



Langage C  
Claude Delannoy  
Delannoy, Claude (1946-....)  
Eyrolles, Paris  
Best of  
ISBN: 978-2-212-12445-3

## Table des Matières

Langage C

Claude Delannoy

Eyrolles

Avant propos	1
A qui s'adresse ce livre ?	1
Structure de l'ouvrage	2
A propos de la norme ANSI/ISO	5
A propos de la fonction main	5
Remerciements	6
Chapitre 1 - Généralités	7
1 - Historique du langage C	7
2 - Programme source, module objet et programme exécutable	8
3 - Compilation en C : existence d'un préprocesseur	9
4 - Variable et objet	10
4.1 Définition d'une variable et d'un objet	10
4.2 Utilisation d'un objet	11
5 - Lien entre objet, octets et caractères	12
6 - Classe d'allocation des variables	13
Chapitre 2 - Les éléments constitutifs d'un programme source	15
1 - Jeu de caractères source et jeu de caractères d'exécution	16
1.1 Généralités	16
1.2 Commentaires à propos du jeu de caractères source	17
1.3 Commentaires à propos du jeu minimal de caractères d'exécution	18
2 - Les identificateurs	19
3 - Les mots clés	20
4 - Les séparateurs et les espaces blancs	21
5 - Le format libre	22
6 - Les commentaires	23
7 - Notion de token	24
7.1 Les différentes catégories de tokens	24
7.2 Décomposition en tokens	26
Chapitre 3 - Les types de base	27
1 - Les types entiers	27
1.1 Les six types entiers	28

1.2 Représentation mémoire des entiers et limitations	29
1.3 Critères de choix d'un type entier	32
1.4 Écriture des constantes entières	34
1.5 Le type attribué par le compilateur aux constantes entières	34
1.6 Exemple d'utilisation déraisonnable de constantes hexadécimales	36
1.7 Pour imposer un type aux constantes entières	37
1.8 En cas de dépassement de capacité dans l'écriture des constantes entières	37
2 - Les types caractère	38
2.1 Les deux types caractère	38
2.2 Caractéristiques des types caractère	39
2.3 Écriture des constantes caractère	42
2.4 Le type des constantes caractère	45
3 - Le fichier limits.h	47
3.1 Son contenu	47
3.2 Précautions d'utilisation	48
4 - Les types flottant	49
4.1 Rappels concernant le codage des nombres en flottant	49
4.2 Le modèle proposé par la norme	50
4.3 Les caractéristiques du codage en flottant	50
4.4 Représentation mémoire et limitations	53
4.5 Écriture des constantes flottantes	54
4.6 Le type des constantes flottantes	55
4.7 En cas de dépassement de capacité dans l'écriture des constantes	55
5 - Le fichier float.h	55
6 - Déclarations des variables d'un type de base	57
6.1 Rôle d'une déclaration	58
6.2 Initialisation lors de la déclaration	59
6.3 Les qualifieurs const et volatile	60
Chapitre 4 - Les opérateurs et les expressions	63
1 - Généralités	64
1.1 Les particularités des opérateurs et des expressions en C	64
1.2 Priorité et associativité	65
1.3 Pluralité	66
1.4 Conversions implicites	67
1.5 Les différentes catégories d'opérateurs	67
2 - Les opérateurs arithmétiques	68
2.1 Les différents opérateurs numériques	68
2.2 Comportement en cas d'exception	72
3 - Les conversions numériques implicites	78
3.1 Introduction	78
3.2 Les conversions numériques d'ajustement de type	79
3.3 Les promotions numériques	85

3.4	<i>Combinaisons de conversions</i>	91
3.5	<i>Cas particulier des arguments d'une fonction</i>	94
4	<i>Les opérateurs relationnels</i>	95
4.1	<i>Généralités</i>	95
4.2	<i>Les six opérateurs relationnels du langage C</i>	96
4.3	<i>Leur priorité et leur associativité</i>	97
5	<i>Les opérateurs logiques</i>	99
5.1	<i>Généralités</i>	99
5.2	<i>Les trois opérateurs logiques du langage C</i>	100
5.3	<i>Leur priorité et leur associativité</i>	102
5.4	<i>Les opérandes de &amp;&amp; et de (...) ne sont évalués que si nécessaire</i>	103
6	<i>Les opérateurs de manipulation de bits</i>	103
6.1	<i>Présentation des opérateurs de manipulation de bits</i>	103
6.2	<i>Les opérateurs « bit à bit »</i>	104
6.3	<i>Les opérateurs de décalage</i>	107
6.4	<i>Applications usuelles des opérateurs de manipulation de bits</i>	109
7	<i>Les opérateurs d'affectation et d'incrémentement</i>	112
7.1	<i>Généralités</i>	112
7.2	<i>La lvalue</i>	113
7.3	<i>L'opérateur d'affectation simple</i>	114
7.4	<i>Tableau récapitulatif : l'opérateur d'affectation simple</i>	116
7.5	<i>Les opérateurs d'affectation élargie</i>	117
8	<i>Les opérateurs de cast</i>	119
8.1	<i>Généralités</i>	119
8.2	<i>Les opérateurs de cast</i>	120
9	<i>Le rôle des conversions numériques</i>	122
9.1	<i>Conversion d'un type flottant vers un autre type flottant</i>	123
9.2	<i>Conversion d'un type flottant vers un type entier</i>	123
9.3	<i>Conversion d'un type entier vers un type flottant</i>	124
9.4	<i>Conversion d'un type entier vers un autre type entier</i>	124
9.5	<i>Cas particuliers des conversions d'entier vers caractère</i>	125
9.6	<i>Tableau récapitulatif des conversions numériques</i>	127
10	<i>L'opérateur conditionnel</i>	129
10.1	<i>Introduction</i>	129
10.2	<i>Rôle de l'opérateur conditionnel</i>	130
10.3	<i>Contraintes et conversions</i>	130
10.4	<i>La priorité de l'opérateur conditionnel</i>	132
11	<i>L'opérateur séquentiel</i>	133
12	<i>L'opérateur sizeof</i>	134
12.1	<i>L'opérateur sizeof appliqué à un nom de type</i>	135
12.2	<i>L'opérateur sizeof appliqué à une expression</i>	136
13	<i>Tableau récapitulatif : priorités et associativité des opérateurs</i>	138

14 - Les expressions constantes	139
14.1 Introduction	139
14.2 Les expressions constantes d'une manière générale	140
Chapitre 5 - Les instructions exécutables	143
1 - Généralités	143
1.1 Rappels sur les instructions de contrôle	144
1.2 Classification des instructions exécutables du langage C	145
2 - L'instruction expression	146
2.1 Syntaxe et rôle	146
2.2 Commentaires	146
3 - L'instruction composée ou bloc	148
3.1 Syntaxe d'un bloc	148
3.2 Commentaires	148
3.3 Déclarations dans un bloc	150
3.4 Cas des branchements à l'intérieur d'un bloc	151
4 - L'instruction if	151
4.1 Syntaxe et rôle de l'instruction if	151
4.2 Exemples d'utilisation	152
4.3 Cas des if imbriqués	155
4.4 Traduction de choix en cascade	157
5 - L'instruction switch	158
5.1 Exemple introductif	158
5.2 Syntaxe usuelle et rôle de switch	160
5.3 Commentaires	161
5.4 Quelques curiosités de l'instruction switch	163
6 - Choix entre if et switch	165
7 - Les particularités des boucles en C	166
7.1 Rappels concernant la programmation structurée	166
7.2 Les boucles en C	167
8 - L'instruction do... while	168
8.1 Syntaxe	169
8.2 Rôle	169
8.3 Exemples d'utilisation	170
9 - L'instruction while	172
9.1 Syntaxe	172
9.2 Rôle	173
9.3 Lien entre while et do... while	174
9.4 Exemples d'utilisation	175
10 - L'instruction for	176
10.1 Introduction	177
10.2 Syntaxe	177
10.3 Rôle	178

10.4 Lien entre for et while	179
10.5 Commentaires	179
10.6 Exemples d'utilisation	181
11 - Conseils d'utilisation des différents types de boucles	183
11.1 Boucle définie	183
11.2 Boucle indéfinie	184
12 - L'instruction break	185
12.1 Syntaxe et rôle	185
12.2 Exemple d'utilisation	185
12.3 Commentaires	186
13 - L'instruction continue	187
13.1 Syntaxe et rôle	188
13.2 Exemples d'utilisation	188
13.3 Commentaires	189
14 - Quelques schémas de boucles utiles	191
14.1 Boucle à sortie intermédiaire	191
14.2 Boucles à sorties multiples	194
15 - L'instruction goto et les étiquettes	196
15.1 Les étiquettes	196
15.2 Syntaxe et rôle	197
15.3 Exemples et commentaires	198
Chapitre 6 - Les tableaux	203
1 - Exemple introductif d'utilisation d'un tableau	204
2 - Déclaration des tableaux	205
2.1 Généralités	206
2.2 Le type des éléments d'un tableau	206
2.3 Déclarateur de tableau	207
2.4 La dimension d'un tableau	209
2.5 Classe de mémorisation associée à la déclaration d'un tableau	211
2.6 Les qualifieurs const et volatile	212
2.7 Nom de type correspondant à un tableau	213
3 - Utilisation d'un tableau	214
3.1 Les indices	214
3.2 Un identificateur de tableau n'est pas une lvalue	215
3.3 Utilisation d'un élément d'un tableau	216
3.4 L'opérateur sizeof et les tableaux	216
4 - Arrangement d'un tableau et débordement d'indice	218
4.1 Les éléments d'un tableau sont alloués de manière consécutive	218
4.2 Aucun contrôle n'est effectué sur la valeur de l'indice	218
5 - Cas des tableaux de tableaux	219
5.1 Déclaration des tableaux à deux indices	219
5.2 Utilisation d'un tableau à deux indices	220

5.3 Peut-on parler de lignes et de colonnes d'un tableau à deux indices ?	222
5.4 Arrangement en mémoire d'un tableau à deux indices	222
5.5 Cas des tableaux à plus de deux indices	223
6 - Initialisation de tableaux	224
6.1 Initialisation par défaut des tableaux	225
6.2 Initialisation explicite des tableaux	226
Chapitre 7 - Les pointeurs	231
1 - Introduction à la notion de pointeur	232
1.1 Attribuer une valeur à une variable de type pointeur	232
1.2 L'opérateur* pour manipuler un objet pointé	233
2 - Déclaration des variables de type pointeur	234
2.1 Généralités	235
2.2 Le type des objets désignés par un pointeur	236
2.3 Déclarateur de pointeur	237
2.4 Classe de mémorisation associée à la déclaration d'un pointeur	239
2.5 Les qualifieurs const et volatile	240
2.6 Nom de type correspondant à un pointeur	244
3 - Les propriétés des pointeurs	245
3.1 Les propriétés arithmétiques des pointeurs	246
3.2 Lien entre pointeurs et tableaux	248
3.3 Ordre des pointeurs et ordre des adresses	254
3.4 Les restrictions imposées à l'arithmétique des pointeurs	254
4 - Tableaux récapitulatifs : les opérateurs +, -, &, * et [ ]	258
5 - Le pointeur null	260
6 - Pointeurs et affectation	262
6.1 Prise en compte des qualifieurs des objets pointés	262
6.2 Les autres possibilités d'affectation	265
6.3 Tableau récapitulatif	265
6.4 Les affectations élargies += et -= et les incréments ++ et --	266
7 - Les pointeurs génériques	267
7.1 Généralités	267
7.2 Déclaration du type void *	268
7.3 Interdictions propres au type void *	269
7.4 Possibilités propres au type void *	270
8 - Comparaisons de pointeurs	271
8.1 Comparaisons basées sur un ordre	271
8.2 Comparaisons d'égalité ou d'inégalité	272
8.3 Récapitulatif : les comparaisons dans un contexte pointeur	273
9 - Conversions de pointeurs par cast	274
9.1 Conversion d'un pointeur en un pointeur d'un autre type	274
9.2 Conversions entre entiers et pointeurs	277
9.3 Récapitulatif concernant l'opérateur de cast dans un contexte pointeur	278

Chapitre 8 - Les fonctions	279
1 - Les particularités des fonctions en C	280
1.1 Une seule sorte de module en C : la fonction	281
1.2 Fonction et transmission des arguments par valeur	282
1.3 Les variables globales	283
1.4 Les possibilités de compilation séparée	283
2 - Exemple introductif de la notion de fonction en langage C	284
3 - Définition d'une fonction	286
3.1 Les deux formes de l'en-tête	287
3.2 Les arguments apparaissant dans l'en-tête	288
3.3 La valeur de retour	289
3.4 Classe de mémorisation d'une fonction : extern et static	293
3.5 L'instruction return	294
4 - Déclaration et appel d'une fonction	298
4.1 Déclaration sous forme de prototype	299
4.2 Déclaration partielle (déconseillée)	300
4.3 Portée d'une déclaration de fonction	301
4.4 Redéclaration d'une fonction	303
4.5 Une définition de fonction tient lieu de déclaration	303
4.6 En cas d'absence de déclaration	305
4.7 Utilisation de la déclaration dans la traduction d'un appel	305
4.8 En cas de non-concordance entre arguments muets et arguments effectifs	308
4.9 Les fichiers en-tête standard	310
4.10 Nom de type correspondant à une fonction	311
5 - Le mécanisme de transmission d'arguments	311
5.1 Cas où la transmission par valeur est satisfaisante	311
5.2 Cas où la transmission par valeur n'est plus satisfaisante	313
5.3 Comment simuler une transmission par adresse avec des pointeurs	315
6 - Cas des tableaux transmis en arguments	317
6.1 Règles générales	318
6.2 Exemples d'applications	322
6.3 Pour qu'une fonction dispose de la dimension d'un tableau	325
6.4 Quelques conseils de style à propos des tableaux en argument	327
7 - Cas particulier des tableaux de tableaux transmis en arguments	329
7.1 Application des règles générales	329
7.2 Artifices facilitant la manipulation de tableaux de dimensions variables	333
8 - Les variables globales	340
8.1 Exemples introductifs d'utilisation de variables globales	341
8.2 Les déclarations des variables globales	344
8.3 Portée des variables globales	350
8.4 Variables globales et édition de liens	351
8.5 Les variables globales sont de classe statique	353

8.6 Initialisation des variables globales	353
9 - Les variables locales	356
9.1 La portée des variables locales	356
9.2 Classe d'allocation et initialisation des variables locales	358
10 - Tableau récapitulatif : portée, accès et classe d'allocation des variables	364
11 - Pointeurs sur des fonctions	365
11.1 Déclaration d'une variable pointeur sur une fonction	366
11.2 Affectation de valeurs à une variable pointeur sur une fonction	367
11.3 Appel d'une fonction par le biais d'un pointeur	370
11.4 Exemple de paramétrage d'appel de fonctions	372
11.5 Transmission de fonction en argument	374
11.6 Comparaisons de pointeurs sur des fonctions	375
11.7 Conversions par cast de pointeurs sur des fonctions	376
Chapitre 9 - Les entrées-sorties standard	379
1 - Caractéristiques générales des entrées-sorties standard	380
1.1 Mode d'interaction avec l'utilisateur	380
1.2 Formatage des informations échangées	381
1.3 Généralisation aux fichiers de type texte	382
2. Présentation générale de printf	383
2.1 Notions de format d'entrée, de code de format et de code de conversion	383
2.2 L'appel de printf	384
2.3 Les risques d'erreurs dans la rédaction du format	387
3 - Les principales possibilités de formatage de printf	390
3.1 Le gabarit d'affichage	391
3.2 Précision des informations flottantes	394
3.3 Justification des informations	396
3.4 Gabarit ou précision variable	397
3.5 Le code de format g	399
3.6 Le drapeau + force la présence d'un signe « plus »	402
3.7 Le drapeau espace force la présence d'un espace	403
3.8 Le drapeau 0 permet d'afficher des zéros de remplissage	403
3.9 Le paramètre de précision permet de limiter l'affichage des chaînes	405
3.10 Cas particulier du type unsigned short int : le modificateur h	406
4. Description des codes de format des fonctions de la famille printf	407
4.1 Structure générale d'un code de format	407
4.2 Le paramètre drapeaux	408
4.3 Le paramètre de gabarit	409
4.4 Le paramètre de précision	410
4.5 Le paramètre modificateur h//L	411
4.6 Les codes de conversion	412
4.7 Les codes utilisables avec un type donné	414
5 - La fonction putchar	415

5.1 Prototype	416
5.2 L'argument de putchar est de type int	416
5.3 La valeur de retour de putchar	417
6 - Présentation générale de scanf	417
6.1 Format de sortie, code de format et code de conversion	418
6.2 L'appel de scanf	418
6.3 Les risques d'erreurs dans la rédaction du format	420
6.4 La fonction scanf utilise un tampon	423
6.5 Notion de caractère invalide et d'arrêt prématuré	425
6.6 La valeur de retour de scanf	426
6.7 Exemples de rencontre de caractères invalides	428
7 - Les principales possibilités de scanf	430
7.1 La présentation des informations lues en données	431
7.2 Limitation du gabarit	433
7.3 La fin de ligne joue un rôle ambigu : séparateur ou caractère	434
7.4 Lorsque le format impose certains caractères dans les données	435
7.5 Attention au faux gabarit du code C	436
7.6 Les codes de format de la forme %[...]	437
8. Description des codes de format des fonctions de la famille de scanf	439
8.1 Récapitulatif des règles utilisées par ces fonctions	439
8.2 Structure générale d'un code de format	441
8.3 Les paramètres * et gabarit	441
8.4 Le paramètre modificateur h//L	442
8.5 Les codes de conversion	442
8.6 Les codes utilisables avec un type donné	445
8.7 Les différences entre les codes de format en entrée et en sortie	446
9. La fonction getchar	446
9.1 Prototype et valeur de retour	447
9.2 Précautions	447
Chapitre 10 - Les chaînes de caractères	451
1 - Règles générales d'écriture des constantes chaîne	452
1.1 Notation des constantes chaîne	452
1.2 Concaténation des constantes chaîne adjacentes	454
2 - Propriétés des constantes chaîne	455
2.1 Conventions de représentation	455
2.2 Emplacement mémoire	457
2.3 Cas des chaînes identiques	458
2.4 Les risques de modification des constantes chaîne	458
2.5 Simulation d'un tableau de constantes chaîne	460
3 - Créer, utiliser ou modifier une chaîne	461
3.1 Comment disposer d'un emplacement pour y ranger une chaîne	462
3.2 Comment agir sur le contenu d'une chaîne	463

3.3 Comment utiliser une chaîne existante	467
4 - Entrées-sorties standard de chaînes	469
4.1 Généralités	469
4.2 Écriture de chaînes avec puts	470
4.3 Écriture de chaînes avec le code de format %s de printf ou fprintf	471
4.4 Lecture de chaînes avec gets	473
4.5 Lecture de chaînes avec le code de format %s dans scanf ou fscanf	476
4.6 Comparaison entre gets et scanf dans les lectures de chaînes	477
4.7 Limitation de la longueur des chaînes lues sur l'entrée standard	478
5 - Généralités concernant les fonctions de manipulation de chaînes	482
5.1 Ces fonctions travaillent toujours sur des adresses	482
5.2 Les adresses sont toujours de type char *	483
5.3 Certains arguments sont déclarés const, d'autres pas	483
5.4 Attention aux valeurs des arguments de limitation de longueur	485
5.5 La fonction strlen	486
6 - Les fonctions de copie de chaînes	487
6.1 Généralités	487
6.2 La fonction strcpy	488
6.3 La fonction strncpy	491
7 - Les fonctions de concaténation de chaînes	495
7.1 Généralités	495
7.2 La fonction strcat	495
7.3 La fonction strncat	499
8 - Les fonctions de comparaison de chaînes	502
8.1 Généralités	502
8.2 La fonction strcmp	503
8.3 La fonction strncmp	504
9 - Les fonctions de recherche dans une chaîne	505
9.1 Les fonctions de recherche d'un caractère : strchr et strchr	505
9.2 La fonction de recherche d'une sous-chaîne : strstr	509
9.3 La fonction de recherche d'un caractère parmi plusieurs : strpbrk	509
9.4 Les fonctions de recherche d'un préfixe	511
9.5 La fonction d'éclatement d'une chaîne : strtok	512
10 - Les fonctions de conversion d'une chaîne en un nombre	516
10.1 Généralités	516
10.2 La fonction de conversion d'une chaîne en un double : strtod	518
10.3 Les fonctions de conversion d'une chaîne en entier : strtol et strtoul	522
10.4 Cas particulier des fonctions atof, atoi et atol	528
11 - Les fonctions de manipulation de suites d'octets	529
11.1 Généralités	530
11.2 Les fonctions de recopie de suites d'octets	530
11.3 La fonction memcmp de comparaison de deux suites d'octets	531

11.4 La fonction memset d'initialisation d'une suite d'octets	532
11.5 La fonction memchr de recherche d'une valeur dans une suite d'octets	533
Chapitre 11 - Les types structure, union et énumération	535
1 - Exemples introductifs	536
1.1 Exemple d'utilisation d'une structure	536
1.2 Exemple d'utilisation d'une union	539
2 - La déclaration des structures et des unions	540
2.1 Définition conseillée d'un type structure ou union	541
2.2 Déclaration de variables utilisant des types structure ou union	544
2.3 Déclaration partielle ou déclaration anticipée	547
2.4 Mixage entre définition et déclaration	548
2.5 L'espace de noms des identificateurs de champs	549
2.6 L'espace de noms des identificateurs de types	550
3 - Représentation en mémoire d'une structure ou d'une union	551
3.1 Contraintes générales	551
3.2 Cas des structures	552
3.3 Cas des unions	554
3.4 L'opérateur sizeof appliqué aux structures ou aux unions	554
4 - Utilisation d'objets de type structure ou union	555
4.1 Manipulation individuelle des différents champs d'une structure ou d'une union	556
4.2 Affectation globale entre structures ou unions de même type	557
4.3 L'opérateur & appliqué aux structures ou aux unions	559
4.4 Comparaison entre pointeurs sur des champs	559
4.5 Comparaison des structures ou des unions par == ou != impossible	560
4.6 L'opérateur ->	560
4.7 Structure ou union transmise en argument ou en valeur de retour	561
5 - Exemples d'objets utilisant des structures	565
5.1 Structures comportant des tableaux	565
5.2 Structures comportant d'autres structures	566
5.3 Tableaux de structures	567
5.4 Structure comportant des pointeurs sur des structures de son propre type	568
6 - Initialisation de structures ou d'unions	569
6.1 Initialisation par défaut des structures ou des unions	569
6.2 Initialisation explicite des structures	570
6.3 L'initialisation explicite d'une union	574
7 - Les champs de bits	575
7.1 Introduction	575
7.2 Exemples introductifs	575
7.3 Les champs de bits d'une manière générale	576
7.4 Exemple d'utilisation d'une structure de champs de bits dans une union	580
8 - Les énumérations	581
8.1 Exemples introductifs	581

8.2 Déclarations associées aux énumérations	584
Chapitre 12 - La définition de synonymes avec typedef	589
1 - Exemples introductifs	590
1.1 Définition d'un synonyme de int	590
1.2 Définition d'un synonyme de int*	590
1.3 Définition d'un synonyme de int[3]	591
1.4 Définition d'un synonyme d'un type structure	592
2 - L'instruction typedef d'une manière générale	593
2.1 Syntaxe	593
2.2 Définition de plusieurs synonymes	594
2.3 Imbrication des définitions de synonyme	594
3 - Utilisation de synonymes	594
3.1 Un synonyme peut s'utiliser comme spécificateur de type	594
3.2 Un synonyme n'est pas un nouveau type	595
3.3 Un synonyme peut s'utiliser à la place d'un nom de type	596
4 - Les limitations de l'instruction typedef	596
4.1 Limitations liées à la syntaxe de typedef	596
4.2 Cas des tableaux sans dimension	598
4.3 Cas des synonymes de type fonction	598
Chapitre 13 - Les fichiers	601
1 - Généralités concernant le traitement des fichiers	602
1.1 Notion d'enregistrement	602
1.2 Archivage de l'information sous forme binaire ou formatée	603
1.3 Accès séquentiel ou accès direct	603
1.4 Fichiers et implémentation	604
2 - Le traitement des fichiers en C	605
2.1 L'absence de la notion d'enregistrement en C	605
2.2 Notion de flux	606
2.3 Distinction entre fichier binaire et fichier formaté	607
2.4 Opérations applicables à un fichier et choix du mode d'ouverture	610
2.5 Accès séquentiel et accès direct	612
2.6 Le tampon et sa gestion	613
3 - Le traitement des erreurs de gestion de fichier	613
3.1 Introduction	613
3.2 La détection des erreurs en C	614
4 - Les entrées-sorties binaires : fwrite et fread	618
4.1 Exemple introductif de création séquentielle d'un fichier binaire	619
4.2 Exemple introductif de liste séquentielle d'un fichier binaire	621
4.3 La fonction fwrite	623
4.4 La fonction fread	627
5 - Les opérations formatées avec fprintf, fscanf, fputs et fgets	632
5.1 Exemple introductif de création séquentielle d'un fichier formaté	632

5.2 Exemple introductif de liste séquentielle d'un fichier formaté	635
5.3 La fonction <i>fprintf</i>	637
5.4 La fonction <i>fscanf</i>	640
5.5 La fonction <i>fputs</i>	644
5.6 La fonction <i>fgets</i>	647
6 - Les opérations mixtes portant sur des caractères	651
6.1 La fonction <i>fputc</i> et la macro <i>putc</i>	651
6.2 La fonction <i>fgetc</i> et la macro <i>getc</i>	654
7 - L'accès direct	659
7.1 Exemple introductif d'accès direct à un fichier binaire existant	659
7.2 La fonction <i>fseek</i>	661
7.3 La fonction <i>ftell</i>	663
7.4 Les possibilités de l'accès direct	665
7.5 Détection des erreurs supplémentaires liées à l'accès direct	667
7.6 Exemple d'accès indexé à un fichier formaté	670
7.7 Les fonctions <i>fsetpos</i> et <i>fgetpos</i>	673
8 - La fonction <i>fopen</i> et les différents modes d'ouverture d'un fichier	673
8.1 Généralités	673
8.2 La fonction <i>fopen</i>	674
9 - Les flux prédéfinis	677
Chapitre 14 - La gestion dynamique	679
1 - Intérêt de la gestion dynamique	679
2 - Exemples introductifs	681
2.1 Allocation et utilisation d'un objet de type double	681
2.2 Cas particulier d'un tableau	682
3 - Caractéristiques générales de la gestion dynamique	683
3.1 Absence de typage des objets	684
3.2 Notation des objets	685
3.3 Risques et limitations	685
3.4 Limitations	687
4 - La fonction <i>malloc</i>	687
4.1 Prototype	687
4.2 La valeur de retour et la gestion des erreurs	688
5 - La fonction <i>free</i>	689
6 - La fonction <i>calloc</i>	690
6.1 Prototype	690
6.2 Rôle	690
6.3 Valeur de retour et gestion des erreurs	691
6.4 Précautions	692
7 - La fonction <i>realloc</i>	692
7.1 Exemples introductifs	692
7.2 Prototype	694

7.3 Rôle	694
7.4 Valeur de retour	695
7.5 Précautions	695
8 - Techniques utilisant la gestion dynamique	696
8.1 Gestion de tableaux dont la taille n'est connue qu'au moment de l'exécution	696
8.2 Gestion de tableaux dont la taille varie pendant l'exécution	697
8.3 Gestion de listes chaînées	699
Chapitre 15 - Le préprocesseur	703
1 - Généralités	704
1.1 Les directives tiennent compte de la notion de ligne	704
1.2 Les directives et le caractère#	704
1.3 La notion de token pour le préprocesseur	705
1.4 Classification des différentes directives du préprocesseur	706
2 - La directive de définition de symboles et de macros	706
2.1 Exemples introductifs	707
2.2 La syntaxe de la directive #define	710
2.3 Règles d'expansion d'un symbole ou d'une macro	713
2.4 L'opérateur de conversion en chaîne : #	725
2.5 L'opérateur de concaténation de tokens : ##	728
2.6 Exemple faisant intervenir les deux opérateurs # et ##	730
2.7 La directive #undef	730
2.8 Précautions à prendre	731
2.9 Les symboles prédéfinis	736
3 - Les directives de compilation conditionnelle	737
3.1 Compilation conditionnelle fondée sur l'existence de symboles	737
3.2 Compilation conditionnelle fondée sur des expressions	740
3.3 Imbrication des directives de compilation conditionnelle	747
3.4 Exemples d'utilisation des directives de compilation conditionnelle	748
4 - La directive d'inclusion de fichier source	750
4.1 Généralités	750
4.2 Syntaxe	751
4.3 Précautions à prendre	752
5 - Directives diverses	755
5.1 La directive vide	755
5.2 La directive #line	756
5.3 La directive #error	756
5.4 La directive #pragma	757
Chapitre 16 - Les déclarations	759
1 - Généralités	760
1.1 Les principaux éléments : déclarateur et spécificateur de type	760
1.2 Les autres éléments	761
2 - Syntaxe générale d'une déclaration	763

2.1 <i>Forme générale d'une déclaration</i>	764
2.2 <i>Spécificateur de type structure</i>	766
2.3 <i>Spécificateur de type union</i>	768
2.4 <i>Spécificateur de type énumération</i>	769
2.5 <i>Déclarateur</i>	770
3 - Définition de fonction	772
3.1 <i>Forme moderne de la définition d'une fonction</i>	772
3.2 <i>Forme ancienne de la définition d'une fonction</i>	773
4 - Interprétation de déclarations	774
4.1 <i>Les règles</i>	774
4.2 <i>Exemples</i>	775
5 - Écriture de déclarateurs	777
5.1 <i>Les règles</i>	777
5.2 <i>Exemples</i>	778
Chapitre 17 - Fiabilisation des lectures au clavier	781
1 - Généralités	781
2 - Utilisation de scanf	783
3 - Utilisation de gets	784
4 - Utilisation de fgets	786
4.1 <i>Pour éviter le risque de débordement en mémoire</i>	786
4.2 <i>Pour ignorer les caractères excédentaires</i>	787
4.3 <i>Pour traiter l'éventuelle fin de fichier et paramétrer la taille des chaînes lues</i>	788
Chapitre 18 - Les catégories de caractères et les fonctions associées	791
1 - Généralités	791
1.1 <i>Dépendance de l'implémentation et de la localisation</i>	791
1.2 <i>Les fonctions de test</i>	792
2 - Les catégories de caractères	792
3 - Exemples	795
3.1 <i>Pour obtenir la liste de tous les caractères imprimables et leur code</i>	795
3.2 <i>Pour connaître les catégories des caractères d'une implémentation</i>	796
4 - Les fonctions de transformation de caractères	797
Chapitre 19 - Gestion des gros programmes	799
1 - Utilisation de variables globales	800
1.1 <i>Avantages des variables globales</i>	800
1.2 <i>Inconvénients des variables globales</i>	801
1.3 <i>Conseils en forme de compromis</i>	802
2 - Partage d'identificateurs entre plusieurs fichiers source	803
2.1 <i>Cas des identificateurs de fonctions</i>	803
2.2 <i>Cas des identificateurs de types ou de synonymes</i>	804
2.3 <i>Cas des variables globales</i>	805
Chapitre 20 - Les arguments variables	809
1 - Écriture de fonctions à arguments variables	809

1.1 Exemple introductif	810
1.2 Arguments variables, forme d'en-tête et déclaration	811
1.3 Contraintes imposées par la norme	813
1.4 Syntaxe et rôle des macros <i>va_start</i> , <i>va_arg</i> et <i>va_end</i>	813
2 - Transmission d'une liste variable	815
3 - Les fonctions <i>vprintf</i> , <i>vfprintf</i> et <i>vsprintf</i>	817
Chapitre 21 - Communication avec l'environnement	819
1 - Cas particulier des programmes autonomes	820
2 - Les arguments reçus par la fonction <i>main</i>	820
2.1 L'en-tête de la fonction <i>main</i>	820
2.2 Récupération des arguments reçus par la fonction <i>main</i>	821
3 - Terminaison d'un programme	823
3.1 Les fonctions <i>exit</i> et <i>atexit</i>	823
3.2 L'instruction <i>return</i> dans la fonction <i>main</i>	824
4 - Communication avec l'environnement	825
4.1 La fonction <i>getenv</i>	825
4.2 La fonction <i>system</i>	826
5 - Les signaux	826
5.1 Généralités	826
5.2 Exemple introductif	827
5.3 La fonction <i>signal</i>	829
5.4 La fonction <i>raise</i>	832
Chapitre 22 - Les caractères étendus	833
1 - Le type <i>wchar_t</i> et les caractères multioctets	834
2 - Notation des constantes du type <i>wchar_t</i>	835
3 - Les fonctions liées aux caractères étendus <i>mblen</i> , <i>mbtowc</i> et <i>wctomb</i>	836
3.1 Généralités	836
3.2 La fonction <i>mblen</i>	837
3.3 La fonction <i>mbtowc</i>	837
3.4 La fonction <i>wctomb</i>	838
4 - Les chaînes de caractères étendus	839
5 - Représentation des constantes chaînes de caractères étendus	839
6 - Les fonctions liées aux chaînes de caractères étendus : <i>mbstowcs</i> et <i>wcstombs</i>	840
6.1 La fonction <i>mbstowcs</i>	840
6.2 La fonction <i>wcstombs</i>	841
Chapitre 23 - Les adaptations locales	843
1 - Le mécanisme de localisation	843
2 - La fonction <i>setlocale</i>	844
3 - La fonction <i>localeconv</i>	846
Chapitre 24 - La récursivité	849
1 - Notion de récursivité	849
2 - Exemple de fonction récursive	850

3 - L'empilement des appels	851
4 - Autre exemple de récursivité	853
Chapitre 25 - Les branchements non locaux	857
1 - Exemple introductif	857
2 - La macro setjmp et la fonction longjmp	858
2.1 Prototypes et rôles	858
2.2 Contraintes d'utilisation	859
Chapitre 26 - Les incompatibilités entre C et C++	861
1 - Les incompatibilités raisonnables	862
1.1 Définition d'une fonction	862
1.2 Les prototypes en C++	862
1.3 Fonctions sans valeur de retour	862
1.4 Compatibilité entre le type void * et les autres pointeurs	863
1.5 Les déclarations multiples	863
1.6 L'instruction goto	864
1.7 Initialisation de tableaux de caractères	864
2 - Les incompatibilités incontournables	864
2.1 Fonctions sans arguments	864
2.2 Le qualifieur const	865
2.3 Les constantes de type caractère	866
Annexe A - La bibliothèque standard	867
1 - Généralités	868
1.1 Les différents fichiers en-tête	868
1.2 Redéfinition d'une macro standard par une fonction	869
2 - Assert.h : macro de mise au point	869
3 - Ctype.h : tests de caractères et conversions majuscules - minuscules	870
3.1 Les fonctions de test d'appartenance d'un caractère à une catégorie	870
3.2 Les fonctions de transformation de caractères	871
4 - Erro.h : gestion des erreurs	871
4.1 Constantes prédéfinies	871
4.2 Macros	871
5 - Locale.h : caractéristiques locales	872
5.1 Types prédéfinis	872
5.2 Constantes prédéfinies	872
5.3 Fonctions	872
6 - Math.h : fonctions mathématiques	873
6.1 Constantes prédéfinies	873
6.2 Traitement des conditions d'erreur	873
6.3 Fonctions trigonométriques	874
6.4 Fonctions hyperboliques	874
6.5 Fonctions exponentielle et logarithme	875
6.6 Fonctions puissance	875

6.7 <i>Autres fonctions</i>	876
7 - <i>Setjmp.h : branchements non locaux</i>	876
7.1 <i>Types prédéfinis</i>	876
7.2 <i>Fonctions et macros</i>	876
8 - <i>Signal.h : traitement de signaux</i>	877
8.1 <i>Types prédéfinis</i>	877
8.2 <i>Constantes prédéfinies</i>	877
8.3 <i>Fonctions et traitement de signaux</i>	877
9 - <i>Stdarg.h : gestion d'arguments variables</i>	878
9.1 <i>Types prédéfinis</i>	878
9.2 <i>Macros</i>	878
10 - <i>Stddef.h : définitions communes</i>	878
10.1 <i>Types prédéfinis</i>	878
10.2 <i>Constantes prédéfinies</i>	879
10.3 <i>Macros prédéfinies</i>	879
11 - <i>Stdio.h : entrées-sorties</i>	879
11.1 <i>Types prédéfinis</i>	879
11.2 <i>Constantes prédéfinies</i>	879
11.3 <i>Fonctions d'opérations sur les fichiers</i>	880
11.4 <i>Fonctions d'accès aux fichiers</i>	881
11.5 <i>Fonctions d'écriture formatée</i>	882
11.6 <i>Fonctions de lecture formatée</i>	883
11.7 <i>Fonctions d'entrées-sorties de caractères</i>	884
11.8 <i>Fonctions d'entrées-sorties sans formatage</i>	886
11.9 <i>Fonctions agissant sur le pointeur de fichier</i>	887
11.10 <i>Fonctions de gestion des erreurs d'entrée-sortie</i>	888
12 - <i>Stdlib.h : utilitaires</i>	889
12.1 <i>Types prédéfinis</i>	889
12.2 <i>Constantes prédéfinies</i>	889
12.3 <i>Fonctions de conversion de chaîne</i>	889
12.4 <i>Fonctions de génération de séquences de nombres pseudo-aléatoires</i>	892
12.5 <i>Fonctions de gestion de la mémoire</i>	892
12.6 <i>Fonctions de communication avec l'environnement</i>	893
12.7 <i>Fonctions de tri et de recherche</i>	894
12.8 <i>Fonctions liées à l'arithmétique entière</i>	895
12.9 <i>Fonctions liées aux caractères étendus</i>	895
12.10 <i>Fonctions liées aux chaînes de caractères étendus</i>	896
13 - <i>String.h : manipulations de suites de caractères</i>	897
13.1 <i>Types prédéfinis</i>	897
13.2 <i>Constantes prédéfinies</i>	897
13.3 <i>Fonctions de copie</i>	897
13.4 <i>Fonctions de concaténation</i>	898

13.5 Fonctions de comparaison	898
13.6 Fonctions de recherche	899
13.7 Fonctions diverses	901
14 - Time.h : gestion de l'heure et de la date	901
14.1 Types prédéfinis	901
14.2 Constantes prédéfinies	902
14.3 Fonctions de manipulation de temps	902
14.4 Fonctions de conversion	903
Annexe B - La norme C99	905
1 - Contraintes supplémentaires	905
1.1 Type de retour d'une fonction	905
1.2 Déclaration implicite d'une fonction	905
1.3 Instruction return	906
2 - Division d'entiers	906
3 - Emplacement des déclarations	906
4 - Commentaires de fin de ligne	907
5 - Tableaux de dimension variable	908
5.1 Dans les déclarations	908
5.2 Dans les en-têtes de fonctions et leurs prototypes	908
6. Nouveaux types	909
6.1 Nouveaux type entier long long	909
6.2 Types entiers étendus	910
6.3 Nouveaux types flottants	911
7 - Le type booléen	911
8 - Les types complexes	911
9 - Nouvelles fonctions mathématiques	913
9.1 Généralisation aux trois types flottants	913
9.2 Nouvelles fonctions	913
9.3 Fonctions mathématiques génériques	915
10 - Les fonctions en ligne	916
11 - Les caractères étendus	917
12 - Les pointeurs restreints	917
13 - La directive #pragma	918
14 - Les calculs flottants	918
14.1 La norme IEEE 754	918
14.2 Choix du mode d'arrondi	919
14.3 Gestion des situations d'exception	919
14.4 Manipulation de l'ensemble de l'environnement de calcul flottant	920
15 - Structures incomplètes	920
16 - Autres extensions de la norme C99	920
Index	921

