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

À propos du cours

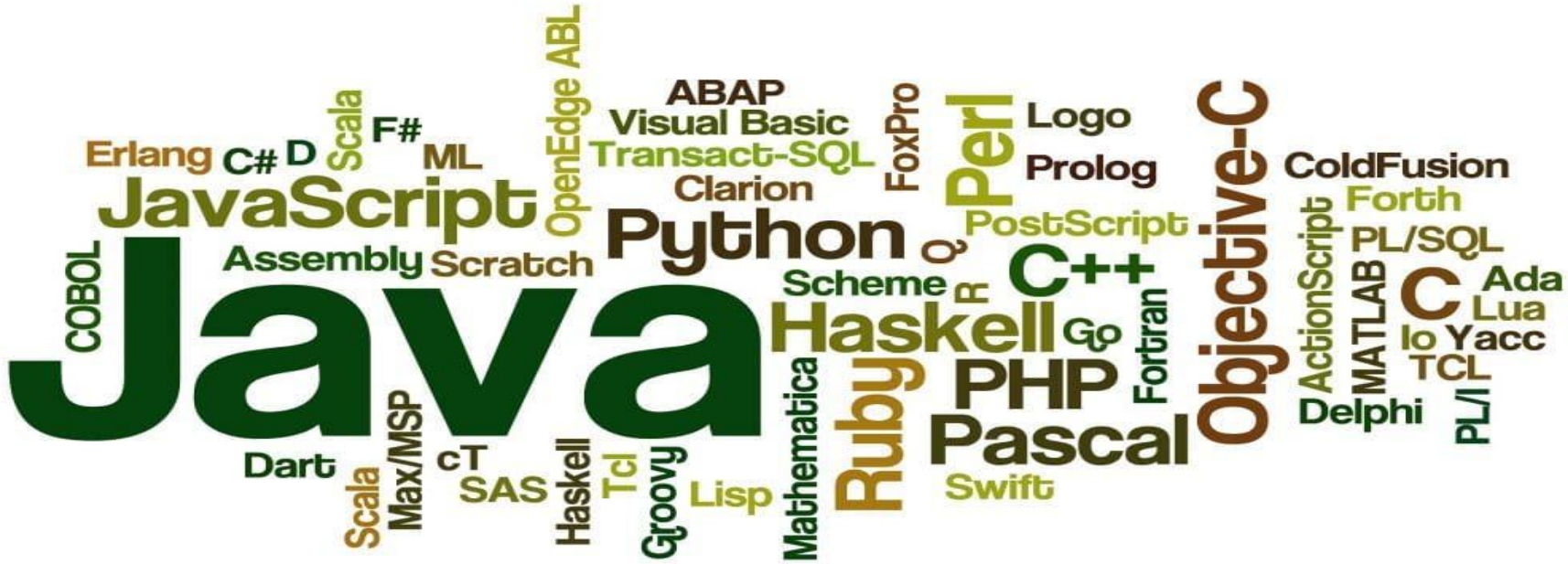
- **Crédits**: 02
- **Coefficients**:02

Partie logiciel (Software)

Langage de programmation

Un langage de programmation sert à écrire des instructions que l'ordinateur peut exécuter pour accomplir des tâches. Il existe plusieurs langages de programmation, comme C/C++, Pascal, Java, Fortran, Matlab, etc.

Exemples



Langages de Programmation

Partie logiciel (Software)

Langage de programmation C

C: est un langage de programmation créé dans les années 1970 par Dennis Ritchie. Il est utilisé pour écrire des programmes qui s'exécutent directement sur l'ordinateur.

Pour exécuter un programme en C, on utilise des logiciels appelés IDE (Environnement de Développement Intégré). Ces IDE permettent d'écrire, compiler et exécuter le code.

Exemples :

- CodeBlocks;
- Dev-C++;
- Visual Studio / Visual Studio Code;
- Eclipse avec plugin C/C++.

Structure d'un algorithme

Un algorithme est généralement présenté sous une forme structurée afin d'être clair, lisible et compréhensible. On adopte une organisation standard composée de trois parties principales :

- **En-tête :**Contient l'identificateur de l'algorithme, c'est un nom qui identifie l'algorithme;
- **Déclaration :**Partie où l'on définit les variables, leurs types et éventuellement les constantes;
- **Corps (ou bloc d'instructions) :**Partie principale qui décrit les instructions à exécuter étape par étape, elle est délimitée par les mots-clés **Début** et **Fin.**

L'algorithme	En C

Algorithme id_Algorithme ; // En-tête
Variables <Déclaration des variables> ;
Constantes < Déclaration des constantes> ; //Déclaration
Début
<instruction_1> ;
.
. // Corps de l'algorithme
.
<instruction_N> ;
Fin.
	#include<stdio.h>
	int main( ) {
	// Déclaration
	//instructions
	return 0;
	}

Structure d'un algorithme

Expressions

Une expression est une combinaison de constantes, de variables et d'opérateurs (arithmétiques, relationnels, logiques, etc.) qui peut être évaluée pour produire une valeur.

Expressions arithmétiques

Une expression arithmétique est constituée d'opérandes numériques (valeurs, variables, constantes) reliés par des opérateurs arithmétiques.

Opération	Algorithmique	En C
Addition	+	+
Soustraction	-	-
Multiplication	*	*
Division réelle	/	/
Division entière	div	/
Modulo	mod	%

Expressions

Expressions relationnelles

Une expression relationnelle utilise des opérateurs relationnels (ou de comparaison).
Le résultat est booléen (vrai ou faux)

Opération	Algorithmique	En C
Égalité	=	==
Différence	≠	!=
Supériorité stricte	>	>
Infériorité stricte	<	<
Supérieur ou égal	>=	>=
Inférieur ou égal	<=	<=

Expressions

Expressions logiques (booléennes)

Une expression logique est une combinaison de variables de type booléen et d’opérateurs booléens. Elle peut combiner des expressions relationnelle. Le résultat est booléen.

Opération	Algorithmique	En C
Négation	Non	!
Conjonction	Et	&&
Disjonction	Ou	||

Exemples de fonctions courantes en algorithmique et en langage C

Fonction (algorithmique)	Rôle/ Description	Équivalent en langage C	Bibliothèque C nécessaire	Exemple d'utilisation en C
abs(x)	Donne la valeur absolue de x	abs(x)/ fabs(x)	<stdlib.h> / <math.h>	val = abs(-5); // 5
racine(x)	Calcule la racine carrée de x	sqrt(x)	<math.h>	r = sqrt(9); // 3
puiss(x, y)	Calcule x à la puissance y	pow(x, y)	<math.h>	p = pow(2, 3); // 8

ent(x)	Donne la partie entière de x	floor(x) / ceil(x)	<math.h>	e = floor(3.7); // 3
				
				e = ceil(3.7); // 4

arrondi(x)	Arrondit x à l'entier le plus proche	round(x)	<math.h>	a = round(3.6); // 4
alea(n)	Génère un nombre aléatoire inférieur à n	rand() % n	<stdlib.h>	x = rand() % 10;

Types de variable

Les types de variable sont les catégories fondamentales de données utilisées en algorithmique (programmation). Voici les principaux types avec leur représentation en mémoire :

- **Entier (int, long)** : Représente l'ensemble Z des entiers. Exemple : -3, 0, 20075 .
- **Réel (float, double)** : Représente l'ensemble R des réels. Exemple : 3.14, -0.5, 123.789.
- **Caractère (char)** : Peut être tous les symboles que nous trouvons sur le clavier d'un ordinateur : espace, lettres, chiffres, signes de ponctuation et caractères spéciaux (#, @, &, etc.).
- **Chaîne de caractères (char id_variable[n], n étant le nombre de caractères)** : suite de caractères, stockée en mémoire sous forme de tableau de charactères.
- **Booléen (bool)** : Une donnée de type booléen peut avoir deux valeur « vrai » ou « faux » (true, false). Stocké sur 1 bit théoriquement (Le bit (abréviation de *binary digit*) est la plus petite unité d'information utilisée en informatique. Il ne peut prendre que deux valeurs 0 ou 1, un octet ou Byte (en anglais) est une suite de 8 bits).

```
En C, les mots-clés comme int float char _Bool if else
for while do return main scanf printf sont sensibles à
la casse; il faut toujours les écrire en minuscules.
```

Notion de variable et de constante

Variable

Une variable est une zone mémoire identifiée par un nom (identificateur) et contenant une valeur modifiable durant l'exécution du programme.

Déclaration des variables

En Algorithme

```
Variables id_variable : type ;
```

Exemple :

Variables a,b,c : entier;
S:réel;
nom : chaîne de caractères [20] ;
lettre : Caractère ;
test : Booleen ;

En C :

```
type id_variable;
```

Exemple :

```
int a,b,c;
float s;
char nom[20];
char lettre;
bool teste;
```

Les variables de même type peuvent être déclarées ensemble

Notion de variable et de constante

Constante

Une constante est une valeur fixe qui ne peut pas être modifiée après sa déclaration.

Déclaration des constantes

En Algorithme

```
Constantes id_constante = valeur ;
```

Exemple :

Constantes PI = 3.1415 ;

En C :

```
const type nom = valeur;
```

Exemple :

const float PI = 3.1415 ;

En C, chaque déclaration de constante doit être précédée par le mot-clé const sauf si les constantes ont le même type. Exemple const int a=15 , b=10 , c = 20 ;

Les instructions de base sont les éléments essentiels pour écrire un algorithme.Elles servent à manipuler les données, effectuer des calculs et échanger des informations avec l'utilisateur. On distingue principalement trois instructions :la lecture, l'écriture et l'affectation.

Instruction de lecture

La lecture est une instruction qui permet de saisir des données au clavier et de les stocker dans des variables pour être utilisées dans l'algorithme. Seules les variables peuvent être lues .

Syntaxe :

Lire (id_variable) ;

Instruction de lecture en c

Pour indiquer le type de la variable

scanf("spécificateur de format", & id_variable);

Pour indiquer l'adresse de la variable

Exemples :

scanf("%d", & id_variable); // pour un entier ou un booléen, on peut utiliser aussi %i
scanf("%f", & id_variable); // pour un réel
scanf("%c", & id_variable); // pour un caractère
scanf("%s", id_variable); // pour une chaîne de caractères

Remarque : La lecture d'une chaîne de caractères se fait sans indiquer son adresse (sans &).

Instruction d'écriture

L'écriture est une instruction qui permet d'afficher des messages ou des résultats à l'écran.

Syntaxe :

Ecrire ("Message à afficher") ; // affiche un message
Ecrire (id_variable) ; // affiche la valeur de la variable
Ecrire (expression) ; // affiche la valeur de l'expression
Ecrire (valeur) ; // affiche la valeur

En C :

```
printf("Message");
printf("spécificateur de format", id_variable);
printf("spécificateur de format", expression );
printf("spécificateur de format", valeur);
```

Instruction d'écriture

Exemples **:**

Ecrire ("Le résultat est : ", R) ;

En C :

printf("Le résultat est : %d\n", id_variable);

Algorithme LectureAffichage ;
Variables x : entier ;
Début
Écrire("Donner un entier :") ;
Lire(x) ;
Écrire("La valeur de x est :", x) ;
Fin.

En C :

```
#include <stdio.h>
int main() {
int x;
printf("Donner un entier : ");
scanf("%d", &x);
printf("La valeur de x est : %d\n", x);
return 0;}
```

Instruction d'affectation

L'affectation est une instruction qui permet de donner une valeur à une variable. Cette valeur peut être direct, celle d'une variable ou le résultat d'une expression.

Syntaxe :
id_variable ← valeur ; id_variable ← id_variable; id_variable ← expression ;
En C :
id_variable = valeur ; id_variable = id_variable; id_variable = expression ;

Remarques :
- Lorsque l'on donne une valeur pour la première fois à une variable en utilisant une affectation, cette opération s'appelle **l'initialisation**.
- Lorsque l'on augmente la valeur d'une variable numérique d'une certaine quantité, le plus souvent de 1, cette opération s'appelle **l'incrémentation**.

← =

Instruction d'affectation

Exemple : Calcul de la moyenne d'une matière

Algorithme Moyenne ;
Variables TD, Ex, moy : réel ;
Début
Écrire (" Donner la note de TD :") ;
Lire(TD) ;
Écrire (" Donner la note de l'examen :") ;
Lire(Ex) ;
moy ← (TD * 2 + Ex *3) / 5 ;
Écrire (" Moyenne = ", moy) ;
Fin.

En C :

```
#include <stdio.h>
int main() {
float TD, Ex, moy;
printf("Donner la note de TD : ");
scanf("%f", &TD);
printf("Donner la note de l'examen
: ");
scanf("%f", &Ex);
moy = (TD * 2 + Ex *3) / 5;
printf(" Moyenne = %.2f\n ", moy);
return 0;
}
```

Exercice 01 :

Écrire un programme en langage **C** qui demande à l'utilisateur de saisir **deux nombres**.

Le programme doit calculer et afficher :

•la **somme** ;le **produit** ;la **moyenne** des deux nombres.

Exercice 01 :

Écrire un programme en langage **C** qui demande à l'utilisateur de saisir **un nombre entier strictement positif (x > 0)**.

Le programme doit calculer et afficher :

•le **double** du nombre ;

•le **carré** du nombre ;

•la **valeur absolue** du nombre ;

•la **racine carrée** du nombre.