Les types de variables

char	-127 à 128
int / long	-2 147 483 648 à 2 147 483 647
float	3.4 x 10 ^ -38 à 3.4 x 10 ^ 38
double	1.7 x 10 ^ -308 à 1.7 x 10 ^ 308
unsigned char	0 à 255
unsigned int / long	0 à 294 967 295
La table ASCII : correspondance nombre <-> caractère	
Constante : mot clé **const** avant le type.	

Les symboles d'affichage

%ld	Entier long (long)
%d	Entier court (int, [unsigned] char)
%c	Caractère (char)
%lu	Entier long non signé (unsigned long)
%u	Entier court non signé (unsigned int)
%lf	Flottant long (long double)
%f	Flottant (float, double)
%p	Adresse (un pointeur)
%s	Chaine de caractères (char[], string)

Les 5 opérations de base

+ - *	Addition, soustraction, multiplication
/	Division (résultat de la division euclidienne)
%	Modulo (reste de la division euclidienne)

Les pointeurs

int *pointeur = NULL;	Création du pointeur. **Un pointeur sans valeur vaut NULL**
pointeur	Adresse de la variable pointée.
***pointeur**	Contenu de la variable pointée.
&pointeur	Adresse du pointeur.

L'allocation dynamique

L'opérateur **sizeof** retourne la taille d'un(e) type/structure

malloc(nombre_octets);	Alloue de la mémoire	**NULL**
free(pointeur)	Libère de la mémoire	-

Exemple : int *var = malloc (sizeof *var * 3);

Les structures

Définition

```
typedef struct Structure Structure;
struct Structure
{
    /* variables */
};
```

Remarques

L'instruction **typedef** crée un alias d'une structure

. a une plus grande priorité que *
*(structure).var peut s'écrire structure->var

Les conditions

Les opérateurs de comparaison

==	Égalité
!=	Différence
> <	Comparaisons strictes
>= <=	Comparaisons larges

Les opérateurs logiques

&&	ET logique
||	OU logique
!(expression)	NON logique (négation)

Les structures conditionelles

```
if ( <condition_1> )
{
    <instruction_1>
}
else if ( <condition_2> )
{
    <instruction_2>
}
else // Sinon...
{
    <instruction_3>
}
```

```
switch(age)
{
    case <egalité_1>:
        <instruction_1>
    break;

    default:
        <default_instruction>
}
```

Les structures itératives (boucles)

```
while (<condition>)
{
    <instruction>
}
```

```
do
{
    // Exécuté au moins 1 fois
    <instruction>
} while (<condition>);
```

```
for (<initialisation> ; <condition> ; <incrément>)
{
    <instruction>
}
```

La librairie mathématique

Attention : cette librairie travaillent avec des nombres décimaux !

# include <math.h>	Directive préprocesseur
fabs(nombre)	Valeur absolue d'un décimal
Valeur absolue d'un entier : **abs(nombre)** - dispo dans <stdlib.h>	
ceil(nombre)	Retourne le nombre entier supérieur
floor(nombre)	Nombre entier inférieur
pow(nombre, exposant)	Calcul de puissances
sqrt(nombre)	Racine carée
sin(x), cos(x), tan(x) **asin(x), acos(x), atan(x)**	Fonctions trigonométriques. 1 radiant = 1 degré * 180 / Pi
exp(nombre)	Exponentielle d'un nombre décimal
log(nombre)	Logarithme népérien (de Néper)
log10(nombre)	Logarithme décimal (en base 10)
# define PI 3.14159265359	Définir le nombre Pi

Les chaînes de caractères

chaîne = tableau de char terminé par '\0' (caractère nul)

# include <string.h>	Directive préprocesseur	
strlen(chaine)	Calcule la longueur d'une chaîne	-
strcpy(copie, chaine)	Copie d'une chaîne dans une autre	-
strcat(chaine_1, chaine_2)	Ajoute chaine_2 après chaine_1	-
strcmp(chaine_1, chaine_2)	Comparaison binaire de chaînes	-
strchr(chaine, caractere)	Recherche de la première occurence	**NULL**
strrchr(chaine, caractere)	Recherche de la dernière occurence	**NULL**
strpbrk(chaine, liste_caracteres)	Recherche de plusieurs caractères	**NULL**
strstr(chaine, chaine_recherchee)	Recherche une chaîne dans une autre	**NULL**
sprintf(chaine, format, ...)	<stdio.h> - écrit une chaîne formatée	-

Les fichiers

# include <stdio.h>	Directive préprocesseur	
EOF est un nombre indiquant une erreur ou la fin du fichier		
fopen(nom_fichier, mode)	Ouvre un fichier	**NULL**
fclose(pfichier)	Ferme un fichier	**EOF**
fputc(caractere, pfichier)	Ecrit un caractère	**EOF**
fputs(chaine, pfichier)	Ecrit une chaîne	**EOF**
fprintf(pfichier, format, ...)	Ecrit une chaîne formatée	-
fgetc(pfichier)	Lit un caractère	**EOF**
fgets(&chaine, taille_max, pfichier)	Lit une chaîne	**NULL**
fscanf(pfichier, format, ...)	Lit une chaîne formatée	-
ftell(pfichier)	Position du pointeur de fichier	**-1**
fseek(pfichier, deplacement, origine)	Modifie la position du pointeur origine = SEEK_(SET|CUR|END)	**-1**
rewind(pfichier)	Replace le pointeur au début	**void**
rename(ancien_nom, nouveau_nom)	Renomme un fichier	**-1**
remove(nom_fichier)	Supprime un fichier	**-1**

Les modes d'ouverture d'un fichier

Mode	Lecture	Ecriture	Le fichier doit exister	Suppresion du contenu	Pointeur
r	oui	non	oui	non	**début**
r+	oui	oui	oui	non	**début**
w	non	oui	non	oui	**début**
w+	oui	oui	non	oui	**début**
a	non	oui	non	non	**fin**
a+	oui	oui	non	non	**fin**

Le préprocesseur

Les définitions

# define CONSTANTE [valeur]	Définition appliquée avant la compilation N'utilise pas de mémoire
	4 constantes prédéfinies :
__FILE__	Nom du fichier actuel
__LINE__	Numéro de la ligne actuelle
__DATE__	Date de la compilation
__TIME__	Heure de la compilation

Les conditions

# ifdef CONSTANTE	Indique si une constante est définie
# ifndef CONSTANTE	Indique si une constante n'est pas définie

Application

```
# ifndef DEF_FICHIER_H
# define DEF_FICHIER_H
    /* Contenu du fichier .h
    (prototypes, defines...) */
# endif
```

Condition du préprocesseur
(pour éviter les inclusions infinies)

Quelques définitions

stderr, stdin, stdout	<stdio.h>
SEEK_SET, SEEK_CUR, SEEK_END	<stdio.h>
EXIT_SUCCESS, EXIT_FAILURE	<stdlib.h>
size_t	<stdio.h>, <stdlib.h>, <time.h>
NULL	<stdlib.h>, <time.h>

par Savageman

Aide mémoire pour le langage C