Les **entrées-sorties** élémentaires

C utilise des fonctions

- Le langage C ne possède pas d'instruction d'e/s.
- Utiliser les fonctions de la bibliothèque <stdio.h> (à inclure)
  - stdin scanf()
  - stdout printf()

LIRE ET ÉCRIRE DES DONNÉES

- La bibliothèque standard *<stdio>* contient un ensemble de fonctions qui assurent la communication de la machine avec le monde extérieur:

printf()	écriture formatée de données
scanf()	lecture formatée de données
putchar()	écriture d'un caractère
getchar()	lecture d'un caractère

Ecriture: printf()

Impression formatée sur sortie standard d'un nombre variable de paramètres.

printf(``format``, liste_de_variables);

- **format:**chaîne de caractères imprimables et format d'écriture des variables

  %d, %f, %.2f, %s, …
- ***liste_de_variables***: liste de variables ou d'expressions, séparés par des virgules.
- **Exemple:**

printf(``prix : %.2f DH à payer avant le %d juin \n``, p, j);

Ecriture: printf();

- ***Spécificateurs de format pour printf ()***
- ***SYMBOLE TYPE IMPRESSION COMME***

%d ou %i	**int**	entier relatif
%u	**int**	entier naturel (unsigned)
%o	**int**	entier exprimé en octal
%x	**int**	entier exprimé en hexadécimal
%c	**int**	caractère
%f	**double**	rationnel en notation décimale
%e	**double**	rationnel en notation scientifique
%s	**char**	chaîne de caractères

Pour pouvoir traiter correctement les arguments du type **long**, il faut utiliser les spécificateurs **%ld, %li, %lu, %lo, %lx**.

printf() Suite

Un indicateur de format se construit ainsi:

%[flag][largeur][.précision][modificateur]type

- **flag**: **-** cadre à gauche, + fait précéder de son signe,…
- **largeur** minimum d'impression
- **La précision** indique le nombre maximum de caractère d'une chaîne à imprimer. Soit le nombre de chiffres à imprimer à droite du point décimal d'une valeur à virgule flottante, soit le nombre de chiffre à imprimer pour un entier.
- **Modificateur**: h (short), l (long pour entiers), L (doubles pour réels)
- **Le type**:

d: entier signé u: entier non signé o: octal

x: hexadécimal e: réel not. exp. f: réel avec virgule

c: caractère s: chaîne de caractère

Printf() exemples

- printf(``%d, %o, %x, %c\n``,75,75,75,75);
  Donne: 75, 113, 4b, K

- %f\t%f\n donne 2 doubles séparés par une tabulation et suivis d'un retour à la ligne.

- %6.1f\n affiche un flottant avec une largeur d'au moins 6 caractères, une virgule décimale et un chiffre fractionnaire.

- Traduction des instructions : écriture

Syntaxe en algo	Syntaxe en C
Ecrire()	printf()

Exemple :

Type de la valeur	Syntaxe en algo	Syntaxe en C
Une chaîne	Ecrire("Bonjour")	printf("Bonjour") ;
X est entier	Ecrire(X)	printf("%d", X) ;
Y est un réel	Ecrire(Y)	printf("%f", Y) ;
Z est un caractère	Ecrire(Z)	printf("%c", Z) ;
Une expression	Ecrire(" La valeur de X =", X , " et de Y =",Y)	printf(" La valeur de X = %d et de Y =%f ", X, Y);

Pour pouvoir utiliser la fonction printf(), il faut ajouter au début de votre fichier :

#include<stdio.h>

- Exercice

Traduire en C

```
Algorithme echange
Variables A, B, C : Entier
Début
        A ← 3
        B ← 2
        Ecrire(" Avant échange")
        Ecrire(" La valeur de A =", A , " et de B =", B)
        C ← B
        B ← A
        A← C
        Ecrire(" Après échange")
        Ecrire(" La valeur de A =", A , " et de B =", B)
Fin
```

- Corrigé

Traduire en C

```
Algorithme echange
Variables A, B, C : Entier
Début
        A ← 3
        B ← 2
        Ecrire(" Avant échange")
        Ecrire(" La valeur de A =", A , " et de B =", B)
        C ← B
        B ← A
        A← C
        Ecrire(" Après échange")
        Ecrire(" La valeur de A =", A , " et de B =", B)
Fin
```

```
#include<stdio.h>
#include<stdlib.h>

int main ( )
{
        int  A, B, C;
        A  = 3;
        B  = 2 ;
        printf(" Avant échange");
        printf("La valeur de A =%d et de B =%d",A, B) ;
        C = B ;
        B = A ;
        A = C ;
        printf(" Après échange") ;
        printf("La valeur de A =%d et de B =%d",A, B);

        return EXIT_SUCCESS;
}
```

Lecture: scanf()

- Lecture formatée de l'entrée standard d'un nombre variable d'arguments.

  scanf(``format``,liste_d_adresses_de_variables);

- Le format est du type %d, %f, %c, etc …
- Les adresses sont séparées par des virgules.

  &variable, nom d'un tableau de caractères, pointeur.

- **Exemple:**

```
scanf(``%d heures %d minutes``, &h, &m);
scanf("%i %i %i", &JOUR, &MOIS, &ANNEE);
```

- Traduction des instructions : lecture

Syntaxe en algo	Syntaxe en C
Lire()	scanf()

Exemple :

Type de la valeur	Syntaxe en algo	Syntaxe en C
X est entier	Lire(X)	scanf("%d", &X) ;
Y est un réel	Ecrire(Y)	printf("%f", &Y) ;
Z est un caractère	Ecrire(Z)	printf("%c", &Z) ;

Pour pouvoir utiliser la fonction scanf(), il faut ajouter au début de votre fichier :

#include<stdio.h>

- Exercice

Traduire en C

```
Algorithme echange
Variables A, B, C : Entier
Début
        Ecrire("Donner A")
        Lire(A)
        Ecrire("Donner B")
        Lire(B)
        C ← B
        B ← A
        A← C
        Ecrire("Après échange")
        Ecrire("La valeur de A =", A , " et de B =", B)
Fin
```

- Corrigé

Traduire en C

```
Algorithme echange
Variables A, B, C : Entier
Début
        Ecrire("Donner A")
        Lire(A)
        Ecrire("Donner B")
        Lire(B)
        C ← B
        B ← A
        A← C
        Ecrire("Après échange")
        Ecrire("La valeur de A =", A , " et de B =", B)
Fin
```

```
#include<stdio.h>
#include<stdlib.h>
int main ( )
{
        int  A, B, C;
        printf("Donner A");
         scanf(" %d",&A);
        printf("Donner B");
        scanf("%d ",&B) ;
        C = B ;
        B = A ;
        A = C ;
        printf("Après échange") ;
        printf("La valeur de A =%d et de B =%d",A, B);
        return EXIT_SUCCESS;
}
```

Lecture: scanf()

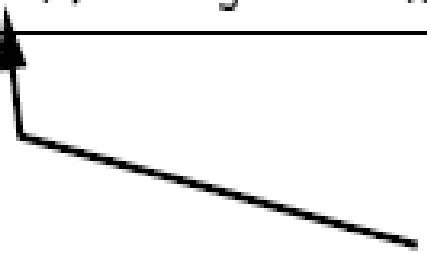
Il est possible d'ignorer des chaînes de caractères par *:

scanf(``%d %*s %d %*s``,&h;&m);

- Attention, les arguments sont toujours des adresses (voir passage de paramètres +loin)

Lecture: scanf()

Variable	Instruction
`int i;`	`scanf("%i", &i);`
`double d;`	`scanf("%lf", &d);`
`float f;`	`scanf("%f", &f);`
`short int h;`	`scanf("%hi", &h);`
`long int l;`	`scanf("%li", &l);`
`char c;`	`scanf("%c", &c);` ou `c=getchar();`



Remarquer le &

Lecture: scanf()

- ***1. Le type long***

Si nous voulons lire une donnée du type **long**, nous devons utiliser les spécificateurs **%ld**, **%li**, **%lu**, **%lo**, **%lx**. (Sinon, le nombre est simplement coupé à la taille de **int**).

- ***2. Le type double***

Si nous voulons lire une donnée du type **double**, nous devons utiliser les spécificateurs **%le** ou **%lf**.

- ***3. Le type long double***

Si nous voulons lire une donnée du type **long double**, nous devons utiliser les spécificateurs **%Le** ou **%Lf**.

- ***4. Indication de la largeur maximale***

Pour tous les spécificateurs, nous pouvons indiquer la *largeur maximale* du champ à évaluer pour une donnée. Les chiffres qui passent au-delà du champ défini sont attribués à la prochaine variable qui sera lue !

- ***Exemple***

Soient les instructions:

int A,B;

scanf("%4d %2d", &A, &B); Si nous entrons le nombre **1234567**, nous obtiendrons les affectations suivantes: **A=1234 B=56** le chiffre 7 sera gardé pour la prochaine instruction de lecture.

- ***5. Les signes d'espacement***

Lors de l'entrée des données, une suite de signes d'espacement (espaces, tabulateurs, interlignes) est évaluée comme un seul espace. Dans la chaîne de format, les symboles **\t**, **\n**, **\r** ont le même effet qu'un simple espace.

Écriture d'un caractère

- **La commande,**

putchar('a');

- ***Exemples***

```
char B = '\a';
 putchar('x');      /* afficher la lettre x */
 putchar('?');       /* afficher le symbole ?*/
 putchar('\n');     /* retour à la ligne */
 putchar(65);       /* afficher le symbole avec */ /*
                    le code 65 (ASCII: 'A') */
putchar(B);                 /* beep sonore */
putchar(EOF);    /* marquer la fin du fichier */
```

Lecture d'un caractère

C = getchar();

getchar lit les données de la zone tampon de *stdin* et fournit les données seulement après confirmation par 'Enter'. La bibliothèque *<conio.h>* contient une fonction du nom **getch()** qui fournit immédiatement le prochain caractère entré au clavier.

- La fonction **getch()** n'est pas compatible avec ANSI-C et elle peut seulement être utilisée sous MS-DOS