

prépaBAC



1^{re}
GÉNÉRALE

NSI

Numérique
& sciences
informatiques

SPÉCIALITÉ

**Cours &
entraînement**

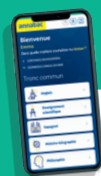
**NOUVEAU
BAC**

Pour maîtriser
ta spécialité!

- ✓ Un **cours** complet et visuel
- ✓ Des **projets** guidés
- ✓ **150 exos & sujets** guidés
- ✓ Des **corrigés** expliqués

INCLUS

Des mémos visuels 



OFFERT Un abonnement
au site de révision **annabac**



NSI

Numérique & sciences informatiques

SPÉCIALITÉ

Cours & entraînement

► Céline Adobet

Professeure agrégée de mathématiques
Docteur en mathématiques
Enseignante en informatique à l'IUT d'Orléans
Formatrice au DIU « Enseigner l'informatique au Lycée »

► Guillaume Connan

Professeur agrégé de mathématiques
Enseignant de mathématiques et d'informatique
en classes préparatoires au lycée Externat-Chavagnes
de Nantes
Formateur au DIU « Enseigner l'informatique au Lycée »

► Gérard Rozsavolgyi

Professeur agrégé de mathématiques
Responsable de la Licence professionnelle métiers de
l'informatique à l'IUT d'Orléans
Formateur au DIU « Enseigner l'informatique au Lycée »

► Laurent Signac

Docteur en informatique
Enseignant en informatique à l'école nationale
supérieure d'ingénieurs de Poitiers
Formateur au DIU « Enseigner l'informatique au Lycée »

Mode d'emploi

Voici les ressources que vous trouverez dans ce Prépabac :

Dans chaque chapitre

Un cours visuel

- ▶ 1. Des fiches synthétiques et illustrées
- ▶ 2. Des méthodes détaillées
- ▶ 3. Une carte mentale récapitulative

Un entraînement progressif

- ▶ 1. Des quiz pour se tester
- ▶ 2. Des exercices guidés
- ▶ 3. Des sujets de type bac

Des corrigés détaillés

Nous vous recommandons d'utiliser régulièrement l'ouvrage, en fonction de vos besoins. Ainsi, vous pourrez aborder vos contrôles et vos projets en toute sérénité et acquérir les compétences nécessaires pour la Terminale.

Les auteurs



Céline
Adobet



Guillaume
Connan



Gérard
Rozsavolgyi



Laurent
Signac

SOMMAIRE

1 Représentation des données : types et valeurs de base

COURS & MÉTHODES

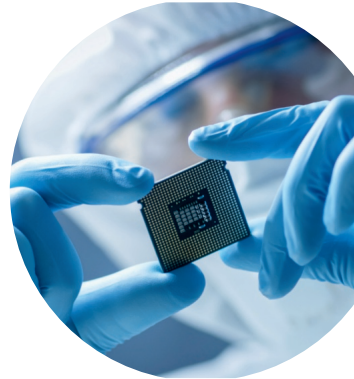
1	Représentation des entiers naturels en binaire	10
2	Représentation des entiers relatifs	12
3	Codage des nombres à virgule	14
4	Les enjeux du codage des nombres	16
5	Opérations fondamentales de l'algèbre booléenne	18
6	Fonctions logiques composées et lois de l'algèbre booléenne	20
7	Codage du texte	22

MÉMO VISUEL

EXERCICES & SUJETS	SE TESTER • S'ENTRAÎNER	26
--------------------	-------------------------	----

OBJECTIF BAC	33
--------------	----

CORRIGÉS	35
----------	----



2 Langages et programmation

COURS & MÉTHODES

8	De l'algorithme au calcul mécanique	46
9	Variables, affectations et opérations	48
10	Fonctions	50
11	Instructions conditionnelles	52

12	Boucles bornées	54
----	-----------------	----

13	Boucles non bornées	56
----	---------------------	----

MÉMO VISUEL	58
-------------	----

EXERCICES & SUJETS	SE TESTER • S'ENTRAÎNER	60
--------------------	-------------------------	----

OBJECTIF BAC	70
--------------	----

CORRIGÉS	72
----------	----



3 Représentation des données, types construits

COURS & MÉTHODES

14 Tuples, listes et itérations 84

15 Dictionnaires 86

16 Structures imbriquées et compréhensions 88

MÉMO VISUEL

EXERCICES & SUJETS SE TESTER • S'ENTRAÎNER 92

OBJECTIF **BAC** 100

CORRIGÉS 104



4 Données en tables

COURS & MÉTHODES

17 Manipulation de fichiers CSV 112

18 Operations sur les tables 114

19 Jointures de tables 116

MÉMO VISUEL

EXERCICES & SUJETS SE TESTER • S'ENTRAÎNER 120

OBJECTIF **BAC** 124

CORRIGÉS 128



5 Interface homme-machine et Web

COURS & MÉTHODES

20 Éléments d'interaction dans une page web 138

21 Interaction client-utilisateur, éléments de JavaScript 140

22 Interactions client-serveur – HTTP 142

23 Développement web côté serveur : PHP 144

24 Les grandes évolutions du Web 146

MÉMO VISUEL

EXERCICES & SUJETS SE TESTER • S'ENTRAÎNER 150

OBJECTIF **BAC** 155

CORRIGÉS 157



6 Architectures matérielles et systèmes d'exploitation



COURS & MÉTHODES

25	Histoire des ordinateurs	172
26	Architecture de von Neumann	174
27	Mémoire et langage machine	176
28	Linux et Bash	178
29	Arborescences et flux	180
30	Droits et permissions sous Unix	182
31	Réseaux et modèles en couches	184

MÉMO VISUEL

		186
--	--	-----

EXERCICES & SUJETS

SE TESTER • S'ENTRAÎNER	188
-------------------------	-----

OBJECTIF BAC	195
--------------	-----

CORRIGÉS

	202
--	-----

7 Algorithmes fondamentaux

COURS & MÉTHODES

32	La machine de Turing	214
33	Parcours séquentiels d'une liste	216
34	Recherche dichotomique dans une liste triée	218
35	Tri par insertion	220
36	Tri par sélection	222

MÉMO VISUEL

	224
--	-----

EXERCICES & SUJETS

SE TESTER • S'ENTRAÎNER	226
-------------------------	-----

OBJECTIF BAC	232
--------------	-----

CORRIGÉS

	235
--	-----



8 Quelques algorithmes avancés

COURS & MÉTHODES

37 Introduction à l'algorithme des k plus proches voisins 246

38 Implémentation de l'algorithme des k plus proches voisins 248

39 Exemple d'algorithme glouton : le rendu de monnaie 250

40 Intelligence artificielle, *machine learning* et *deep learning* 252

MÉMO VISUEL

EXERCICES & SUJETS SE TESTER • S'ENTRAÎNER 256

OBJECTIF **BAC** 266

CORRIGÉS 270



9 Projets informatiques



MÉTHODES

41 Gestion de projet et travail collaboratif 282

42 Systèmes de gestion de version 284

EXEMPLES DE PROJETS

43 Réaliser un catalogue de films en PHP 286

44 Créer des questionnaires d'évaluation 294

45 Réaliser un logiciel de traitement d'images 302

ANNEXES

Chronologie de l'histoire de l'informatique 313

Index 317

Le site de vos révisions

L'achat de ce Prépubac vous permet de bénéficier d'un **ACCÈS GRATUIT*** à toutes les **ressources** d'annabac.com (vidéos, fiches, quiz, sujets corrigés...) et à ses **parcours de révision** personnalisés.

Pour profiter de cette offre, rendez-vous sur **www.annabac.com** dans la rubrique « Je profite de mon avantage client ».



* Selon les conditions précisées sur le site.

1 Représentation des données : types et valeurs de base

En électronique, les opérations logiques de l'algèbre de Boole sont effectuées par des circuits qui combinent les signaux logiques présentés à leurs entrées sous forme de tensions : les portes logiques.

FICHES DE COURS

1	Représentation des entiers naturels en binaire	10
2	Représentation des entiers relatifs	12
3	Codage des nombres à virgule	14
4	Les enjeux du codage des nombres	16
5	Opérations fondamentales de l'algèbre booléenne	18
6	Fonctions logiques composées et lois de l'algèbre booléenne	20
7	Codage du texte	22

MÉMO VISUEL

24

EXERCICES & SUJETS

SE TESTER	Exercices 1 à 7	26
S'ENTRAÎNER	Exercices 8 à 21	28
OBJECTIF BAC	Exercice 22 • Sujet guidé	33

CORRIGÉS

Exercices 1 à 22	35
------------------	----

1

Représentations des entiers naturels en binaire

En bref

Nous sommes habitués à utiliser l'écriture en base 10 des entiers. Mais plus généralement, tout entier peut s'écrire dans une autre base, comme la base 2 ou la base 16, couramment utilisées en informatique car plus adaptées à la mise en mémoire des nombres.

I Écriture d'un entier en base β

■ **Définition :** Une **base** d'un **système de numération positionnel** est un entier naturel β supérieur ou égal à deux.

Dire qu'un nombre s'écrit $a_n a_{n-1} a_{n-2} \dots a_1 a_0_\beta$ en base β signifie qu'il est égal à :

$$a_n \times \beta^n + a_{n-1} \times \beta^{n-1} + a_{n-2} \times \beta^{n-2} + \dots + a_1 \times \beta + a_0$$

où les entiers naturels a_i sont strictement inférieurs à β .

■ **Exemple :** 237_{10} est égal à $2 \times 10^2 + 3 \times 10 + 7$ et $237_8 = 2 \times 8^2 + 3 \times 8 + 7 = 159_{10}$.



À NOTER

Dans l'écriture 237_{10} , le 10 en indice indique que le nombre est écrit en base 10.

II Les bases privilégiées en informatique

En informatique, on travaille en base 2 (les bits), en base 16 (adresses mémoire, couleurs HTML) et parfois en base 8 (droits des fichiers UNIX).

1 Écriture en base 16

Pour écrire un nombre en base 16, il faut disposer d'un caractère pour chacun des entiers de 0 à 15. Or, on ne dispose pas d'assez de chiffres pour écrire les 16 chiffres de la base 16. On complète donc les chiffres de 0 à 9 par les six premières lettres de l'alphabet : A, B, C, D, E et F.

Base 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Base 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11

2 Écriture en base 2

En base 2, tous les nombres sont représentés avec les deux symboles 0 et 1.

Base 10	0	1	2	3	4	5	6	7	8	9	10
Base 2	0	1	10	11	100	101	110	111	1000	1001	1010

■ **Exemple :** 1101_2 est égal à $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$.

III Conversions de la base 10 vers la base 2

1 Algorithme de soustraction

Pour convertir un nombre de la base 10 vers la base 2, on retranche du nombre la plus grande puissance de 2 possible.

Exemple :

$$58_{10} - 1 \times 2^5 = 58_{10} - 32_{10} = 26_{10} \quad 1$$

$$26_{10} - 1 \times 2^4 = 10_{10} \quad 1$$

$$10_{10} - 1 \times 2^3 = 2_{10} \quad 1$$

$$2_{10} - 0 \times 2^2 = 2_{10} \quad 0$$

$$2_{10} - 1 \times 2^1 = 0 \quad 1$$

$$0_{10} - 0 \times 2^0 = 0 \quad 0$$

Donc $58_{10} = 111010_2$.



À NOTER

En Python, on utilise des préfixes pour indiquer dans quelle base les nombres sont donnés.

0x pour la base 16, 0b pour la base 2 et 0o pour la base 8.

2 Algorithme de divisions

Pour convertir un nombre de la base 10 vers la base 2, on effectue des divisions successives de ce nombre par 2. En lisant les restes de bas en haut on obtient le résultat.

Exemple : $58_{10} = 111010_2$.



À NOTER

On utilise ce même algorithme de la division pour la conversion de la base 10 en n'importe quelle autre base.

$$\begin{array}{r}
 58 \mid 2 \quad 29 \mid 2 \quad 14 \mid 2 \quad 7 \mid 2 \quad 3 \mid 2 \quad 1 \mid 2 \\
 \underline{0} \quad \underline{1} \quad \underline{0} \quad \underline{1} \quad \underline{1} \quad \underline{0} \\
 \text{Lecture : } 111010
 \end{array}$$

IV Autres conversions

■ De la base 2 vers la base 16, on groupe les bits par paquets de 4, quitte à rajouter des 0 à gauche.

Exemple : $10101000011_2 = \underbrace{0101}_5 \underbrace{0100}_4 \underbrace{0011}_3 = 543_{16}$.

■ De la base 16 vers la base 2, on transforme chaque caractère par un groupe de 4 bits.

Exemple : $A3C_{16} = \underbrace{1010}_{A_{16}=10_{10}} \underbrace{0011}_3 \underbrace{1100}_{C_{16}=12_{10}} = 101000111100_2$.

■ De la base 16 à la base 10, on écrit les caractères en base 10 et on utilise la définition avec les puissances de la base de départ, ici 16.

Exemple :

$$\begin{aligned}
 A3C_{16} &= 10_{10} \times 16^2 + 3_{10} \times 16^1 + 12_{10} \times 16^0 \\
 &= 10_{10} \times 256_{10} + 3_{10} \times 16_{10} + 12_{10} \\
 &= 2\,620_{10}.
 \end{aligned}$$



À NOTER

On obtient par le même algorithme la conversion de n'importe quelle base vers la base 10.

2

Représentation des entiers relatifs

En bref

Le signe d'un nombre peut prendre deux valeurs (positif ou négatif), il suffit donc d'un bit pour le représenter. Mais les ordinateurs représentent les entiers négatifs à l'aide d'un codage plus approprié qui facilite les opérations arithmétiques : le complément à 2^n .

I Représentation des nombres entiers – convention

1 Le binaire non signé

■ Les entiers naturels sont codés sur machine en base 2 sur un nombre arbitraire de bits. Pour simplifier nos illustrations, nous considérerons des entiers codés sur **8 bits**.



À NOTER

On peut travailler sur 8 bits en Python grâce à la fonction `int8` de la bibliothèque `numpy`.

■ Dans la représentation binaire **non signée**, les nombres entiers naturels sont écrits en base 2.

Exemple : Le nombre « dix » en base 10, s'écrit 1010 en base 2. Dans la mémoire, sur 8 bits, il est codé 00001010.

■ Pour une mémoire à 8 bits, tous les entiers naturels de **0 à 255** peuvent donc être représentés de cette façon.

2 Le binaire signé

■ Dans la représentation en binaire signé, le bit de poids fort (le plus à gauche) sert à représenter le signe : 0 pour un entier positif et 1 pour un entier négatif.

■ Utiliser les n autres bits pour représenter la valeur absolue du nombre en binaire non signé pose problème : on se retrouverait avec deux zéros et l'addition ne fonctionnerait plus.

Exemple : Sur quatre bits, si on utilise l'algorithme d'addition habituel, $3 + (-2)$ serait égal à -5 :

$$\begin{array}{r} 0011 \rightarrow 3 \\ + 1010 \rightarrow -2 \\ \hline 1101 \rightarrow -5 \end{array}$$

■ On code les nombres signés avec le complément à 2^n .

II Le complément à 2ⁿ

Ce codage permet de représenter les entiers relatifs tout en ne changeant pas d'algorithme d'addition.

1 Définitions

■ Prendre le **complément à 1** d'un nombre consiste à inverser les 0 et les 1 de son écriture binaire. Par la suite on notera $\bar{b}$ (on dit « b barre ») le complément à 1 du bit b .

Exemples : Le complément à 1 de 1001 est 0110 ; $\bar{1} = 0$ et $\bar{0} = 1$.

■ L'**opposé** d'un nombre x est le nombre y vérifiant $x + y = 0$.

Exemple : L'opposé de 3 est -3 car $3 + (-3) = 0$.

■ Le **complément à 2ⁿ** d'un nombre, c'est le nombre qu'il faut lui ajouter pour obtenir 2^n .

2 Opposé d'un nombre en 8 bits

■ On remarque astucieusement qu'un nombre d'un octet plus son complément à 1 s'écrit 1111 1111.

Par exemple sur 8 bits : $1001\ 1010 + 0110\ 0101 = 1111\ 1111$.

On note $\bar{b}$ le complément à 1 du bit b (ainsi $\bar{1} = 0$ et $\bar{0} = 1$).

Alors, pour un nombre sur 8 bits :

$$(b_7b_6b_5b_4b_3b_2b_1b_0 + \bar{b}_7\bar{b}_6\bar{b}_5\bar{b}_4\bar{b}_3\bar{b}_2\bar{b}_1\bar{b}_0) + 1 = 11111111 + 1 = (1)00000000$$

$$\text{donc } b_7b_6b_5b_4b_3b_2b_1b_0 + \bar{b}_7\bar{b}_6\bar{b}_5\bar{b}_4\bar{b}_3\bar{b}_2\bar{b}_1\bar{b}_0 + 1 = 0.$$

■ Par convention l'**opposé d'un nombre positif** est son complément à $2^{\text{taille des entiers}}$ c'est-à-dire son complément à 1, plus 1.

Exemple : On cherche l'opposé du nombre positif 3.

3 s'écrit 0000 0011 sur 8 bits. Le complément à 1 est donc 1111 1100.

L'opposé (-3) s'écrit alors $1111\ 1100 + 1 = \mathbf{1111\ 1101}$.

■ Pour connaître le nombre que représente un entier négatif, on effectue la démarche inverse : on lui retranche 1 puis on prend son complément à 1.

Exemple : On cherche à quel entier relatif correspond 1011 0101. C'est un nombre négatif car son premier bit vaut 1.

On lui retranche 1 : $1011\ 0101 - 1 = 1011\ 0100$.

On prend le complément à 1 de ce dernier nombre : 0100 1011.

On convertit en base 10 (en lisant de droite à gauche) : $2^0 + 2^1 + 2^3 + 2^6 = 75$.

1011 0101 est donc l'écriture sur 8 bits de **-75**.

3

Codage des nombres à virgule

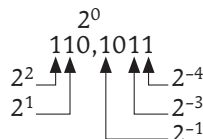
En bref Tout comme les nombres entiers relatifs, les nombres à virgule peuvent aussi être représentés en base 2.

I Conversion de la base 2 à la base 10

Les chiffres à gauche de la virgule correspondent à des puissances de 2 positives, ceux situés à droite correspondent à des puissances de 2 négatives.

Exemple :

$$\text{On a donc } 110,1011_2 = 2^2 + 2^1 + 2^{-1} + 2^{-3} + 2^{-4} \\ = 6,6875_{10}.$$



II Conversion de la base 10 vers la base 2

■ Commencer par convertir la partie entière → FICHE 1.

■ Pour convertir la partie décimale, procéder par multiplications par 2 successives. Après chaque multiplication le résultat est reporté sans sa partie entière. Le calcul se poursuit jusqu'à ce qu'on obtienne 1.

Exemple :

$$0,6875 \times 2 = 1,375$$

$$0,375 \times 2 = 0,75$$

$$0,75 \times 2 = 1,5$$

$$0,5 \times 2 = 1$$

Les parties entières (0 ou 1) des quatre résultats donnent les chiffres après la virgule de l'écriture binaire de 0,6875, à lire de haut en bas :

$$0,6875_{10} = 0,1011_2.$$

III Écritures infinies

Certains nombres ont une écriture décimale infinie périodique (par exemple $\frac{7}{11} = 0,636363\dots$). Et certains ont une écriture binaire infinie périodique, alors même que leur écriture décimale est finie.

Exemple :

$$0,2 \times 2 = 0,4$$

$$0,4 \times 2 = 0,8$$

$$0,8 \times 2 = 1,6$$

$$0,6 \times 2 = 1,2$$

$$0,2 \times 2 = 0,4$$

...

Ici, la ligne $0,2 \times 2 = 0,4$ a été obtenue deux fois. Aussi, les chiffres 0011 situés après la virgule vont se répéter indéfiniment. L'écriture en binaire de 0,2 est donc infinie et périodique : $0,2_{10} = 0,00110011\dots_2$



À NOTER

De tous les nombres ne comportant qu'un seul chiffre après la virgule en base 10, seul 0,5 a une écriture binaire finie.