Entrées / Sorties

Les instructions d'entrées / sorties permettent de communiquer avec le programme, par le biais de saisies au clavier, d'affichages à l'écran, ou de fichiers.

Une instruction d'entrée lit des données fournies par l'utilisateur et les communique au programme. Une instruction de sortie communique des données à l'utilisateur.

Les données échangées peuvent se présenter sous un format prédéfini : il s'agit alors d'entrées / sorties formatées . Ce format est décrit dans une chaîne de format .

Voici les instructions d'entrées / sorties les plus utilisées :

Instruction	**Type**	**Action**
printf("*chaine_de_format*", *variable1*, …) ;	sortie formatée à l'écran	écrit dans le buffer de sortie standard **stdout** (écran) ; voir *Fiche 4* ***printf***
fprintf(*descripteur_de_fichier*, "*chaine_de_format*", *variable1,* …) ;	sortie formatée dans un fichier	écrit dans un fichier ; analogue à **printf**
putchar(*caractère*) ; Ex : **char c='k' ; putchar('a') ; putchar(c) ; putchar(64) ;** a k @	sortie non formatée à l'écran	écrit dans le buffer de sortie standard **stdout**
puts(*adresse_de_chaîne*) ;	sortie non formatée à l'écran	écrit dans le buffer de sortie standard **stdout** , le contenu de la chaîne jusqu'au 1er **\0** inclus, remplacé alors par un **\n**
fputs(*adresse_de_chaîne* , *descripteur_ de_fichier*) ;	sortie non formatée dans un fichier	écrit dans un fichier le contenu de la chaîne jusqu'au 1er **\0** non inclus
scanf("*chaine_de_format*", &*variable1*, …) ;	entrée formatée à partir du clavier	lit des données à partir du buffer d'entrée standard **stdin** (clavier) jusqu'au 1er **\n** non inclus; *voir Fiche 5* ***scanf***
fscanf(*descripteur_de_fichier*, "*chaine_de_format*", &*variable1,* …) ;	entrée formatée à partir d'un fichier	lit des données dans un fichier ; analogue à **scanf**
getchar(*caractère*) ; /* caractère lu et non récupéré */ ***variable_char* = getchar(*caractère*) ; /* caractère lu et récupéré dans *variable_char* */**	entrée non formatée à partir du clavier	lit un caractère à partir du buffer d'entrée standard **stdin** jusqu'au 1er **\n** non inclus
gets(*adresse_de_chaîne*) ;	entrée non formatée à partir du clavier	lit une chaîne de caractères à partir du buffer d'entrée standard **stdin** jusqu'au 1er **\n** inclus, remplacé alors par un **\0** et met le tout dans la chaîne
fgets(*adresse_de_chaîne*, *nombre,* *descripteur_de_fichier*) ;	entrée non formatée à partir d'un fichier	lit dans un fichier au maximum **(*nombre*-1)** caractères, ou s'arrête au 1er **\n** inclus ; rajoute ensuite **\0** et met le tout dans la chaîne

Attention : **scanf** et **getchar** lisent ce qui se trouve avant le 1er **\n** rencontré dans le buffer **stdin** , mais laissent, après avoir lu, le **\n** dans **stdin** . Donc, à la lecture suivante, si le buffer n'a pas été vidé, le 1er caractère rencontré étant le **\n** de la lecture précédente, **scanf** ou **getchar** considèreront que la nouvelle saisie a déjà été effectuée et ne la proposeront pas à l'utilisateur, d'où … un dysfonctionnement du programme ! Il existe 4 solutions à ce problème :

- ➔ insérer un **getchar()** « à vide » avant la deuxième instruction de saisie, pour récupérer le **\n**
- ➔ insérer un **fflush(stdin)** avant la deuxième instruction de saisie pour vider le buffer d'entrée
- ➔ commencer la chaîne de format du **scanf** suivant par un espace (**" %c"** au lieu de **"%c"** par exemple)
- ➔ empêcher la bufférisation dans **stdin** à l'aide de l'instruction **system("stty raw")** , quitte à la rétablir ultérieurement avec **system("stty cooked")**