Programmation Langage C
Structures de contrôle

Youssef ALJ

2 mars 2020

1. Structures de contrôle
   - if … else
   - Opérateurs de test
   - Division euclidienne
   - switch … case
   - ”if” et ”else if”

- instruction1 n'est réalisée que si la condition est réalisée.

```
if (condition){
    instruction1;
}
```

Exemple

```c
void main()
{
    int var;
    printf("veuillez saisir un entier\n");
    scanf("%d", &var);
    if(var >= 0){
        printf("le nombre saisi est positif");
    }
}
```

if . . . else

```
if (condition){
    instruction1;
}
else{
    instruction2;
}
```

- On exécute instruction1 si la condition est réalisée.
- Sinon on exécute instruction2.

Exemple

```c
void main()
{
    int var;
    printf("veuillez saisir un entier\n");
    scanf("%d", &var);
    if(var >= 0){
        printf("le nombre saisi est positif");
    }
    else{
        printf("le nombre saisi est negatif");
    }
}
```

if . . . else if . . . else

```
if (condition1){
    instruction1;
}
else if (condition2){
    instruction2;
}

else if(condition3){
    instruction3;
}

else{
    instruction_par_defaut;
}
```

- Si condition1 est vérifiée on exécute instruction1.
- Si condition2 est vérifiée on exécute instruction2.
- Si condition3 est vérifiée on exécute instruction3.
- Si aucune des conditions n'est vérifié alors on exécute instruction par défaut.

Opérateurs

Opérateur	Signification
==	test d'égalité
! =	test de différence
>	supérieur
>=	supérieur ou égal
<	inférieur
<=	inférieur ou égal
&&	ET logique
||	OU logique
!	NOT logique
$\wedge$	XOR logique

A	B	A ET B	A ou B	$A \oplus B$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Exemple

Comment tester si deux nombrer sont égaux avec l'opérateur XOR (en C ^) ?

Comment tester si deux nombrer sont égaux avec l'opérateur XOR (en C ^)?

```c
#include<stdio.h>

int main()
{
   int x = 10;
   int y = 10;
   // pour deux nombres egaux l'operateur xor renvoir zero
   // ici z vaut 0
   // int z = x ^ y;
   // printf("z=%d", z);
   if ( !(x ^ y) )
      printf(" x is equal to y ");
   else
      printf(" x is not equal to y ");
   return 0;
}
```

Que fait le programme suivant ?

```c
// dans_le_mille.c
#define CIBLE_1 1000
#define CIBLE_2 100
#include <stdio.h>
#include <math.h>
int main() {
    float x, y;
    int danslemille, dehors, total_points=0;
    printf("x ? "); scanf("%f", &x); printf("x = %.2f\n", x);
    printf("y ? "); scanf("%f", &y); printf("y = %.2f\n", y);
    float d = sqrt(x*x + y*y);
    danslemille = (d < 1);
    dehors = (d > 3);
    if (danslemille) total_points = CIBLE_1;
    else if (!dehors) total_points = CIBLE_2;
    printf("total points = %d\n", total_points);
    return (0);
}
```

Théorème de la division euclidienne

Pour tout $(a, b) \in \mathbb{N} \times \mathbb{N}^*$, il existe un unique couple $(q, r) \in \mathbb{N} \times \mathbb{N}$ tel que :

$$a = bq + r \qquad \text{avec } 0 \leq r < b$$

Exemple

Le reste de la division euclidienne de tout entier a par 2 est soit 0 soit 1.

Conséquence immédiate

- un entier a est pair si le reste de division euclidienne de a par 2 est 0. Il est impair sinon.

Division euclidienne en C

- le quotient de la division euclidienne d'un entier a par un entier b est obtenu avec l'opérateur /.
- le reste est obtenu avec %.

Exemple

```c
#include<stdio.h>

int main()
{
    int a,b,q,r;
    printf("veuillez saisir a\n");
    scanf("%d",&a);
    printf("veuillez saisir b\n");
    scanf("%d",&b);
    // calcul du quotient de a par b
    q = a/b;
    // calcul du reste de a par b
    r = a%b;
    printf("Le quotient est q=%d\n", q);
    printf("Le reste est r=%d\n", r);
    return 0;
}
```

Exercice

- On veut vérifier si un nombre donné est multiple de 3.
- On veut vérifier si un nombre donné est pair ou pas.

- Une solution afin d'éviter les imbrications des instructions if.
- Si variable prend valeur1 on exécute instruction10 et instruction11.

```
switch(variable){
case valeur1:
     instruction10;
     instruction11;
     break;
case valeur2:
     instruction20;
     instruction21;
     break;
default :
     instruction_par_defaut;
}
```

Pourquoi utiliser ”else if” au lieu de plusieurs ”if” ?

```c
#include <stdio.h>
int main()
{
    int i;
    printf("veuillez saisir i\n");
    scanf("%d", &i);
    if(i > 0)
    {
        printf("i > 0\n");
    }
    if(i > 1)
    {
        printf("i > 1\n");
    }
    if(i > 2)
    {
        printf("i > 2\n");
    }
}
```

Ce programme va afficher :
veuillez saisir i
9
$i > 0$
$i > 1$
$i > 2$

```c
#include <stdio.h>
int main()
{
    int i;
    printf("veuillez saisir i\n");
    scanf("%d", &i);
    if(i > 0)
    {
        printf("i > 0\n");
    }
    else if(i > 1)
    {
        printf("i > 1\n");
    }
    else if(i > 2)
    {
        printf("i > 2\n");
    }
}
```

Ce programme va afficher :
veuillez saisir i
9
$i > 0$

- On n'écrit pas : `if (12 <= x <= 14)`
- On écrit plutôt : `if ((12 <= x)&& (x <= 14))`