Récapitulatif des bases du C

Types de données : *type nomDeLaVariable = saValeur ;*

int nombres entiers.	Format : **%d**	Exemple : *int age = 25 ;*
char: caractère unique.	Format : **%c**	Exemple : *char lettre = 'A' ;* // on met des ' '
float, **double**: nombres à virgule flottante.	Format : **%f** **%lf**	Exemple : *float pi = 3.14f ;* *double db = 4.58 ;*
char[]: chaîne de caractères.	Format : **%s**	Exemple : *char nom[50] = "Alice" ;* *char nom[] = "Alice" ;* // on met des " "

Commentaires

// pour un commentaire sur une ligne.

/* */ pour plusieurs lignes ou un bout de ligne.

Exemple :

```
// Ceci est un commentaire
int age = /* un autre commentaire */ 2 ;
/* un commentaire
sur plusieurs
lignes */
```

Incrementation/Decrementation

```
int i = 0 ;
i++ ;    // equivalent à i = i + 1; ou i += 1;
i-- ;    // equivalent à i = i -1 ; ou i -= 1;
```

Fonction main

Point d'entrée du programme.

```
int main()
{
   // Code à exécuter
   return 0;
}
```

Le nom de cette fonction est : ***main***
Le type de valeur que renvoi cette fonction est un entier : ***int***
Le début et la fin de la fonction se trouve entre les accolades ***{ }***
return est le mot clé qui permet de sortir de la fonction et de renvoyer quelque chose.
Ici comme on attend un entier on renvoit 0.

Opérateurs

Opérations arithmétiques :

+ **-** ***** **/** **%**

Exemple : *int somme = 5 + 3;*
int reste = 5%2 ; //reste sera égal à 1
int reste2 = 6%2 ; //reste2 sera égal à 0

Opérations de comparaison :

== **!=** **<** **>** **<=** **>=**

Exemple: *if (a == b) {}*

Opérateurs logiques :

&& (et) **||** (ou)

Exemple : *if (a > 0 && b > 0) {}*
// si a supérieur à 0 et b supérieur à 0

Caractères spéciaux

\n	passage à la ligne
\t	tabulation
\0	caractère de fin de chaîne

Récapitulatif des bases du C

Entrée/Sortie

printf affichage Exemple: *int age = 12 ; char nom[] = "Tom" ;*

printf("Age: %d Nom : %s", age, nom);

// On veut afficher qqch dans la console. On peut afficher des variables du programme grâce à %x avec x correspondant au type de la variable.

// Dans cet exemple on aura en console : Age : 12 Nom : Tom

scanf lecture Exemple: *int age = 0 ;*

scanf_s *scanf_s("%d", &age);* //on demande à l'utilisateur de donner son age, l'utilisateur doit rentrer un entier (%d).

Structures de contrôle :

/!\ pas de ; on le met seulement pour les instructions à l'intérieur des blocs { }

if - else if - else *« si – sinon si – sinon »*

for *« pour tel valeur à tel valeur avec un pas de tant »* : boucle avec compteur.
Pour effectuer un groupe d'instructions un nombre de fois connu à l'avance.

while *« tant que »* : boucle conditionnelle.
Pour effectuer un groupe d'instruction de manière répétée jusqu'à ce qu'une certaine condition devienne fausse.

```c
int age = 12;

if (age < 3) // si la condition est vérifiée on rentre dans le bloc
{
    printf("Vous êtes un bébé, vous avez %d ans.", age);
}
else if (age < 13) // si la condition précédente n'est pas vérifiée et que celle ci est vrai on rentre dans ce bloc
{
    printf("Vous êtes un enfant, vous avez %d ans.", age);
}
else if (age < 18) // si les 2 conditions précédentes ne sont pas vérifiées et que celle ci est vrai on rentre dans ce bloc
{
    printf("Vous êtes un adolescent, vous avez %d ans.", age);
}
else // sinon (donc aucunes des conditions précédentes ont été vérifiéesà on rentre dans ce bloc
{
    printf("Vous êtes un adulte, vous avez %d ans.", age);
}
```

```c
// pour (condition initiale; condition d'arrêt; pas)
for (int i = 0; i < 5; i++)
{
// on teste la condition au début du bloc
    printf("%d ", i);
// la pas s'incrémente à la fin du bloc
}
// sera affiché 0 1 2 3 4
```

```c
int cpt = 0;
// tant que (condition respectée)
while (cpt <= 4)
{
    // alors on fait ce qui est dans les accolades
    cpt++;
    printf("cmt = %d ", cpt);
}
// affichera cmt = 1 cmt = 2 cmt = 3 cmt = 4 cmt = 5
```

break	arrête une **boucle**
continue	passe à **l'itération suivante** d'une **boucle**
return	sort d'une **fonction**

Récapitulatif des bases du C

Tableaux : *type nomDeLaVariable[taille] = { les valeurs séparées par des , } ;*

```
int tb_entier[4] = {2 , 1, 6, 12} ;

char chaine[6] = "Hello" ;        // /!\ Si la chaîne de caractère contient 5 caractères, il faut rajouter 1 à la taille pour le
                                  caractère de fin de chaîne \0

float tableau_valeurs[] = { 4.5f , 8.2f , -7.9f , 54.7f , -56.8f } ; // ce tableau aura une taille de 5 éléments, il ne sera pas possible d'en rajouter.
```

Pour accéder à un élément du tableau, il faut utiliser les **[ ]** et mettre à l'intérieur l'indice de l'élément dans le tableau.
/!\ les indices commencent à 0.

Exemple :

```
float tableau_valeurs[] = { 4.5f , 8.2f , -7.9f , 54.7f , -56.8f };
for (int i = 0; i < 5; i++)
{
        printf("%f\n", tableau_valeurs[i]);
}
```

Pour modifier une valeur de tableau il faut faire par exemple : `tableau_valeurs[0] = 47.2f;`

Portée d'une variable

Variable locale

Une variable est dite locale quand elle est créée dans un bloc **{ }**
Elle est détruite à la sortie du bloc.

Exemple :

```
int main()
{
  int val1 = 4 ; // val1 est créée
  {
     int val2 = 2 ; // val2 est créée
  } // val2 est détruite
  val2 = 3 ; // erreur val2 n'existe plus
  return 0 ;
} // val1 est détruite
```

Fonctions :

Définition d'une fonction :

```
typeDeRetour nomDeLaFonction(arguments)
{ // début de la fonction
        // … instructions à l'intérieur de la fonction
        return variable ou valeur du type de retour ;
} // fin de la fonction
```

Prototype d'une fonction :

```
typeDeRetour nomDeLaFonction(arguments) ;
```

Une fonction est un **outil**. Si on en a besoin, il faut l'utiliser, pour cela il faut **l'appeler** et récupérer ce qu'elle nous **renvoi** (return) dans une variable.
Une fonction n'a pas toujours besoin de renvoyer quelque chose, dans ce cas le type de retour sera **void**. On pourra tout de même terminer la fonction par **return**, mais sans valeur à la suite.

Exemples :

```
int somme (int a, int b)
{
  int resultat = a + b ;
  return resultat ;
}
int main()
{
   int val1 = 2 ;
   int val2 = 3 ;
  int addition = somme(val1, val2);
  affichage() ;
  return 0 ;
}
```

```
void affichage ()
{
  printf("blabla \n");
  return ;
}
```

Laureen Fabre