*Pointeurs*

Pointeur : c’est une variable spéciale, qui est au lieu de stocker une valeur, il stock l’adresse mémoire d’une autre variable.

Exemple: int tab[100]; tab équivalent á &tab[0]; tab + i équivalent á &tab[i]; \*(tab + i) équivalent á tab[i];

*Allocation dynamique:*

*Malloc*

void\* malloc (size\_t size); malloc renvoie l'adresse du 1ère variable de bloc réservé, a comme paramètre le nombre d'octet à réserver ;il renvoie NULL si il y’a pas de l'espace a réservé; par exemple si on veut réserver 19 case de type float;

float\* tab = (float\*)malloc(19\*sizeof(float)); équivalent á float tab[19];

*Calloc*

void \*calloc (size\_t nb, size\_t size);comme malloc sauf que les cases sont initialisées par 0;

float\* tab = (float\*)calloc(19,sizeof(float));tous les case de tableau sont initialise par 0;

*Realloc*

void\* realloc (void \*p, size\_t newsize);sert à changer la taille (reallouer) d'un pointeurs, par exemple on veut changer la taille de 19 à 29;

tab = (float\*)realloc(tab,29\*sizeof(float));

*Free*

void free( void \*p );sert á libérer l'espace sur lequel p pointe( n’est pas libérer p); chaque utilisation de malloc ou calloc entraine obligatoirement libération de mémoire ave free() ;

free(tab);tab = NULL;

*Variable globale:* c'est une variable connue pour toutes les fonctions, se déclare endors des fonctions.

*Variable locale:* accessible juste au sein de sa fct;

*Variable static:* garde le comportement global dans les prochaines visites de la faction ou elle était déclarée;

*Chaines de caractères*

Chaine statique: char chaine[32];c'est un tableau de caractères;

gets(chaine) équivalente á scanf("%s", chaine);

puts(chaine) équivalente á printf("%s", chaine);

Chaine dynamique: char\* chaine = (char\*)malloc(32\*sizeof(char));

Chaine constantes: char\* chaine = "good days”; on ne peut pas modifier le contenu de cette chaine;

Size\_t strlen(const char \* s), strlen(chaine):renvoie le nombre de caractère d’une chaine de caractère donnée;

char \* strcpy (char \*dest, char \*src) équivalent à dire (dest recoit src);renvoie un ptr sur dest; la taille de dest doit être plus grand que src;

-char \* strdup (const char \* s): renvoie un ptr sur une copie de la chaine s;

-char \* strcat (char \*dest, const char \*src): sert à coller src à la fin de dest; il faut que la taille de dest soit suffisante pour accepter la concaténation;

-int strcmp(const char \*s1, const char \*s2);renvoie 1 si s1 > s2(code ASCI) ,0 en cas d’egalité,-1 s1 < s2;

-char \*strtok(char \*str, const char \*delim);renvoie un ptr sur la 1ere partie séparer par delim de la chaine str, on appelle une autre fois par strtok(NULL,delim) on aura l 2eme partie.....renvoie NULL si on arrive à la fin;

-int sprintf(char \*str, const char \*format, ...):sert à ecrire \*format dans str; exemple :

sprintf(chaine,"toto note = %d",note) 🡪 toto égale á 20 le contenu de chaine donc🡪 toto note 20

sscanf(const char \*str, const char \*format, ...): sert á lire á partir de str, exemple :

sscanf(chaine,"%d %s %c",&nb,ch,&c); chaine = 77 toto R, donc nb = 77,ch = toto, c = R;

*Gestion des fichiers:*

Déclaration de fichier: FILE\* fic;

fic = fopen("empalecmentPhysiqueDeFichier,"mode d'ouverture");fic point sur le début de fichier;

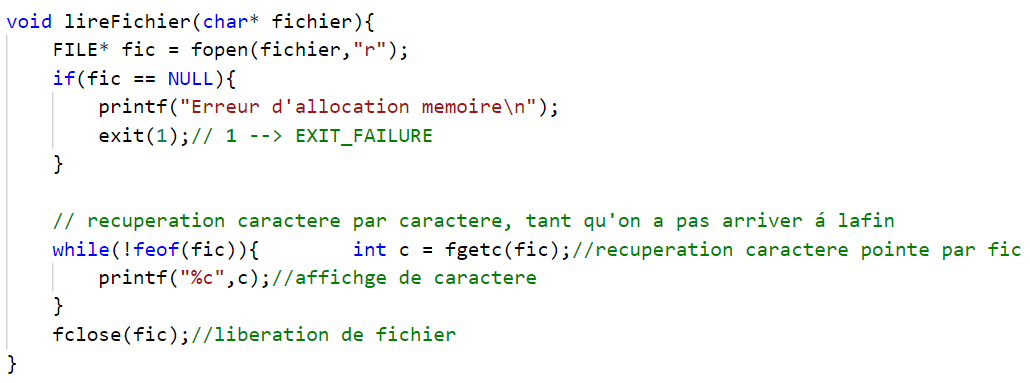
EX text.txt "r","w","a","r+","rb","r+b"...........;

fclose(fic); fic = NULL 🡪 pour libérer l'espace occupé par fic;

*Accès au contenu de fichier:*

-int fputc(int c, FILE\* fic):écrire caractère c dans fic;

-int fgetc(FILE \*fich):renvoie code asci de caractère pointé par fic; renvoid EOF en cas d'erreur ou la fin de fichier;



-feof(fic): renvoie true c'est on a arrivé à la fin de fichier 🡪 Généralement utiliser comme condition pour arrêter la lecture depuis un fichier.

-int fputs(const char\* chaine, FILE\* fic), écrire chaine dans le fichier fic; renvoie une valeur positive en cas de succès, EOF = -1 en cas d'echec;

-char\* fgets(char\* chaine, size\_t taille, FILE\* fic), retourne null a la fin de fichier ou en cas d'erreur; écrire ce qui est pointé par fic + la taille dans chaine;

fread(void \*Zone, size\_t Taille de l’enregistr, size\_t Nbr, FILE \*fp) ;

fwrite(void \*Zone, size\_t Taille de l’enregistr, size\_t Nbr, FILE \*fp) ;

Exemple

fread(tab,sizeof(int),10,fic) lire de fichier est mettre le resultat dans un tableau d'entiers de taille 10,

fwrite(tab,sizeof(int),10,fic):mettre un tableau d'entier de taille 10 dans un fichier;

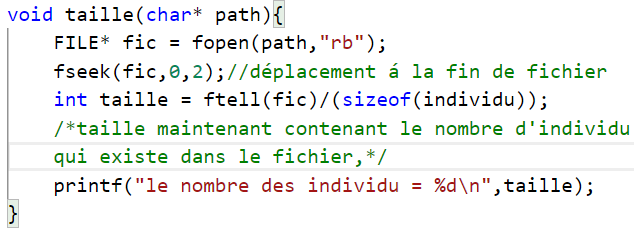
-int fseek(FILE \*fp, long int deplacement, int origine);

0 debut de fichier, 1 est l’emplacement courant de fp,2 c’est la fin de fichier;

-void rewind(FILE \*fich); retour au début du fichier équivalente avec fseek(fic,0,0)

-long ftell(FILE \*fich) retourne le nombre d’octet entre le point courant du pointeur et le début du fichier;

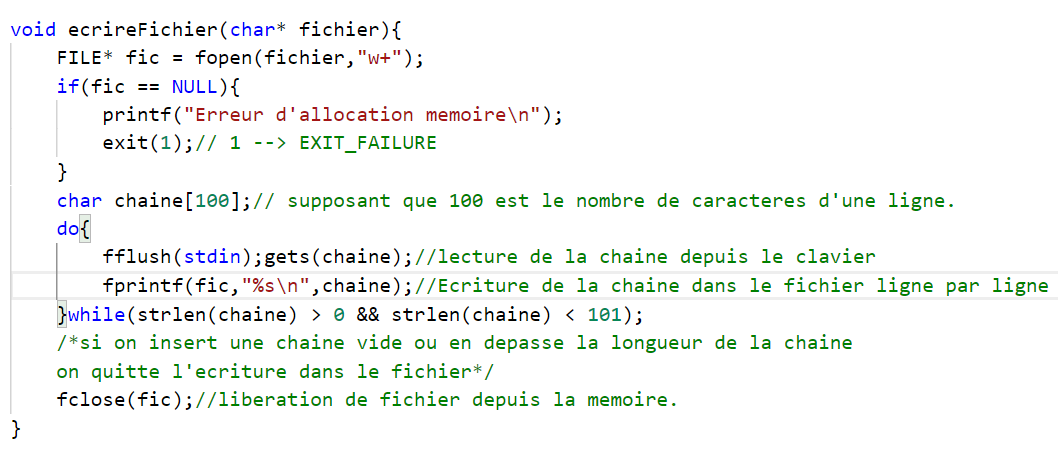
Ce programme permet de savoir le nombre des éléments d’une structure donnée.



-int rename(const char\* ancienNom, const char\* nouveauNom);Renomme un fichier;

-int remove(const char\* fichierASupprimer);supprimer un fichier;

-fprintf(flot,"format de contrôle",var1,var2,...):écrire dans un fichier en respectant le format, et c’est la meilleur façon d’écrire des chaine de caractères dans un fichier.



-fscanf(flot,"format de contrôle",var1,var2,...):lire du fichier et les mettre dans les variables,var1,var2....

FILE \*stdin, \*stdout, \*stderr;

stdin entree standart,(clavier);

stdout sortie standar,(ecran);

stderr sortie pour les messages des erreurs;

\*\*->structure

example:

typedef struct etudiant

{

char nom[max];

int code;

int nbNotes;

float\* note;

}etd,\*etud;

etd\* tab = (etd\*)malloc(\*N\*sizeof(etd));

tab[0] et de type etd pour afficher le nom de tab 0 en utilise : tab[0].nom;

tab[0].note[0] est le contenu de premier note de tab[0];