Langage C

Notes de cours.
https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c

Code minimum :

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


int main(int argc, char *argv[])
{

    printf("Salut é !\n"); // Sous windows pas d'accents!

    return 0;

}
```

Commentaires :

```
/*
Commentaires sur plusieurs lignes.
*/

instruction // Commentaire sur une ligne.
```

Commande minimum pour compilation :

```
gcc nom_du_programme.c
```

Commande d'exécution :

```
./a.out
```

Types des variables :

Nom du type	**Min**	**Max**
unsigned char	0	255
unsigned int	0	65 535
unsigned long	0	4 294 967 295
signed char	-127	127
int	-32 767	32 767
long	-2 147 483 647	2 147 483 647
float	-1 x1037	1 x1037
double	-1 x1037 // erreur???	1 x1037 // erreur???

Afficher une variable :

Format	**Type attendu**
%d	int
%ld	long
%f	float, double

```c
int main(int argc, char *argv[])
{
    const int PV_INITIAUX = 300; // déclaration et affectation
    int pv_max, pv_actuels = PV_INITIAUX;
    int niveau = 0;

    printf("Vous êtes niveau %d et vous avez %d PV.\n", niveau, pv_actuels);

    return 0;

}
```

Récupérer une saisie :

Format	**Type attendu**
%d	int
%ld	long
%f	float
%lf	double

```c
int age = 0;
scanf("%d", &age);
```

Les types d'opérations :

Opération	**Signe**
Addition	+
Soustraction	-
Multiplication	*
Division	/
Modulo	%

L'incrémentation :

nombre = nombre + 1;	nombre++;

La décrémentation :

nombre = nombre - 1;	nombre--;

Les autres raccourcis :

nombre = nombre + 2;	nombre += 2;
nombre = nombre - 2;	nombre -= 2;
nombre = nombre * 2;	nombre *= 2;
nombre = nombre / 2;	nombre /= 2;
nombre = nombre% 2;	nombre%= 2;

```c
int main(int argc, char *argv[])
{
    int premier_nbr = 0, deuxieme_nbr = 0, resultat = 0;
```

```
    printf("%d, %d, %d\n", premier_nbr, deuxieme_nbr, resultat);


    printf("Entrez un premier nombre : \n");
    scanf("%d", &premier_nbr);

    printf("Entrez un deuxième nombre : \n");
    scanf("%d", &deuxieme_nbr);

    resultat = premier_nbr + deuxieme_nbr;

    printf("%d + %d = %d\n", premier_nbr, deuxieme_nbr, resultat);
    return 0;

}
```

Les bibliothèques:

Nom de la bibliothèque	**Exemple de contenu**
stdio.h	scanf, printf
math.h	fabs, ceil, floor, pow, sqrt, sin, cos, tan, asin, acos, atan, exp, log, log10
stdlib.h	abs, system("clear")
time.h	srand, rand

les conditions :

Symbole	**Signification**
==	est égal à
>	est supérieur à
<	est inférieur à
>=	est supérieur ou égal à
<=	est inférieur ou égal à
!=	est différent de
if	si
else	sinon
else if	sinon, si
&&	et
||	ou
!	non if (!(age < 18))

```
if (age >= 18)
{
  printf ("Vous etes majeur !");
}
```

```
int main(int argc, char *argv[])
{
    int age = 0;

    printf("Quel est votre age ?\n");
```

```
    scanf("%d", &age);

    if (age > 18)
    {
        printf("Vous êtes majeur.\n");
    }
    else if (age == 18)
    {
        printf("Vous êtes tout juste majeur.\n");
    }
    else
    {
        printf("Vous êtes mineur.\n");
    }

    return 0;

}
```

Les booléens :

```
int main(int argc, char *argv[])
{
    int age = 0, majeur = 0;

    printf("Quel est votre age ?\n");
    scanf("%d", &age);

    majeur = age >= 18;

    if (majeur)
    {
        printf("Vous êtes majeur.\n");
    }

    else
    {
        printf("Vous êtes mineur.\n");
    }

    return 0;

}
```

La condition switch et boucle while (ex : menu) :

```
int main(int argc, char *argv[])
{
    int choix_menu = 3, nombre = 0;

    while (choix_menu)
    {
        printf("### Menu ###\n0. Quitter\n1. Lire\n2. Ecrire\n\nEntrez votre
choix : ");
        scanf("%d", &choix_menu);

        switch (choix_menu)
        {
            case 0:
                printf("\nBye.\n");
                break;
```

```
            case 2:
                printf("\nEntrez votre nombre : \n");
                scanf("%d", &nombre);
                break;
            default:
                printf("\nVotre nombre : %d.\n", nombre);
                break;
        }
    }

    return 0;
}
```

Les conditions condensées : ternaires :

Forme if, else	**Ternaire**

`if (majeur)`
`  age = 18;`
 	
`else`
 	
`  age = 17;`
	`age = (majeur) ? 18 : 17;`

Boucle do… while :

```
int main(int argc, char *argv[])
{
    int compteur = 45;

    do // La boucle fera toujours au moins 1 tour.
    {
        printf("Salut les Zeros !\n");
        compteur++;
    } while (compteur < 10);

    return 0;
}
```

Boucle for :

```
int main(int argc, char *argv[])
{
    int compteur;

    for (compteur = 0 ; compteur < 10 ; compteur ++)
    {
        printf("Compteur = %d.\n", compteur);
    }
    return 0;
}
```

Les fonctions :

```
int triple(int nbr)
{
    return nbr * 3; // c'est le retour qui définit le type de la fonction.
}


int main(int argc, char *argv[])
```

```
{
    int number;

    printf("Entrez un nombre : \n");
    scanf("%d", &number);

    printf("Le triple de %d est : %d.\n", number, triple(number));
    return 0;
}
```

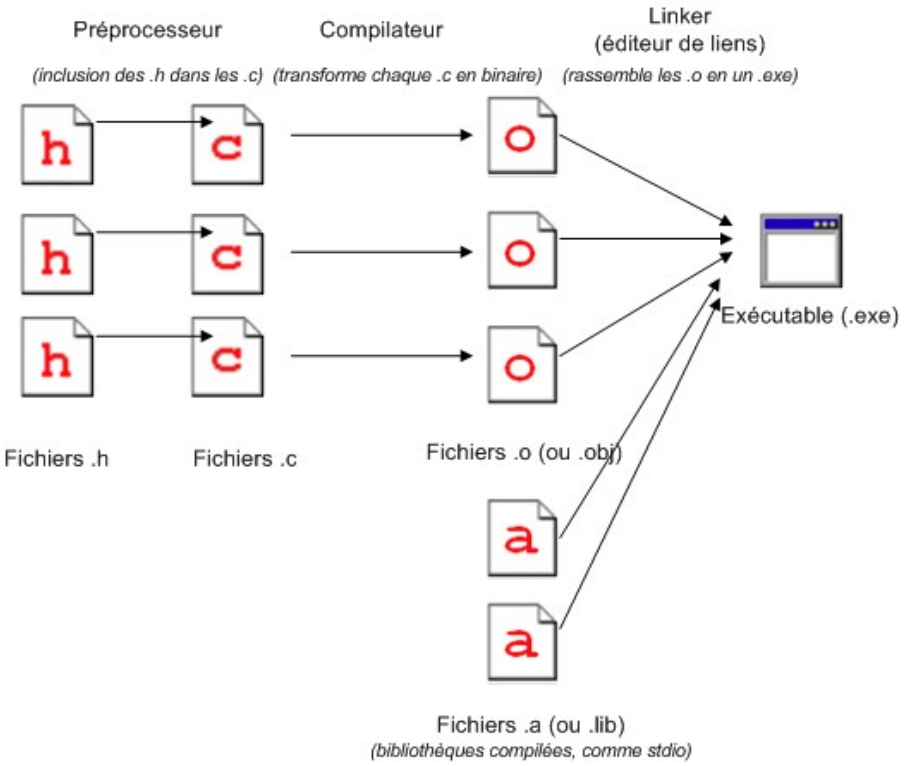
Les prototypes de fonctions :

```
int triple(int); // Prototype de de la fonction.


int main(int argc, char *argv[])
{
    int number;

    printf("Entrez un nombre : \n");
    scanf("%d", &number);

    printf("Le triple de %d est : %d.\n", number, triple(number));
    return 0;
}


int triple(int nbr)
{
    return nbr * 3;
}
```

Les modules :

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


#include "fonctions.h" // Les prototypes dans le fichier header.
#include "fonctions.c" // Les fonctions dans le fichier source.


int main(int argc, char *argv[])
{
    int number;

    printf("Entrez un nombre : \n");
    scanf("%d", &number);

    printf("Le triple de %d est : %d.\n", number, triple(number));
    return 0;
}
```

La compilation :



Variable globale :

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


int result = 0; // Variable globale.
// Sencée être accessible de tous les fichiers ! Comment ???


void triple(int);


int main(int argc, char *argv[])
{
    int number;

    printf("Entrez un nombre : \n");
    scanf("%d", &number);

    triple(number);

    printf("Le triple de %d est : %d.\n", number, result);
    return 0;
}


void triple(int nbr)
{
    result = nbr * 3;
}
```

Variable globale accessible uniquement dans un fichier :

```
static int resultat = 0;
```

Variable statique à une fonction :

```
int triple(int nombre)
{
    static int resultat = 0; // La variable resultat est créée la première fois
que la fonction est appelée

    resultat = 3 * nombre;
    return resultat;
} // La variable resultat n'est PAS supprimée lorsque la fonction est terminée.
```

Fonction locale à un fichier :

```
static int triple(int nombre)
{
    // Instructions
}
```

```
static int triple(int nombre); // Proto
```

Les pointeurs :

```c
int main(int argc, char *argv[])
{
    int number = 15;

    printf("L'adresse de la variable number est : %p.\n", &number);

    return 0;
}
```

Adresse	Valeur
&number	number
0x7fffc16c0754	15

```c
int *monPointeur = NULL;
// Le type du pointeur dépend du type de la variable vers la quelle il pointe.
```

```c
int main(int argc, char *argv[])
{
    int age = 15;
    int *pointeur_sur_age = &age;

    printf("L'adresse de la variable age est : %p.\n", pointeur_sur_age);
    printf("La valeur de la variable age est : %d.\n", *pointeur_sur_age);

    return 0;
}
```

```c
void triple(int *);


int main(int argc, char *argv[])
{
    int age = 2;

    triple(&age);

    printf("Le triple de \"age\" est : %d.\n", age);

    return 0;
}


void triple(int *pointeur)
{
    *pointeur *= 3;
}
```

```c
void triple(int *);


int main(int argc, char *argv[])
{
    int age = 2;
    int *pointeur_sur_age = &age;
```

```
    triple(pointeur_sur_age);

    printf("Le triple de \"age\" est : %d.\n", age);

    return 0;
}


void triple(int *pointeur)
{
    *pointeur *= 3;
}
```

Les tableaux :

```
int tableau[4];

tableau[0] = 10;
tableau[1] = 23;
tableau[2] = 505;
tableau[3] = 8;
```

```
printf("%d", tableau); // retourne l'adresse de la prmière case.
```

tableau[1]	*(tableau + 1)

En C99 la taille du tableau peut être dynamique mais pas en C89 :

```
int taille = 5;
int tableau[taille];
```

```
int main(int argc, char *argv[])
{
    const int NBR_CELLULES = 4 ;
    int notes_fr[NBR_CELLULES];

    notes_fr[0] = 12;
    notes_fr[1] = 14;
    notes_fr[2] = 16;
    notes_fr[3] = 18;

    for (int counter = 0; counter < NBR_CELLULES; counter ++)
    {
        printf("Note de français n° %d : %d\n", counter + 1,
notes_fr[counter]);
    }

    return 0;
}
```

```
int main(int argc, char *argv[])
{
    const int NBR_CELLULES = 4 ;

    int notes_fr[4] = {12, 14, 16, 18}; // La constante a provoqué une erreur.

    for (int counter = 0; counter < NBR_CELLULES; counter ++)
    {
```

```
        printf("Note de français n° %d : %d\n", counter + 1,
notes_fr[counter]);
    }

    return 0;
}
```

```
void modify_array(int *, int);


int main(int argc, char *argv[])
{
    const int NBR_CELLULES = 4 ;

    int notes_fr[NBR_CELLULES];

    modify_array(notes_fr, NBR_CELLULES);

    for (int counter = 0; counter < NBR_CELLULES; counter ++)
    {
        printf("Note de français n° %d : %d\n", counter + 1,
notes_fr[counter]);
    }

    return 0;
}


void modify_array(int array[], int taille)
{
    for (int i = 0; i < taille; i ++)
    {
        array[i] = 10 + (i * 2);
    }
}
```