LANGAGE DE PROGRAMMATION: C

F. Karami

La syntaxe du langage

- Instructions, Expressions et Opérateurs
- Les structures de contrôle
- La récursivité
- Les conversions de types
- Principales fonctions d' entrées-sorties standard

Instructions, Expressions et Opérateurs

- Les instructions :
  - **Une instruction** représente une tâche à accomplir par l'ordinateur. En langage C, on écrit une instruction par ligne et elle se termine par un point virgule(à l'exception de #define et #include). Par exemple:

    x=2+3;

    est une instruction d'affectation. Elle demande à l'ordinateur d'ajouter 2 et 3 et d'attribuer le résultat à la variable x.
  - **Les blocs:**
    - Un bloc (ou instructions composées) est un groupe d'instructions entre accolades:

```
{
  printf("Hello");
  printf("world!");
}
```

```
{printf("Hello");
  printf("world!");}
```

Instructions, Expressions et Opérateurs

- Les expressions:

  Une expression est une combinaison d'opérateurs et d'opérandes(variables, constantes). Autrement tout ce qui représente une valeur numérique. Une expression génère toujours un résultat d'un type bien défini qu'on appellera le *type de l'expression*.

- **Les expressions Simples**
  - L'expression simple est constituée d'une seule variable, d'une constante. Par exemple:
    - PI //constante dont la valeur est définie par #define
    - Taux //variable
- **Les expressions complexes**
  - Les expressions complexes sont constituées d'expressions plus simples avec des opérateurs. Par exemple:
    - 2+8
    - 8+(5*taux)+(taux*taux)/cout

Instructions, Expressions et Opérateurs

- Les opérateurs: opérateur d'affectation
  - L'opérateur d'affectation est le signe(=). Dans le programme C : x=y;

    ne signifie pas x égal y. Elle indique à l'ordinateur d'affecter la valeur de la variable y à x. Cette instruction doit être composée d'une expression à droite du signe égale, et d'un nom de variable à gauche de ce signe:

    variable=expression;
  - Exemple:
    - x=6+(y=4+5);
    - a=b=2;

Instructions, Expressions et Opérateurs

- Les opérateurs: opérateur d'affectation
  - **Opération et affectation combinées op=**

    var += exp ; équivalent à var = var + (exp) ;

    var -= exp ; var = var - (exp) ;

    var *= exp ; var = var * (exp) ;

    var /= exp ; var = var / (exp) ;

    var %= exp ; var = var % (exp) ;
  - **Attention** :

    pas d'espace entre l'opérateur op et le égal =,

    x *= y+1 est équivalent à x = x*(y+1) et pas x = x*y+1.

Instructions, Expressions et Opérateurs

- Opérateurs arithmétiques

Opérateur	Traduction	Exemple	Résultat
+	Addition	`x + y`	l'addition de x et y
-	Soustraction	`x - y`	la soustraction de x et y
*	Produit	`x * y`	la multiplication de x et y
/	Division	`x / y`	le quotient de x et y
%	Reste	`x % y`	Reste de la division euclidienne de x par y
+(unaire)	Signe positif	`+x`	la valeur de x
-(unaire)	Signe négatif	`-x`	la négation arithmétique de x
++(unaire)	Incrément	`x++` ou `++x`	x est incrémenté ($x = x + 1$). L'opérateur préfixe `++x` (resp. suffixe `x++`) incrémente x avant (resp. après) de l'évaluer
--(unaire)	Decrément	`x--` ou `--x`	x est décrementé ($x = x - 1$). L'opérateur préfixe `--x` (resp. suffixe `x--`) décrémente x avant (resp. après) de l'évaluer

Instructions, Expressions et Opérateurs

- Opérateurs arithmétiques
  - Remarques :
    - Les opérandes de ces opérateurs arithmétiques peuvent appartenir à tout type arithmétique seul l'opérateur % requiert des types entiers.
    - Le résultat d'une division d'entiers est aussi un entier,
      Exemple :
      - 6 / 4 // Resultat: 1
      - 6 % 4 // Resultat: 2
      - 6.0 / 4.0 // Resultat: 1.5

Instructions, Expressions et Opérateurs

- Opérateurs arithmétiques
  - Remarques
    - Les opérateurs unaires opèrent sur une seule variable ou opérande.
    - Concernant l'incrémentation pré/postfixe, voici un petit exemple: Supposons que la valeur de N soit égale à 3 :
      - Incrémentation postfixe : X = N++ ;

        Résultat : N = 4 et X = 3
      - Incrémentation préfixe : X = ++N ;

        Résultat : N = 4 et X = 4
      - Décrémentation postfixe : X = N-- ;

        Résultat : N = 2 et X = 3
      - Décrémentation préfixe : X = --N ;

        Résultat : N = 2 et X = 2

Instructions, Expressions et Opérateurs

- Opérateurs de comparaison

  - Toute comparaison est une expression de type **int** qui renvoie la valeur 0 (faux) ou 1 (vraie). Il faut que les opérandes soient du même type arithmétique (ou des pointeurs sur des objets de même type).
  - Attention : il ne faut pas confondre l'opérateur d'égalité (==) avec celui d'affectation (=).
  - Les différents opérateurs de comparaison sont détaillés dans le tableau ci-dessous.

Instructions, Expressions et Opérateurs

- Opérateurs de comparaison

Opérateur	Traduction	Exemple	Résultat
<	inférieur	x < y	1 si x est inférieur à y
<=	inférieur ou égal	x <= y	1 si x est inférieur ou égal à y
>	supérieur	x > y	1 si x est supérieur à y
>=	supérieur ou égal	x >= y	1 si x est supérieur ou égal à y
==	égalité	x == y	1 si x est égal à y
!=	inégalité	x != y	1 si x est différent de y

Instructions, Expressions et Opérateurs

- Opérateurs de comparaison

```
#include <stdio.h>
main (){
   int x=14,y=1;   // x est différent de y
   if (x = y)        //erreur!!! il faudrait écrire ' if (x == y)'
     printf("x est égal à y (%d=%d)\n",x,y);
  else
     printf("x est différent de y (%d!=%d)\n",x,y);

system(« pause»);
}
```

Instructions, Expressions et Opérateurs

- Opérateurs logiques

Les opérateurs logiques, permettent de combiner le résultat de plusieurs expressions de comparaison en une seule expression logique. Les opérandes des opérateurs logiques peuvent être n' importe quel scalaire. Toute valeur différente de 0 est interprétée comme vraie (et 0 correspond à ' faux'). Comme pour les expressions de comparaisons les expressions logiques renvoient une valeur entière (0 =faux ; 1=vraie).

Remarque :

les opérateurs && et || évaluent les opérandes de gauche à droite et le résultat est connu dès l' opérande de gauche. Ainsi, l' opérande de droite n' est évaluée que si celle de gauche est vraie dans le cas de l' opérateur && (respectivement fausse dans le cas de l' opérateur ||).

Exemple:

(i < max) && (f(14) == 1), la fonction f n' est appelée que si i < max.

Instructions, Expressions et Opérateurs

- Opérateurs logiques

Opérateur	Traduction	Exemple	Résultat
&&	ET logique	x && y	1 si x et y sont différents de 0
||	OU logique	x || y	1 si x et/ou y sont différents de 0
!	NON logique	!x	1 si x est égal à 0. Dans tous les autres cas, 0 est renvoyé.

- Exemples
  - L'expression : 32 && 40 vaut 1
  - L'expression : !65.34 vaut 0
  - L'expression : !!0 vaut 0

Instructions, Expressions et Opérateurs

Autres opérateurs

- **Opérateur séquentiel (,)**
  - **<expr1> , <expr2>,., <exprN>**
  - Exprime des calculs successifs dans une même expression. Le type et la valeur de l'expression sont ceux du dernier opérande.
  - **Exemple : x = 5 , x + 6;** L'expression **a pour valeur 11**
- **Opérateur conditionnel (? :)**
  - **<expression> ? <expr1>: <expr2>**
  - <expression> est évaluée. Si sa valeur est non nulle, alors la valeur de <expr1> est retournée. Sinon, c'est la valeur de <expr2> qui est renvoyée.
  - **Exemple : max = a > b ? a : b**
  - si a est le plus grand, alors affectation à max du contenu de a sinon affectation du contenu de b

Instructions, Expressions et Opérateurs

- **Autres opérateurs**

Op.	Traduction	Exemple	Résultat
()	Appel de fonction	`f(x,y)`	Exécute la fonction f avec les arguments x et y
(type)	cast	`(long)x`	la valeur de x avec le type spécifié
`sizeof`	taille en bits	`sizeof(x)`	nombre de bits occupé par x
? :	Evaluation conditionnelle	`x?:y:z`	si x est différent de 0, alors y sinon z
,	séquencement	`x,y`	Evalue x puis y

Exercices

1) Soit les déclarations suivantes :

```
int n=10, p=4;
Long q=2;
Float x=1.75;
```

Donner le type et la valeur de chacune des expressions suivantes

a) n+q
b) n+x
c) n%p+q
d) n<p
e) q+3 *(n>p)
f) x*(q==2)
g) x*(q=2)
h) (q-2)&&(n-10)

Exercices

1) Soit les déclarations suivantes :

```
int n=10, p=4;
long q=2;
float x=1.75;
```

Donner le type et la valeur de chacune des expressions suivantes

a) n+q
b) n+x
c) n%p+q
d) n<p
e) q+3 *(n>p)
f) x*(q==2)
g) x*(p==2)
h) (q-2)&&(n-10)

a) long 12
b) float 11,75
c) long 4
d) int 0
e) long 5
f) float 1,75
g) float 0
h) Int 0

Exercices

2. n étant de type int, écrire une expression qui prend la valeur :

-1 si n est négatif

0 si n est nul

1 si n est positif

Exercices

2. n étant de type int, écrire une expression qui prend la valeur :

-1 si n est négatif

0 si n est nul

1 si n est positif

```
n ? (n>0 ? 1: -1) : 0
```

Les structures de contrôle

- On appelle structure de contrôle toute instruction servant à contrôler le déroulement de l'enchaînement des instructions à l'intérieur d'un programme, ces instructions peuvent être des instructions conditionnelles ou itératives.
- Parmi les structures de contrôle, on distingue :
  - structures de choix
    - if….else (choix conditionnel)
    - switch (choix multiple)
  - structures répétitives ou itérative ou boucle
    - for
    - while
    - do…while
  - Branchement inconditionnel
    - Break, goto, continue

Les structures de contrôle : if-else

- La construction **if-else** (*si-sinon*) est la construction logique de base du langage C qui permet d'exécuter un bloc d'instructions selon qu'une condition est vraie ou fausse.
- **Syntaxe**
  - **Forme 1**
    - **if** (expression)
      instruction1;

    la forme if est ici dans sa forme la plus simple.

    Si expression est vraie, instruction1 est exécutée.

    Si expression est fausse, instruction1 est ignorée.

Les structures de contrôle : if-else

- **Syntaxe: Forme 2**
  - ❖ **if** (expression)
    instruction1;
    else
    instruction2;

Si expression est vraie, instruction1 est exécutée, sinon c' est instruction2 qui est exécutée.

- **Forme 3**
  - ❖ **if** (expression1)
    instruction1;
    else if (expression2)
    instruction2;
    else
    instruction3;

Les instructions if sont imbriquées. Si expression1 est vraie, instruction1 est exécutée. Dans le cas contraire expression2 est évaluée si cette dernière est vraie instruction2 est exécutée. Si les deux expressions sont fausses, c' est instruction3 qui est exécuté

Les structures de contrôle : if-else

- Exemples:
  - Exemple 1
    - if (salaire >45.000)
      tax=0.30;
      else
      tax=0.25;
  - Exemple 2
    - if (age <18)
      printf('' mineur' ');
      else if (age <65)
      printf('' adulte' ');
      else
      printf('' personne agée' ');

Les structures de contrôle : if-else

```
#include <stdio.h>
main (){
    int i;
    printf("Tapez un nombre entier positif ou negatif: ");
    scanf("%d", &i);
 if (i<0) {
        i=-i;
        printf("J'ai remis i à une valeur positive.\n");
   }
 else
        printf("Vous avez tapé un nombre positif.\n");

system(« pause»);
}
```

Les structures de contrôle : Switch

- L'instruction switch est l'instruction de contrôle la plus souple du langage C. Elle permet à votre programme d'exécuter différentes instructions en fonction d'une expression qui pourra avoir plus de deux valeurs.

- On l'appelle aussi l'instruction d'aiguillage. Elle teste si une expression prend une valeur parmi une suite de constantes, et effectue le branchement correspondant si c'est le cas.

Les structures de contrôle : Switch

- Syntaxe

```
switch (<variable>) {
    case <valeur 1> : <action 1>; break;
    case <valeur 2> : <action 2>; break;
      ...
    default : <action n>;
}
```

Les structures de contrôle : Switch

- Remarques :
  - Le fonctionnement de cette instruction est le suivant :
    - expression est évaluée ;
    - s' il existe un énoncé case avec une constante qui est égale à la valeur de l' expression, le contrôle est transféré à l' instruction qui suit cet énoncé;
    - si un tel case n' existe pas, et si énoncé default existe, alors le contrôle est transféré à l' instruction qui suit l' énoncé default ;
    - si la valeur de l' expression ne correspond à aucun énoncé case et s' il n' y a pas d' énoncé default, alors aucune instruction n' est exécutée.
  - Attention.
    - Lorsqu' il y a branchement réussi à un case, toutes les instructions qui le suivent sont exécutées, jusqu' à la fin du bloc ou jusqu' à une instruction de rupture (break).

Les structures de contrôle : Switch

```
#include<stdio.h>
    main() {
 int a,b,y;
 char operateur;
  printf( '' Entrez un opérateur (+, -,* ou /):' ' );
  scanf( '' %c' ' ,& operateur);
  printf( '' Entrez deux entiers:' ' );
  pcanf( '' %d' ' ,&a); Scanf( '' %d' ' ,&b);
 switch(operateur){
        case ' -' : y=a-b; printf( '' %d' ' ,y);break;
        case ' +' : y=a+b; printf( '' %d' ' ,y); break;
        case ' *' : y=a*b; printf( '' %d' ' ,y); break;
        case ' /' : y=a/b; printf( '' %d' ' ,y); break;
        default : printf( '' opérateur inconnu\n' ' );break;
}
}
```