



Extrait du référentiel : BTS Systèmes Numériques option A (Informatique et Réseaux)		Niveau(x)
S4. Développement logiciel S4.4. Programmation procédurale	Manipulations de données (« quoi ») en pseudo-langage et/ou en langage C	4
	Transcription d'algorithmes (« comment ») en pseudo-langage et/ou en langage C	4
	Développement de programmes « console » avec gestion des arguments de la ligne de commande	3

### Objectifs du TP :

- Les types de programme :
  - en fenêtres
  - en console
- La compilation
- Les composants élémentaires du C
- Le code minimal :
  - bibliothèque : stdio.h (directive de préprocesseur)
  - fonction
  - instruction
  - argument
- Commentaire, déclaration de variables, affectation
  - Récupérer et afficher une saisie clavier (printf, scanf)

### Support d'activité :

- Logiciels : Visual Studio ou CodeBlocks
- Ce document au format PDF

**VOUS RÉDIGEREZ UN COMPTE-RENDU NUMÉRIQUE**

Le langage de programmation en C est à la base des systèmes d'exploitation que nous connaissons aujourd'hui ou au moins du noyau de ces systèmes comme par exemple Unix/Linux. Le Langage C a justement été créé pour un seul et unique but au départ, développer un système d'exploitation (Unix) mais au fil du temps, grâce à sa puissance, il a été adopté par une large communauté de développeurs ce qui a permis au langage d'évoluer et surtout d'être standardisé.

Ce langage est **multi plate-forme**, c'est-à-dire qu'un programme que vous créez par exemple sous Linux pourrais tout à fait être recompilé sous Windows, ... sans devoir changer grand-chose (voir rien du tout) dans le code source si vous respectez les normes en vigueur ANSI/ISO.

Vérifiez la « popularité » du langage C avec l'adresse ci-dessous :

<https://www.tiobe.com/tiobe-index/>

Comme vous pouvez le constater le langage C occupe une bonne place dans le classement.

## LES TYPES DE PROGRAMME

Il existe deux types de programmes :

- les programmes avec fenêtres ;
- les programmes en console.

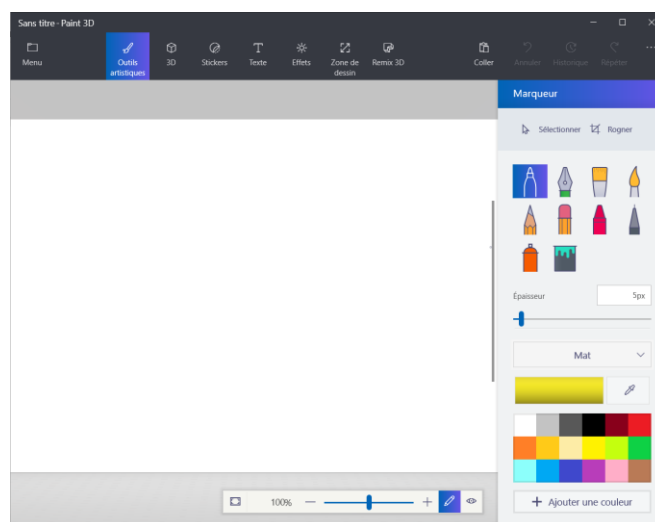
## LES PROGRAMMES EN FENÊTRES

Ce sont les programmes que vous utilisez souvent.

Ci-dessous, deux exemples de programme en fenêtres.



Le programme Calculatrice



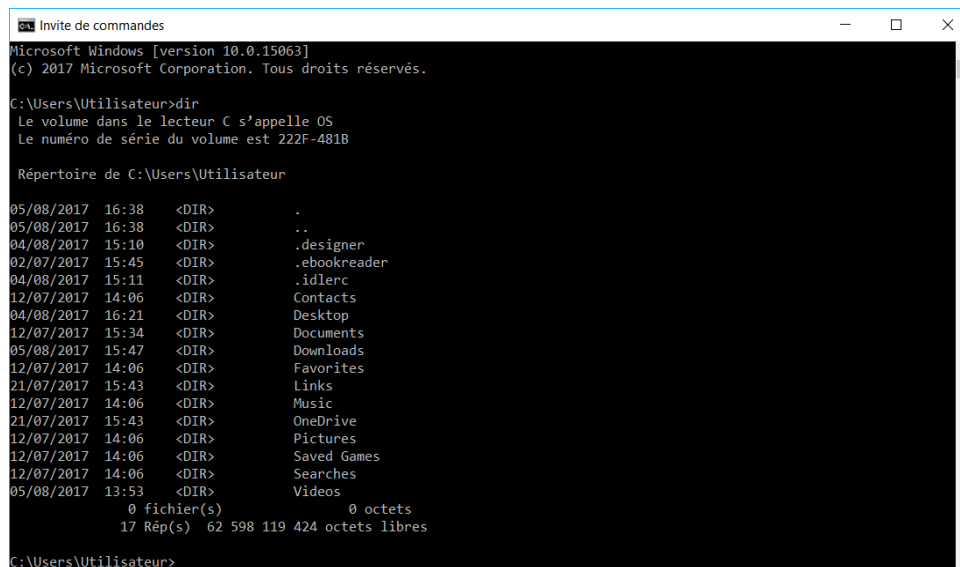
Le programme Paint 3D

Il est possible de créer des programmes avec des fenêtres en C, mais... c'est compliqué ! Pour débiter, il est préférable de commencer par créer des programmes en console.

### LES PROGRAMMES EN CONSOLE

Les programmes en console ont été les premiers à apparaître. À cette époque, l'ordinateur ne gérait que le noir et le blanc et il n'était pas assez puissant pour créer des fenêtres comme on le fait aujourd'hui.

Bien entendu, le temps a passé depuis. Windows a rendu l'ordinateur « grand public » principalement grâce à sa simplicité et au fait qu'il n'utilisait que des fenêtres. Windows est devenu tellement populaire qu'aujourd'hui beaucoup de monde a oublié ce qu'était la console.



```

Invite de commandes
Microsoft Windows [version 10.0.15063]
(c) 2017 Microsoft Corporation. Tous droits réservés.

C:\Users\Utilisateur>dir
Le volume dans le lecteur C s'appelle OS
Le numéro de série du volume est 222F-481B

Répertoire de C:\Users\Utilisateur

05/08/2017  16:38    <DIR>          .
05/08/2017  16:38    <DIR>          ..
04/08/2017  15:10    <DIR>          .designer
02/07/2017  15:45    <DIR>          .ebookreader
04/08/2017  15:11    <DIR>          .idlerc
12/07/2017  14:06    <DIR>          Contacts
04/08/2017  16:21    <DIR>          Desktop
12/07/2017  15:34    <DIR>          Documents
05/08/2017  15:47    <DIR>          Downloads
12/07/2017  14:06    <DIR>          Favorites
21/07/2017  15:43    <DIR>          Links
12/07/2017  14:06    <DIR>          Music
21/07/2017  15:43    <DIR>          OneDrive
12/07/2017  14:06    <DIR>          Pictures
12/07/2017  14:06    <DIR>          Saved Games
12/07/2017  14:06    <DIR>          Searches
05/08/2017  13:53    <DIR>          Videos

               0 fichier(s)                0 octets
              17 Rép(s) 62 598 119 424 octets libres

C:\Users\Utilisateur>

```

La console sous Windows

### LA COMPILATION

Le C est un langage **compilé** (par opposition aux langages interprétés). Cela signifie qu'un programme C est décrit par un fichier texte, appelé **fichier source**. Ce fichier n'étant évidemment pas exécutable par le microprocesseur, il faut le traduire en langage machine. Cette opération est effectuée par un programme appelé **compilateur**. La compilation se décompose en fait en 4 phases successives :

- 1. Le traitement par le préprocesseur** : le fichier source est analysé par le **préprocesseur** qui effectue des transformations purement textuelles (remplacement de chaînes de caractères, inclusion d'autres fichiers source . . . ).
- 2. La compilation** : la compilation proprement dite traduit le fichier généré par le préprocesseur en **assembleur**, c'est-à-dire en une suite d'instructions du microprocesseur qui utilisent des mnémoniques rendant la lecture possible.
- 3. L'assemblage** : cette opération transforme le code assembleur en un **fichier binaire**, c'est-à-dire en instructions directement compréhensibles par le processeur. Généralement, la compilation et l'assemblage se font dans la foulée, sauf si l'on spécifie explicitement que l'on veut le code assembleur. Le fichier produit par l'assemblage est appelé **fichier objet**.
- 4. L'édition de liens** : un programme est souvent séparé en plusieurs fichiers source, pour des raisons de clarté mais aussi parce qu'il fait généralement appel à des bibliothèques de fonctions standard déjà écrites. Une fois chaque code source assemblé, il faut donc lier entre eux les différents fichiers objets. L'édition de liens produit alors un fichier dit **exécutable**.



# Travaux Pratiques

## La programmation en C (les bases)

TP1 sur la programmation  
en C-les bases.doc

1<sup>ère</sup> année

Page:4/8

Les différents types de fichiers utilisés lors de la compilation sont distingués par leur suffixe. Les fichiers source sont suffixés par **.c**, les fichiers prétraités par le préprocesseur par **.i**, les fichiers assembleur par **.s**, et les fichiers objet par **.o**. Les fichiers objets correspondant aux bibliothèques pré-compilées ont pour suffixe **.a**.

### LES COMPOSANTS ÉLÉMENTAIRES DU C

Un programme en langage C est constitué des six groupes de composants élémentaires suivants :

- les identificateurs,
- les mots-clefs,
- les constantes,
- les chaînes de caractères,
- les opérateurs,
- les signes de ponctuation.

On peut ajouter à ces six groupes les commentaires, qui sont enlevés par le préprocesseur.

Le rôle d'un **identificateur** est de donner un nom à une entité du programme. Plus précisément, un identificateur peut désigner :

- un nom de variable ou de fonction,
- un type défini par typedef, struct, union ou enum,
- une étiquette.

Un identificateur est une suite de caractères parmi :

- les lettres (minuscules ou majuscules, mais non accentuées),
- les chiffres,
- le "blanc souligné" ou underscore (**\_**).

Le premier caractère d'un identificateur ne peut pas être un chiffre. Par exemple, var1, tab 23 ou deb sont des identificateurs valides ; par contre, 1i et i;j ne le sont pas. Il est cependant déconseillé d'utiliser **\_** comme premier caractère d'un identificateur car il est souvent employé pour définir les variables globales de l'environnement C.

Les majuscules et minuscules sont différenciées.

Le compilateur peut tronquer les identificateurs au-delà d'une certaine longueur. Cette limite dépend des implémentations, mais elle est toujours supérieure à 31 caractères. (Le standard dit que les identificateurs externes, c'est-à-dire ceux qui sont exportés à l'édition de lien, peuvent être tronqués à 6 caractères, mais tous les compilateurs modernes distinguent au moins 31 caractères).

Un certain nombre de mots, appelés **mots-clefs**, sont réservés pour le langage lui-même et ne peuvent pas être utilisés comme identificateurs. L'ANSI C compte 32 mots clefs :

auto	const	double	float	int	short	struct	unsigned
break	continue	else	for	long	signed	switch	void
case	default	enum	goto	register	sizeof	typedef	volatile
char	do	extern	if	return	static	union	while

### LE CODE MINIMAL

Le code minimal « ne fera rien » de particulier mais il est indispensable.

Selon votre **EDI** (Environnement de Développement Intégré) ou IDE (en anglais) , la méthode pour créer un nouveau projet n'est pas la même.

Exemple de code minimal :

```
#include <stdio.h>
int main()
{
    printf("BTS SNIR\n");
    return 0;
}
```

#### Question 1

Testez le code ci-dessus avec votre EDI (Visual Studio ou CodeBlocks par exemple).  
Modifiez le texte affiché puis testez à nouveau le code.

On a introduit dans ce programme minimal la notion d'**Interface Homme-Machine** (IHM).

L'utilisateur visualise une information sur l'écran, puis l'utilisateur par une action sur le clavier, fournit une information au programme.

### ANALYSE DU CODE

La première ligne :

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

C'est une ligne spéciale que l'on peut voir en haut des fichiers source. Cette ligne est facilement reconnaissable car elles commencent par un « # ». Cette ligne spéciale se nomme **directive de préprocesseur**. Il peut y avoir plusieurs directives de préprocesseur. Ce sont des lignes qui seront lues par un programme appelé préprocesseur, un programme qui se lance au début de la compilation.

« include » signifie « inclure » en français. Cette ligne demande d'inclure un fichier au projet, c'est-à-dire d'ajouter un fichier pour la compilation.

Ce fichier s'appelle « stdio.h ». On l'appelle une **bibliothèque**. Cette bibliothèque contient du code qui va vous permettre (entre autres) d'afficher du texte à l'écran.

Les lignes suivantes :

```
int main()
{
    printf("BTS SNIR\n");
    return 0;
}
```

Un programme en langage C est constitué de **fonctions**.



# Travaux Pratiques

## La programmation en C (les bases)

TP1 sur la programmation  
en C-les bases.doc

1<sup>ère</sup> année

Page: 6/8

Une fonction permet de rassembler plusieurs commandes à l'ordinateur. Regroupées dans une fonction, les commandes permettent de faire quelque chose de précis. Par exemple, vous pouvez créer une fonction « ouvrir\_fichier » qui contiendra une suite d'instructions pour l'ordinateur lui expliquant comment ouvrir un fichier.

L'avantage, c'est qu'une fois la fonction écrite, vous n'aurez plus qu'à écrire « ouvrir\_fichier », et votre ordinateur saura comment faire sans que vous ayez à tout répéter !

La fonction de votre programme s'appelle « **main** ». C'est un nom de fonction particulier qui signifie « **principal** », **c'est toujours par la fonction main que le programme commence.**

Une fonction a un début et une fin, délimités par des accolades { et }. Toute la fonction main se trouve donc entre ces accolades.

Les instructions situées entre les accolades forment un « **bloc** ». Un bloc peut lui-même contenir d'autres blocs.

Votre fonction contient deux lignes. On appelle ces lignes des **instructions**.

Chaque instruction est une commande à l'ordinateur. Chacune de ces lignes demande à l'ordinateur de faire quelque chose de précis.

La ligne : `printf("BTS SNIR!\n");` demande à afficher le message « BTS SNIR! » à l'écran. Quand votre programme arrivera à cette ligne, il va donc afficher un message à l'écran, puis passer à l'instruction suivante.

Cette ligne appelle en fait une fonction prédéfinie (fournie avec le langage, et donc que vous n'avez pas à écrire vous-même) nommée **printf**. Ici, cette fonction reçoit un **argument** qui est : "BTS SNIR!\n".

Les guillemets servent à délimiter une « chaîne de caractères » (suite de caractères).

La notation `\n` est conventionnelle : elle représente un caractère de fin de ligne, c'est-à-dire un caractère qui, lorsqu'il est envoyé à l'écran, provoque le passage à la ligne suivante. Vous verrez que, de manière générale, le langage C prévoit une notation de ce type (`\` suivi d'un caractère) pour un certain nombre de caractères dits « de contrôle », c'est-à-dire ne possédant pas de graphisme particulier.

### Remarques :

Toute instruction se termine obligatoirement par un **point-virgule** « ; ». C'est d'ailleurs comme ça qu'on reconnaît ce qui est une instruction et ce qui n'en est pas une. Si vous oubliez de mettre un point-virgule à la fin d'une instruction, votre programme ne compilera pas !

Les **caractères spéciaux** sont des lettres spéciales qui permettent d'indiquer qu'on veut aller à la ligne, faire une tabulation, etc.

Ils sont faciles à reconnaître : c'est un ensemble de deux caractères. Le premier d'entre eux est toujours un anti-slash ( `\` ), et le second un nombre ou une lettre. Voici deux caractères spéciaux courants que vous aurez probablement besoin d'utiliser, ainsi que leur signification :

`\n` : retour à la ligne

`\t` : tabulation

## COMMENTAIRE, DÉCLARATION DE VARIABLES, AFFECTATION

### Question 2

Testez le code ci-page suivante.

```
#include <stdio.h> /* bibliotheque d'entrees-sorties standard */

int main()
{
    int heure = 24, mois = 12;

    printf("Il y a %d heures dans une journee et %d mois dans une annee.\n", heure, mois);

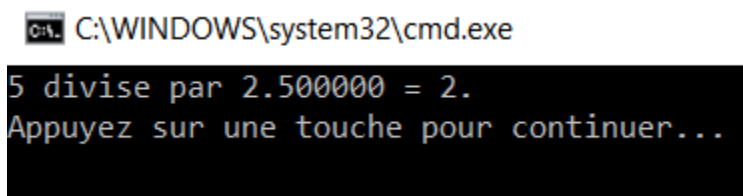
    return 0;
}
```

Dans ce nouveau programme, vous avez introduit trois nouvelles notions :

- la notion de commentaire « **/\* bibliotheque d'entrees-sorties standard \*/** »
- la notion de déclaration de variables : les variables sont les données que manipulera le programme lors de son exécution (ici, heure et mois). Ces variables sont rangées dans la mémoire vive de l'ordinateur. Elles doivent être déclarées au début du programme.
- la notion d'affectation, symbolisée par le signe « = ».

### Question 3

Modifiez le programme précédent afin que le résultat corresponde à l'image ci-après.



**Exemple attendu d'affichage du résultat en console**

### Remarque :

Le tableau ci-dessous vous sera certainement utile pour « typer » vos variables.

Type	Lettre
int	%d
long	%ld
float/double	%f / %lf
char	%c
string (char*)	%s
pointeur (void*)	%p
short	%hd
entier hexadécimal	%x

## RÉCUPÉRER UNE SAISIE

### Question 4

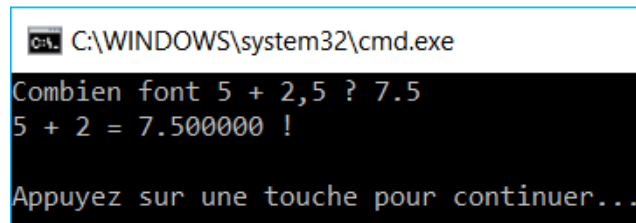
Testez le programme ci-dessous.

```
#include <stdio.h> /* bibliotheque d'entrees-sorties standard */

int main()
{
    int nombre = 0; // On initialise la variable nombre à 0
    printf("Combien font 5 + 2 ? ");
    scanf("%d", &nombre); // On demande d'entrer le nombre avec scanf
    printf("5 + 2 = %d !\n\n", nombre);
    return 0;
}
```

### Question 5

Modifiez le programme précédent afin que le résultat corresponde à l'image ci-après.

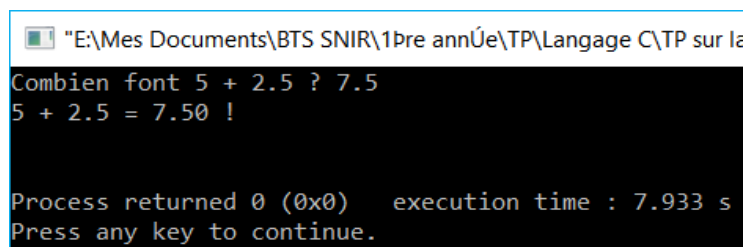


```
C:\WINDOWS\system32\cmd.exe
Combien font 5 + 2,5 ? 7.5
5 + 2 = 7.500000 !
Appuyez sur une touche pour continuer...
```

Exemple attendu d'affichage du résultat en console

### Question 6

Le résultat obtenu par le code précédent est trop « précis ». Modifiez à nouveau le programme précédent afin que le résultat corresponde à l'image ci-dessous.



```
E:\Mes Documents\BTS SNIR\1pre annUe\TP\Langage C\TP sur la
Combien font 5 + 2.5 ? 7.5
5 + 2.5 = 7.50 !
Process returned 0 (0x0) execution time : 7.933 s
Press any key to continue.
```

### Remarque :

%n.f permet une précision avec n chiffre(s) après la virgule.