

Langage C

Déclarations de variables

```
int nomEntier; (4 octets)
char nomCaractere; (1 octet)
float nomVirguleFlottante; (4 octets)
double nomVirguleFlottanteDoublePrecision; (8 octets)
```

Types abstraits

```
entier (4 octets)
réel (8 octets)
booléen (1 octet)
caractère (1 octet)
```

Opérations de base

```
nom_de_variable = valeur_à_affecter;
scanf("%d", &variableNumeriqueSaisieParLUtilisateur);
printf("On peut afficher du texte et une %d à l'écran", variableNumérique);
```

Déclaration de fonctions

```
type nomFonction(parametres){
    instructions;
    return valeurRetour;
}
```

Appel de fonction

```
[maVariable] = nomFonction(parametres);
(L'affectation de la valeur de retour à une variable est facultative)
```

Conditionnelles et boucles

<pre>if (condition) { instruction(s); } else { instruction(s) 2; }</pre>	<pre>while (condition) { instruction(s); } do { instruction(s); } while (condition)</pre>	<pre>int variable; for (variable = 0; variable < 42; variable++) { instruction(s); }</pre>
--	---	---

Formats

int “%d”, char “%c”, float “%f” (“%Xf” pour X après la virgule), string “%s” (“%Xs” pour X lettres)

Structures

```
struct NomStructure { ... }; typedef struct NomStructure nomType;  
...  
nomType.nomChamp = valeur;
```

Tableaux

```
type monTableau[nombreCases] = {valeur1, valeur2, ...};  
On peut utiliser des tableaux à deux dimensions. La syntaxe est la même.  
type monTableau[nombreCases][nombreCases2] = {{v1, v2, v3}, {v4,v5,v6}};
```

Pointeurs

maFonction(&variable) permet de communiquer l'adresse de la variable et non sa valeur.

<pre>void <i>maFonction</i>(<i>type</i> * <i>adresse</i>){ ... *<i>adresse</i> = <i>valeur</i>; ... }</pre>	<pre>void <i>maFonction</i>(<i>typeComposé</i> * <i>adresse</i>){ ... (*<i>adresse</i>).<i>champ1</i> = <i>valeur</i>; ... }</pre>
---	--

Un tableau est un pointeur.

Chaînes

Une chaîne de caractères est un tableau de caractères se terminant par ‘\0’.
Penser à inclure <string.h> pour des fonctions utiles (strcpy, strcmp, etc...).