R101 — Introduction au développement

Initiation à la programmation en langage C — Les variables

Clément Pagès
clement.pages@u-pec.fr

Université Paris-Est Créteil
BUT Informatique 1re année

2025-2026

Licence CC-BY-SA

Sommaire

1 Notion de variable

2 Des variables qui varient

3 Variables booléennes

Licence CC-BY-SA

Qu'est-ce qu'une variable ?

Définition

Une variable est un emplacement mémoire qui peut contenir des données (nombres, caractères, textes, ou autres données plus compliquées). Les variables sont identifiées par leur nom.

Licence CC-BY-SA

Qu'est-ce qu'une variable ?

Définition

Une variable est un emplacement mémoire qui peut contenir des données (nombres, caractères, textes, ou autres données plus compliquées). Les variables sont identifiées par leur nom.

- En C, les variables sont typées.
- Chaque variable doit être déclarée au début du bloc de code dans lequel elle se situe.
- On *peut* attribuer aux variables une valeur pour l'initialiser, mais ce n'est pas obligatoire.

Licence CC-BY-SA

Déclaration de variables I

```c
#include <stdio.h> // Pour avoir la fonction printf
int main(void) {
int x; // Déclaration d'une variable de type entier.
float y, z; // Variable de type flottant.
x = 456;
y = 0.5;
z = 1.0/3.0;
printf("Valeur de x : %d.\n", x);
printf("Valeur de x en hexadecimal : %x.\n", x);
printf("y=%f et z=%.3f", y, z);
}
```

Déclaration de variables II

Initialisation et déclaration des variables

Il est recommandé d'initialiser une variable au moment où elle est déclarée :

```c
int x = 456;
```

- Une variable déclarée mais non initialisée peut contenir n'importe quelle valeur.

Types de variables

Type de donnée	Signification	Taille (en octets)	Plage de valeurs acceptée
char	Caractère	1	-128 à 127
unsigned char	Caractère non signé	1	0 à 255
short int short	Entier court	2	-32 768 à 32 767
unsigned short int unsigned short	Entier court non signé	2	0 à 65 535
int	Entier	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	-32 768 à 32 767 -2 147 483 648 à 2 147 483 647
unsigned int	Entier non signé	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	0 à 65 535 0 à 4 294 967 295
long int long	Entier long	4	-2 147 483 648 à 2 147 483 647
unsigned long int unsigned long	Entier long non signé	4	0 à 4 294 967 295
float	Flottant (réel)	4	$3.4*10^{-38}$ à $3.4*10^{38}$
double	Flottant double	8	$1.7*10^{-308}$ à $1.7*10^{308}$
long double	Flottant double long	10	$3.4*10^{-4932}$ à $3.4*10^{4932}$

Licence CC-BY-SA

Sommaire

1 Notion de variable

2 Des variables qui varient

3 Variables booléennes

Licence CC-BY-SA

Opérations arithmétiques

Additions, soustractions, multiplications Opérateurs `+` `-` `*`.

La division Opérateur `/`.

Le modulo Opérateur `%`. Reste de la division euclidienne entre deux entiers.

Licence CC-BY-SA

Opérations arithmétiques

Additions, soustractions, multiplications Opérateurs `+` `-` `*`.

La division Opérateur `/`.

Le modulo Opérateur `%`. Reste de la division euclidienne entre deux entiers.

Plus de détails sur les divisions

- La division de deux entiers donne le quotient entier, en ignorant le reste.
- La division de deux flottants donne un quotient flottant.

```c
printf("15 / 6 = %d\n", 15 / 6);          // Affiche : 15 / 6 = 2
printf("15.0 / 6.0 = %f\n", 15.0 / 6.0); // Affiche : 2.500000
printf("15 % 6 = %d\n", 15 % 6);          // Affiche : 15 % 6 = 3
```


Licence CC-BY-SA

Opérations arithmétiques

Additions, soustractions, multiplications Opérateurs `+` `-` `*`.

La division Opérateur `/`.

Le modulo Opérateur `%`. Reste de la division euclidienne entre deux entiers.

Plus de détails sur les divisions

- La division de deux entiers donne le quotient entier, en ignorant le reste.
- La division de deux flottants donne un quotient flottant.

```c
printf("15 / 6 = %d\n", 15 / 6);          // Affiche : 15 / 6 = 2
printf("15.0 / 6.0 = %f\n", 15.0 / 6.0); // Affiche : 2.500000
printf("15 % 6 = %d\n", 15 % 6);          // Affiche : 15 % 6 = 3
```

Priorités opératoires

Le langage C respecte les priorités des opérations usuelles.

Licence CC-BY-SA

Opérations sur les variables

Il est possible d'effectuer les opérations arithmétiques avec des variables, ou d'affecter le résultat d'opérations à des variables.

Licence CC-BY-SA

Opérations sur les variables

Il est possible d'effectuer les opérations arithmétiques avec des variables, ou d'affecter le résultat d'opérations à des variables.

C

```c
int a = 2;
int b = 7;
int somme = a + b;

printf("%d + %d = %d\n", a, b,
                         somme);
```

Console

```
2 + 7 = 9
```


Licence CC-BY-SA

Opérations sur les variables

Il est possible d'effectuer les opérations arithmétiques avec des variables, ou d'affecter le résultat d'opérations à des variables.

```c
int a = 2;
int b = 7;
int somme = a + b;

printf("%d + %d = %d\n", a, b,
                         somme);
```

```
2 + 7 = 9
```

Console

```c
int a = 8;
int b = 33;

printf("b/a*3 + 19%4 = %d\n",
        b/a*3 + 19%4);
```

Console

```
b/a*3 + 19%4 = 15
```


Licence CC-BY-SA

Opérations sur les variables

Modifier la valeur d'une variable

Au cours de l'exécution d'un programme, les valeurs affectées aux variables peuvent évoluer.

Licence CC-BY-SA

Opérations sur les variables

Modifier la valeur d'une variable

Au cours de l'exécution d'un programme, les valeurs affectées aux variables peuvent évoluer.

Avec des variables de type `int`

```c
int a = 2;
printf("a = %d\n", a);
a = a-7;
printf("a = %d\n", a);
```

```
a = 2
a = -2
```


Licence CC-BY-SA

Opérations sur les variables

Modifier la valeur d'une variable

Au cours de l'exécution d'un programme, les valeurs affectées aux variables peuvent évoluer.

Avec des variables de type `int`

```c
int a = 2;
printf("a = %d\n", a);
a = a-7;
printf("a = %d\n", a);
```

```Console
a = 2
a = -2
```

Avec des variables de type `char`

```c
char c = 75;
printf("c = %c\n", c);
c = c-10;
printf("c = %c\n", c);
```

```Console
c = K
c = A
```


Licence CC-BY-SA

Opérations sur les variables

Syntaxe raccourcie

Il existe une syntaxe raccourcie pour modifier la valeur d'une variable. Son utilisation est recommandée car elle améliore la lisibilité du code source.

Syntaxe raccourcie	Syntaxe longue
`var += valeur`	`var = var+valeur`
`var -= valeur`	`var = var-valeur`
`var *= valeur`	`var = var*valeur`
`var /= valeur`	`var = var/valeur`
`var %= valeur`	`var = var%valeur`
`var++`	`var = var+1`
`var--`	`var = var-1`

Licence CC-BY-SA

Afficher les valeurs des variables

Pour afficher du texte dans la console, on utilise la fonction `printf`.

Type	Code du format
Caractère	`%c`
Entier signé	`%d` ou `%i`
Entier signé long	`%ld` ou `%li`
Entier non signé	`%u`
Flottant	`%f`

- Représentation hexadécimale des entiers : `%x`
- Écriture scientifique des flottants : `%e`
- Précision dans l'écriture décimale des flottants : `%.2f` affichera deux chiffres significatifs.

Licence CC-BY-SA

Récupérer des saisies clavier I

La bibliothèque `stdio.h` fournit la fonction `scanf`.

C

```c
#include <stdio.h>

int main(void) {
    int annee=0;

    printf("Quelle est votre année de naissance ? ");
    scanf("%d", &annee); // Observez la présence du synbole &.
                         // Cf. cours sur les pointeurs.
    printf("Vous êtes né.e en %d.", annee);
    return 0;
}
```

Console

```
Quelle est votre année de naissance ? 2007
Vous êtes né.e en 2007.
```


Licence CC-BY-SA

Récupérer des saisies clavier II

Type	Code
`char`	`%c`
`int`	`%d` ou `%i`
`long int`	`%ld` ou `%li`
`unsigned int`	`%u`
`float`	`%f`

Utilisation de la fonction `scanf`

Fonction simple *en apparence*, mais son utilisation sécurisée est très compliquée.
Dans cette ressource introductive, on ignore ce point.

Licence CC-BY-SA

Sommaire

1 Notion de variable

2 Des variables qui varient

3 Variables booléennes

Licence CC-BY-SA

Le type booléen

Définition

Une variable ou une expression est de type booléen si elle peut prendre deux valeurs : `vrai` (ou plutôt `true`) ou `faux` (`false`).

Licence CC-BY-SA

Le type booléen

Définition

Une variable ou une expression est de type booléen si elle peut prendre deux valeurs : `vrai` (ou plutôt `true`) ou `faux` (`false`).

Exemples

- L'expression `18<25` vaut `true`.
- `char c = 'r'; c<='a'` vaut `false`.

Licence CC-BY-SA

Le type booléen

Définition

Une variable ou une expression est de type booléen si elle peut prendre deux valeurs : `vrai` (ou plutôt `true`) ou `faux` (`false`).

Exemples

- L'expression `18<25` vaut `true`.
- `char c = 'r'; c<='a'` vaut `false`.

Les booléens en C

- Le Langage C n'a pas de type booléen prédéfini.
- À la place, on utilise des valeurs entières.
  - Si une expression prend la valeur `0`, elle est interprétée comme `false`.
  - Toute autre valeur est interprétée comme `true`.

Licence CC-BY-SA

Opérateurs logiques

Opérateurs logiques en C

- ET logique : `&&`
- OU logique : `||`
- Négation logique : `!`

Opérateurs de comparaison en C

- Inégalités strictes et larges : `<` `<=` `>` `>=`
- Égalité : `==`
- Non égalité : `!=`

Licence CC-BY-SA

Opérateurs logiques

Opérateurs logiques en C

- ET logique : `&&`
- OU logique : `||`
- Négation logique : `!`

Opérateurs de comparaison en C

- Inégalités strictes et larges : `< <= > >=`
- Égalité : `==`
- Non égalité : `!=`

Un peu de calcul

Évaluer les expressions logiques suivantes.

```c
int x = 4;
(x>=0) && (x<100)
!(x>100)
(x>0) || (x<=-2)
!( (x>=0) && (x<100) )
!x
```


Licence CC-BY-SA

La bibliothèque `stdbool.h`

- `stdbool.h` est la bibliothèque standard en C pour définir le type `bool`. Une variable de type `bool` peut prendre les valeurs `true` et `false`.
- Pour l'utiliser, écrire la ligne `#include <stdbool.h>` au début du code source.
- Son utilisation est recommandée pour améliorer la lisibilité du code source...
- ... mais cela ne crée pas un « vrai » type booléen, car l'instruction `bool var = 4;` est considérée comme valide.

Licence CC-BY-SA