Type	**Occupation mémoire**	**Plage de valeurs**
char	1 octet	-128 à 127
unsigned char	1 octet	0 à 255
int	2 ou 4 octets	Selon l'architecture
unsigned int	2 ou 4 octets	Selon l'architecture
short	2 octets	-32768 à 32767
unsigned short	2 octets	0 à 65535
long	4 octets	-2147483648 à 2147483647
unsigned long	4 octets	0 à 4294967295
long long	8 octets	-2^{63} à $2^{63}-1$
unsigned long long	8 octets	0 à $2^{64}-1$
Type à virgule flottante		
float	4 octets	3.4×10^{-38} à 3.4×10^{38} (IEEE 754)
double	8 octets	1.7×10^{-308} à 1.7×10^{308} (IEEE 754)
long double	10 octets	3.4×10^{-4932} à 3.4×10^{4932} (IEEE 754)

Tab. 1 – Types de données élémentaires

Format	**Type de donnée**	
%d	Entier	`short, int`
%i	Entier	`short, int`
%u	Entier non signé	`unsigned short, unsigned int`
%o	Entier octal	`short, int`
%x, %X	Entier hexadécimal	`short, int`
%l, %ld, %li, %lu, %lo, %lx, %lX	Entier long	`long`
%lld, %lli, %llu, %llo, %llx, %llX	Entier long 64 bits	`long long`
%c	Caractère ASCII	`char, unsigned char`
%f, %F, %g, %G	Flottant	`float, double`
%e, %E	Flottant (exponentiel)	`float, double`
%Lf, %LF, %Lg, %LG, %Le, %LE	long double	`long double`
%s	Chaine de caractère	`char *`
%p	Pointeur	`(type) *`

Tab. 2 – (Quelques) Formatages de la fonction `printf`

Séquence d'échappement	**Action**
\n	Nouvelle ligne (`new line`)
\t	Tabulation horizontale
\v	Tabulation verticale
\b	Retour d'un caractère arrière (`backspace`)
\r	Retour chariot (`carriage return`)
\f	Saut de page (`form feed`)
\a	Signal sonore (`alarm`)
\'	Affiche une apostrophe
\"	Affiche un guillemet
\\	Affiche un `Backslash`
\ddd	Affiche les codes ASCII en octale
\xddd	Affiche les codes ASCII en hexadécimale

Tab. 3 – Séquences d'échappement

```c
#include <stdio.h>
int main(void) {
  printf("Hello World\n");
  return 0;
}
```

FIG. 1 – Premier programme

```c
#include <stdio.h>
/* Ceci est un commentaire */
void p01(void) {
  printf("Hello World\n");
}
void p02(void) {
  char chaine[] = "Hello World\n";
  int i;
  for(i = 0; i < 12; i++)
    putchar(chaine[i]);
}
void p03(char ch[]) {
  printf("%s", ch);
}
void p04(char ch[]) {
  while(*ch != '\0')
    putchar(*ch++);
}
int main(void) { /* Main est la fonction principale */
  char chaine[] = "Hello World\n";
  p01();
  p02();
  p03(chaine);
  p04(chaine);
  return 0;
}
```

FIG. 2 – Différentes méthodes pour imprimer une chaîne de caractères

```c
main() {
  int i;
  char num[] = "123456"; /* Mettez votre numero d'etudiant a la place */
  for(i = 0; num[i] != '\0'; i++)
    putchar('a' + num[i] - '0');
  putchar('\n');
}
```

FIG. 3 – Que fait ce programme ? (exo01.c)

Devoir 01 :

1. Envoyer par email le résultat de la compilation (`gcc -Wall exo01.c -o exo01`) et de l'exécution du programme de la figure 3. Que fait ce programme ?
2. Écrire les fonctions :
   - `int prod(int n)` retournant le produit des n premiers entiers strictement positifs ;
   - `int somme(int n)` retournant la somme des n premiers entiers ;
   - `int prodp(int n)` retournant le produit des n premiers entiers strictement positifs et pairs ;
   - `int sommep(int n)` retournant la somme des n premiers entiers pairs ;
   - `int prodi(int n)` retournant le produit des n premiers entiers impairs ;
   - `int sommei(int n)` retournant la somme des n premiers entiers impairs ;
   - `int puissance(int x, int n)` retournant x à la puissance n.

   Utiliser ces fonctions dans un même programme affichant le retour de chacune d'entre-elles en utilisant la fonction `printf`. Envoyer le code source par email (en attachement).