebdomadaire de la première période
  - 1 h classe entière + 1,5 h en groupes

# L'option informatique au lycée

- Activités (première période)
  - Programmation
  - Informatique et société

# Algorithmique ou programmation ?

- Algorithmique
  - utilisation du papier et du crayon.
  - expression « libre »
- Programmation
  - utilisation de l'ordinateur
  - expression contrainte par le langage utilisé.

# Enseignement optionnel complémentaire d'informatique en second cycle long Objectifs du programme (extraits)

L'informatique sera replacée dans le cadre d'un processus général de résolution de problèmes, où l'on distingue et approfondit différentes étapes :

- énoncé d'un problème ;
- découverte et expression d'un procédé de résolution ;
- codage dans un langage de programmation ;
- validation du programme ainsi obtenu ;
- utilisation effective pour l'application retenue.

Cette démarche met en jeu des qualités universelles d'analyse, de synthèse, de rigueur d'expression et d'efficacité. L'intérêt de l'informatique est d'aider à les révéler et à les consolider chez les élèves grâce à son caractère constructif, concret, et à sa rapidité de calcul.

(...)

# PROGRAMME D'ENSEIGNEMENT DE LA CLASSE DE SECONDE (extraits)

## **2 - Analyse et programmation :**

L'enseignement doit mettre l'élève à même de :

- lire un texte et spécifier clairement le problème à résoudre ;
- proposer et formuler avec précision une méthode de résolution ;
- la rédiger dans un langage de programmation ;
- la mettre en oeuvre sur un ordinateur.

Pour cela, le professeur développera des exemples typiques, en insistant sur les aspects méthodologiques. En travaux pratiques, les élèves pourront reprendre ces exemples et traiter, de façon analogue, d'autres problèmes de difficultés comparables.

*2.1. - Traitement séquentiel*

*2.2. - Traitement conditionnel*

*2.3. - Traitement itératif simple*

*2.4. - Traitement itératif général (première approche)*

(on ne sait pas calculer à l'avance le nombre de répétitions, mais on connaît une condition d'arrêt)

Les notions de tableau, de fichier de données, de procédures et de fonctions sont au programme de première.

Mise en relief de ce que l'on faisait avec  
ce qui nous réunit aujourd'hui

Mise en relief de ce que l'on faisait avec  
ce qui nous réunit aujourd'hui

- Il s'agissait de faire de la programmation

Mise en relief de ce que l'on faisait avec  
ce qui nous réunit aujourd'hui

- Il s'agissait de faire de la programmation
  - avec des élèves volontaires,

Mise en relief de ce que l'on faisait avec  
ce qui nous réunit aujourd'hui

- Il s'agissait de faire de la programmation
  - avec des élèves volontaires,
  - sans faire de mathématiques.

Les débats de l'époque **et de maintenant**

# Les débats de l'époque **et de maintenant**

- Quelle méthode ?

# Les débats de l'époque **et de maintenant**

- Quelle méthode ?
  - **impérative** ou fonctionnelle ?

# Les débats de l'époque **et de maintenant**

- Quelle méthode ?
  - **impérative** ou fonctionnelle ?
- Quelle façon de traiter la répétition ?

# Les débats de l'époque **et de maintenant**

- Quelle méthode ?
  - **impérative** ou fonctionnelle ?
- Quelle façon de traiter la répétition ?
  - **itératif** ou récursif ?

# Les débats de l'époque **et de maintenant**

- Quelle méthode ?
  - **impérative** ou fonctionnelle ?
- Quelle façon de traiter la répétition ?
  - **itératif** ou récursif ?
- Quel langage ?

# Les débats de l'époque **et de maintenant**

- Quelle méthode ?
  - **impérative** ou fonctionnelle ?
- Quelle façon de traiter la répétition ?
  - **itératif** ou récursif ?
- Quel langage ?
  - BASIC, LSE, PASCAL, LOGO ...?

# Les débats de l'époque **et de maintenant**

- Quelle méthode ?
  - **impérative** ou fonctionnelle ?
- Quelle façon de traiter la répétition ?
  - **itératif** ou récursif ?
- Quel langage ?
  - BASIC, LSE, PASCAL, LOGO ...?
  - **Scratch, Xcas, Linotte, Maxima, Python, Scilab, Execalgo, Algobox, R ...**

# Les débats de l'époque **et de maintenant**

- Quelle méthode ?
  - **impérative** ou fonctionnelle ?
- Quelle façon de traiter la répétition ?
  - **itératif** ou récursif ?
- Quel langage ?
  - BASIC, LSE, PASCAL, LOGO ...?
  - **Scratch, Xcas, Linotte, Maxima, Python, Scilab, Execalgo, Algobox, R ...**
- Le GOTO

# Les difficultés observées chez les élèves

# Les difficultés observées chez les élèves

- Programmer est un exercice difficile

# Les difficultés observées chez les élèves

- Programmer est un exercice difficile
- Nouveaux environnements
  - contexte de rédaction, contexte d'exécution

# Les difficultés observées chez les élèves

- Programmer est un exercice difficile
- Nouveaux environnements
  - contexte de rédaction, contexte d'exécution
- Nouveaux concepts
  - variable, entrée-sortie, choix, boucle ...

# Les difficultés observées chez les élèves

- Programmer est un exercice difficile
- Nouveaux environnements
  - contexte de rédaction, contexte d'exécution
- Nouveaux concepts
  - variable, entrée-sortie, choix, boucle ...
- Nouveau vocabulaire

# Les difficultés observées chez les élèves

- Programmer est un exercice difficile
- Nouveaux environnements
  - contexte de rédaction, contexte d'exécution
- Nouveaux concepts
  - variable, entrée-sortie, choix, boucle ...
- Nouveau vocabulaire
- L'élève doit se construire des représentations mentales correctes.

# Le concept de variable

- Les « vraies variables »
  - Celles dont la valeur change effectivement dans le programme. Exemple :  
`cumul <- cumul + note`
- Les « fausses variables »
  - Ne passent que de l'état « n'a pas de valeur » à l'état « a une valeur »

Extrait du sujet du DNB – France – septembre 2008

# Extrait du sujet du DNB – France – septembre 2008

Choisir un nombre

- a) Calculer le carré de ce nombre
- b) Multiplier par 10
- c) ajouter 25

Ecrire le résultat

# Extrait du sujet du DNB – France – septembre 2008

Choisir un nombre

- a) Calculer le carré de ce nombre
- b) Multiplier par 10
- c) ajouter 25

Ecrire le résultat

`N,C,M,R : nombres`

`lire N`

`C <- N*N`

`M <- C*10`

`R <- M+25`

`afficher R`

# Extrait du sujet du DNB – France – septembre 2008

Choisir un nombre

- a) Calculer le carré de ce nombre
- b) Multiplier par 10
- c) ajouter 25

Ecrire le résultat

`N,C,M,R : nombres`

```
lire N
```

```
C <- N*N
```

```
M <- C*10
```

```
R <- M+25
```

```
afficher R
```

`N,R : nombres`

```
lire N
```

```
R <- N*N
```

```
R <- R*10
```

```
R <- R+25
```

```
afficher R
```

# Le concept de boucle

# Le concept de boucle

- Une boucle **doit** avoir (au moins) une **condition de sortie**
  - cela suppose la maîtrise des opérateurs  $=$   $>$   $<$  ,  
[ **négation**, **et**, **ou** ]

# Le concept de boucle

- Une boucle **doit** avoir (au moins) une **condition de sortie**
  - cela suppose la maîtrise des opérateurs  $=$   $>$   $<$  ,  
[ **négation**, **et**, **ou** ]
- La condition de sortie contient obligatoirement une « vraie variable »
  - il est délicat d'aborder le concept de boucle si le concept de « vraie variable » n'est pas acquis.

# Algorithme d'échange de deux variables

```
| Variables  
| a, b, c : nombres  
| Début  
|   Ecrire('Saisissez deux valeurs entières ' )  
|   Ecrire('Valeur n°1 ? ' )  
|   Lire(a)  
|   Ecrire('Valeur n°2 ? ' )  
|   Lire(b)  
|   Ecrire('Valeur n°1=',a,' Valeur n°2= ',b)  
|   c <- a  
|   a <- b  
|   b <- c  
|   Ecrire('Valeur n°1=',a,' Valeur n°2= ',b)  
|   Fin
```

# Affectation de variable problème de représentation

	A	B	C
1		7=A1	
2			

# Affectation de variable problème de représentation

- Dans un tableur, on entre  
7 dans la cellule A1  
la formule =A1 dans B1

	A	B	C
1	7	=A1	
2			

# Affectation de variable problème de représentation

- Dans un tableur, on entre  
7 dans la cellule A1  
la formule =A1 dans B1

	A	B	C
1	7	=A1	
2			

- « B1 ← A1 »











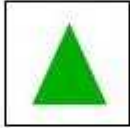
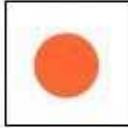

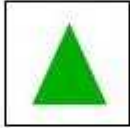
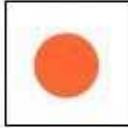

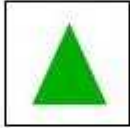
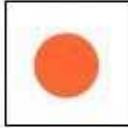
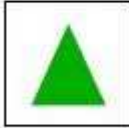
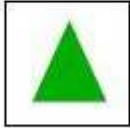
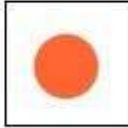
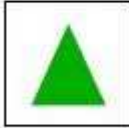
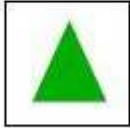
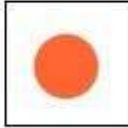
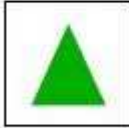
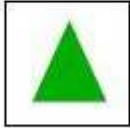
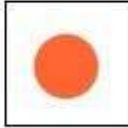









# Affectation de variable problème de représentation

- Dans un tableur, on entre  
7 dans la cellule A1  
la formule =A1 dans B1

	A	B	C
1	7	=A1	
2			

- « B1 ← A1 »
- Une modification de la valeur de A1 implique une modification de la valeur de B1 !

# Echange de deux variables

<i>Algorithme</i>	<i>En machine</i>						
<i>Initialisations</i>	<table><tr><td><i>A</i></td><td><i>B</i></td><td><i>C</i></td></tr><tr><td></td><td></td><td></td></tr></table>	<i>A</i>	<i>B</i>	<i>C</i>			
<i>A</i>	<i>B</i>	<i>C</i>					
							
<b><math>C \leftarrow A</math></b>	<table><tr><td><i>A</i></td><td><i>B</i></td><td><i>C</i></td></tr><tr><td></td><td></td><td></td></tr></table>	<i>A</i>	<i>B</i>	<i>C</i>			
<i>A</i>	<i>B</i>	<i>C</i>					
							
<b><math>A \leftarrow B</math></b>	<table><tr><td><i>A</i></td><td><i>B</i></td><td><i>C</i></td></tr><tr><td></td><td></td><td></td></tr></table>	<i>A</i>	<i>B</i>	<i>C</i>			
<i>A</i>	<i>B</i>	<i>C</i>					
							
<b><math>B \leftarrow C</math></b>	<table><tr><td><i>A</i></td><td><i>B</i></td><td><i>C</i></td></tr><tr><td></td><td></td><td></td></tr></table>	<i>A</i>	<i>B</i>	<i>C</i>			
<i>A</i>	<i>B</i>	<i>C</i>					
							

# Une vraie variable dans un si ... alors ... sinon

- **Algorithme du maximum**

```
| a, b, max : nombres  
| Lire(a)  
| Lire(b)  
| si a>=b alors max <- a  
| sinon max <- b  
| Ecrire('Le maximum est',max)
```

- **OU**

```
| a, b, max : nombres  
| Lire(a)  
| Lire(b)  
| max <- a  
| si b>max alors max <- b  
| Ecrire('Le maximum est',max)
```

# Le feu tricolore

```
| * etat est une variable qui ne peut prendre que  
| les valeurs 'vert', 'orange' ou 'rouge'  
| lire etat  
| si etat='vert' alors etat <- 'orange'  
| sinon  
|   si etat= 'orange' alors etat <- 'rouge'  
|   sinon  
|     etat <- 'vert'  
|     * fin si etat=orange  
|     * fin si etat=vert  
| afficher etat
```

# Pour aborder la répétition

- La structure répète  $n$  fois est intéressante du point de vue pédagogique

```
| lire etat  
| répète 20 fois  
|   si etat='vert' alors etat <- 'orange'  
|   sinon  
|     si etat= 'orange' alors etat <- 'rouge'  
|     sinon  
|       etat <- 'vert'  
|     * fin si etat=orange  
|   * fin si etat=vert  
| afficher etat  
| * fin du répète
```

# Vers la boucle *Pour*

```
| Pour i de 1 à n  
|   Traitement  
|   * Fin du Pour
```

- C'est le système qui gère ici la variable *i* !
- est équivalent à

```
| i <- 1  
| répète n fois  
|   Traitement  
|   i <- i+1  
|   * fin du répète
```

# Forme algorithmique d'une boucle

- Forme « large » qui facilite l'expression.

```
| boucle  
|   TraitementA  
|   quand ConditionDeSortie sors  
|   TraitementB  
|   * Fin du bloc boucle
```

- TraitementA est exécuté une fois de plus que Traitement B

# Calcul de la moyenne d'une suite de notes entrées au clavier ; on termine en entrant -1

```
| Compteur : entier  
| Note,Cumul,Moyenne : décimaux  
| Compteur <- 0  
| Cumul <- 0  
| boucle  
|   ecris "Note numero ",Compteur+1," (-1 pour finir) "  
|   lis Note  
|   quand Note = -1 sors  
|   Compteur <- Compteur+1  
|   Cumul <- Cumul + Note  
|   * Fin du bloc boucle  
| Afficher Cumul/Compteur
```

# Algorithme du jeu consistant à trouver la nombre choisi au hasard par l'ordinateur

```
| atrouver,compt,n:entiers  
| atrouver <- hasard(100)  
| compteur <- 0  
| boucle  
|   compteur <- compteur + 1  
|   écris "Propose un nombre entre 0 et 100 "  
|   lis n  
|   quand n=atrouver sors  
|   si n>atrouver alors  
|     écris "Le nombre proposé est TROP GRAND"  
|   sinon  
|     écris "Le nombre proposé est TROP PETIT"  
|   * fin du si  
|   * fin du bloc boucle  
|   écris "Tu as trouvé en ",compteur," coups"
```

*Le Tant que*

# *Le Tant que*

- Le *Tant que* est une structure de programmation qui exprime les boucles :
  - à **une** condition de sortie
  - quand cette condition est en **début** de boucle

# *Le Tant que*

- Le *Tant que* est une structure de programmation qui exprime les boucles :
  - à **une** condition de sortie
  - quand cette condition est en **début** de boucle

| **Tant que** (non ConditionDeSortie)

# *Le Tant que*

- Le *Tant que* est une structure de programmation qui exprime les boucles :
  - à **une** condition de sortie
  - quand cette condition est en **début** de boucle

```
| Tant que (non ConditionDeSortie)  
|   Traitement
```

# *Le Tant que*

- Le *Tant que* est une structure de programmation qui exprime les boucles :
  - à **une** condition de sortie
  - quand cette condition est en **début** de boucle

```
| Tant que (non ConditionDeSortie)  
|   Traitement  
|   * Fin Tant que
```

# *Le Tant que*

- Le *Tant que* est une structure de programmation qui exprime les boucles :
  - à **une** condition de sortie
  - quand cette condition est en **début** de boucle

```
| Tant que (non ConditionDeSortie)  
|   Traitement  
|   * Fin Tant que
```

- Il y a une vraie difficulté à traduire une boucle qui n'a pas ces caractéristiques avec un *Tant que*!

# Boucles à deux conditions de sortie

```
| boucle
```

```
|   TraitementA
```

```
|   quand ConditionDeSortie1 sors
```

```
|   TraitementB
```

```
|   quand ConditionDeSortie2 sors
```

```
|   TraitementC
```

```
|   * Fin du bloc boucle
```

```
| * ici il y a un test pour savoir par où on est sorti
```

## Exemples

- Jeu du nombre à trouver en limitant le nombre de coups  
 Algorithme 5 page 25 du document d'accompagnement
- Recherche de la primalité d'un nombre en s'arrêtant à la racine carrée
- Algorithme 3 page 23 du document d'accompagnement

## Algorithme 3 : déterministe à pas constant sur un intervalle borné (page 23/33)

- Cet algorithme **simple** repose sur le balayage à pas constant de l'intervalle d'étude et par la comparaison systématique de deux images “successives” ou plutôt du signe de leur différence. Si ce signe ne change jamais, la fonction est peut-être monotone, sinon elle ne l'est certainement pas et on pourra arrêter le balayage en affichant les valeurs considérées .

### Algorithme 3 - Page 23/33

```
| Variables
| a, b les bornes de l'intervalle d'étude [a ; b]
| f, la fonction à étudier
| N, le nombre d'intervalles
| x, les valeurs successives de la variable
| Initialisation
| pas prend la valeur (b-a)/N
| sens prend la valeur signe de la différence f (b) - f (a)
| x prend la valeur a
| Traitement
|
| Pour k variant de 1 à N
|   Si ( f (x+pas)-f(x) n'est pas de même signe que sens ) alors
|     Affiche " la fonction n'est pas monotone "
|     Affiche x et x+pas
|     Fin du programme
|   x prend la valeur x+pas
|   * Fin du Pour
| Affiche " La fonction semble monotone ".
```

## Algorithme 5 : jeu du nombre à deviner - Page 25/33

```
| Variables
| N nombre choisi par l'utilisateur
| Initialisation
| S, un nombre entier au hasard entre 10 et 100
| essai prend la valeur 1
| Traitement
| Tant que essai est inférieur ou égal à 6
|     Saisir N
|     Si N est supérieur à S alors
|         Affiche " c'est moins "
|     Si N est inférieur à S
|         Affiche " c'est plus "
|     Si n=S alors
|         Affiche " gagné "
|         fin de programme
|     essai prend la valeur essai+1
|     * Fin du Tant que
| Sortie
| Affiche " perdu ".
```

# Traduction avec *Tant que*

- Le langage dispose de l'instruction *break*

```
| boucle  
|   TraitementA  
|   quand ConditionDeSortie sors  
|   TraitementB  
|   * Fin du bloc boucle
```

**se traduit en langage C**

```
| while (1==1) {  
|   TraitementA  
|   if (ConditionDeSortie) then break;  
|   TraitementB  
|   }
```

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*

On peut utiliser une variable « drapeau »

```
• etat <- 0
| Tant que (etat=0)
|   TraitementA
|   si ConditionDeSortie alors etat <-1
|   sinon
|     TraitementB
|     * fin du si
| * Fin Tant que
```

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*

On peut aussi répéter deux fois l'écriture de *TraitementA*

```
| boucle  
|   TraitementA  
|   quand ConditionDeSortie sors  
|   TraitementB  
|   * Fin boucle
```

est équivalent à

```
|   TraitementA  
|   Tant que (Non ConditionDeSortie)  
|   TraitementB  
|   TraitementA  
|   * Fin Tant que
```

Traduction avec *Tant que*

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*
- Utiliser *GOTO* ou *Aller à*

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*
- Utiliser *GOTO* ou *Aller à*
  - | \* **Boucle**

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*
- Utiliser *GOTO* ou *Aller à*
  - | \* **Boucle**
  - |       :DEBUT1

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*
- Utiliser *GOTO* ou *Aller à*

```
| * Boucle  
|   :DEBUT1  
|   TraitementA
```

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*
- Utiliser *GOTO* ou *Aller à*

```
| * Boucle  
|   :DEBUT1  
|   TraitementA  
|   IF ConditionDeSortie Aller à SORTIE1
```

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*
- Utiliser *GOTO* ou *Aller à*

```
| * Boucle  
|   :DEBUT1  
|   TraitementA  
|   IF ConditionDeSortie Aller à SORTIE1  
|   TraitementB
```

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*
- Utiliser *GOTO* ou *Aller à*

```
| * Boucle  
|   :DEBUT1  
|   TraitementA  
|   IF ConditionDeSortie Aller à SORTIE1  
|   TraitementB  
|   Aller à DEBUT1
```

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*
- Utiliser *GOTO* ou *Aller à*

```
| * Boucle  
|   :DEBUT1  
|   TraitementA  
|   IF ConditionDeSortie Aller à SORTIE1  
|   TraitementB  
|   Aller à DEBUT1  
| * Fin boucle
```

# Traduction avec *Tant que*

- Le langage ne dispose pas de l'instruction *break*
- On ne veut pas utiliser de variable « drapeau »
- On ne veut pas répéter deux fois l'écriture de *TraitementA*
- Utiliser *GOTO* ou *Aller à*

```
| * Boucle  
|   :DEBUT1  
|   TraitementA  
|   IF ConditionDeSortie Aller à SORTIE1  
|   TraitementB  
|   Aller à DEBUT1  
|   * Fin boucle  
| :SORTIE1
```

# Conseils pour la classe de seconde

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.
- Le programme du B.O. indique l'algorithme de la résolution par dichotomie. Le prendre comme objectif de façon à amener l'élève à le construire lui-même.

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.
- Le programme du B.O. indique l'algorithme de la résolution par dichotomie. Le prendre comme objectif de façon à amener l'élève à le construire lui-même.

1/ Calculs sans vraie variable (périmètre et surface d'un cercle, exercice du DNB...)

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.
  - Le programme du B.O. indique l'algorithme de la résolution par dichotomie. Le prendre comme objectif de façon à amener l'élève à le construire lui-même.
- 1/ Calculs sans vraie variable (périmètre et surface d'un cercle, exercice du DNB...)
- 2/ Des *si alors sinon* simples (dire si un nombre est pair, positif, dire si un point est sur un cercle, si un triangle est isocèle ...)

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.
- Le programme du B.O. indique l'algorithme de la résolution par dichotomie. Le prendre comme objectif de façon à amener l'élève à le construire lui-même.

1/ Calculs sans vraie variable (périmètre et surface d'un cercle, exercice du DNB...)

2/ Des *si alors sinon* simples (dire si un nombre est pair, positif, dire si un point est sur un cercle, si un triangle est isocèle ...)

Manipulation de vraies variables

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.
- Le programme du B.O. indique l'algorithme de la résolution par dichotomie. Le prendre comme objectif de façon à amener l'élève à le construire lui-même.

1/ Calculs sans vraie variable (périmètre et surface d'un cercle, exercice du DNB...)

2/ Des *si alors sinon* simples (dire si un nombre est pair, positif, dire si un point est sur un cercle, si un triangle est isocèle ...)

Manipulation de vraies variables

3/ Permuter deux variables

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.
- Le programme du B.O. indique l'algorithme de la résolution par dichotomie. Le prendre comme objectif de façon à amener l'élève à le construire lui-même.

1/ Calculs sans vraie variable (périmètre et surface d'un cercle, exercice du DNB...)

2/ Des *si alors sinon* simples (dire si un nombre est pair, positif, dire si un point est sur un cercle, si un triangle est isocèle ...)

Manipulation de vraies variables

3/ Permuter deux variables

4/ Fonctionnement du feu tricolore (sans répétition puis avec répétition)

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.
- Le programme du B.O. indique l'algorithme de la résolution par dichotomie. Le prendre comme objectif de façon à amener l'élève à le construire lui-même.

1/ Calculs sans vraie variable (périmètre et surface d'un cercle, exercice du DNB...)

2/ Des *si alors sinon* simples (dire si un nombre est pair, positif, dire si un point est sur un cercle, si un triangle est isocèle ...)

Manipulation de vraies variables

3/ Permuter deux variables

4/ Fonctionnement du feu tricolore (sans répétition puis avec répétition)

5/ Jeu du "trop grand trop petit" sans limitation du nombre de coups pour mettre en évidence la stratégie de dichotomie. Difficile car la condition de sortie est au milieu.

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.
- Le programme du B.O. indique l'algorithme de la résolution par dichotomie. Le prendre comme objectif de façon à amener l'élève à le construire lui-même.

1/ Calculs sans vraie variable (périmètre et surface d'un cercle, exercice du DNB...)

2/ Des *si alors sinon* simples (dire si un nombre est pair, positif, dire si un point est sur un cercle, si un triangle est isocèle ...)

Manipulation de vraies variables

3/ Permuter deux variables

4/ Fonctionnement du feu tricolore (sans répétition puis avec répétition)

5/ Jeu du "trop grand trop petit" sans limitation du nombre de coups pour mettre en évidence la stratégie de dichotomie. Difficile car la condition de sortie est au milieu.

6/ Calcul d'un zéro de fonction par dichotomie. Commencer avec une boucle *pour*.

# Conseils pour la classe de seconde

- Mettre en place les concepts progressivement. Donner une certaine autonomie à l'élève.
- Le programme du B.O. indique l'algorithme de la résolution par dichotomie. Le prendre comme objectif de façon à amener l'élève à le construire lui-même.

1/ Calculs sans vraie variable (périmètre et surface d'un cercle, exercice du DNB...)

2/ Des *si alors sinon* simples (dire si un nombre est pair, positif, dire si un point est sur un cercle, si un triangle est isocèle ...)

Manipulation de vraies variables

3/ Permuter deux variables

4/ Fonctionnement du feu tricolore (sans répétition puis avec répétition)

5/ Jeu du "trop grand trop petit" sans limitation du nombre de coups pour mettre en évidence la stratégie de dichotomie. Difficile car la condition de sortie est au milieu.

6/ Calcul d'un zéro de fonction par dichotomie. Commencer avec une boucle *pour*.

7/ On s'arrête quand on est suffisamment proche de 0.

# Point commun à la programmation et aux mathématiques

# Point commun à la programmation et aux mathématiques

- la recherche et la manipulation  
d'**invariants**.

# Point commun à la programmation et aux mathématiques

- la recherche et la manipulation d'**invariants**.
- Chercher un programme, c'est chercher un invariant.