API C

Basisconcepten C

Fundamentele datatypes	int, (long) double, char
int sizeof(*type/uitdrukking)*	lengte in bytes van type of uitdrukking
void (*type*) *uitdrukking*	expliciet uitdrukking tot type converteren
void printf(*formaatstring, argumenten…*)	printen van boodschap
formaatstring: "%[-][w][.p]k"	[-]: linkse alignatie [w]: minimale breedte [p]: max. aantal karakters/cijfers na komma k: specifiek: c -> karakter, s -> string, [l]f -> double/float, [l]d -> decimaal getal, [l]o -> octaal getal, [l]x -> hexadecimaal getal, e/E -> reëel getal in wet. notatie
int scanf(*formaatstring*, *argumenten…*)	inlezen van boodschap, argumenten -> &, negeert witruimte, behalve bij char
int getchar()	inlezen van één karakter
int putchar()	schrijven van één karakter

Pointers

void qsort(void *base, size_t item, size_t size, int (*compar)(const void *, const void *))	Sorteren van array: base -> pointer naar eerste element in array, item -> grootte array, size -> grootte van één element uit array, compar -> functie die twee elementen met elkaar vergelijkt
char * gets(char *s)	leest een array van chars in (C-string) ! kan buiten gereserveerd geheugen gaan !
char * fgets(char *s, int n, FILE *stream)	zelfde als gets, maar kan niet meer dan *int n* karakters inlezen, nullkarakter inbegrepen, *FILE *stream* -> stdin
size_t strlen(const char *s)	geeft de lengte van een C-string (zonder nullchar) terug
char * strcpy(char *dest, const char *src)	kopieert de inhoud van *src* naar *dest* ! alloceert geen geheugen !
int strcmp(const char *s1, const char *s2)	vergelijkt *s1* en *s2*: 0 -> gelijk, <0: s1 alfabetisch kleiner dan s2, >0: omgekeerd

char * strcat(char *dest, const char *src)	voegt twee C-strings samen in *dest* ! alloceert geen geheugen !
char * strncpy(char *dest, const char *src, size_t n)	zelfde als strcpy, met een maximum aantal karakters (n) die van *src* gekopieerd worden
char * strncat(char *dest, const char *src, size_t n)	zelfde als strcat, met een maximum aantal karakters (n) die van *src* gekopieerd worden

Dynamisch geheugenbeheer

void * malloc(size_t totaal_aantal_bytes)	reserveert *totaal_aantal_bytes* voor variabele waarop opgeroepen
void * calloc(size_t aantal, size_t bytes)	reserveert en initialiseert op 0 *aantal* keer *bytes* voor variabele waarop opgeroepen
void free(void *toegewezen_pointer)	geeft het geheugen dat voorheen gealloceerd was voor *toegewezen_pointer* terug vrij
void * realloc(void *toegewezen_pointer, size_t totaal_aantal_bytes)	poging tot hergebruik van *totaal_aantal_bytes* van *toegewezen_pointer*

Bit manipulation

& (AND)	1 als beide 1, 0 anders
| (OR)	0 als beide 0, 1 anders
^ (XOR, exclusive OR)	0 als beide 1 of beide 0, 1 anders
~ (NOT, 1 complement)	0 -> 1 en 1 -> 0
>> i	verplaatst de bits *i* keer naar rechts en voegt links nullen/enen toe unsigned types: nullen signed types: tekenbit of nullen
<< i	verplaatst de bits *i* keer naar links en voegt rechts nullen toe