Ministère de l'enseignement supérieur et de la recherche scientifique
Université Abderrahmane MIRA de Bejaia

Faculté de Technologie
Département de Technologie



Structure des ordinateurs et applications

Présentation

✓ Dr. DJAFRI Ghani
✓ ghani.djafri@univ-bejaia.dz



À propos du cours

- **Crédits**: 02
- **Coefficients**: 02

Les expressions arithmétiques, relationnelles et logiques

Les expressions arithmétiques, relationnelles et logiques

Expressions arithmétiques

C'est une combinaison d'opérande(s) (les éléments sur lesquels portent les calculs: valeur, constante, variable) et d'opération(s) arithmétiques. Exemple : **b := (a * 2) - 4;**

Les opérations arithmétiques

Opération	Opérateur
Addition	+
Soustraction	-
Multiplication	*
Division	/
Modulo (reste de la division entière)	%

Expressions relationnelles

Une expression relationnelle est une expression qui permet de comparer deux valeurs à l'aide d'un opérateur de comparaison. Son résultat est une valeur logique : Vrai ou Faux.

Les opérateurs relationnels

Signification	Opérateur
Egal à	==
Strictement inférieur à	<
Inférieur ou égal à	<=
Strictement supérieur à	>
Supérieur ou égal à	>=
Différent de	!=

Remarque importante : • = est l'opérateur d'affectation : a = 5; • == est l'opérateur de comparaison : a == 5

Expressions logiques (booléennes)

C'est une combinaison de variables de type booléen (True : Vrai ou False : Faux) et d'opérateurs booléens (Non, Et, Ou).

L'opérateur ET logique (&&)

est utilisé pour vérifier si deux conditions sont vraies en même temps. . Il donne :

✓ vrai seulement si les deux expressions sont vraies.

✓ False dans tous les autres cas.

A	B	A && B
False	False	False
False	True	False
True	False	False
True	True	True

Expressions logiques (booléennes)

❑L'opérateur OU logique (| |)

L'opérateur OU est utilisé pour vérifier si au moins une des deux conditions est vraie. Il donne :

✓ vrai si au moins une des deux conditions est vraie.

✓ False seulement si les deux sont fausses.

A	B	A || B
False	False	False
False	True	True
True	False	True
True	True	True

Expressions logiques (booléennes)

❑L'opérateur NON logique (!)

L'opérateur NON sert à inverser une valeur logique (booléenne).

✓ Si la condition est vraie, alors non la rend Fausse;

✓ Si la condition est Fausse, alors non la rend vraie.

A	! A
False	True
True	False

Les expressions arithmétiques, relationnelles et logiques

✞ Les fonctions

Liste de fonctions standards (ou prédéfinies) applicables sur des entiers ou des réels.

Fonction	Bibliothèque	Description	Exemple
abs(x)	<stdlib.h>	Valeur absolue d'un entier	abs(-5) ⟶ 5
sqrt(x)	<math.h>	Racine carrée	sqrt(25) ⟶ 5
pow(x,y)	<math.h>	Puissance x^y	pow(2,3) ⟶ 8
sin(x)	<math.h>	Sinus de x (en radians)	sin(x)
cos(x)	<math.h>	Cosinus de x (en radians)	cos(x)
tan(x)	<math.h>	Tangente de x (en radians)	tan(x)
exp(x)	<math.h>	Exponentielle e^x	exp(1)
log(x)	<math.h>	Logarithme népérien	log(x)
log10(x)	<math.h>	Logarithme décimal	log10(100) ⟶ 2
ceil(x)	<math.h>	Plus petit entier supérieur ou égal à x	ceil(3.2) ⟶ 4
floor(x)	<math.h>	Plus grand entier inférieur ou égal à x	floor(3.8) ⟶ 3
fabs(x)	<math.h>	Valeur absolue d'un réel	fabs(-3.5) ⟶ 3.5
fmod(x,y)	<math.h>	Reste de la division de deux réels	fmod(7.5,2) ⟶ 1.5

Remarque : les fonctions de la bibliothèque <math.h> nécessitent généralement l'inclusion de cette bibliothèque au début du programme :

La priorité dans les expressions en langage C

Lorsqu'une expression contient plusieurs opérateurs, le langage C respecte un **ordre de priorité** pour déterminer dans quel ordre les opérations doivent être effectuées.

Priorité	Opérateurs	Description	Associativité
1	`() ; fonctions`	Parenthèses ; fonctions	Gauche ⟶ droite
2	`!`	NON logique	Droite ⟶ gauche
3	`* / %`	Multiplication, division, modulo	Gauche ⟶ droite
4	`+ −`	Addition, soustraction	Gauche ⟶ droite
5	`< <= > >=`	Opérateurs relationnels	Gauche ⟶ droite
6	`== !=`	Égalité, différence	Gauche ⟶ droite
7	`&&`	ET logique	Gauche ⟶ droite
8	`||`	OU logique	Gauche ⟶ droite
9	`= += −= *= /= %=`	Affectation	Droite ⟶ gauche

Évaluation des expressions

L'évaluation d'une expression consiste à calculer, au fur et à mesure, les résultats des calculs jusqu'à obtenir un résultat finale. Cela se fait en plusieurs étapes :

- Écrire l'expression sous forme linéaire (Il faut noter qu'en algorithmique, les expressions s'écrivent sous forme linéaire:
- $\frac{(x+z)}{(y*2)} \rightarrow (x + z)/(y * 2)$;
- Remplacer les identifiants (c'est à dire les noms) des variables et des constantes par leurs valeurs ;
- Évaluer (Calculer) étape par étape chacune des sous-expressions en commençant par les sous-expressions qui sont dans les parenthèses les plus internes.
- Indiquer à chaque calcul, le rang d'évaluation.

Exercice N°01 : (Evaluation des expressions)

Evaluer les expressions suivantes en respectant l'ordre de priorité des opérateurs :

- Expression 1 : $E1 := \left(\frac{70}{5} + 3 * 2\right) \% (4 + 1) - 6;$
- Expression 2: $E2 := \left(\frac{x}{y}\right) + \left(\frac{w*z+6}{3}\right) + 2z - \frac{v}{u}$; $avec$: $x = 8, y = 4, z = 6, w = 3, v = 10, u = 5$;
- Expression 3: $E3 := ((a + b) \% 3 = 0)$ && $((c - a) < b)$ $(!(a >= c))$, avec : a = 6 ; b = 3 ; c = 10.

- $E5 = ((a + b) - c)\%3 == 0 \&\& c * a - b < c \,||\, !(a >= c)$ $avec\ a = 6\ ; b = 10\ ; c = 4.$
- *E6 = (2 * x + 3 * y) /x + 6 * y - (5 % y * 2 * x) avec x=2, y=4.*

Exercice N°02 *: (Expressions arithmétiques en Algorithme/langage C) a) Traduire les expressions suivantes en langage C:*

$\boldsymbol{y1}=b^2-4\text{ac};\ \boldsymbol{y2}=2yx2\ ;\boldsymbol{y3}=\frac{-b-\sqrt{d}}{2a};\boldsymbol{y4}=e^{3a}+|b|;$

$\boldsymbol{y5}=\sqrt{|\text{a}-\text{b}|}+|\ c+d|\ ;\boldsymbol{y6}=\sqrt{\frac{x\times y}{z\times w}-\frac{|u-v|}{t}}\ ;$

$\boldsymbol{y7}=-4e^{2b}+ba^2-b\left(\frac{a}{b}\right)(b^2).\ \ \boldsymbol{y8}=\frac{a^2+\sqrt{e^a+c}}{3\sqrt{b}}\ ;$

$\boldsymbol{Y9}=\frac{-\sqrt{a}+e^{3b}+\sqrt{a^2+ab}}{2a+|b|}$

b*) Donner l'expression arithmétique correspondante à l'expression suivante écrite en langage C :*

Z0

$= Sqrt(abs(2 * x - 1 + y/2)) / (Sqr(x) - 2 * x * y) + 2 * Sqrt(x + y);$

Z1=(Sqrt(Abs(2 * x - 3)) / (x + 1)) + Sqrt(3 * x + 1) ;

Z2=Sqrt(abs(x - y + z/2)) / (x * y - z) + abs(x - z) / Sqrt(y);