

PROGRAMMATION C

tawfiq@caditazi.fr



Objectifs

Définitions

Les outils

La mémoire

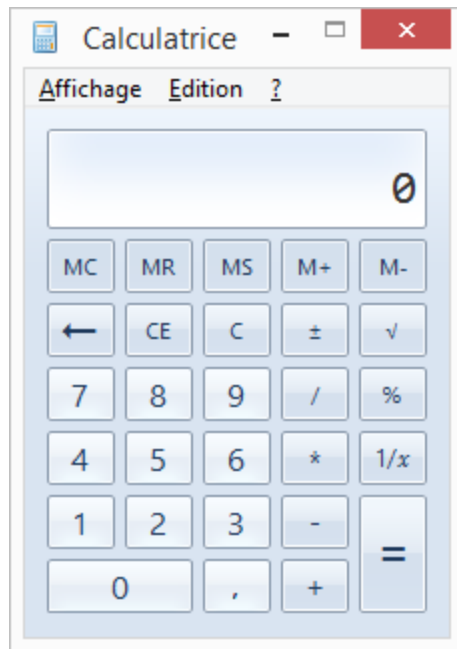
Les conventions
de
programmation

Le langage C

DÉFINITIONS

Programme

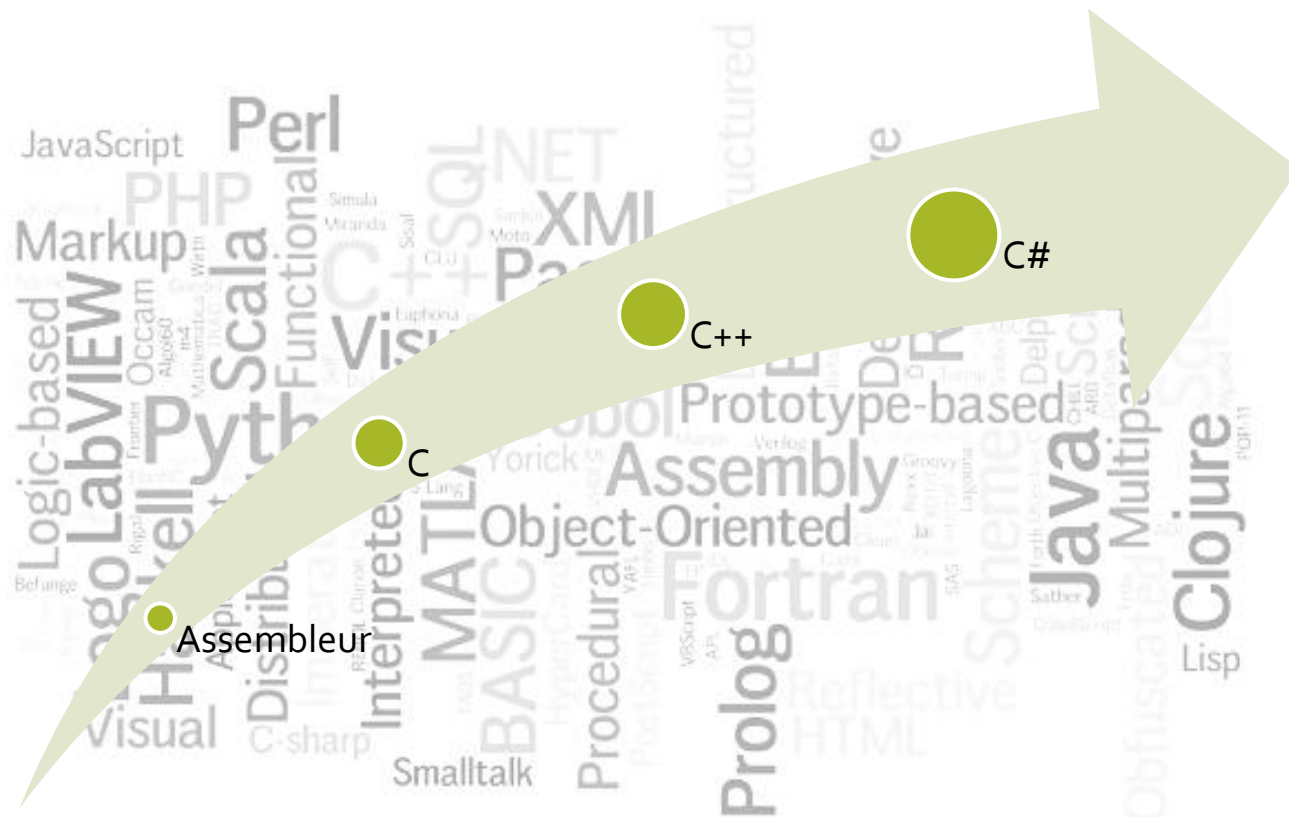
- Ensemble d'instructions exécutées par le processeur de l'ordinateur.





Programmation

- Ecriture d'un ensemble d'instructions dans un langage spécifique.





Programmation

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main()
5 {
6     int age = 0;
7
8     printf("Bonjour, entrez votre age\r\n");
9     scanf("%d", &age);
10    printf("Vous avez %d an(s) \r\n", age);
11
12    system("pause");
13 }
```

```
001010111010010101101011
10101001111101010011010
101101011101010110100101
011010111101010011111010
10011101010011101011010
```

Langage de programmation

- Compréhensible par un développeur

Langage machine

- Instructions binaires interprétées par la machine



Compilation

- **Compilation** : Transformation d'instructions écrites dans un langage de programmation en langage machine.
- **Compilateur** : Programme de compilation. Écrit les instructions en langage machine dans un fichier exécutable (« .exe » sous Windows).





Code source

- Ensemble de fichiers contenant des instructions écrites dans un langage de programmation.

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

int main()
{
    int age = 0;

    printf("Bonjour, entrez votre age\r\n");
    scanf("%d", &age);
    printf("Vous avez %d an(s) \r\n", age);

    system("pause");
}
```


LES OUTILS



IDE

- Un IDE (Integrated Development Environment) ou encore EDI (Environnement de Développement Intégré) est un logiciel utilisé pour le développement de logiciels. Il intègre plusieurs outils nécessaires aux développeurs par exemple :
 - Éditeur de texte (écriture du code source)
 - Compilateur (création de l'exécutable)
 - Débogueur (test du programme)

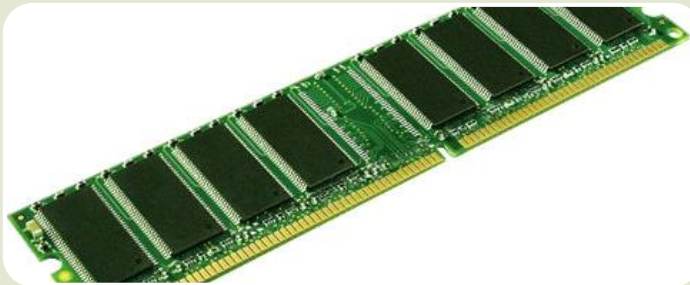


Visual Studio

LA MÉMOIRE



RAM vs ROM



Mémoire vive

- Random Access Memory (RAM)
- Volatile
- Accès très rapide

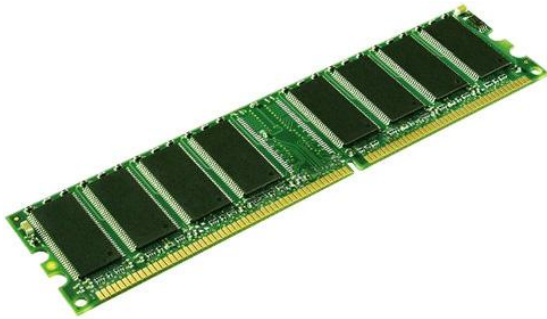


Mémoire morte

- Read-Only Memory (ROM)
- Non volatile
- Accès plus lent



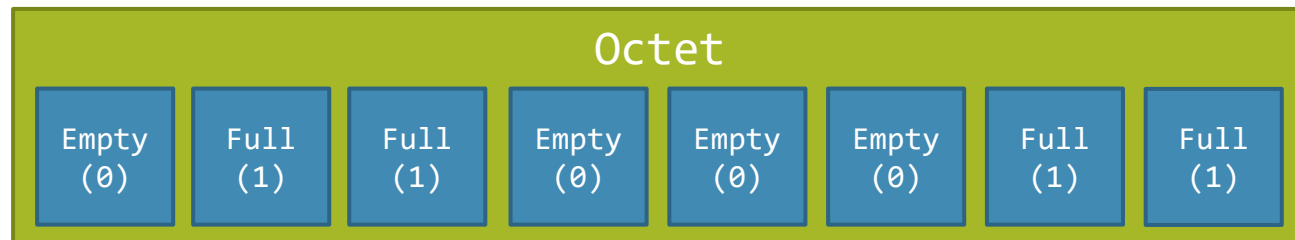
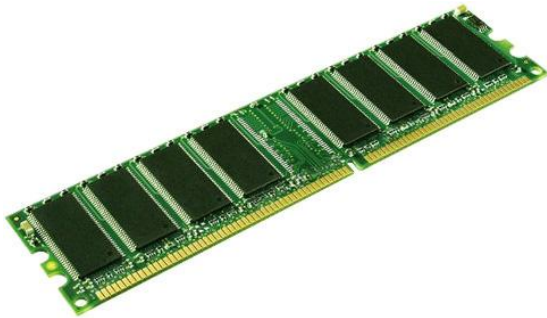
La mémoire vive



Adresse	Valeur
0X0000	18
0X0001	3,14
0X0002	42
0X0003	7
0X0004	24
0X0005	118 218



La mémoire vive



128 64 32 16 8 4 2 1

0 64 32 0 0 0 2 1



Les variables

```
int maVariable;  
maVariable = 5;
```

- Initialiser une variable

Adresse	Valeur
0X0000	5
0X0001	3,14
0X0002	42
0X0003	7
0X0004	24
0X0005	118 218



Les variables

Type	Limite	Octet	Espace
char	-128 128	1	Entier relatif
int	$-2^{15} 2^{15}$ $-2^{31} 2^{31}$	2 octet (Linux) 4 octet (Windows)	Entier relatif
long	$-2^{31} 2^{31}$	4 octet	Entier relatif
long long	$-2^{63} 2^{63}$	8 octet	Entier relatif
float	3.4E +/- 38 (7 chiffres)	4 octet	Décimal
double	1.7E +/- 308 (15 chiffres)	8 octet	Décimal
unsigned char	0 255	1 octet	Entier naturel
unsigned int	0 2^{32}	4 octet	Entier naturel
unsigned long	0 2^{32}	4 octet	Entier naturel
unsigned long long	0 2^{64}	8 octet	Entier naturel

LES CONVENTIONS DE PROGRAMMATION



Indentation

- L'indentation consiste en l'ajout de tabulations ou d'espaces dans un fichier, pour une meilleure lecture et compréhension du code.

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

    if (variable > 3)
    {
        printf("Hello world!\n");
    }

    return 0;
}
```

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

    if (variable > 3)
    {
        printf("Hello world!\n");
    }

    return 0;
}
```



Accolades

- Il existe en règle général deux façons d'écrire les accolades. On pourra opter pour l'une ou pour l'autre en fonction de ses préférences personnelles, mais aussi du langage de programmation utilisé.

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

    if (variable > 3)
    {
        printf("Hello world!\n");
    }

    return 0;
}
```

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

    if (variable > 3) {
        printf("Hello world!\n");
    }

    return 0;
}
```

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

    if (variable > 3)
    {
        printf("Hello world!\n");
    }

    return 0;
}
```



Interlignes et espaces

- N'hésitez pas à aérer votre code.

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

    if (variable > 3)
    {
        printf("Hello world!\n");
    }

    return 0;
}
```

```
int main(int argc, char *argv[])
{
    int variable;
    variable=2;
    if (variable>3)
    {
        printf("Hello world!\n");
    }
    return 0;
}
```



Commentaires

```
/*
  Fonction principale de mon programme
  @param int argc Le nombre d'arguments
  @param char* argv Les arguments passés en paramètre
  @return void
*/
int main(int argc, char *argv[])
{
    int variable;
    variable = 2;

    // Si ma variable est supérieur à 3
    if (variable > 3)
    {
        printf("Hello world!\n"); // Affiche Hello world! dans la console
    }

    return 0;
}
```



Règles de nommage

- Dans un programme informatique, le développeur pourra nommer comme il le souhaite de nombreux éléments (variables, fonctions, propriétés, méthodes, librairies...).

Lettres
minuscules,
majuscules et
chiffres.

Commence par
une lettre

Pas d'espace

Pas d'accents

Respecte la casse

Compréhensible
facilement (ex :
dateNaissance)

CODER EN ANGLAIS ?

LE LANGAGE C

Introduction



La fonction main

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

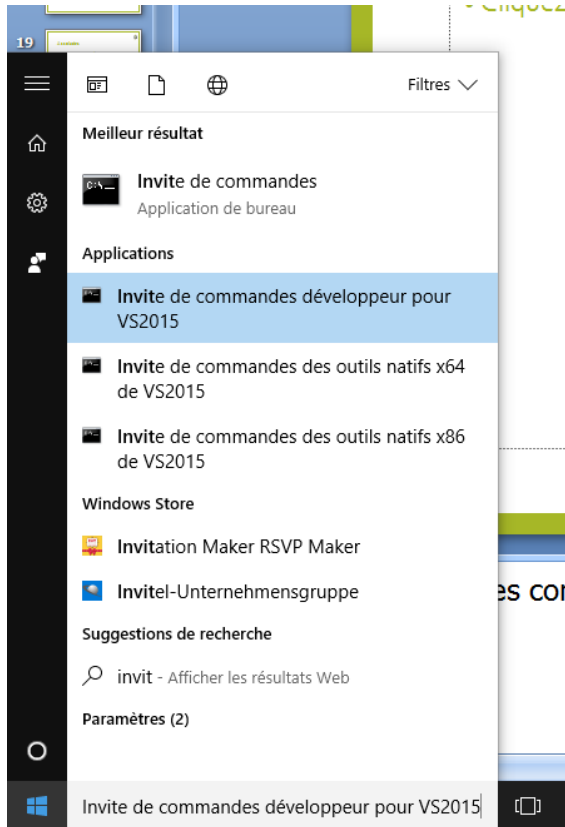
```
#include <stdlib.h>
```

```
int main(int argc, char *argv[])  
{  
    printf("Hello world!\n");  
    system("pause");  
    return 0;  
}
```

- Copiez ce code dans notepad++



La compilation



- Enregistrer votre fichier sous le nom « main.c »
- Lancer l'invite de commande développeur pour VS2015
- Se positionner sur votre bureau :
- Cd c:\users\votre_compte\Desktop
- Ecrire : cl.exe main.c



Exercice 1

Écrivez un programme affichant votre nom dans la console.



Interface utilisateur

- Écrire dans la console :

```
printf("Hello world!\n");
```

- Écrire un nombre entier :

```
int monAge = 99;
```

```
printf("J'ai %d ans !", monAge);
```

- Écrire un nombre décimale :

```
float pi = 3.14;
```

```
printf("Pi = %f", pi);
```

Type	Lettre
int	%d
long	%ld
long long	%lld
float/double	%f / %lf
char	%c
string (char*)	%s
pointeur (void*)	%p



Interface utilisateur

Caractère spéciaux	Description
\»	Guillemet
\\	Antislash
\a	Signal sonore
\b	Retour en arrière
\n	Retour à la ligne
\t	Tabulation

```
int moi = 99;
```

```
int toi = 82;
```

```
printf("J'ai %d ans !Et toi ? \nJ'ai %d ans.", moi, toi);
```



Exercice 2

Créer 3 variables : jour, mois, année.

Afficher dans la console le message ":

« Je suis né le xx/xx/xxxx »



Interface utilisateur

- Lire un nombre entier

```
int monAge = 0;  
scanf("%d", &monAge);  
printf("\nJ'ai %d ans !", monAge);
```

- Lire un nombre décimale :

```
float pi = 3.14;  
scanf("%f", &pi);  
printf("Pi = %f", pi);
```

NE PAS OUBLIER LE SIGNE « & »



Exercice 3

1 -

Écrivez un programme demandant à l'utilisateur son âge, puis afficher ensuite la valeur saisie.

2-

Demandez à l'utilisateur son jour de naissance, puis dans un second temps son mois de naissance et enfin son année de naissance.

Afficher ensuite son jour de naissance comme suis :

« Vous êtes le le 10/3/1999 »

3 -

Écrivez un programme qui demande à l'utilisateur de saisir le nombre d'or avec 11 chiffres après la virgule, puis afficher la saisie de l'utilisateur



Calculs

- Addition

```
int addition = 1 + 2;  
addition += 2;
```

- Soustraction

```
int soustraction = 1 - 2;  
soustraction -= 2;
```

- Multiplication

```
int multiplication = 2 * 2;  
multiplication *= 2;
```

- Division

```
int division = 8 / 2;  
division /= 2;
```

- Modulo

```
int modulo = 10 % 3;  
modulo %= 1;
```




Calculs

- Fonctions courantes de math.h

Fonctions	Description
sin(x), cos(x), tan(x)	Trigonométrie
asin(x), acos(x), atan(x)	
exp(x)	Exponentielle
log(x)	Logarithme népérien
log10(x)	Logarithme à base 10
pow(x,y)	Puissance
sqrt(x)	Racine carré
ceil(x)	Arrondit entier supérieur
floor(x)	Arrondit entier inférieur
fabs(x)	Valeur absolue



Exercice 4

Écrire un programme demandant à l'utilisateur de saisir 2 chiffres puis afficher le résultat de la multiplication de ces 2 chiffres.



Conditions

Opérateur	Description
==	égale
&&	et
	ou
<	inférieur
<=	inférieur égale
>	supérieur
>=	supérieur égale
!=	différent



Table de vérité

ET	V	F
V	V	F
F	F	F

OU	V	F
V	V	V
F	V	F



Exercice conditions

```
int a = 10;  
int b = 20;  
int c = 45;
```

```
if (a < b || c == 45 && b > c || c % 4 == 2) {  
    printf("OK\n");  
}  
else {  
    printf("KO\n");  
}
```



Exercice conditions

```
int a = 10;  
int b = 20;  
int c = 45;
```

```
if ((a < b || c == 45) && b > c || c % 4 == 2) {  
    printf("OK\n");  
}  
else {  
    printf("KO\n");  
}
```



Conditions

```
int moi = 5, toi = 6;

if (moi > toi)
{
    printf("Je suis le plus vieux.");
}
else if (moi < toi)
{
    printf("Tu es le plus vieux.");
}
else
{
    printf("Nous avons le meme age !");
}
```



Conditions ternaires

- Affectation

```
int a = 5;  
int b;
```

```
b = a > 2 ? 6 : 9;
```

```
if (a > 2) {  
    b = 6;  
}  
else {  
    b = 9;  
}
```

- Expression

```
printf("%d\n", a < 2 ? 6 : 9);
```

```
int toto = 17;  
printf((toto >= 18) ? "Toto est majeure." : "Toto est  
mineure.\n");
```




Conditions

- Condition unique

```
int couleur = 2;
switch (couleur)
{
    case 0:
        printf("Rouge");
        break;
    case 1:
        printf("Vert");
        break;
    case 2:
        printf("Bleu");
        break;
    default:
        printf("Blanc");
        break;
}
```



Exercice conditions : Exercice 4.5

En utilisant un switch **ET** un if faire sorte :

Demandez à l'utilisateur s'il veut prendre une pause

1 pour oui

0 pour non

S'il saisie 1 afficher « oui mais en fait non pas tout de suite »

S'il saisie 0 afficher « non, c'est bien continue a travailler »

S'il saisie autre chose, lui demander s'il est bon avec un grand C ?



Boucles

- Tant que

```
int compteur = 10;

while (compteur > 0)
{
    printf("%d !!!", compteur);
    compteur--;
}

printf("Bonne année !!!");
```



Boucles

- Faire tant que

```
int choix = 0;
do
{
    printf("Quelle est la réponse à l'univers ?");
    scanf("%d", &choix);
} while (choix != 42);
```



Boucles

- Pour

```
for (int i = 10; i > 0; i--)  
{  
    printf("%d !!!\n", i);  
}  
printf("Bonne année !!!\n");
```

```
for (int i = 0; i < 4; i++)  
{  
    printf("%d\n", i);  
}  
printf("Nous irons au bois...");
```



Exercice 4.6

Reprendre l'exercice 4.5

Et redemandez à l'utilisateur s'il veut une pause jusqu'à ce qu'il dise « Non ».

Lorsqu'il aura dit non dites lui : « Et ben tu vois t'as envi de bosser »



Exercice 5

Réaliser un menu à l'aide d'une boucle

Pensez à utiliser
`system(« cls »);`

Choisissez une option :

- 1 : Option 1
- 2 : Option 2
- 0 : Quitter

Si je choisie l'option 1 je printf « Option 1 »

Si je choisie l'option 2 je printf « Option 2 »

En boucle jusqu'à :

Si je choisie l'option 0 je quitte le programme



Exercice Calcules

- Demander à l'utilisateur deux variables (x et y)
- Afficher le menu suivant
 - 1 : addition ($x + y$)
 - 2 : soustraction
 - 3 : Division
 - 4 : multiplication
 - 5 : Modulo
 - 6 : puissance (x^y) : ne pas utiliser la bibliothèque maths, faire la boucle nécessaire à la résolution de la puissance
- Afficher le résultat du calcul (par exemple $x + y = \text{resultat}$ si le choix est 1)
- Demande si l'utilisateur souhaite recommencer ? 1 : on recommence / 0 : on termine le programme
 - Si on recommence
 - Demande si on doit réutiliser le résultat
 - 0 – Non – x et y saisies initialement sont conservés
 - 1 - $x = \text{résultat}$
 - 2 - $y = \text{résultat}$



Exo 7 : Jeu mystère 2 joueurs

- Jeux à deux joueurs
 - Joueur 1 saisie une valeur comprise entre 0 et 100
 - Effacer la valeur (system(« cls »))
 - Joueur 2 doit deviner la valeur
 - La machine indique « plus » si la valeur est en dessous de la valeur proposée par joueur 1 ou « moins » si la valeur est au-dessus.
 - Lorsque le joueur 2 saisit la bonne valeur, l'ordinateur lui indique qu'il a gagné en précisant le nombre de coups joués



Exo 8 : Jeu mystère : résolution automatique

- Vous proposez une valeur entre 0 et 100.
 - L'ordinateur doit deviner la valeur que vous avez saisie.
 - Affichez dans la console pour chaque essai si l'ordinateur est $>$, $<$ ou égale au nombre que vous avez proposé.
- Lorsque l'ordinateur trouve la bonne solution, afficher le nombre d'essais.



Exo 9 : Exercice les étoiles : 1

**Demander un chiffre à l'utilisateur
entre 1 et 10**

- *s'il propose 1 écrire dans la console :*
*
- *s'il propose 4 écrire dans la console :*

- *s'il propose 2 écrire dans la console :*

*

**

*

- *s'il propose 3 écrire dans la console :*

*

**

**

*

*

**

**

*

Etc.



Exo 10 : Exercice les étoiles : 2

Demander un chiffre impair à l'utilisateur entre 1 et 9

- S'il répond 1 écrire dans la console : *
- S'il répond 9 écrire dans la console :

- S'il répond 3 écrire dans la console :
*

- S'il répond 5 écrire dans la console :

*



Exo 11 : Jeux des allumettes

- Vous démarrez avec 10 allumettes
 - |||||
 - Chaque joueur à tour de rôle peu enlever 1, 2 ou 3 étoiles
 - Le joueur ayant retiré la dernière allumette a perdu
-
- Commencez le jeu en demandant le nombre d'allumettes
 - Affichez le nombre d'allumettes entre chaque tour
 - Indiquez à la fin de la partie si Joueur 1 ou Joueur 2 a gagné



Exo 12 - Jeux des allumettes

- Faite un joueur 2 joué par l'ordinateur qui gagne à tout les coups s'il commence.



Exo 13 - 421

- Vous avez 4 essais pour réaliser avec 3 dés un jet donnant 421.
- Le programme lance les dés et affiche le résultat.
- Si c'est un 421, on affiche "Gagné", sinon on demande à l'utilisateur d'appuyer n'importe quelle touche.
- Au bout du 4ème essai, au lieu de proposer de relancer, on affichera "Perdu" et le programme se terminera.

`#include <time.h>` (ajouter cette ligne en haut)

`srand(time(NULL));` (ajouter cette ligne une seule fois juste après les déclaration de variables)

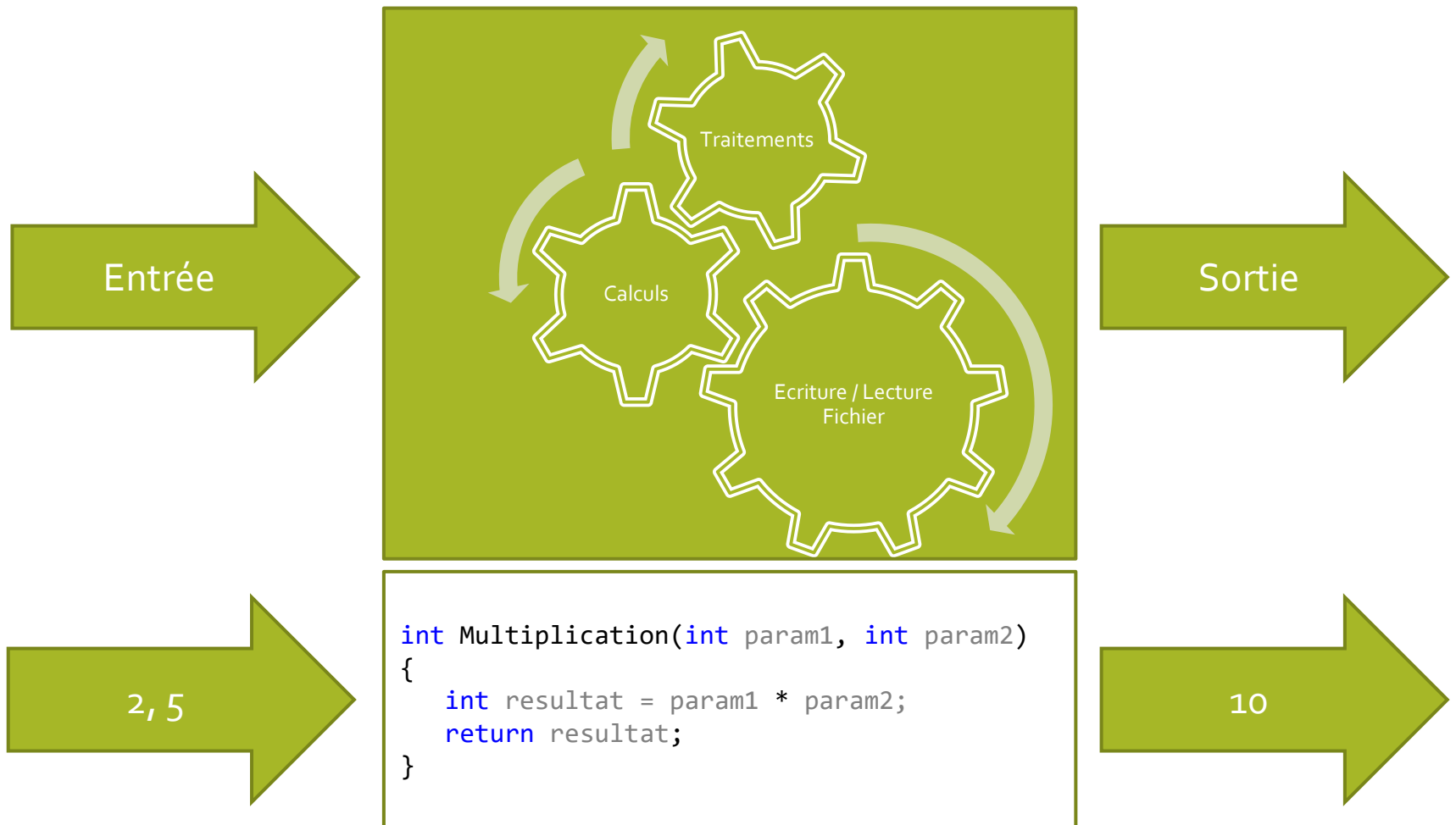
`xxx= rand()%(max-min) + min;` (à chaque fois que vous avez besoin
d'un nombre aléatoire appelez cette formule)

LE LANGAGE C

Les fonctions



Les fonctions





Les fonctions

Signature

Type de retour

Nom de la fonction

Paramètres

```
int Multiplication(int param1, int param2)
```

```
{
```

```
    // ...
```

```
    return 0;
```

```
}
```

Corps /
Instructions

LE TYPE DE RETOUR « VOID » NE RETOURNE RIEN



Les fonctions

- Appeler une fonction

```
int resultat = Multiplication(2, 5);
```

- Les prototypes (annoncer une fonction)

```
int Multiplication(int param1, int param2);
```

UNE FONCTION DOIT ÊTRE DÉCLARÉE AVANT D'ÊTRE APPELÉE



Exercice 11

- Créer un programme permettant d'effectuer une multiplication.
 - Le programme demande à l'utilisateur de saisir une première valeur, puis une seconde valeur.
 - Ensuite le programme affiche le résultat de la multiplication de ces deux valeurs.

FAITES APPEL À UNE FONCTION POUR EFFECTUER LA MULTIPLICATION



Exo 14 - TP Sous-programme

- Créer un menu qui selon le choix de l'utilisateur appellera les sous-programmes suivants
- Créer un sous-programme qui retourne pi avec ses 8 premières décimales
 - paramètres : 0, sortie : double
- Créer un sous-programme qui transforme des Celsius en Fahrenheit -- $\text{Celsius} = (\text{Fahrenheit} - 32) \times \frac{5.0}{9}$
 - Paramètres : 1 double, sortie : 1 double
- Créer un sous-programme qui transforme des Fahrenheit en Celsius
 - paramétré : 1 double, sortie : 1 double
- Écrire un sous-programme qui donne la puissance du premier paramètre par le deuxième paramètre
 - paramétré : 2 entiers, sortie : 1 entier
- Créer un sous-programme qui renvoie le périmètre d'un cercle en fonction de son diamètre (utiliser le sous-programme pi)
 - paramétré : 1 double, sortie : 1 double
- Créer un sous-programme qui renvoie l'aire d'un triangle en fonction de sa base et de sa hauteur
- Créer un sous-programme qui renvoie le volume d'un cylindre en fonction de son rayon et de sa hauteur



Exo 15 – Consommation à la pompe

- Demander à l'utilisateur
 - son compteur de départ (km) (dernier passage à la pompe)
 - son compteur à l'arrivé (km) (actuellement à la pompe)
 - le prix du litre d'essence
 - Le prix payé à la pompe
 - Ecrire un sous programme qui permet de calculer la distance parcourue.
 - Ecrire un sous programme qui permet de calculer sa consommation du l/100 km
-
- 77312 km ; 77726 km ; 1.450 €/l ; 55.26 € -> 9.21 € / 100 km



Exo 16 - Affichage de toutes les combinaisons de couleurs

```
ST TEST TEST TEST 5 6
ST TEST TEST TEST 5 7
ST TEST TEST TEST 5 8
ST TEST TEST TEST 5 9
ST TEST TEST TEST 5 10
ST TEST TEST TEST 5 11
ST TEST TEST TEST 5 12
ST TEST TEST TEST 5 13
ST TEST TEST TEST 5 14
ST TEST TEST TEST 5 15
ST TEST TEST TEST 6 0
ST TEST TEST TEST 6 1
ST TEST TEST TEST 6 2
ST TEST TEST TEST 6 3
ST TEST TEST TEST 6 4
ST TEST TEST TEST 6 5
ST TEST TEST TEST 6 7
ST TEST TEST TEST 6 8
ST TEST TEST TEST 6 9
ST TEST TEST TEST 6 10
```

- Utiliser le sous-programme ci-dessous pour afficher toutes les combinaisons de couleurs de fond avec les couleurs de texte
- Les variables Fond et Texte sont comprises entre 0 et 11
- Il est nécessaire d'inclure Windows.h
`#include <Windows.h>`

```
int Color(int Texte, int Fond)
{
    HANDLE H = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(H, Fond * 16 + Texte);
}
```

LE LANGAGE C

Les tableaux



Les tableaux

- Un tableau a une dimension définie
- Les tableaux en mémoire :
 - Ses cases doivent être contiguës
 - Chaque case du tableau contient un nombre du même type

Dans cet exemple, nous créons un tableau de taille 2 avec des données de type int (4 octets)

Adresse	Valeur
0x0000	12
0x0001	14
0x0002	1
0x0003	
0x0004	
0x0005	
0x0006	2
0x0007	
0x0008	
0x0009	
0x0010	55



Les tableaux

```
// int tableau[3] = { 10, 20, 30 }  
// int tableau[3] = { 0 } ==> { 0, 0, 0 }  
// int tableau[3] = { 10 } ==> { 10, 0, 0 }  
int tableau[3];
```

```
tableau[0] = 10;  
tableau[1] = 20;  
tableau[2] = 30;
```

```
printf("Valeur de la première case : %d", tableau[0]);
```

UN TABLEAU COMMENCE À L'INDICE ZÉRO

LA TAILLE D'UN TABLEAU NE PEUT PAS ÊTRE DÉFINIE PAR UNE VARIABLE



Les tableaux

- On peut également déclarer des tableaux à plusieurs dimensions
- En mémoire, l'ensemble des éléments d'un tableau multidimensionnel sont enregistrés à la suite de façon contiguë.

```
int tableau[2][4] = { { 1, 2, 3, 4 }, { 4, 3, 2, 1 } };  
printf("Premiere valeur : %d", tableau[0][0]);
```



Parcourir un tableau

```
int i = 0;  
while (i < taille) {  
    //traitement  
    i++;  
}
```



Exercice 17

1. Une fonction qui initialise un tableau avec des valeurs aléatoires
2. Une fonction qui initialise un tableau en indiquant une valeur
3. Une fonction qui fait la somme des valeurs d'un tableau
4. Une fonction qui renvoie le nombre d'éléments pair d'un tableau
5. Une fonction qui renvoie la plus petite valeur d'un tableau
6. Une fonction qui renvoie la position de la plus petite valeur d'un tableau
7. Définir une fonction qui inverse l'ordre des éléments du tableau



Exercice 18 – partir d'un tableau de 10 cases

1. Rechercher une valeur dans un tableau. Retourner la position de la première occurrence de la valeur dans le tableau. Si la valeur n'existe pas, retourner -1
2. Rechercher une valeur dans un tableau. Retourner la position de la dernière occurrence de la valeur dans le tableau. Si la valeur n'existe pas, retourner -1
3. Demander à l'utilisateur de saisir un entier. Compter le nombre de fois que cette valeur est présente dans le tableau.
4. Fusionner 2 tableaux dans un troisième ayant suffisamment de cases – les valeurs d'un tableau de 3 cases et d'un tableau de 5 cases se retrouvent dans un tableau de 8 cases
5. Demander à l'utilisateur 2 positions de cases. Echanger les valeurs de ces deux positions
6. Calculer la moyenne d'un tableau d'entier
7. Demander à l'utilisateur une valeur. Faire en sorte que cette valeur soit la première valeur du tableau et décaler la première case du tableau dans la deuxième etc... La dernière valeur du tableau est perdue



Exercice 3 - Tries

1. Tester si un tableau est ordonné
2. Ordonner un tableau
(https://fr.wikipedia.org/wiki/Tri_%C3%A0_bulles)
3. Rechercher une valeur dans un tableau ordonné (dichotomique)
4. Rechercher une valeur dans un tableau non ordonné (tri + dichotomie, sans modifier le tableau initial)

LE LANGAGE C

Le préprocesseur



Le préprocesseur

- Il s'agit d'un programme qui est exécuté automatiquement avant la compilation et qui transforme votre fichier source à partir d'un certain nombre de directives.
- Les directives préprocesseur sont précédées de #
- Il en existe 3 :
 - l'incorporation de fichiers source (directive #include) ;
 - la définition de symboles (directive #define) ;
 - la compilation conditionnelle.



#include

- Elle permet d'incorporer, avant compilation, le texte figurant dans un fichier quelconque.
- cette directive possède deux syntaxes
 - Inclusion d'une bibliothèque standard :

#include <nom_fichier>

- Inclusion de votre propre bibliothèque

#include "nom_fichier"

Un fichier incorporé peut lui-même comporter des #include



#define

- Définition de symboles

#define nbmax 5

- Lorsque le symbole nbmax sera rencontré par le préprocesseur dans votre code source, il sera substitué par 5 avant la compilation



La compilation conditionnelle

- Un certain nombre de directives permettent d'incorporer ou d'exclure des portions du fichier source dans le texte qui est analysé par le préprocesseur



La compilation conditionnelle

- Existence ou non d'un symbole :
 - Incorpore le code si le symbole est défini :

```
#ifdef symbole  
.....  
#else  
.....  
#endif
```

- Incorpore le code si le symbole n'est pas défini :

```
#ifndef symbole  
.....  
#else  
.....  
#endif
```

Exemple :

```
#define MISEAUPPOINT  
.....  
#ifdef MISEAUPPOINT  
instructions 1  
#else  
instructions 2  
#endif
```



La compilation conditionnelle

- Test d'une expression pour incorporer ou non le code

```
#if condition
```

```
.....
```

```
#else
```

```
.....
```

```
#endif
```

Exemple :

```
#define CODE 1
```

```
.....
```

```
#if CODE == 1
```

```
instructions 1
```

```
#endif
```

```
#if CODE == 2
```

```
instructions 2
```

```
#endif
```

LE LANGAGE C

Headers



Vos propres bibliothèques

- Vous pouvez regrouper des fonctions par en faire une bibliothèque de fonction
- Pour cela il faut créer un fichier portant le même nom que votre fichier .c avec l'extension .h
- Ce fichier .h doit contenir les prototypes des fonctions du fichier .c, c'est-à-dire la signature de la fonction se terminant par un « ; »
- Si le fichier .c a besoin de symbole les définitions de symboles (#define) se font dans le .h
- Le fichier .c doit incorporer (#include) le fichier .h qui lui correspond
- Tout fichier .c ayant besoin d'une fonction présente dans un autre fichier .c doit incorporer (#include) le fichier .h nécessaire



Vos propres bibliothèques

- Un fichier .h peut être incorporé plusieurs fois
- Par contre en C, il est interdit de déclarer plusieurs fois la même fonction
- Il y a donc un problème, si mon programme appelle plusieurs fois

```
#include malibrary.c
```

- Les déclarations de cette bibliothèque seront présentes plusieurs fois et cela plantera
- C'est pour cela que pour vos bibliothèques il faut incorporer ce qu'on appelle un **garde-fou**

```
// monfichier.h
#ifndef MONFICHIER
#define MONFICHIER
// ...
#endif // MONFICHIER
```



Vos propres bibliothèques - Exemple

- Je regroupe dans un fichier « mescalculs.c » les fonction suivantes :

```
int addition(int a, int b) {  
    return a + b;  
}
```

```
int soustraction(int a, int b) {  
    return a - b;  
}
```

Je désire en faire une bibliothèque



Vos propres bibliothèques - Exemple

- Je crée un fichier mescalculs.h (le fichier .h doit porter le même nom que le fichier .c)
- contenant tout d'abord le garde-fou

```
//fichier mescalculs.h  
#ifndef MESCALCULS  
#define MESCALCULS
```

```
#endif
```

Le symbole doit porter le nom du fichier en majuscule



Vos propres bibliothèques - Exemple

- Entre le `#define` et le `#endif` je dois indiquer les prototypes des fonctions de mon fichier `mescalculs.c`

```
//fichier mescalculs.h
#ifndef MESCALCULS
#define MESCALCULS

int addition(int a, int b);
int soustraction(int a, int b);

#endif
```

- Si j'avais eu besoin d'autre symbole, commune taille de tableau par exemple je l'aurais indiqué juste au-dessus des prototypes
- Les noms de variables ne sont pas obligatoires pour les prototypes :

```
int soustraction(int, int);
```



Vos propres bibliothèques - Exemple

- Tout en haut de mon fichier « mescalculs.c » j'incorpore mon fichier « mescalculs.h » entre guillemets

```
//fichier mescalculs.c  
#include "mescalculs.h"
```

```
int addition(int a, int b) {  
    return a + b;  
}
```

```
int soustraction(int a, int b) {  
    return a - b;  
}
```



Vos propres bibliothèques - Exemple

- Si dans un autre fichier exemple.c j'ai besoin de faire une addition dans un sous-programme, il faudra que je fasse une incorporation en haut de ce fichier du fichier « mescalculs.h »

```
//fichier exemple.c  
#include "mescalculs.h"
```

```
int unSousProgramme() {  
    int c;
```

```
    c = addition(10, 5);  
    return c;  
}
```

LE LANGAGE C

Le Debug



Debug

- Positionner un point d'arrêt en cliquant sur la barre grise à gauche du code (ou en appuyant sur F9 au niveau de votre curseur)

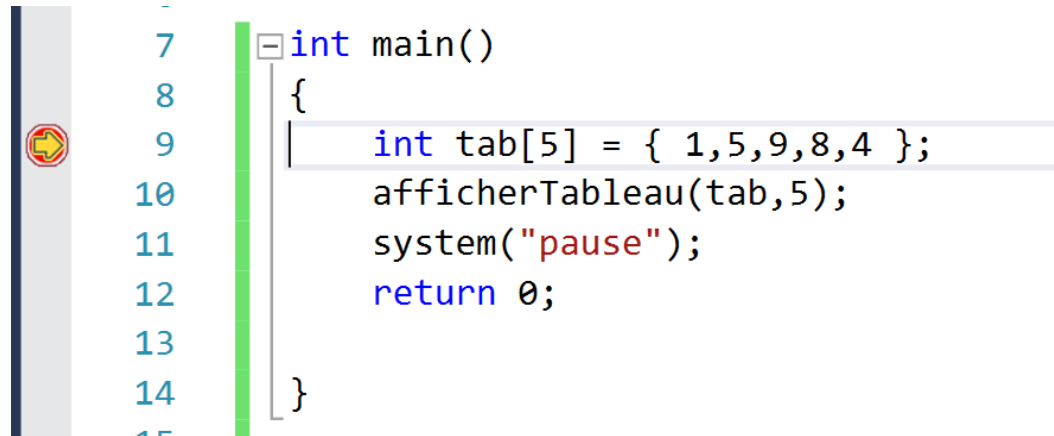
The image shows a code editor with a dark blue vertical bar on the left and a light gray vertical bar on the right. A red circle with a white center is positioned on the gray bar, indicating a breakpoint. The code is as follows:

```
6
7
8
9  int tab[5] = { 1,5,9,8,4 };
10 afficherTableau(tab,5);
11 system("pause");
12 return 0;
13
14 }
```




Debug

- Exécuter votre code avec F5
- L'exécution du programme s'arrêtera lorsqu'il arrivera à un point d'arrêt.



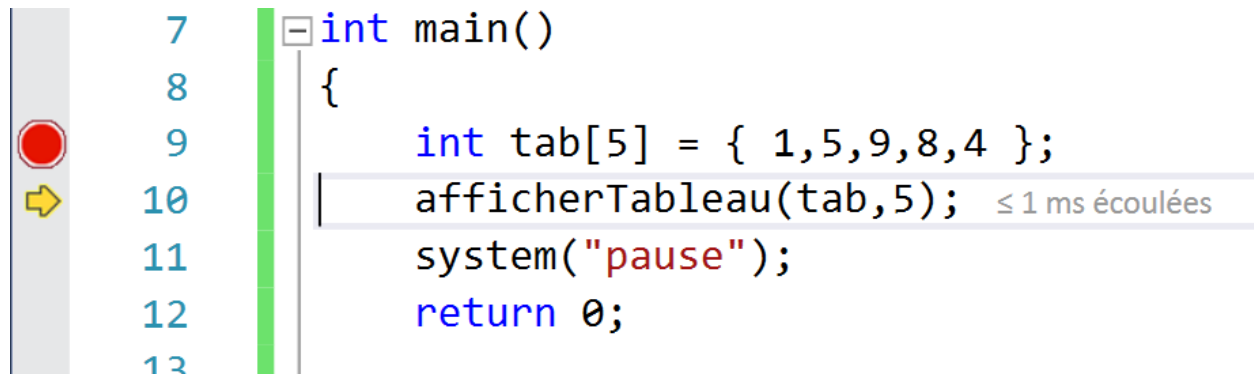
```
7  int main()  
8  {  
9  |  int tab[5] = { 1,5,9,8,4 };  
10 |  afficherTableau(tab,5);  
11 |  system("pause");  
12 |  return 0;  
13 |  
14 |  
15 }
```

The image shows a code editor window with a C program. A breakpoint, represented by a red circle with a yellow arrow, is set on line 9. The code defines an array 'tab' and calls 'afficherTableau' and 'system("pause")'. The editor has a dark theme with a light blue line number margin.



Debug

- Vous avez la possibilité de poursuivre l'exécution en pas à pas grâce à F10 ou F11



```
7  int main()  
8  {  
9      int tab[5] = { 1,5,9,8,4 };  
10     afficherTableau(tab,5); ≤ 1 ms écoulées  
11     system("pause");  
12     return 0;  
13 }
```

- F10 permet de faire du pas à pas sortant c'est-à-dire qu'il ne montrera pas l'exécution du code dans la fonction afficherTableau
- F11 est pour du pas à pas entrant, il montrera le code en pas à pas de la fonction afficherTableau

LE LANGAGE C

Les chaînes de caractères



Les chaînes de caractères

- Les chaînes de caractères sont des tableaux de `char`
- On utilise `%d` pour afficher le code ASCII et `%c` pour afficher le caractère correspondant

```
char lettre = 'A';
```

```
scanf("%c", &lettre);  
printf("ASCII : %d\n", lettre);  
printf("Lettre : %c\n", lettre);
```



Les chaînes de caractères

- `\0` indique la fin de la chaîne de caractères
- Un tableau contenant 'Bonjour' devra donc être déclaré avec une taille de 8 cases

```
// char chaine[] = "Bonjour";  
char chaine[8];
```

```
chaine[0] = 'B';  
chaine[1] = 'o';  
chaine[2] = 'n';  
chaine[3] = 'j';  
chaine[4] = 'o';  
chaine[5] = 'u';  
chaine[6] = 'r';  
chaine[7] = '\0';
```

```
printf("%s", chaine);
```

Adresse	Valeur
0x0000	12
0x0001	14
0x0002	66 → 'B'
0x0003	111 → 'o'
0x0004	110 → 'n'
0x0005	106 → 'j'
0x0006	111 → 'o'
0x0007	117 → 'u'
0x0008	114 → 'r'
0x0009	0 → '\0'
0x0010	55



Les chaînes de caractères

- Il est possible également de demander à l'utilisateur de saisir une chaîne de caractères

```
char nom[100];  
char prenom[100];  
  
printf("Veuillez saisir votre nom : ");  
scanf("%s", nom);  
printf("Veuillez saisir votre prenom : ");  
scanf("%s", prenom);  
printf("Bonjour %s %s", prenom, nom);
```



Exercices

- Rechercher la taille d'une chaîne de caractères (strlen)
- Copier une chaîne de caractères dans une autre (strcpy)
- Comparer deux chaînes de caractères (strcmp)
- Rechercher un caractère dans une chaîne de caractères (position du caractère) (strchr)
- Recherche une chaîne de caractères dans une chaîne de caractères (retourne la position du caractère) (strstr)
- Mettre une chaîne de caractères en minuscule (tolower)
- Mettre une chaîne de caractères en majuscule (toupper)
- strcmp : i = insensitive : int strcmp(char chaine1[], char chaine2[])
- strncmp : int strncmp(char chaine1[], char chaine2[], int maxlen)
- strrchr : int strrchr(char chaine1[], char chr) (reverse)

LE LANGAGE C

Les pointeurs



Les pointeurs

- Dans chaque adresse on peut stocker un nombre (seulement un nombre d'où le code ASCII)

`int secondes = 47811;`

- Lors d'une déclaration de variable, une zone mémoire est réservée pour la variable par le système d'exploitation
- lors de l'exécution le nom de la variable `secondes` sera remplacé par son adresse : `0x0008`

Adresse	Valeur
0x0000	12
0x0001	14
0x0002	55
0x0003	0
0x0004	0
0x0005	0
0x0006	0
0x0007	477
0x0008	47811
0x0009	487
0x0010	777



Les pointeurs

- Affichage de la valeur de secondes :
`printf("La variable secondes vaut : %d", secondes);`
- Ceci affichera :
- La variable secondes vaut : 47811
- Affichage de l'adresse de secondes :
`printf("L'adresse de la variable secondes est : %p", &secondes);`
- Ceci affichera :
- L'adresse de la variable secondes est : 0x0008

Adresse	Valeur
0X0000	12
0X0001	14
0X0002	55
0X0003	0
0X0004	0
0X0005	0
0X0006	0
0X0007	477
0X0008	47811
0X0009	487
0X0010	777



Les pointeurs

`secondes`

désigne la valeur de la variable

`&secondes`

désigne l'adresse de la variable



Les pointeurs

- Les pointeurs sont des variables permettant de stocker des adresses
- Création d'un pointeur :

```
int *unPointeur;
```

- Initialisation d'un pointeur ne contenant pas encore d'adresse :

```
int *unPointeur = NULL;
```

- Affectation d'un pointeur

```
int secondes = 47811;
```

```
int *unPointeurSurSeconde = &secondes;
```

- le pointeur `unPointeurSurSeconde` pointe sur la variable `secondes`



Les pointeurs

- Illustration de l'exemple précédent en mémoire :

`unPointeurSurSeconde` est à l'adresse `0x0003` et pointe sur l'adresse `0x0008`

`Seconde`

Adresse	Valeur
0x0000	12
0x0001	14
0x0002	55
0x0003	0x0008
0x0004	0
0x0005	0
0x0006	0
0x0007	477
0x0008	47811
0x0009	487
0x0010	777



Les pointeurs

- Comment afficher la valeur d'un pointeur ?

```
printf("%d", unPointeurSurSeconde);
```

- Ceci affichera la valeur décimal de 0x0003, un entier.
- Ce n'est pas ce que nous voulons, nous voulons afficher 47811
- Pour cela il faut précéder le nom d'un pointeur par *

```
printf("%d", *unPointeurSurSeconde);
```

- Ceci affichera 47811



Les pointeurs

- Illustration de l'exemple précédent en mémoire :

```
printf("%p", unPointeurSurSeconde);
```

```
printf("%d", *unPointeurSurSeconde);
```

Adresse	Valeur
0x0000	12
0x0001	14
0x0002	55
0x0003	0x0008
0x0004	0
0x0005	0
0x0006	0
0x0007	477
0x0008	47811
0x0009	487
0x0010	777



Les pointeurs

- sur une variable, comme la variable secondes:
 - secondes signifie : « Je veux la valeur de la variable secondes »,
 - **&**secondes signifie : « Je veux l'adresse à laquelle se trouve la variable secondes » ;
- sur un pointeur, comme pointeurSurSecondes :
 - *****pointeurSurSecondes signifie : « Je veux la valeur de la variable qui se trouve à l'adresse contenue dans pointeurSurSecondes ».
 - pointeurSurSecondes signifie : « Je veux la valeur de pointeurSurSecondes » (cette valeur étant une adresse),



Les pointeurs

Utiliser les pointeurs dans une fonction

```
int main()
{
    int nombre = 5;
    ajouterDix(&nombre);
    printf("%d", nombre);
}
```

← Affiche 15

```
void ajouterDix(int *pointeurSurNombre)
{
    *pointeurSurNombre = *pointeurSurNombre + 10;
}
```



Les pointeurs

Utiliser les pointeurs dans une fonction

```
int main()
{
    int nombre = 5;
    int* pointeurSurNombre = &nombre;
    doubler(pointeurSurNombre);
    printf("%d", nombre);
}
```

← Affiche 10

```
void doubler(int *pointeurSurNombre)
{
    *pointeurSurNombre = *pointeurSurNombre * 2;
}
```



Les pointeurs

```
int main()
{
    int secondes = 47811;
    int minutes = 0;

    decoupeMinutes(&minutes, &secondes);

    printf("%d minutes et %d secondes", minutes, secondes);
}

void decoupeMinutes(int *pointeurMinutes, int *pointeurSecondes)
{
    *pointeurMinutes = *pointeurSecondes / 60;
    *pointeurSecondes = *pointeurSecondes % 60;
}
```



Les pointeurs – Exercice 1

```
int    main()
{
    int A = 1;
    int B = 2;
    int C = 3;
    int *P1;
    int *P2;

    P1 = &A;
    P2 = &C;
    *P1 = *P2 + 1;
    P1 = P2;
    P2 = &B;
    C = *P2 * 2;
    ++*P2;
    A = *P2 * *P1;
    P1 = &A;
    *P1 = (*P2)++;
    return 0;
}
```

	A	B	C	P1	P2
Initialisation					
P1 = &A					
P2 = &C					
*P1 = *P2 + 1					
P1 = P2					
P2 = &B					
C = *P2 * 2					
++*P2					
A = *P2 * *P1					
P1 = &A					
*P1 = (*P2)++					



Les pointeurs – Exercice 1

```
int    main()
{
    int A = 1;
    int B = 2;
    int C = 3;
    int *P1;
    int *P2;

    P1 = &A;
    P2 = &C;
    *P1 = *P2 + 1;
    P1 = P2;
    P2 = &B;
    C = *P2 * 2;
    ++*P2;
    A = *P2 * *P1;
    P1 = &A;
    *P1 = (*P2)++;
    return 0;
}
```

	A	B	C	P1	P2
Initialisation	1	2	3	-	-
P1 = &A	1	2	3	&A	-
P2 = &C					&C
*P1 = *P2 + 1	4				
P1 = P2				&C	
P2 = &B					&B
C = *P2 * 2			4		
++*P2		3			
A = *P2 * *P1	12				
P1 = &A				&A	
*P1 = (*P2)++	3	4			



Les Pointeurs - Exercice

- Soit :

```
int A[9] = { 12, 23, 34, 45, 56, 67, 78, 89, 90 };  
int *P;  
P = A;
```

- Indiquer les valeurs de

- a) $*P+2$
- b) $*(P+2)$
- c) $A+3$
- d) $P+(*P-10)$
- e) $*(P+*(P+8)-A[7])$



Les Pointeurs - Exercice

- Soit :

```
int A[9] = { 12, 23, 34, 45, 56, 67, 78, 89, 90 };  
int *P;  
P = A;
```

- Indiquer les valeurs de

- a) $*P+2$ \leftarrow 14
- b) $*(P+2)$ \leftarrow 34
- c) $A+3$ \leftarrow Adresse de la 4^{ème} case
- d) $P+(*P-10)$ \leftarrow Adresse la 3^{ème} case du tableau
- e) $*(P+*(P+8)-A[7])$ \leftarrow 23



Les Pointeurs – Exercice 1

- Écrire un sous-programme qui divise le paramètre entier x par le paramètre entier y. qui affecte le troisième paramètre, pointeur sur entier en lui donnant le résultat de la division entière et qui renvoi le reste de la division.
- Écrire un sous-programme qui découpe des minutes en jours / heures / minutes : On lui fournit trois paramètres de type pointeurs sur entier dont le paramètre minutes avec le nombre de minutes à découper. Les paramètres sont alors affectés par les nouvelles valeurs.
- Idem avec des octets. On fournit 3 paramètres : octet, kilooctet et mégaoctet. Le paramètre octet contient la valeur qui servira de base de calcul
- fournir a un sous-programme 2 paramètres a et b. Mettre dans a la valeur de b et dans b la valeur de a
- Écrire un sous-programme qui affiche la table ASCII : le caractère, sa valeur hexa et sa valeur décimale

```
int division(int,int, int*);
```

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

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

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


Les Pointeurs – Exercice

Écrire un sous-programme qui vérifie si une chaîne de caractère est un palindrome. En utilisant des pointeurs

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

```
int estPalindrome(char * pChar) {
}
```

```
int main()
{
    char chaine[100] = "laval";
    char * pC = chaine;
    int resultat = estPalindrome(pC);

    printf(resultat ?
    "Oui c'est un palindrome" :
    "Non, ce n'est pas un palindrome");
    printf("\n");
}
```

Les Pointeurs – Exercice

Écrire un sous-programme qui compte le nombre de mot d'une chaîne de caractères

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

```
int compterNombreMots(char * pChar) {

}
```

```
int main() {
char chaine[100] = "    il    est
etudiant    ";
int resultat = compterNombreMots(chaine);
printf("resultat : %d\n", resultat);
system("pause");
return 0;
}
```



Les Pointeurs – Exercice 3

- Refaire les exercices sur les chaînes de caractères mais avec des pointeurs



Les Pointeurs – Exercice 4

- Pendu
 - Exercice code à remplir



TP Jeu de l'oie

- 63 cases
- La partie se termine lorsque on arrive sur la case 63
- Lorsqu'on est à la position 0 si on fait 9 avec
 - 6 et 3 on va directement à la case 26
 - 5 et 4 on va directement à la case 53
- Case : multiple de 9
 - avance une deuxième fois du même nombre ajouté
- case 6 avancer à la case 12
- cases 42 (labyrinthe) on retourne à la case 30
- case 58 (tête de mort) revenir à 0
- case 31 correspondant au puits attendra qu'un autre joueur arrive au même numéro et prendra sa place.
- 52 correspondant la prison attendra qu'un autre joueur vienne au même numéro pour repartir.
- si on dépasse la case 63 on recule du nombre de cases en trop.

```
int nbJoueurs = 4;  
int* tabJoueurs = NULL;  
tabJoueurs = (int*)malloc(nbJoueurs * sizeof(int));
```



Devoir surprise !

LE LANGAGE C

Allocation dynamique



Allocation dynamique

- L'allocation dynamique permet de définir la taille d'une variable en mémoire après compilation
- Selon la machine, la taille d'une variable peut être différente

Malloc

- Memory-alloc
- N'initialise pas la mémoire

Calloc

- Clear-alloc
- Initialise la mémoire

Realloc

- Re-alloc
- Redéfini la taille de la variable en conservant les données

Free

- Libérer la mémoire



Allocation dynamique

```
int* pointeurSurAge = NULL;

printf("Sur votre machine, le type int a une taille de %d octets en mémoire.\n", sizeof(int));

pointeurSurAge = malloc(sizeof(int));

if (pointeurSurAge != NULL)
{
    printf("Quel est votre age ?\n");
    scanf("%d", pointeurSurAge);
}
else
{
    printf("Erreur lors de l'allocation dynamique de mémoire !\n");
}

free(pointeurSurAge);
```



Allocation dynamique

- Exercice : Écrire un programme demandant à l'utilisateur de saisir le nombre de matchs joués par son équipe puis le score de chaque match (enregistrer dans 2 tableaux d'entiers différents). Enfin, afficher le meilleur score de l'équipe (par exemple +3 buts d'écart).



Exercice Gestion Matches

- Demander à l'utilisateur le nombre de matches qu'il souhaite
- Demander à l'utilisateur le score domicile puis extérieur de chaque match
- Afficher le nombre de matches gagnés par l'équipe domicile

LE LANGAGE C

Les structures



Les structures



Nouveaux types de variables

Assemblage de variables (ayant des types différents)



Positionnées dans les fichier .h



Les structures

```
typedef struct
{
    char Nom[100];
    char Prenom[100];
    int age;
    double taille;
} Personne;
```

```
Personne personne;
personne.age = 25;
```

```
personne.taille = 1.81;
```

```
strcpy(personne.Nom, "Doe");
strcpy(personne.Prenom, "John");
```



Les structures

- On peut envoyer à une fonction un pointeur sur structure :

```
void AjouterPersonne(Personne* ptrPersonne)
```

- Attention cependant, la syntaxe suivante ne fonctionnera pas pour modifier l'un des éléments de la structure :

```
*ptrPersonne.age = 20;
```

```
*ptrPersonne.taille = 1.78;
```

- Il faudra alors utiliser l'une des 2 syntaxes suivantes :

```
(*ptrPersonne).age = 20;
```

```
ptrPersonne->taille = 1.78;
```



Les structures

```
PERSONNE * pPersonne = calloc(1000, sizeof(PERSONNE));
```

```
strcpy(pPersonne->Nom, "BRESSON");  
strcpy(pPersonne->Prenom, "Benjamin");  
pPersonne->age = 15;  
pPersonne->taille = 1.51;
```

```
strcpy((pPersonne+1)->Nom, "LOUIS");  
strcpy((pPersonne + 1)->Prenom, "Nicolas");  
(pPersonne + 1)->age = 19;  
(pPersonne + 1)->taille = 1.80;
```




Les structures

- Enumération :

```
typedef enum Avancement Avancement;  
enum Avancement  
{  
    DEBUT = 0,  
    CAHIER_DES_CHARGES = 25,  
    DEVELOPPEMENT = 50,  
    TEST = 75,  
    PRODUCTION = 100  
};
```



Exercice Bibliothèque

LE LANGAGE C

Les fichiers



Les fichiers - Ouverture

- La fonction **fopen** attend 2 paramètres (le **nom du fichier** et le **mode d'ouverture**) et renvoi un pointeur sur une structure **FILE**.

	lecture	écriture	Création fichier	Position curseur	Vidage du fichier
r	Oui	Non	Non	Au début	Non
w	Non	Oui	Oui	-	Oui
a	Oui	Oui	Oui	A la fin	Non
r+	Oui	Oui	Non	Au début	Non
w+	Oui	Oui	Oui	-	Oui
a+	Oui	Oui	Oui	À la fin	Non



Les fichiers - Ouverture

```
FILE* fichier = NULL;
fichier = fopen("personnes.csv", "r+");

if (fichier == NULL)
{
    printf("Erreur lors de l'ouverture du fichier personnes.csv");
}
```

- Cette instruction va créer un pointeur sur un structure file et la fonction fopen va ouvrir (ou créer) le fichier personnes.csv
- Le fichier personnes.csv sera positionné dans le même répertoire que le fichier .exe de l'application

On peut ouvrir un fichier à n'importe quel emplacement sur le disque dur



Les fichiers - Fermeture

- Ne jamais oublier de fermer un fichier ouvert (évite les fuites mémoires)

```
fclose(fichier);
```



Les fichiers - Opérations

- Renommer / Déplacer

```
rename("personne.csv",  
"dossier\\personne.xls"); Supprimer  
remove(fichier);
```

Remove ne place **PAS** le fichier dans la corbeille



Les fichiers - Ecriture

- `fputc`

```
fputc('H', fichier);
```

- `fputs`

```
fputs("John,Doe", fichier);
```

- `fprintf`

```
fprintf(fichier, "Ceci est un %s", test);
```




Les fichiers - Lecture

- `fgetc`

```
lettre = fgetc(fichier);  
do  
{  
    lettre = fgetc(fichier);  
    printf("%c", lettre);  
} while (lettre != EOF);
```

- `fgets`

```
printf("%s", ligne);  
while (fgets(ligne, TAILLE_MAX_LIGNE, fichier) != NULL)  
{  
    printf("%s", ligne);  
}
```

- `fscanf`

```
fscanf(fichier, "%s;%s;%s", personne.Nom, personne.Prenom, personne.Adresse);
```



Les fichiers - Déplacement

- `ftell`

```
long position = ftell(fichier);
```

- `fseek`

```
// positionner le curseur en position début de fichier + 2
```

```
fseek(fichier, 2, SEEK_SET);
```

- `rewind`

```
rewind(fichier);
```

Constante	Position
<code>SEEK_SET</code>	Début
<code>SEEK_CUR</code>	Courante
<code>SEEK_END</code>	Fin

Les déplacements de curseur ne sont pas possibles
avec les modes append (a, a+)



Exercice

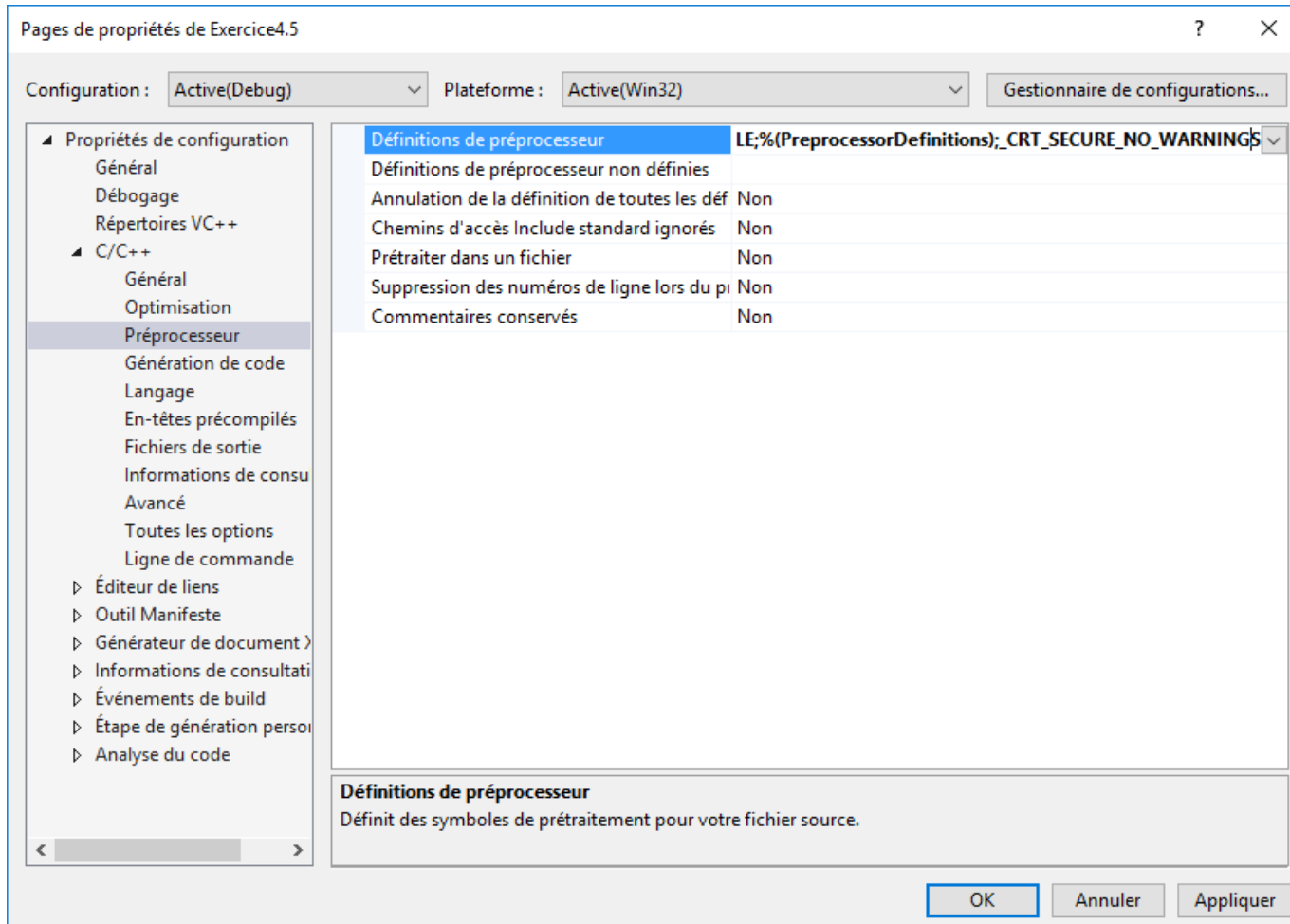
1. Créer un fichier .txt
2. Ajouter dans ce fichier une liste de 3 personnes (à la main dans le fichier)
 1. Les informations sur les personnes (Nom, Prénom, Email) sont séparées par le caractère « ; »
 2. Chaque personne est séparée par un retour à la ligne « \n »
3. Demander à l'utilisateur d'ajouter une nouvelle personne et ajouter cette personne dans le fichier
4. Afficher l'ensemble des données contenues dans le fichier sous la forme : CADI TAZI Prenom (email)

APPENDICE



Erreur sur le scanf :

- Ajouter `:_CRT_SECURE_NO_WARNINGS` aux définitions du préprocesseur dans les propriétés du projet





Visual Studio 2017

- Créer un projet Application Console Win32 C++
- Supprimer tous les fichiers dans le header et source

