16-10-93

Résumé* → Langage C

"The C programming language" Brian Kernighan
Dennis Ritchie

Prentice Hall software series, 1988

chap. 1

- commentaires sur le passage "par valeur" comparativement à "par adresse" p. 27

- définition de "void" p. 30

- différences entre "déclaration" et "définition" p. 33

chap. 2

+++ - convention importante : constantes en majuscules
variables en minuscules p. 35

- les expressions reliés par && et || sont évaluées de gauche à droite et s'arrêtent dès que le résultat vrai ou faux est connu p. 41

* toutes les annotations faites ds le livre ne sont pas rapportées ici. Seulement les principales. Celles que je considérais comme surprenantes ou non habituelles pour moi au moment de faire le résumé!!!

+++ – Les expressions relationnelles ont une valeur 1 si elles sont vraies et 0 si fausses donc .... d = c >='0' && c<='9' .. c'est bon ! p.44

– définition du cast p.45

– Cas classique : p.46

```
n=5;        n=5;
x=n++;      x=++n;
.... x vaut 5   ... x vaut 6
```

+++ – Autre cas classique `s[j]=s[i]; j++;` équivaut à `s[j++]=s[i];` p.47

– Ne pas mélanger && et & ainsi que || et | p.49

+++ – `expr1 op= expr2` c'est équivalent à `expr1 = expr1 op expr2` p.50

`x *= y+1` " " " `x = x * (y+1)`

chap. 3

– définition de ; comparativement au Pascal p.55

+++ – Association du else avec le if... p.56

+++ – Les else-if p.57

chap. 4

- Fonction type et la valeur de retour par défaut... p. 70

+++ - la grande règle du "scope", la partie du programme de laquelle une variable est vue p. 80

+++ - pour limiter le "scope" d'une variable ou fonction : static p. 83

- on ne peut définir une fonction dans une autre fonction p. 84

- initialisation des variables statiques / automatiques p. 85

+++ - initialisation des tableaux p. 86

- truc pour éviter des inclusions multiples... p. 91

chap 5

+++ - Attention à ça ++*ip équivaut à ++(*ip) incrémente *ip p. 95
(*ip)++ n'équivaut pas à *ip++ qui incrémente le pointeur.

+++ - *(pa+i) c'est un "objet" plus loin que ça pointe p. 98

- Ne pas oublier pa = &a[0] équivaut à pa = a p. 99

+++ - passage de paramètres f(&a[2]), f(a+2) c'est ok p. 100
f(int an[]) f(int *a) c'est ok

- remarque sur l'adresse ∅. ∅ n'est jamais une adresse valide. p. 102

+++ - arithmétique des pointeurs 1- ==, !=, <, >= etc... marche pour les membres d'un même tableau — p.102

2- rôle de ptrdiff_t — p.103

3- Les opérations autorisées avec les pointeurs — p.103

+++ - Ça m'épate `char *pmessage; pmessage = "allo les filles";` — p.104

+++ - Différence `char amessage[] = "allo les filles"; char *pmessage = "allo les filles";` — p.104

+++ - Les étapes de `while ( (*s++ = *t++) != '\0') ....` — p.105

+++ - Push et pull avec des pointeurs ..... inverse ces deux classiques — p.106

- exemple d'initialisation d'un tableau multi-dimensionel pour de "petits" entiers — p.111

+++ - Les tableaux multi-dimensionels sont stockés par.... — p.112

+++ - un tableau multi-dimensionel en paramètre d'une fonction — p.112

+++ - différence entre `int a[10][30], *b[10]` — p.113

+++ - encore un mystère du tableau multi-dimensionel .... — p.114

+++ - `(*++argv)[0]` → augmentation du pointeur argv — p.117

`*++argv[0]` → augmentation du pointeur argv[0]

- appel d'une fct avec un pointeur de fct ! p.119

+++ - différence entre un pointeur de fct et une fct qui retourne un pointeur p.120, 122

- décoder des affaires compliquées p.122, 126

chap. 6

+++ définition d'une structure (avec ou sans nom) et initialisation p.128

- comparaison entre structures p.129

+++ - deux façon d'accéder aux membres d'une structure p.131

- truc pour définir les fonctions relatives à la taille des structures p.135

+++ - boucle avec un pointeur ← très intéressant p.137

++ - low, high deux pointeurs sur le début et la fin d'une table trouver le milieu ? p.138

- histoire de l'alignement p.138

- définition "récursive" : structure contenant un pointeur sur elle-même p.140

- définition de "hash table" p.144

- boucle "standard" pour parcourir une liste chainée p.145

- définition de "union" p.147

+++ - truc pour déterminer la valeur de flags p.149

- définition du bit field p.150

chap. 7

- différence entre prog <infile et prog | anotherprog p.152

- truc pour obtenir une longueur de champ variable à l'impression p.154

- truc pour lire un format d'entrée variable p.159

- remarque sur exit p.163
p.164

chap. 8

- définition du File descriptor p.169, 170

- définition des "low level read/write" p.170

- open, creat, close, unlink p.172-174

- structure "FILE" - contenu p.176

- définition d'un directory, inode p.179

- exemple d'un appel de fct avec un pointeur de fct p.182

- 2 points de philosophie pour l'écriture des programmes p.184

16-10-93

Problèmes classiques de C

1- {} non fermées !
2- pile trop petite → pb apparaît à l'entrée d'une fct
3- nb de fichiers ouverts trop grands
4- débordement de tableaux...

Questions

1- p.90 l'opérateur ##
2- p.103 je ne comprends pas l'intérêt...
3- p.111 False → 0 True → 1 ou vrai quelque soit l'implantation ?

```
int fi();           /* function returning int */
int ai[];           /* array of int */
int *pi;            /* pointer to int */

int *fpi();         /* function returning pointer to int */
int aai[][];        /* array of array of int */
int *api[];         /* array of pointer to int */
int (*pfi)();       /* pointer to function returning int */
int (*pai)[];       /* pointer to array of int */
int **ppi;          /* pointer to pointer to int */

int (*fpfi())();    /* function returning pointer to function returning int */
int (*fpai())[];    /* function returning pointer to array of int */
int **fppi();       /* function returning pointer to pointer to int */
int aaai[][][];     /* array of array of array of int */
int *aapi[][];      /* array of array of pointer to int */
int (*apfi[])();    /* array of pointer to function returning int */
int (*apai[])[];    /* array of pointer to array of int */
int **appi[];       /* array of pointer to pointer to int */
int *(*pfpi)();     /* pointer to function returning pointer to int */
int (*paai)[][];    /* pointer to array of array of int */
int *(*papi)[];     /* pointer to array of pointer to int */
int (**ppfi)();     /* pointer to pointer to function returning int */
int (**ppai)[];     /* pointer to pointer to array of int */
int ***pppi;        /* pointer to pointer to pointer to int */

int *(*fpfpi())();  /* function returning pointer to function returning pointer to i
int (*fpaai())[][]; /* function returning pointer to array of array of int */
int *(*fpapi())[];  /* function returning pointer to array of pointer to int */
int (**fppfi())();  /* function returning pointer to pointer to function returning i
int (**fppai())[];  /* function returning pointer to pointer to array of int */
int ***fpppi();     /* function returning pointer to pointer to pointer to int */
int aaaai[][][][];  /* array of array of array of array of int */
int *aaapi[][][];   /* array of array of array of pointer to int */
int (*aapfi[][])(); /* array of array of pointer to function returning int */
int (*aapai[][])[]; /* array of array of pointer to array of int */
int **aappi[][];    /* array of array of pointer to pointer to int */
int *(*apfpi[])();  /* array of pointer to function returning pointer to int */
int (*apaai[])[][]; /* array of pointer to array of array of int */
int *(*apapi[])[];  /* array of pointer to array of pointer to int */
int (**appfi[])();  /* array of pointer to pointer to function returning int */
int (**appai[])[];  /* array of pointer to pointer to array of int */
int ***apppi[];     /* array of pointer to pointer to pointer to int */
int (*(*pfpfi)())();/* pointer to function returning pointer to function returning i
int (*(*pfpai)())[];/* pointer to function returning pointer to array of int */
int **(*pfppi)();   /* pointer to function returning pointer to pointer to int */
```



```
int (*paaai)[][][]; /* pointer to array of array of array of int */
int *(*paapi)[][];  /* pointer to array of array of pointer to int */
int (*(*papfi)[])();/* pointer to array of pointer to function returning int */
int (*(*papai)[])[];/* pointer to array of pointer to array of int */
int **(*pappi)[];   /* pointer to array of pointer to pointer to int */
int *(**ppfpi)();   /* pointer to pointer to function returning pointer to int */
int (**ppaai)[][];  /* pointer to pointer to array of array of int */
int *(**ppapi)[];   /* pointer to pointer to array of pointer to int */
int (***pppfi)();   /* pointer to pointer to pointer to function returning int */
int (***pppai)[];   /* pointer to pointer to pointer to array of int */
int ****ppppi;      /* pointer to pointer to pointer to pointer to int */
```