1) Entrées-sorties et variables

Programmation C (2013-2014)

Guillaume Claret

1 Premier programme

```
// Un commentaire commence par //.
// On inclut les fonctions prédéfinies usuelles :
#include <stdio.h>

// "main" est toujours le nom de la fonction principale :
int main() {
  // Un message de bienvenue. Les instructions se terminent par un ";".
  printf("Bonjour.\n");

  // On fini le programme en retournant la valeur par défaut 0 :
  return 0;
}
```

2 Compilation

Écriture On crée un fichier `ex1.c` que l'on ouvre avec `gedit` (ou un autre éditeur) puis on écrit le programme.

Compilation À la ligne de commande (`-Wall` active tous les warnings, `ex1` est le nom de l'exécutable à générer) :

```
gcc -Wall ex1.c -o ex1
```

Lancement À la ligne de commande :

```
./ex1
```

3 Variables

Les variables permettent de stocker des valeurs et de les modifier.

Déclaration Un type suivit d'une liste de noms. Les noms de variables contiennent des lettres sans accents, des chiffres ou des underscores (symbole _). Ils ne peuvent pas commencer par un chiffre. On peut donner une valeur initiale.

```
int var1, x = 12, n;
```

Types Quelques types standards :

Type	Taille[1]	Contenu	Intervalle
`char`	8 bits	entier	$[-128; 127]$
`short`	16 bits	entier	$[-32.768; 32.67]$
`int`	32 bits	entier	$[-2^{31}; 2^{31} - 1]$
`long`	64 bits	entier	$[-2^{63}; 2^{63} - 1]$
`unsigned char`	8 bits	entier	$[0; 255]$
`unsigned short`	16 bits	entier	$[0; 65.535]$
`unsigned int`	32 bits	entier	$[0; 2^{32} - 1]$
`unsigned long`	64 bits	entier	$[0; 2^{64} - 1]$
`float`	32 bits	flottant	$3,4 \times 10^{-38}$ à $3,4 \times 10^{38}$
`double`	64 bits	flottant	$1,7 \times 10^{-308}$ à $1,7 \times 10^{308}$

Modification La variable à modifier est à gauche. Pour ajouter 3 à `n` :

```
n = n + 3;
```

4 Entrées-sorties

Lire Avec `printf` suivit du message entre guillemets. Le `\n` représente un retour à la ligne :

```
printf("Bonjour.\n");
```

Pour afficher le contenu d'une variable entière :

```
printf("Valeur de x : %d\n", x);
```

Écrire Avec `scanf` suivit d'un format et de variables précédées par des `&`. Pour lire deux entiers :

```
scanf("%d%d", &n1, &n2);
```

Codes spéciaux Certains codes n'ont de sens que pour `printf` ou que pour `scanf` :

Code	Signification
`\n`	retour à la ligne
`\t`	tabulation
`\"`	"
`\\`	\
`%d`	entier
`%u`	entier non-signé
`%f`	flottant
`%.2f`	flottant avec 2 chiffres après la virgule
`%8f`	flottant affiché avec une largeur d'au moins 8 caractères
`%c`	caractère
`%c*`	caractère lu puis oublié (utile pour manger des retours à la ligne)
`%%`	%

1. Peut dépendre de l'architecture, valeurs pour un PC 64 bits sous Linux.