INF111 – Notes langage C

Lucas Pastor

1 Variables

Déclaration de variable de type entier.

```
1    int x;
```

Déclaration de variable de type réel (nombre à virgule).

```
1    double x;
```

Déclaration de variable de type caractère (lettre).

```
1    char x;
```

Déclaration et affectation d'une variable de type entier.

```
1    int x;
2    x = 2;
```

Déclaration et affectation d'une variable de type réel.

```
1    double x;
2    x = 2.5;
```

Déclaration et affectation d'une variable de type caractère.

```
1    char x;
2    x = 'A';
```

2 Affichage et saisie de données

2.1 Affichage

La fonction **printf** permet d'afficher du texte dans le terminal UNIX. Le premier paramètre de **printf** est une chaîne de caractères (entre double quotes).

```
1    printf("Bonjour.");
```

Si l'on souhaite retourner à la ligne après l'affichage de notre texte, on ajoute **n** à la fin de la chaîne de caractères.

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

Si l'on souhaite afficher les valeurs de certaines variables dans notre texte on indique leurs emplacement avec **%** suivi du type à afficher, par exemple **d** pour une valeur entière. Les paramètres de **printf** suivant la chaîne de caractères sont exactement l'ordre des variables à afficher. Par exemple, pour afficher la valeur de x à l'écran.

```
1    int x;
2    x = 2;
3
4    printf("La valeur de x est %d.\n", x);
```

Ce qui affichera exactement « **La valeur de x est 2.** ⏎ ».
On peut afficher plusieurs valeurs d'un coup.

```
1    int x;
2    x = 2;
3
4    double y;
5    y = 2.5;
6
7    printf("X = %d, Y = %f.\n", x, y);
```

2.2 Saisie

Pour permettre à l'utilisateur de saisir des données on utilise **scanf**. La fonction **scanf** s'utilise comme printf. Le premier paramètre est une chaîne de caractères formatée, suivi d'un paramètre associé à la variable. Ne pas oublier le symbole **&** devant la variable.

Saisie d'un nombre entier.

```
1    int x;
2    scanf("%d", &x);
```

Saisie d'un nombre réel.

```
1    double x;
2    scanf("%lf", &x);
```

Saisie d'un caractère. **Attention, il faut mettre un espace devant le symbole %).**

```
1    char x;
2    scanf(" %c", &c);
```

Il est également possible de saisir plusieurs valeurs d'un coup.

```
1    int x;
2    double y;
3
4    scanf("%d %lf", &x, &y);
```

Tableau résumé

Type	Symboles printf et scanf
int	%d ou %i
double	%lf pour scanf, et %f pour printf
char	%c

3 Instructions conditionnelles

Pour permettre à un programme de réagir en fonction de la valeur d'une ou plusieurs variables, on utilise les instructions conditionnelles.

On utilise **if** si on veut exécuter du code seulement sous une certaine condition.

```
1     int x;
2     scanf("%d", &x);
3
4     if (x > 0)
5     {
6         /*
7            Le code dans ce bloque est exécuté
8            uniquement si x est strictement
9            plus grand que zéro.
10        */
11        printf("X est plus grand que zéro.\n");
12    }
```

On utilise **if-else** si on veut exécuter du code sous une certaine condition **OU** exécuter un autre morceau de code si cette condition n'est pas respectée.

```
1     int x;
2     scanf("%d", &x);
3
4     if (x == 0)
5     {
6         /*
7            Le code dans ce bloque est exécuté
8            uniquement si x est égal à zéro.
9         */
10        printf("X égal à zéro.\n");
11    }
12    else
13    {
14        /*
15           Le code dans ce bloque est exécuté
16           uniquement si x est inférieur ou
17           égal à zéro.
18        */
19        printf("X différent de zéro.\n");
20    }
21
22    /*
23       Le programme reprend son
24       fonctionnement normal.
25       Le printf suivant sera toujours exécuté.
26    */
27    printf("Coucou.\n");
```

On utilise **if-elseif-else** si on veut exécuter un morceau de code numéro 1 en fonction d'une condition numéro 1, **OU** un morceau de code numero 2 en fonction d'une condition numéro 2, **OU** etc ..., **OU**, si aucune condition n'est vraie, on passe dans le **else**.

```
1     int x;
2     scanf("%d", &x);
3
4     if (x == 1)
5     {
6         printf("X = 1.\n");
7     }
8     else if (x == 2)
9     {
10        printf("X = 2.\n");
11    }
12    else if (x == 3)
13    {
14        printf("X = 3.\n");
15    }
16    else
17    {
18        /*
19           Le code dans ce bloque est executé
20           uniquement si X est différent
21           de 1 ET différent de 2 ET différent
22           de 3.
23        */
24        printf("X = %d.\n", x);
25    }
```

Tableau résumé

Opérateurs booléens	Symbole
égalité	==
différent	!=
inférieur strict	<
inférieur ou égal	<=
supérieur strict	>
supérieur ou égal	>=
et	&&
ou	||
négation	!

4 Boucle while

La boucle **while** permet d'exécuter du code tant qu'une expression booléenne est vraie.

```
1     int i;
2     i = 1;
3
4     while (i <= 100)
5     {
6         printf("Valeur de i = %d.\n", i);
7
8         /*
9            Ne pas oublier d'incrémenter le
10           compteur de boucle.
11        */
12        i = i + 1;
13    }
```

5 Boucle for

La boucle **for** permet d'exécuter un bloque de code en itérant sur les valeurs d'une variable de contrôle (indice de boucle). La syntaxe est la suivante (ce qui suit est du pseudo code) :

```
1     for (initialisation ; test ; incrémentation)
2     {
3         opérations
4     }
```

On commence la boucle **for** par l'*initialisation*, puis on effectue les différentes *opérations* liées à son bloque, et après chaque tour de boucle on effectue une *incrémentation* (ou *décrémentation*) sur notre variable de contrôle. Enfin, on sort de la boucle lorsque le *test* devient faux.

Ainsi pour afficher les entiers de 1 à n on écrira :

```c
    int i;

    for (i = 1; i <= n; i++)
    {
        printf("%d\n", i);
    }
```

6 Exemples de code complet

Code minimal d'un programme C.

```c
    #include <stdio.h>

    int main(void)
    {
        return 0;
    }
```

Tant que l'utilisateur n'a pas tapé la lettre 's', on continue.

```c
    #include <stdio.h>

    int main(void)
    {
        char lettre;
        lettre = 'a';

        while (lettre != 's')
        {
            printf("Tapez un caractère.\n");
            scanf(" %c", &lettre);
        }

        printf("Fin de programme.\n");

        return 0;
    }
```