PARTIE 2 : LES VARIABLES

Petit rappel du schéma d'une fonction :

```
void                    nom_fonction(void)
{

        Code_ligne1;
        code_ligne2 ;
        etc..
        return ;

}
```

Bon du coup, concrètement ça donne ça dans notre exemple :

```
void        afficher_a(void)
{

        write(1, "a", 1);
        write(1, "b", 1);
        return ;

}
```

Dans :
int main(void)

Nous allons appeler la fonction afficher_a();

```
#include <unsitd.h>

void        afficher_a(void)
{

        write(1, "a", 1);
        write(1, "b", 1);
        return ;

}

int main(void)
{
        afficher_a();
        return (0);
}
```

On commence un peu à voir l'utilité d'une fonction non ? Si on veut écrire plusieurs fois "ab" dans notre programme il suffit de rappeler la fonction afficher_a() Au lieu de sans cesse devoir écrire write(1, "a", 1); puis write(1, "b", 1);

Je te laisse tester le code suivant :

```
#include <unsitd.h>

void        afficher_a(void)
{
            write(1, "a", 1);
            write(1, "b", 1);
            return ;

}

int main(void)
{
            afficher_a();
            afficher_a();
            afficher_a();
            return (0);

}
```

Jusque-là tout va bien ? aucune question sur les fonctions ? n'hésite pas je suis là pour ça !

Maintenant il faut qu'on parle des prototypes. Le prototype d'une fonction c'est la première ligne de la fonction avec un (point-virgule) à la fin.

Par exemple le prototype de la fonction afficher_a(void)

C'est ? …. Devine … :

void   afficher_a(void);

A quoi ça sert ? tu te rappelles que je t'ai dit : on doit toujours mettre les fonction au-dessus de la fonction **int main(void)** ?

Et bah j'ai mentiiiiiiis :D (enfin a moitié).

Il faut savoir que si tu mets ta fonction **afficher_a()** en dessous de la fonction **main(void)** ton ordinateur ne va pas aimer, car il ne connaitra pas la fonction **afficher_a()**

Il ne sait pas où elle est ! Si tu fais ça:

```
#include <unsitd.h>

int main(void)
{
	afficher_a();
	afficher_a();
	afficher_a();
	return (0);
}

void	afficher_a(void)
{
	write(1, "a", 1);
	write(1, "b", 1);
	return ;
}
```

Tu auras une erreur bien moche qui va apparaitre :p.
Pour résoudre ce problème il faut mettre le prototype en haut de la fonction **int main(void)**

NOTE : A partir de maintenant pour designer la fonction **int main(void)** je dirais **main**, par exemple devant le **main** ou devant la fonction **main**

```
#include <unsitd.h>

void			afficher_a(void);

int main(void)
{
	afficher_a();
	afficher_a();
	afficher_a();
	return (0);
}

void	afficher_a(void)
{
	write(1, "a", 1);
	write(1, "b", 1);
	return ;
}
```

Bon il est temps de commencer des exercices !!! 3:)

Pour tous les exercices que tu auras à faire, je te donnerai le prototype et tu devras faire la fonction en suivant le prototype.

Exemple, je te dirai : tu fois faire une fonction qui affiche "ab" avec le prototype suivant :

void afficher_a(void) ;

Tu devras me rendre un fichier contenant:

```
void                    afficher_a(void)
{
        write(1, "ab", 2);
        return ;
}
```

Tu vois ce que je veux dire ?
C'est parti Ouvre le PDF Exercices

Ok bon jusque-là c'était rigolo mais pas pratique. Si je veux additionner des chiffres comment je fais ?

Parce que faire :

write(1, "1 + 1 = 2", 9);

Ce n'est pas réellement une addition, c'est du texte et ça demande à l'homme de faire le calcul.
Ok 1+1 c'est facile mais si je veux faire 195005 * 456 + 7 - 3445 ? on rigole moins hein ?

Pour ça il y a les variables ! Oh ! que c'est bien les variables. Une variable comme son nom l'indique peut varier. Elle sert à stocker des données dans la mémoire de l'ordinateur. Il y a plusieurs types de variables

Les nombres :

int
float
double
Unsigned int

les caracteres :

char

Int ? mais ça dans le main !!

Exactement on y reviendra. Bon on va se concentrer sur les int,
int est une abréviation de **Integer** qui signifie **Entier** en Anglais.

Un nombre entier bah c'est un nombre entier ! 1 2 3 .. 789 ... 1324659780 etc ...

Pour les nombres à virgule il y a les **float**.

Créer une variable se dit déclarer.
On va déclarer une variable. On commence par déclarer la variable pour que le programme sache qu'elle existe comme ça on peut réutiliser la variable dans le code ! (Oui j'ai répété déclarer 3 fois et 4fois variable dans 1 phrase).

C'est important d'apprendre les termes technique comme "déclarer une variable" "fonction" etc.. Car si tu cherches des infos sur internet tu ne liras que ça :p

Donc tout ceci c'est joli mais c'est du blabla !
Comment fait-on ?

```
int main(void)

{

	int nombre;    //ON DECLARE une variable qu'on appelle nombre (tu peux l'appeler comme tu veux)



	nombre = 5;    // nombre vaut désormais 5 (si tu as appelé ta variable z par exemple il faut écrire z = 5)

	nombre = 4;    // nombre valait 5 maintenant il devient 4 et rien d'autre

	nombre = 13;		// nombre vaut mtn 13 et rien d'autre

	return (0);

}
```

/ ! \ Qu'est-ce que les \\ dans le code ? ce sont des commentaires. Dans le langage C on peut utiliser des commentaires. Ils ne sont pas pris en compte par le programme, tu peux essayer si tu veux / ! \

On peut modifier la valeur de la variable nombre, n'importe quand.

Si tu essayes le code juste au-dessus, tu verras que rien ne se passe. C'est tout à fait normal on a donné la valeur 5 puis 4 puis 13 à notre variable mais à aucun moment on a demandé à notre programme de l'afficher à l'écran.

Huuuuuuuuum nan c'est pas la peine d'essayer la fonction write() ce n'est que pour les lettres et les phrases.

Comment on fait ? pour le moment on peut utiliser

printf();

(Uniquement pour tester notre code)

Le '/n' du code en dessous signifie retour à la ligne, c'est comme tapez entrer sur word
On pronounce « back slash n »

```
#include <stdio.h>

int main(void)
{
	int nombre;

	nombre = 5;
	printf("nombre = %d \n", nombre);   // hein ?! :D pour afficher une variable qui contient un nombre avec printf il faut mettre %d qui sera remplacer par nombre
	nombre = 4;
      printf("nombre = %d (2eme essaie) \n", nombre);   //entre guillemet on met la phrase qu'on veut voir afficher, et la ou doit ya voir le nombre on met %d
	nombre = 13;
      printf("nombre = %d (3eme essaie) \n", nombre);                  //ensuite on met une virgule et le nom de notre variable
	return (0);
}
```

On peut maintenant résoudre notre problème de tout à l'heure

```
#include <stdio.h>

int main(void)
{
	int nombre1;
	int nombre2;
	int nombre3;


	nombre1 = 1;
	nombre2 = 1;
	nombre3 = nombre1 + nombre2;
	printf("%d + %d = %d, nombre1, nombre2, nombre3);    // une autre façon d'utiliser printf, on peut mettre plusieurs variable dedans.
	return (0);

}
```

A noter que le ; est TRES important on le met à la fin de chaque ligne quasiment et l'oublier = impossible de compiler le programme.
Une ligne qui ouvre des accolades : { } n'a pas besoin de point-virgule à la fin c'est la seule exception.
C'est pour ça que int main(void) n'a pas de point-virgule a la fin de la ligne

Ce n'est pas mal les variables hein ?

Il est important de comprendre qu'une variable sert à stocker une information c'est pour cela que nous avons créé nombre3. Car nombre3 va prendre le résultat de l'addition entre les deux variables (nombre1 + nombre2). Ce n'est pas obligatoire mais si tu ne stock pas le résultat dans une variable, ton ordinateur va faire le calcul mais n'affichera rien à l'écran.

On peut aussi faire : **nombre2 = nombre1 + nombre2;**

Ça fonctionne aussi bien car nombre2 stockera le résultat après l'addition. On n'a donc pas besoin de créer une variable nombre3 mais c'est plus propre avec une troisième variable.

Par contre nombre2 après l'addition ne vaudra plus 1 mais le nouveau résultat.

Je te laisse essayer avec la multiplication et la soustraction !