

Ministère de l'Enseignement Supérieur et de
la Recherche Scientifique
Université Ziane Achour
Djelfa
Faculté des Sciences et de la
Technologie



وزارة التعليم العالي والبحث العلمي
جامعة زيان عاشور
الجلفة
كلية العلوم والتكنولوجيا

Département de Génie Électrique

Polycopie de Cours

PROGRAMMATION EN C++

Dr. MOHAMMEDI Ridha Djamel

r.mohammedi@univ-djelfa.dz

Avant-propos

Le langage de programmation C++ est actuellement l'un des plus largement utilisés dans le monde, que ce soit pour les applications scientifiques ou le développement de logiciels. Cette polycopie est destinée aux étudiants d'automatique poursuivant leur licence L3. Il est élaboré en conformité avec le programme officiel de la formation.

Ce cours sert d'introduction à la maîtrise de la programmation en langage C++. L'objectif principal est de guider les étudiants à travers les concepts fondamentaux et les exemples pratiques de programmes, leur permettant ainsi de comprendre les subtilités de ce langage.

R.D. Mohammedi

Semestre: 5

Unité d'enseignement: UEF 3.1.2 Matière

2: Programmation en C++

VHS: 22h30 (Cours: 1h30)

Crédits: 2 Coefficient: 1

Objectifs de l'enseignement :

Ce cours permettra à l'étudiant de se familiariser avec les langages de programmation et en particulier le langage C++.

Connaissances préalables recommandées :

Bases en mathématiques, Notions d'algorithmique, Méthodes numériques, Logique binaire.

Contenu de la matière :

Chapitre 1. Présentation du langage C+ (1 Semaine)

Historique, Environnement de développement en C++ (création d'objets, compilation, débogage, exécution ...).

Chapitre 2. Syntaxe élémentaire en langage C+ (1 Semaine)

Instructions Commentaires, Mots clés et mots réservés- Constantes et variables, Types fondamentaux Opérateurs (unitaires, binaires, priorité...).

Chapitre 3. Structures conditionnelles et Boucles (2 Semaines)

If/else, Switch/case, Boucle for, Boucle while, Boucle do/while.

Chapitre 4. Entrées/sorties (2 Semaines)

Flux de sortie pour affichage, Flux d'entrée clavier, Cas des chaînes de caractères, les fichiers.

Chapitre 5. Pointeurs et Tableaux (2 Semaines)

Pointeurs, Références, Tableaux statiques, Tableaux et pointeurs, Tableaux dynamiques, Tableaux multidimensionnels.

Chapitre 6. Fonctions (2 Semaines)

Prototype d'une fonction, Définition d'une fonction, Appel d'une fonction, Passage d'arguments à une fonction, Surchage d'une fonction, Fichiers.

Chapitre 7. Programmation orientée objet en C++ (5 Semaines)

Introduction, Concept de classes et objets, Héritage, Méthodes particulières (constructeurs, destructeurs...), Programmation procédurale ou structurée, Programmation par objets.

Mode d'évaluation : Examen : 100%.

Références bibliographiques :

1. Bjarne Stroustrup, Marie-Cécile Baland, Emmanuelle Burr, Christine Eberhardt, « Programmation: Principes et pratique avec C++ », Edition Pearson, 2012.
2. Jean-Cédric Chappelier, Florian Seydoux, « C++ par la pratique. Recueil d'exercices corrigés et aide-mémoire », PPUR Édition : 3e édition, 2012.
3. Jean-Michel Léry, Frédéric Jacquenot, « Algorithmique, applications aux langages C, C++ en Java », Edition Pearson, 2013.
4. Frédéric DROUILLON, « Du C au C++ - De la programmation procédurale à l'objet », Eni; Édition : 2e édition, 2014.
5. Claude Delannoy, « Programmer en langage C++ », Edition Eyrolles, 2000.

Intitulé de la Licence: Automatique

Sommaire

Chapitre 1: Présentation du C++.....	9
1.1 Historique	9
1.2 Terminologie.....	10
1.3 Outils de programmation en C++	11
1.4 Dev C++	11
Chapitre 2: Syntaxe élémentaire en langage C++	16
2.1 Types de données	16
2.2 Déclaration de variables	16
2.3 Mots réservés	17
2.4 Constantes.....	18
2.5 Instructions d'entrée et de sortie.....	18
2.6 Opérateurs dans le langage C++	18
2.6.1 Opérateur d'affectation.....	18
2.6.2 Opérateurs arithmétiques.....	19
2.6.3 Opérateurs d'affectation composée	19
2.6.4 Opérateurs relationnels.....	19
2.6.5 Opérateurs logiques.....	20
2.6.6 Opérateurs binaires	20
2.6.7 Opérateurs d'incrément et de décrémentation	22
2.6.8 Ordre de priorité des opérations.....	23
2.6.9 L'opérateur sizeof	24
2.6.10 La bibliothèque cmath.....	24
Chapitre 3: Structures conditionnelles et Boucles.....	27
3.1 Introduction.....	27
3.2 Les Conditions	27

3.3	Les Boucles.....	35
Chapitre 4:	Entrées/Sorties.....	48
4.1	Introduction.....	48
4.2	Flux de Sortie pour Affichage.....	48
4.3	Flux d'Entrée Clavier.....	50
4.4	Gestion des Chaînes de Caractères.....	50
4.5	Gestion des Fichiers.....	52
Chapitre 5:	Pointeurs et Tableaux.....	54
5.1	Les tableaux.....	54
Chapitre 6:	Fonctions.....	64
6.1	Introduction.....	64
6.2	Fonctions retournant une valeur.....	64
6.3	Fonctions void.....	67
Chapitre 7:	Programmation orientée objet en C++.....	69
7.1	Introduction.....	69
7.2	Notions de classe et d'objet.....	69
7.2.1	Principes de la programmation orientée objet.....	71
TP N° 01	Introduction à la programmation en C++.....	79
TP N° 02	: Les conditions en C++.....	84
TP N° 03	: Les boucles en C++.....	87
TP N° 04	: Les Tableaux (vecteurs et matrices) en C++.....	93
TP N° 05	: Les Fonctions en C++.....	97
Annexe	103
Bibliographie	106

Chapitre 1 :

Présentation du C++

Chapitre 1: Présentation du C++

1.1 Historique

L'histoire de C++ remonte aux années 1970, lorsque Bjarne Stroustrup, un chercheur en informatique danois, a commencé à travailler sur ce langage de programmation. L'objectif principal de Stroustrup était de créer un langage qui permettrait la programmation orientée objet tout en conservant la compatibilité avec le langage C existant.



Bjarne Stroustrup

C++ a été officiellement publié pour la première fois en 1983 en tant que "C avec des classes". Il a rapidement gagné en popularité parmi les programmeurs en raison de sa puissance et de sa flexibilité. Le langage a évolué au fil des ans avec l'ajout de nombreuses fonctionnalités, y compris les templates, les exceptions, et la gestion automatique de la mémoire grâce à l'utilisation du mot-clé "new".

Dans les années 1990, C++ était devenu le langage de prédilection pour le développement de logiciels système, de jeux vidéo et d'applications graphiques en raison de ses performances élevées et de sa capacité à gérer efficacement les ressources matérielles. Il est également devenu largement utilisé dans le domaine financier pour la création de systèmes de trading haute fréquence.

Au fil du temps, C++ a continué à évoluer avec l'introduction de nouvelles normes, telles que le standard C++98, C++11, C++14, C++17 et C++20. Ces normes ont apporté de nouvelles fonctionnalités au langage, améliorant sa lisibilité, sa sécurité et sa convivialité pour les développeurs.

Aujourd'hui, C++ reste un langage de programmation important dans l'industrie informatique. Il est largement utilisé pour développer des systèmes d'exploitation, des logiciels embarqués, des applications de bureau, des jeux vidéo et bien d'autres applications. Grâce à sa longue histoire et à sa communauté active de développeurs, C++ continue d'évoluer et de prospérer dans le monde de la programmation informatique.

1.2 Terminologie

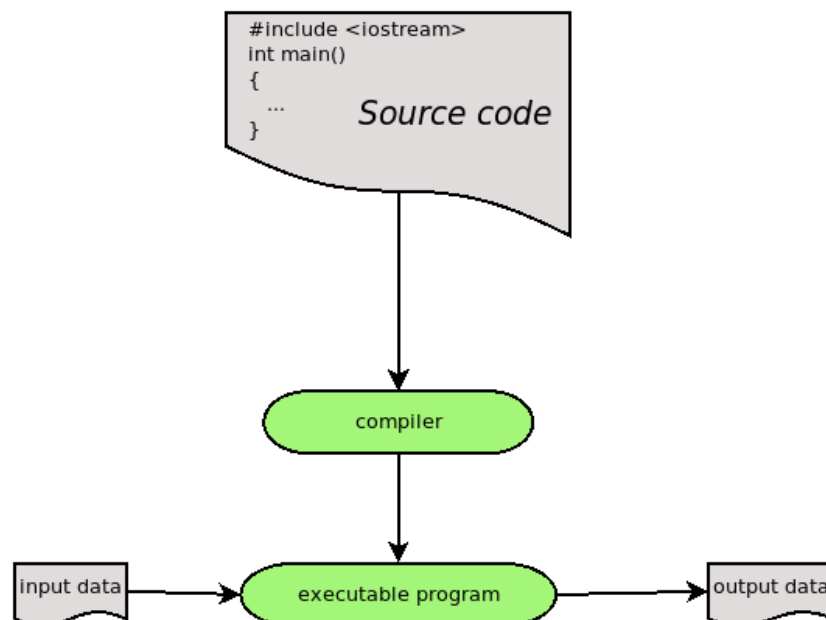
Compilateur : Un outil essentiel en programmation. Il transforme le code source écrit dans un langage de programmation (comme C ou Java) en langage machine, compréhensible par l'ordinateur. Ce processus inclut la vérification de la syntaxe et la génération de code exécutable.

Langage machine : Langage bas niveau composé de binaires (0 et 1). Il est directement exécuté par le processeur d'un ordinateur. Chaque type de processeur a son propre langage machine.

Éditeur de code : Un programme où les développeurs écrivent leur code. Il peut offrir des fonctionnalités comme la coloration syntaxique et la suggestion de code pour faciliter l'écriture et la compréhension du code.





IDE (Environnement de Développement Intégré) : Combinant un éditeur de code avec d'autres outils de développement, comme un débogueur et un compilateur, pour offrir un environnement complet pour la programmation.

Le débogueur : est un outil qui permet de trouver et corriger les bugs, ou erreurs, dans leur code.



1.3 Outils de programmation en C++

Pour programmer en C++, il existe plusieurs IDE (Environnements de Développement Intégrés) populaires, chacun avec ses propres caractéristiques et avantages :

<p>1. Visual Studio:</p> <ul style="list-style-type: none"> • Développé par Microsoft. • Puissant pour le développement Windows. • Excellents outils de débogage et d'analyse. • Prise en charge de nombreux plugins et outils tiers. 	 www.microsoft.com
<p>2. Eclipse CDT (C/C++ Development Tooling):</p> <ul style="list-style-type: none"> • Open source et multiplateforme. • Extensible avec une grande variété de plugins. • Bonnes fonctionnalités pour le développement C/C++. 	 www.eclipse.org
<p>3. Code::Blocks :</p> <ul style="list-style-type: none"> • Gratuit et open source. • Léger et rapide, adapté pour les débutants. • Supporte plusieurs compilateurs comme GCC, Clang. 	 www.codeblocks.org
<p>4. Dev-C++ :</p> <ul style="list-style-type: none"> • Un IDE plus simple et léger. • Bon pour les débutants et les petits projets. • Intègre le compilateur MinGW. 	 www.bloodshed.net

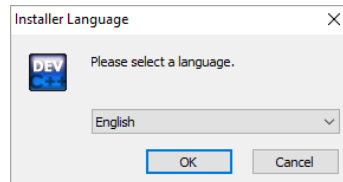
1.4 Dev C++

a) Présentation :

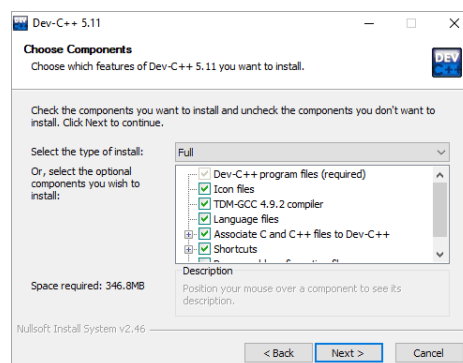
Dev C++ est un environnement de développement intégré (IDE) gratuit pour le langage de programmation C et C++. Il est conçu pour être simple et facile à utiliser, ce qui le rend idéal pour les débutants. Ce logiciel offre diverses fonctionnalités, telles que la coloration syntaxique, le débogage, et la prise en charge de plusieurs langues. Il intègre également un compilateur GCC, permettant ainsi une compilation efficace des programmes. Dev C++ est compatible avec plusieurs systèmes d'exploitation, notamment Windows. Cet IDE est souvent recommandé pour ceux qui débutent en programmation C ou C++, grâce à son interface utilisateur intuitive et ses nombreuses options de personnalisation.

b) Etapes d'installation :

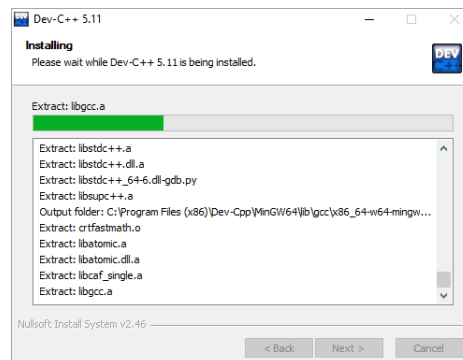
- 1- La première étape c'est de télécharger le logiciel depuis le site officiel¹.
- 2- Sélectionner la langue de notre choix, comme illustré dans la capture d'écran ci-dessous.



- 3- Ensuite, il nous est demandé de sélectionner les composants que nous devons installer dans le cadre de l'installation dev-C++.

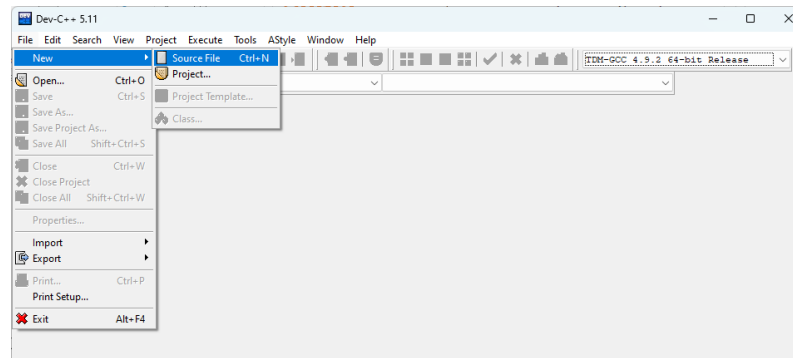


- 4- La capture d'écran suivante montre la progression de l'installation.

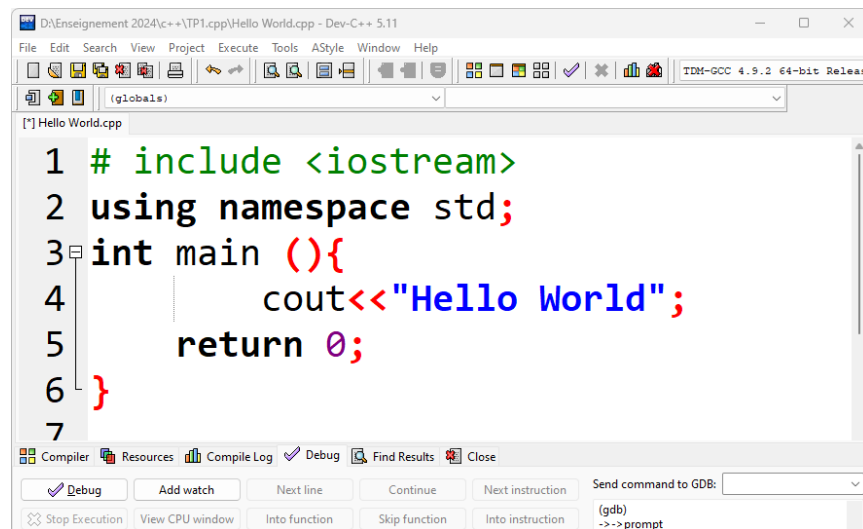


- 5- Pour créer un Nouveau Fichier Source, allez dans "File" > "New" > "Source File" (Fichier > Nouveau > Fichier source).

¹ Site officiel de Dev C++ <https://sourceforge.net/projects/orwelldevcpp/>

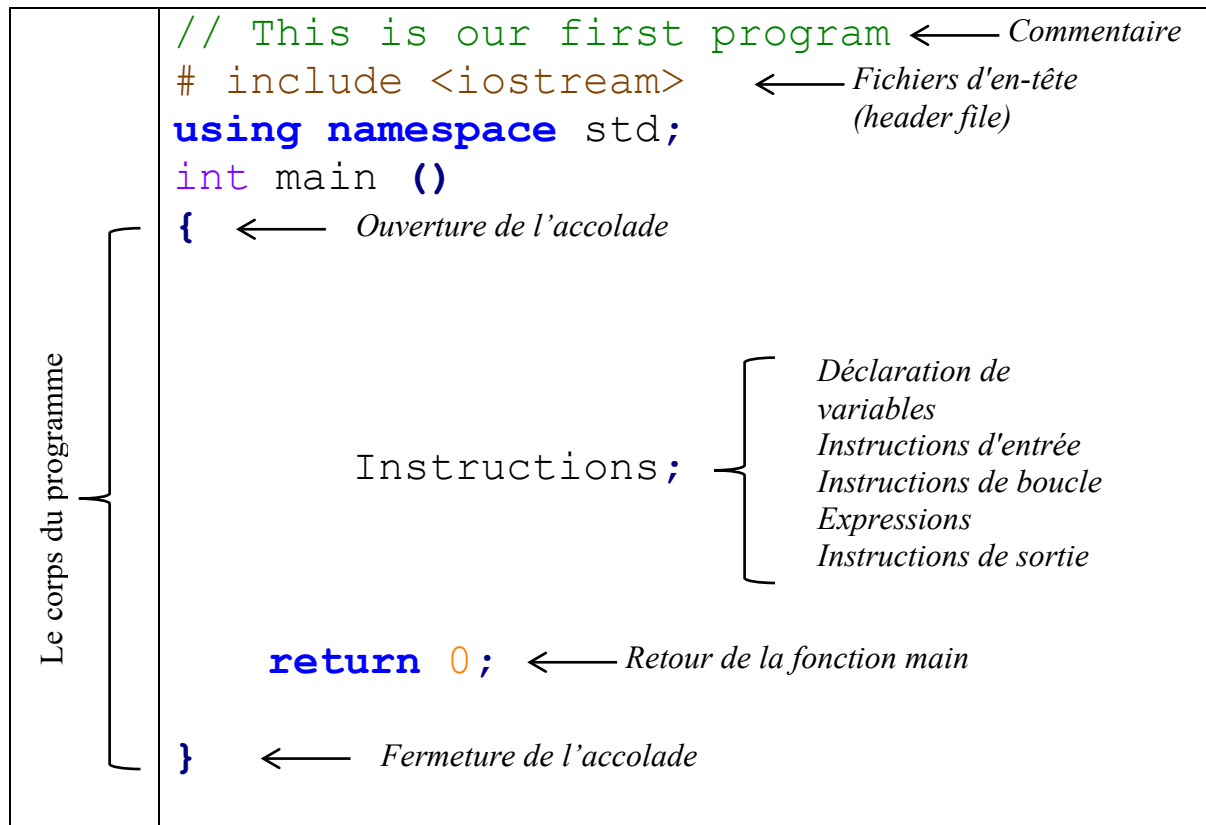


6- Dans l'éditeur de texte, écrivez votre code C++.



- 7- Pour enregistrer le fichier, cliquez sur "File" > "Save" (Fichier > Enregistrer), ou utilisez le raccourci clavier Ctrl+S.
- 8- Pour exécuter le programme, cliquez sur "Execute" > "Run" (Exécuter > Exécuter), ou utilisez le raccourci F10. Si des erreurs de compilation apparaissent, elles seront affichées dans le bas de la fenêtre. Corrigez ces erreurs avant de réessayer.

1.5 La structure d'un programme en C++



- **Fonction principale** : La `main()` est le point de départ de tout programme C++. Elle doit être présente pour que le programme s'exécute.
- **Déclarations de variables** : Ici, on déclare les variables qui seront utilisées dans le programme.
- **Corps de la fonction** : C'est dans cette partie que l'on écrit le code effectuant les opérations du programme.
- **Instructions** : Ce sont des commandes que le programme exécute, telles que les opérations mathématiques ou les appels de fonction.
- **Commentaires** : Ils n'affectent pas l'exécution du programme mais servent à expliquer le code. En C++, ils commencent par `//` pour une ligne ou `/*` et `*/` pour un bloc.
- **Fonctions supplémentaires** : Outre la `main()`, d'autres fonctions peuvent être définies pour organiser et structurer le code.
- **Retour de la fonction main** : En général, la fonction `main()` renvoie un entier. `return 0;` indique que le programme s'est exécuté avec succès.

Chapitre 2 :

Syntaxe élémentaire en langage

C++

Chapitre 2: Syntaxe élémentaire en langage C++

2.1 Types de données

Les types de données de base en C++ Les types de données de base disponibles dans le langage C++ sont indiqués dans le tableau suivant :

Type de Donnée	Plage	Taille	Utilisation
<code>int</code>	-32768 à 32767	2 octets (16 bits)	Pour stocker des entiers sans décimale.
<code>long</code>	-2147483648 à 2147483647	4 octets (32 bits)	Pour stocker des entiers sans décimale.
<code>char</code>	0 à 255	1 octet (8 bits)	Pour stocker des caractères.
<code>float</code>	$\pm 3.4 \times 10^{-38}$ à $\pm 3.4 \times 10^{38}$	4 octets (32 bits)	Pour stocker des nombres à virgule flottante. Il a une précision de 7 chiffres significatifs.
<code>double</code>	$\pm 5.0 \times 10^{-324}$ à $\pm 1.7 \times 10^{380}$	8 octets (64 bits)	Pour stocker des nombres à virgule flottante à double précision. Il a une précision de 17 chiffres significatifs.

2.2 Déclaration de variables

Une variable peut être déclarée de la manière suivante :

```
type nom_variable;
type nom_variable = valeur_initiale;
type nom_variable(valeur);
```

Type est le type de donnée tel que `int`, `float` ou `double`, `char`, etc.

C++ autorise des noms de variables ou d'identifiants longs et descriptifs. Les règles pour former un nom de variable s'appliquent également aux noms de fonctions. Les règles sont:

1. Le premier caractère doit être une lettre, minuscule ou majuscule ;
2. La casse est significative, les lettres majuscules et minuscules sont différentes ;
3. Les noms de variables sont composés de lettres minuscules, de chiffres et du caractère de soulignement ;
4. Les constantes définies sont traditionnellement constituées de caractères majuscules ;

5. Le nombre de caractères autorisés dans un nom de variable dépend du compilateur, mais la variable doit être unique dans les huit premiers caractères pour être sûre à travers différents compilateurs ;
6. Rendez les noms de variables descriptifs ;
7. Ne nommez pas une variable avec le même nom qu'un mot réservé.

2.3 Mots réservés

Les mots réservés pour le langage C++ sont :

volatile	double	int	struct	break	else	long	switch
register	typedef	for	extern	union	char	void	const
unsigned	return	do	sizeof	float	auto	case	static
continue	default	if	signed	short	goto	enum	while

Exemples:

<i>Déclaration</i>	<i>Explication</i>
<code>int my_Age;</code>	Déclarer une variable de type entier.
<code>long fact = 5376894;</code>	Déclarer une variable de type entier avec une valeur initiale.
<code>float moyenne;</code>	Déclarer une variable de type réel.
<code>double x (2.12356724)</code>	Déclarer une variable de type réel avec une valeur initiale.
<code>char ch;</code>	Déclarer une variable de type caractère.
<code>char ch = 'A';</code>	Déclarer une variable de type caractère avec une valeur initiale.
<code>int x1, x2, x3 = 0;</code>	Déclarer plusieurs variables en même temps (dans cet exemple seulement x3 est initialisé à 0)

2.4 Constantes

C++ permet au programmeur de définir des constantes de type décimales, hexadécimales, octales ou de chaîne de caractères. La directive `#define` peut être utilisée pour définir des constantes et elle est placée après les fichiers d'en-tête.

```
#define PI 3.14156
#define MONNOM "MOHAMED ALI"
#define LIMITE 10 #define ESC 0x1B
```

Aussi, nous pouvons utiliser `const` pour définir des constantes comme suit :

```
const int diametre = 10;
const float PI = 3.14159;
const char ch = 'a';
```

2.5 Instructions d'entrée et de sortie

La syntaxe de la déclaration d'entrée est : `cin >> nom_variable;`

Pour lire plusieurs variables : `cin >> var1 >> var2 >> var3 >> ...;`

Exemples :

```
cin >> age;
cin >> x1 >> x2 >> x3;
```

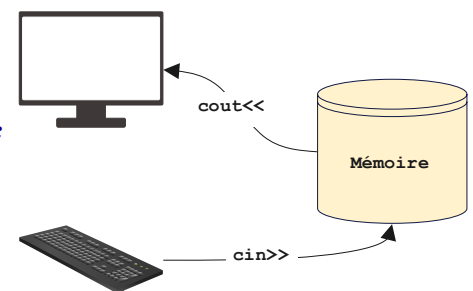
La syntaxe de la déclaration de sortie est : `cout << nom_variable;`

Pour afficher plusieurs variables : `cout << var1 << " " << var2 << " " << var3 << ... ;`

```
cout << var1 << endl << var2 << endl << var3;
```

Exemples :

```
cout << age;
cout << x1 << " " << x2 << " " << x3;
cout << x1 << endl << x2 << endl << x3;
cout << "x1=" << x1;
cout << "Bonjour mes amis";
```



2.6 Opérateurs dans le langage C++

2.6.1 Opérateur d'affectation

Affectation en C++ se réalise à l'aide de l'opérateur (`=`). Cet opérateur assigne la valeur située à sa droite à la variable située à sa gauche. Voici quelques exemples :

<i>Commande</i>	<i>Explication</i>
<code>int x = 5, y = 10, z, w;</code>	// Déclaration et initialisation
<code>z = x;</code>	// affectation
<code>w = x + y;</code>	// affectation
<code>y = (x = 5) + 2;</code>	// assigner la valeur 5 à x, puis assigner x + 2 à y
<code>a = b = c = 5;</code>	// assigner la valeur 5 à a, b et c

2.6.2 Opérateurs arithmétiques

Les opérateurs arithmétiques en C++ sont des symboles qui permettent de réaliser des calculs mathématiques de base. Voici les principaux :

<i>Opérateur</i>	<i>Signification</i>	<i>Exemple</i>	<i>Résultat</i>
<code>+</code>	Addition	<code>5 + 3</code>	<code>8</code>
<code>-</code>	Soustraction	<code>5 - 3</code>	<code>2</code>
<code>*</code>	Multiplication	<code>5 * 3</code>	<code>15</code>
<code>/</code>	Division	<code>5 / 3</code>	<code>1</code>
<code>%</code>	Modulo (reste)	<code>5 % 3</code>	<code>2</code>
<code>++</code>	Incrémentation	<code>f++</code> ou <code>++f</code> (si f vaut 5)	<code>6</code>
<code>--</code>	Décrémentation	<code>g--</code> ou <code>--g</code> (si g vaut 5)	<code>4</code>

2.6.3 Opérateurs d'affectation composée

Les opérateurs d'affectation composée en C++ combinent un opérateur arithmétique avec l'opérateur d'affectation. Ils permettent d'effectuer une opération arithmétique sur une variable et de lui réaffecter le résultat en une seule étape. Voici les principaux :

<i>Opérateur</i>	<i>Expression en C++</i>	<i>Expression équivalente</i>	<i>Explication et utilisation</i>
<code>+=</code>	<code>B += 5;</code>	<code>B = B + 5;</code>	<code>int B = 4 ; B += 5; // B = 9</code>
<code>-=</code>	<code>C -= 6;</code>	<code>C = C - 6;</code>	<code>int C = 10; C -= 6; // C = 4</code>
<code>*=</code>	<code>D *= 2;</code>	<code>D = D * 2;</code>	<code>int D = 10; D *= 2; // D = 20</code>
<code>/=</code>	<code>E /= 3;</code>	<code>E = E / 3;</code>	<code>int E = 21; E /= 3; // E = 7</code>
<code>%=</code>	<code>F %= 4;</code>	<code>F = F % 4;</code>	<code>int F = 10; F %= 4; // F = 2</code>

2.6.4 Opérateurs relationnels

Les opérateurs relationnels sont expliqués dans le tableau suivant :

Opérateur	Utilisation	Exemple	Explication
<	Inférieur à	A < B	// A est inférieur à B.
>	Supérieur à	A > B	// A est supérieur à B.
<=	Inférieur ou égal à	A <= B	// A est inférieur ou égal à B.
>=	Supérieur ou égal à	A >= B	// A est supérieur ou égal à B.
==	Égalité	A == B	// A est égal à B.
!=	Différent de	A != B	// A est différent de B.

2.6.5 Opérateurs logiques

Les opérateurs logiques sont utilisés pour combiner plusieurs conditions (déclarations logiques). Le tableau suivant décrit les opérateurs logiques en C++ :

Opérateur	Utilisation	Exemple
&& (ET logique)	La condition composée est vraie si les deux conditions sont vraies.	((a > b) && (a > c))
 (OU logique)	La déclaration composée est vraie si l'une des conditions ou les deux sont vraies.	((a > b) (a > c))
! (NON logique)	Inverse la condition.	! (a > b)

2.6.6 Opérateurs binaires

Une opération binaire consiste à convertir le nombre en binaire et à effectuer l'opération sur chaque bit individuellement. Les opérateurs binaires sont énumérés dans le tableau ci-dessous:

Opérateur	Description	Exemple de Code	Exemple Numérique	Calcul	Résultat
&	ET binaire	A & B;	5 & 3;	0101 & 0011 = 0001	1
 	OU binaire	A B;	5 3;	0101 0011 = 0111	7
^	OU exclusif binaire <i>VRAI si un seul bit est VRAI</i>	A ^ B;	5 ^ 3;	0101 ^ 0011 = 0110	6
~	Complément à un	~A;	~5;	~0101 = 1010 (Variable)	Variable
<<	Décalage binaire vers la gauche	A << 2;	5 << 2;	0101 << 2 = 10100	20
>>	Décalage binaire vers la droite	A >> 2;	5 >> 2;	0101 >> 2 = 0001	1

&=	ET binaire avec affectation	A &= B;	A = 5; A &= 3;	A = 0101 & 0011 = 0001	1
 =	OU binaire avec affectation	A = B;	A = 5; A = 3;	A = 0101 0011 = 0111	7
^=	OU exclusif binaire avec affectation	A ^= B;	A = 5; A ^= 3;	A = 0101 ^ 0011 = 0110	6
<<=	Décalage vers la gauche avec affectation	A <<= 2;	A = 5; A <<= 2;	A = 0101 << 2 = 10100	20
>>=	Décalage vers la droite avec affectation	A >>= 1;	A = 5; A >>= 1;	A = 0101 >> 1 = 0010	2

Exercice : *Quel sera les résultats de ces deux programmes suivants :*

<pre> 1 #include<iostream> 2 using namespace std; 3 int main() 4 { 5 int A = 2, B = 1, C = 3, D; 6 A <<=2; 7 B >>=2 ; 8 C <<= 1; 9 D = A^B; 10 cout<<"A = "<<A<<endl; 11 cout<<"B = "<<B<<endl; 12 cout<<"C = "<<C<<endl; 13 cout<<"D = "<<D; 14 }</pre>	<pre> 1 #include<iostream> 2 using namespace std; 3 int main() 4 { 5 int A =4, B=5 , C=2; 6 int D,E,F; 7 D = A &B; 8 E = C A; 9 F = C&3; 10 cout <<"D = "<<D<<endl; 11 cout <<"E = "<<E<<endl; 12 cout <<"F = "<<F<<endl; 13 }</pre>
--	---

Solution :

Code 1 : **A = 0010 ; B = 0001 ; C = 0011 ;**

A<<2 → Décalage binaire vers la gauche → **A = 1000** → **A = 8**

B>>2 → Décalage binaire vers la droite → **B = 0000** → **B = 0**

C<<1 → Décalage binaire vers la gauche → **C = 0110** → **C = 6**

D=A^B → OU exclusif binaire → **D = 1000 ⊕ 0000** → **D = 1000 = 8**

Code 2 : **A = 0100 ; B = 0101 ; C = 0010 ;**

D = A & B → **D = 0100** → **D=4**

E = C|A → **E = 0110** → **E=6**

F= C&3 → **F= 0010 & 0011 = 0010** → **F=2**

2.6.7 Opérateurs d'incrément et de décrétement

Ces opérateurs sont utilisés pour augmenter ou diminuer une variable d'une unité **1**. Ils peuvent être placés avant ou après le nom de la variable, comme le montre le tableau suivant.

Opérateur	Pré ou post	Description
++k	Pré-incrément	Augmenter d'abord la valeur de k d'une unité, puis évaluer l'instruction actuelle avec cette nouvelle valeur.
k++	Post-incrément	Utiliser d'abord la valeur actuelle de k pour évaluer les instructions actuelles, puis augmenter k d'une unité.
--k	Pré-décrément	Diminuer d'abord la valeur de k d'une unité, puis évaluer l'instruction.
k--	Post-décrément	Utiliser d'abord la valeur actuelle de k pour évaluer les instructions actuelles, puis diminuer k d'une unité.

Pre-incrément	Post-incrément
<pre>int a = 10, b = 11, c; c = a + ++b; //Résultat: c=10+12=22, b=12</pre>	<pre>int a = 10, b = 11, c; c = a + b++; // Résultat: c=10+11=21, b=12</pre>

Exercice 1: *Quel sera les résultats du programme suivant :*

```

1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5     int a = 6, p = 4, r=3, n =5, A, B, C, K ;
6     A = 6 * ++n ;
7     cout << "A = " << A << "\t n = " << n << endl;
8     K = 5 * a-- ;
9     cout << "K = " << K << "\t a = " << a << endl;
10    B = r++ * r++ ;
11    cout << "B = " << B << "\t r = " << r << endl;
12    C = p-- * p--;
13    cout << "C = " << C << "\t p = " << p << endl;
14 }
```

Remarque: `\t` sert à mettre un espace (tabulation) dans la sortie du programme.

Solution:

$$a = 6, p = 4, r = 3, n = 5$$

$$A = 6 * ++n \rightarrow A = 6 * 6 \rightarrow A = 36 \quad n = 6$$

$$K = 5 * a-- \rightarrow K = 5 * 6 \rightarrow K = 30 \quad a = 5$$

$$B = r++ * r++ \rightarrow B = 3 * 4 \rightarrow B = 12 \quad r = 5$$

$$C = p-- * p-- \rightarrow C = 4 * 3 \rightarrow C = 12 \quad p = 2$$

Exercice 2: Evaluer les expressions suivantes pour :

$$m = 6, n = 2, a = 0, b = 1, c = 0, d = 1, e = 0$$

$$(1) \quad a+=4+ ++m*n ;$$

$$(2) \quad b*=3+ --m*m ;$$

$$(3) \quad c+=2 +m * ++m ;$$

$$(4) \quad d*=2* m*m-- ;$$

$$(5) \quad e-=2* ++m/m-- ;$$

Solution:

$$(1) \quad a+=4+ ++m*n = 4+ 7*2 = 18 ; a+=18 \rightarrow a = 0+18 \rightarrow a = 18 ; m = 7$$

$$(2) \quad b*=3+ --m*m = 3+ 6*6 = 3+36 = 39 \rightarrow b = 1*39 \rightarrow b = 33 ; m = 6$$

$$(3) \quad c+=2 +m * ++m = 2 +7 *7 = 51 \rightarrow c+= 51 \rightarrow c = 51 ; m = 7$$

$$(4) \quad d*=2* m*m-- = 2* 7*7 ; d* = 98 ; d* = 98 ; \rightarrow d = 98 ; m = 5$$

$$(5) \quad e-=2* ++m/m-- = 2* 6/6 = 2 \rightarrow e-=2 \rightarrow e = -2 ; m = 5$$

2.6.8 Ordre de priorité des opérations

En C++, l'ordre de priorité des opérations, également connu sous le nom de précedence des opérateurs, détermine comment les expressions sont évaluées. Voici une liste simplifiée des niveaux de précedence des opérateurs en C++, du plus élevé au plus bas :

<i>Opérateur</i>	<i>Description</i>
()	La plus haute précedence est accordée aux parenthèses. Évaluées de gauche à droite sans imbrication.
++, --	Précedence supérieure aux autres opérateurs arithmétiques. Évalués de gauche à droite.
*, /, %	Évalués après les opérateurs d'incrémentaion et de décrémentation. Évalués de gauche à droite.
+, -	Évalués en dernier. Évalués de gauche à droite si plusieurs sont présents.

2.6.9 L'opérateur sizeof

L'opérateur `sizeof()` retourne la taille, en octets, du type de données ou de la variable spécifiée. Cet opérateur est fréquemment utilisé pour allouer dynamiquement la mémoire ou pour déterminer la taille des types de données dans un programme.

```
#include <iostream>
using namespace std;

int main() {
    int a;
    double b;
    char c;

    cout << "Taille de int : " << sizeof(a) << " octets" << endl;
    cout << "Taille de double : " << sizeof(b) << " octets" << endl;
    cout << "Taille de char : " << sizeof(c) << " octets" << endl;

    return 0;
}
```

Résultat est :

Taille de int : 4 octets

Taille de double : 8 octets

Taille de char : 1 octets

2.6.10 La bibliothèque cmath

La bibliothèque `cmath` en C++ offre une gamme de fonctions mathématiques qui peuvent être utilisées pour effectuer des calculs arithmétiques, des opérations trigonométriques, des opérations exponentielles et logarithmiques, ainsi que diverses autres fonctions mathématiques. Voici un tableau résumant certaines des principales fonctions disponibles dans `cmath` :

<i>La fonction</i>	<i>La signification</i>	<i>Exemple</i>
sin(x)	Le sinus de x (en radian) حساب الجيب بالراديان	sin(2) → 0.909297
cos(x)	Le cosinus de x (en radian) حساب جيب التمام بالراديان	cos(2) → -0.416147
tan(x)	Le tangent de x (en radian) حساب الظل بالراديان	tan(2) → -2.18504
asin(x)	L'arc sinus de x (en radian) الدالة العكسية للجيب	asin(0.2) → 0.201358
acos(x)	L'arc cosinus de x (en radian) الدالة العكسية لجيب التمام	acos(0.2) → 1.36944
atan(x)	L'arc tangent de x (en radian) الدالة العكسية للظل	atan(0.2) → 0.197396
sinh(x) cosh(x)	Le sinus, cosinus et tangent hyperbolique de x	sinh(2) → 3.62686

tanh (x)		
pow (a ,b)	a puissance b (a ^b) حساب الأس	pow(2,3) → 8
sqrt (x)	La racine carrée de x → \sqrt{x} الجذر التربيعي	sqrt(2) → 1.41421
abs (x)	La valeur absolue de x → x	abs(-2) → 2
exp (x)	= e ^x الدالة الأسية ^x	exp(1) → 2.71828
log (x)	Logarithme naturel de x → $\ln(x)$ اللوغاريتم النيبيري	log(2) → 0.693147
log10 (x)	Logarithme à base 10 (أساس 10) اللوغاريتم العشري	log10(2) → 0.30103
round (x)	Arrondir (أقرب عدد صحيح) التدوير	round(1.61) → 2

Exercice 1 : Écrivez un programme pour calculer la surface et la circonférence d'un cercle.

<pre>#include<iostream> #include<cmath> using namespace std; int main() { float r, s, c; const float PI = 3.14159; cout<<"Entez le rayon du cercle: "; cin>>r; s = PI*pow(r,2); c = 2*PI*r; cout<<"Surface ="<<s<<endl; cout<<"Circonference ="<<c; return 0; }</pre>	<p><i>Résultat pour r=2:</i></p> <p>Entez le rayon du cercle: 2</p> <p>Surface =12.5664</p> <p>Circonference =12.5664</p>
---	---

Exercice 2 : Des résistances de 6Ω et 3Ω sont connectées en série à une source de 36v. Écrivez un programme pour trouver le courant total et la tension de chaque résistance.

<pre>#include<iostream> #include<cmath> using namespace std; int main() { float R1 = 6, R2 = 3, V = 36; float I, V1, V2; I = V/(R1 + R2); V1 = I*R1; V2 = I*R2; cout<<"Courant = "<<I<<" A"<<endl; cout<<"V1 = "<<V1<<" V"<<endl; cout<<"V2 = "<<V2<<" V"<<endl; return 0; }</pre>	<p><i>Résultat :</i></p> <p>Courant = 4 A</p> <p>V1 = 24 V</p> <p>V2 = 12 V</p>
--	---

Chapitre 3 :

Structures conditionnelles et Boucles

Chapitre 3: Structures conditionnelles et Boucles

3.1 Introduction

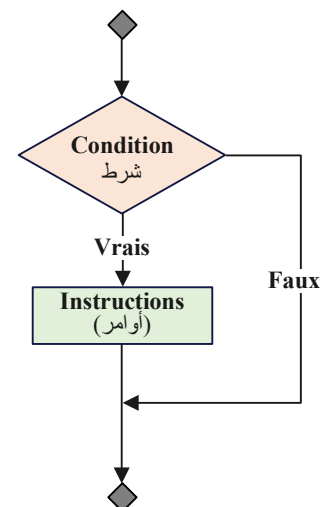
Dans ce chapitre, l'accent est mis sur les structures conditionnelles et les boucles en C++. Il explore d'abord les mécanismes des instructions '**if-else**', essentiels pour diriger le flux d'exécution dans un programme. Ensuite, l'attention se porte sur les boucles '**for**' et '**while**', outils indispensables pour la répétition de tâches. Ce segment approfondit la compréhension de ces éléments fondamentaux, cruciaux pour la programmation efficace en C++.

3.2 Les Conditions

a) La condition if

L'instruction if permet d'exécuter une instruction ou un bloc d'instructions uniquement si la condition spécifiée est vraie. Elle possède la syntaxe suivante :

```
if (condition)
{
    // instructions
}
```



Exemple :

```
#include<iostream>
using namespace std;
int main()
{
    int x;
    cout<<"Donnez x: "; cin>>x;
    if (x%2 == 0)
    {
        cout<<"x est paire";
    }
}
```

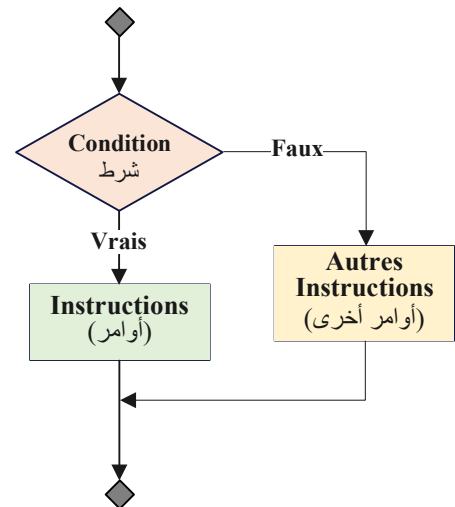
```
#include<iostream>
using namespace std;
int main()
{
    int x;
    cout<<"Donnez x: "; cin>>x;
    if (x%2 != 0)
    {
        cout<<"x est impaire";
    }
}
```

Notez que le signe **!=** signifie **différent de**.

b) La condition if... else

Cette déclaration est utilisée lorsque nous avons deux choix, elle s'écrit comme suit :

```
if (condition)
{
    // instructions
} else
{
    // Autres Instructions
}
```



Exemple 1: Écrire un programme qui demande à l'utilisateur de saisir le nombre x , puis détermine s'il est pair ou impair.

برنامج يطلب من المستخدم إدخال قيمة المتغير x ثم يقوم بالتأكد إذا كان x زوجي أو فردي.

<pre>#include<iostream> using namespace std; int main() { int x; cout<<"Donnez x: "; cin>>x; if (x%2 == 0) { cout<<x<<" est pair"; }else { cout<<x<<" est impair"; } }</pre>	<p><u>Résultat :</u></p> <p>Donnez x: 5</p> <p>5 est impair</p>
--	---

Exemple 1: Écrire un programme qui demande à l'utilisateur de saisir deux nombres n et m , puis vérifier si n est divisible par m .

برنامج يطلب من المستخدم إدخال قيمة المتغيرين n و m ثم يقوم بالتأكد إذا كان n يقبل القسمة على m .

Solution:

<pre>#include<iostream> using namespace std; int main() { int n, m ; cout << "Enter n et m : " ; cin>>n >>m ; if (n % m ==0) { cout<<n<< " est divisible par "<<m ; } else { cout<<n<<" est non divisible par "<< m ; } return (0) ; }</pre>	<p><i>Résultat :</i></p> <p>Enter n et m : 10 5</p> <p>10 est divisible par 5</p>
---	---

c) La condition if.... else if.... else

Cette déclaration est utilisée lorsque nous avons plus de deux choix, elle s'écrit comme suit :

```
if (condition)
{
// Instructions
} else if (condition)
{
// Autres Instructions
} else
{
// Autres Instructions
}
```

Exemple 1: Écrire un programme en C++ qui demande à l'utilisateur de saisir un nombre réel puis vérifier s'il est positif, négatif ou nul.

برنامج يطلب من المستخدم إدخال قيمة متغير حقيقي x ثم يقوم بالتأكد إذا كان x موجب أو سالب أو معدوم.

Solution :

```
#include <iostream>
using namespace std;
int main() {
    float x;
    cout << "Entrez le nombre x : ";
    cin >> x;
    if (x < 0) {
        cout << "x est negatif." << endl;
    } else if (x > 0) {
        cout << "x est positif." << endl;
    } else {
        cout << "x est nul." << endl;
    }
    return 0;
}
```

Résultat :

Entrez le nombre x : 0
x est nul.

Exemple 2: Écrire un programme en C++ pour trouver les racines d'une équation quadratique $ax^2 + bx + c = 0$.

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
    float a,b,c,D,x1,x2,x;
    cout<<"Entrez a : "; cin>>a;
    cout<<"Entrez b : "; cin>>b;
    cout<<"Entrez c : "; cin>>c;
    D = pow(b,2)-4*a*c;
    if (D>0){
        cout<<"Deux Solutions:"<<endl;
        x1 =(-b-sqrt(D))/(2*a);
        x2 =(-b+sqrt(D))/(2*a);
        cout<<"x1 = "<<x1<<endl;
        cout<<"x2 = "<<x2<<endl;
    } else if (D==0){
        cout<<"Double Solution:"<<endl;
        x =-b/(2*a);
        cout<<"x = "<<x<<endl;
    } else{
        cout<<"Pas de Solutions";
    }
    return 0;
}
```

Résultat :

Entrez a : 1
Entrez b : -4
Entrez c : 4
Double Solution:
x = 2

Exemple 3: Écrivez un programme pour trouver le maximum de trois entiers **a**, **b** et **c**.

أكتب برنامج يقوم بإيجاد أكبر عدد من بين ثلاث أعداد **a** ، **b** و **c**

<pre>#include<iostream> using namespace std; int main() { int a, b, c, max; cout<< "Entez les trois nombres : " ; cin>>a>>b>>c; if (a>b && a>c) { max = a; } else if (b>c){ max=b; } else{ max = c; } cout<<"Le maximum est: "<<max; return 0; }</pre>	<p><u>Résultat :</u></p> <p>Entez les trois nombres : 2 6 4 Le maximum est: 6</p>
---	---

Exemple 3: Écrivez un programme en C++ qui reçoit la moyenne d'un élève et évalue cette moyenne : Si $10 \leq \text{moyenne} < 12$, affichez "moyen". Si $12 \leq \text{moyenne} < 15$, affichez "bien". Si $15 \leq \text{moyenne} < 17$, affichez "très bien". Si $17 \leq \text{moyenne} < 20$, affichez "excellent". Dans les autres cas, affichez "Ajourné".

<pre>#include <iostream> using namespace std; int main() { float moyenne; cout << "Entrez la moyenne de l'eleve : "; cin >> moyenne; if (moyenne >= 10 && moyenne < 12) { cout << "moyen" << endl; } else if (moyenne >= 12 && moyenne < 15) { cout << "bien" << endl; } else if (moyenne >= 15 && moyenne < 17) { cout << "tres bien" << endl; } else if (moyenne >= 17 && moyenne < 20) { cout << "excellent" << endl; } else { cout << "Ajourne" << endl; } return 0; }</pre>	<p><u>Résultat :</u></p> <p>Entrez la moyenne de l'eleve : 13 bien</p>
---	---

d) L'opérateur (?:)

En C++, l'opérateur de sélection conditionnelle (?:), également connu comme l'opérateur ternaire, est un opérateur qui permet de choisir entre deux valeurs en fonction d'une condition. Il est souvent utilisé pour des affectations conditionnelles en une seule ligne.

Voici un exemple simple :

<pre>#include<iostream> using namespace std; int main() { int a = 5, b = 10; int max = (a > b) ? a : b; cout<<"Le maximum est: "<<max; return 0; }</pre>	<p><u>Résultat :</u></p> <p>Le maximum est: 10</p>
---	---

Voici d'autres exemples pour l'utilisation de l'opérateur (?:) :

```
m>n? max = m : max = n ;
y=(x>3)? 100:200;
z=(x>3) ? x*x : 2*x+1;
```

Exercice : Écrivez un programme en C++ qui demande à l'utilisateur de saisir trois nombres entiers. Le programme doit utiliser l'opérateur (?:) pour déterminer et afficher le plus grand des trois nombres.

Solution :

<pre>#include<iostream> using namespace std; int main() { int a,b,c; cout<<"Donnez a b c : "; cin>>a>>b>>c; int max1 = (a > b) ? a : b; int max2 = (b > c) ? b : c; int max = (max1 > max2) ? max1 : max2; cout<<"Le maximum est: "<<max; return 0; }</pre>	<p><u>Résultat :</u></p> <p>Donnez a b c : 1 6 4</p> <p>Le maximum est: 6</p>
--	---

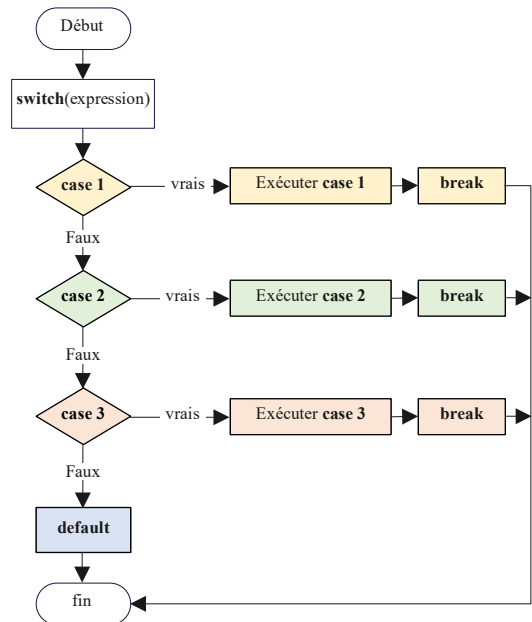
e) switch, case, default

En C++, switch est une structure de contrôle qui permet de sélectionner l'exécution d'une section de code parmi plusieurs choix. Elle est souvent utilisée comme alternative plus lisible

à une série d'instructions **if-else** lorsque l'on doit effectuer différents traitements selon la valeur d'une variable.

```
switch (expression)
{
    case valeur1:
        // bloc de code pour valeur1
        break;
    case valeur2:
        // bloc de code pour valeur2
        break;
    case valeur3:
        // bloc de code pour valeur3
        break;
    .
    .
    .
    .
    .
    default:
        // bloc de code par défaut
}

```



Exemple 1:

Écrivez un programme en C++ qui demande à l'utilisateur de saisir un nombre entre 1 et 7, correspondant à un jour de la semaine. Utilisez l'instruction **switch** pour afficher le nom du jour correspondant au nombre. Si le nombre saisi n'est pas compris entre 1 et 7, affichez un message signalant une erreur.

<pre> #include<iostream> using namespace std; int main() { int jour; cout << "Entrez un numero entre (1-7) : "; cin >> jour; switch (jour) { case 1: cout << "Le jour est samedi"; break; case 2: cout << "Le jour est dimanche"; break; case 3: cout << "Le jour est lundi"; break; case 4: cout << "Le jour est mardi"; break; case 4: cout << "Le jour est mardi"; break; case 5: cout << "Le jour est mercredi"; break; case 5: cout << "Le jour est mercredi"; break; case 6: cout << "Le jour est jeudi"; break; case 7: cout << "Le jour est vendredi"; break; default: cout << "Erreur"; } return 0; } </pre>	<p><u>Résultat :</u></p> <pre> Entrez un numero entre (1-7) : 5 Le jour est mercredi </pre>
---	--

Exemple 2 : Ecrire un programme pour recevoir un opérateur arithmétique et deux entiers. Le programme effectue l'opération arithmétique sur les deux nombres (utilisez l'instruction `switch`).

<pre> #include<iostream> using namespace std; int main() { char op; int x, y; cout << "Entrez l'operateur : " ; cin >> op; cout << "Entrez les deux nombres : " ; cin >> x >> y; switch(op) { case '+': cout << x + y; break; case '-': cout << x - y; break; case '*': cout << x * y; break; case '/': cout << x / y; break; default: cout << "Erreur"; } return 0; } </pre>	<p><u>Résultat :</u></p> <pre> Entrez l'operateur : * Entrez les deux nombres : 5 6 30 </pre>
---	--

Exemple 3 : Ecrire un programme pour trouver la valeur de y (en utilisant l'instruction `switch`).

$$y = \begin{cases} \sqrt{(x+5)^3} & x = -1 \\ 4x^3 + 5x + 4 & x = 0 \\ x - \sin(x) & x = 1 \\ 5 & \text{sinon} \end{cases}$$

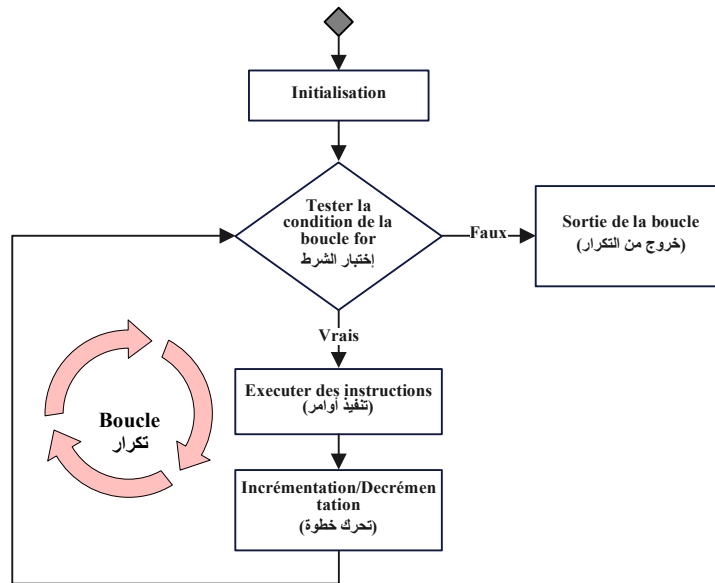
<pre>#include <iostream> #include <cmath> using namespace std; int main() { int x; float y; cout << "Entrez la valeur de x : "; cin >> x; switch(x) { case 0: y = 4*pow(x, 3) + 5 * x + 4; break; case 1: y = x-sin(x); break; default: y = 5; } cout << "y = " << y ; return 0; }</pre>	<p><u>Résultat :</u></p> <p>Entrez la valeur de x : 1</p> <p>y = 0.158529</p>
--	---

3.3 Les Boucles

En C++, les boucles sont des structures de contrôle qui permettent de répéter des instructions ou des séquences de code de manière efficace. Elles jouent un rôle très important dans la simplification du code en évitant les répétitions inutiles. Les trois types de boucles en C++ sont: la boucle `for`, la boucle `while`, et la boucle `do..while`. Chacune a ses propres caractéristiques et est choisie selon les besoins spécifiques du programme.

a) **La boucle `for`** La boucle for est écrite comme suit :

<pre>for (<u>initialisation</u>; <u>condition</u>; <u>incrémentation</u>) { // Instructions (أوامر) }</pre>
--



Exemple 1:

<pre style="font-family: monospace; font-size: 0.9em;"># include <iostream> using namespace std; int main (){ int s,i; s = 0; for (i=0; i<=10; i++){ s = s+i; } cout<< "s = " << s; return 0; }</pre>	<p>$s = 0+1+2+3+4+5+6+7+8+9+10 = 55$</p> <p><i>Résultat :</i></p> <p>s = 55</p>
--	---

Exemple 2: Ecrire un programme en C++ pour calculer la factorielle d'un entier.

<pre style="font-family: monospace; font-size: 0.9em;"># include <iostream> using namespace std; int main (){ int n, f=1,i; cout << "Donnez la valeur de n: "; cin >> n; for(i=1;i<=n;i++) { f = i*f; } cout <<"n! = " << f ; return 0; }</pre>	<p><i>Résultat :</i></p> <p>Donnez la valeur de n: 4</p> <p>n! = 24</p> <p>$f = 1 \times 2 \times 3 \times 4 = 24$</p>
--	---

Opération العملية	Expression العبارة	Equivalent المكافئ
++	i++	i = i+1
--	i--	i = i-1
+=	i+=3	i = i+3

--	i-=3	i = i-3
=	i=2	i = i*2
/=	i/=2	i = i/2

Boucles for imbriquées

En C++, les "boucles imbriquées" désignent une boucle placée à l'intérieur d'une autre boucle.

```
for (int n=0; n<= A; n++)
{
    for (int m=0; m<= B; m++)
    {
        // Instructions;
    }
}
```

Exemple 1: Ecrire un programme qui calcul la valeur de Z à partir de la formule suivante :

$$Z = \sum_{i=0}^5 \sum_{j=0}^4 i \times j$$

Solution:

<pre># include <iostream> using namespace std; int main (){ int i, j, Z=0; for(i=0;i<=5;++i) { for(j=0;j<=4;++j){ Z=Z+ (i*j); } } cout<<"Z = "<<Z; }</pre>	<p><i>Résultat :</i></p> <p>Z = 150</p>
--	--

Exercices sur les boucles for :**Exercice 1 :** Ecrire un programme qui reçoit 10 entiers et trouve la somme et la moyenne.**Solution:**

<pre># include <iostream> using namespace std; int main (){ int i, x ; float sum=0, m; for (i=1; i<=10; i++) { cout<< "donnez la valeur no "<< i<<endl; cin>>x; sum+=x; } m = sum/10; cout<<"sum= "<<sum<<endl; cout<<"moyenne= "<<m; }</pre>	<p><i>Résultat :</i></p> <pre>donnez la valeur no 1 5 donnez la valeur no 2 4 donnez la valeur no 3 3 donnez la valeur no 4 9 donnez la valeur no 5 8 donnez la valeur no 6 3 donnez la valeur no 7 5 donnez la valeur no 8 4 donnez la valeur no 9 9 donnez la valeur no 10 7 sum= 57 moyenne= 5.7</pre>
--	---

Exercice 2 : Ecrire un programme qui calcul la somme suivante :

$$somme = 1^1 + 2^2 + 3^3 + 4^4 + \dots + n^n$$

<pre>#include <iostream> #include <cmath> using namespace std; int main (){ int i,n, s=0; cout<< "donnez la valeur de n: "; cin>>n; for (i=1; i<=n; i++) { s = s+pow(i,i); } cout<<"somme= "<<s<<endl; }</pre>	<p><i>Résultat :</i></p> <pre>donnez la valeur de n: 3 somme= 32</pre>
---	--

Exercice 3 : Ecrire un programme qui calcul la somme suivante en utilisant la boucle **for**:

$$somme = \sum_{k=1}^{50} \frac{k^3}{(k+1)^2}$$

<pre>#include <iostream> #include <cmath> using namespace std; int main (){ float s = 0; for (int k=1; k<=50; k++) { s = s + pow(k,3)/pow((k+1),2); } cout<<"somme= "<<s<<endl; }</pre>	<p><i>Résultat :</i></p> <pre>somme= 1184.93</pre>
--	--

Exercice 4 : Ecrire un programme qui calcul la somme suivante en utilisant la boucle **for**:

$$somme = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$$

```
#include <iostream>
using namespace std;
int main (){
    int n; float f, s=0;
    cout<< "donnez la valeur de n: ";
    cin>>n;
    for (int i=1; i<=n; i++)
    {
        f = 1;
        for (int j=1; j<=i; j++)
        {
            f = f*j;
        }
        s = s + 1/f;
    }
    cout<<"somme= "<<s<<endl;
}
```

Résultat :

donnez la valeur de n: 4
somme= 1.70833

Exercice 5 : Ecrire qui imprime la forme suivante :

```
*
***
*****
*****
*****
*****
```

```
#include <iostream>
using namespace std;
int main() {
    int n = 5, j =1;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            cout << "*";
        }
        cout << endl;
    }
    return 0;
}
```

Résultat :

```
*
**
***
****
*****
```

b) La boucle **while**

La boucle **while** est écrite comme suit :

يبقى التكرار مستمر مادام أن الشرط مُحقق.

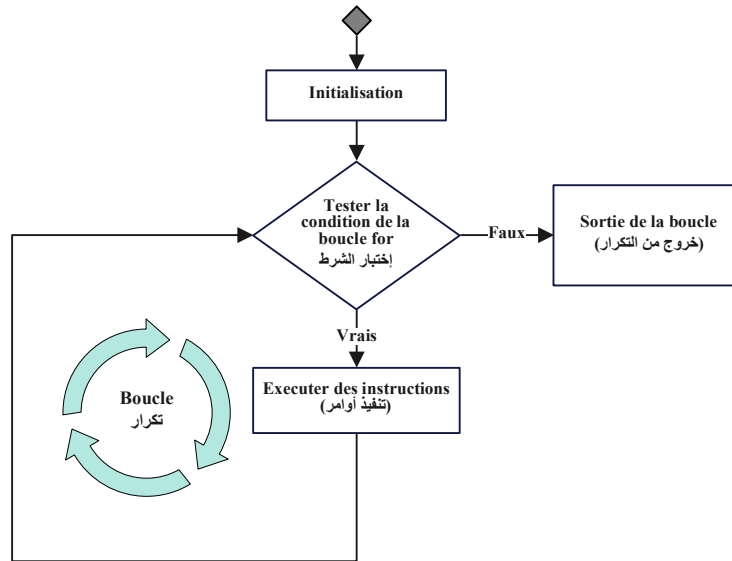
```
while (condition)
```

طرشل

```
{
```

```
    // Instructions (أوامر)
```

```
}
```



Exemple 1:

<pre># include <iostream> # include <cmath> using namespace std; int main () { int k = 1; float r = 0; while (k<=7) { r = r + pow(k,2); k=k+1; } cout<< "r = " << r; return 0; }</pre>	k= 1	r= 0
	k= 1 ≤ 5	r= 0+1 ² = 1
	k= 1+1= 2 ≤ 5	r= 1+2 ² = 5
	k= 2+1= 3 ≤ 5	r= 5+3 ² = 14
	k= 3+1= 4 ≤ 5	r= 14+4 ² = 30
	k= 4+1= 5 ≤ 5	r= 30+5 ² = 55
	k= 5+1= 6 ≤ 5	r= 55+6 ² = 91
	k= 6+1= 7 ≤ 7	r= 91+7 ² = 140
	k= 7+1= 8 < 7	أصبح الشرط غير محقق

الفرق بين for و while : تُستخدم عادةً عندما نعرف مسبقاً عدد المرات التي يُنفذ فيها التكرار، أما **while** تُستخدم عندما لا نعرف مسبقاً عدد المرات التي قد يُنفذ فيها التكرار (يعني مادام أن الشرط مُحقق فالتكرار مُستمر).

Exemple 2:

Ecrire un programme en C++ qui calcul la somme des nombres de 1 à 10 en utilisant la boucle while.

<pre># include <iostream> using namespace std; int main (){ int k = 1, sum = 0; while (k<=10){ sum += k; k=k+1; } cout<< "sum = " << sum; return 0; }</pre>	<table border="1"> <tbody> <tr><td>k= 1</td><td>sum= 0</td></tr> <tr><td>k= 1≤10</td><td>sum= 0+1= 1</td></tr> <tr><td>k= 1+1= 2≤10</td><td>sum= 1+2= 3</td></tr> <tr><td>k= 2+1= 3≤10</td><td>sum= 3+3= 6</td></tr> <tr><td>k= 3+1= 4≤10</td><td>sum= 6+4= 10</td></tr> <tr><td>k= 4+1= 5≤10</td><td>sum= 10+5= 15</td></tr> <tr><td>k= 5+1= 6≤10</td><td>sum= 15+6= 21</td></tr> <tr><td>k= 6+1= 7≤10</td><td>sum= 21+7= 28</td></tr> <tr><td>k= 7+1= 8≤10</td><td>sum= 28+8= 36</td></tr> <tr><td>k= 8+1= 9≤10</td><td>sum= 36+9= 45</td></tr> <tr><td>k= 9+1= 10≤10</td><td>sum= 45+10= 55</td></tr> <tr><td>k= 10+1= 11≤10</td><td>أصبح الشرط غير محقق</td></tr> </tbody> </table>	k= 1	sum= 0	k= 1≤10	sum= 0+1= 1	k= 1+1= 2≤10	sum= 1+2= 3	k= 2+1= 3≤10	sum= 3+3= 6	k= 3+1= 4≤10	sum= 6+4= 10	k= 4+1= 5≤10	sum= 10+5= 15	k= 5+1= 6≤10	sum= 15+6= 21	k= 6+1= 7≤10	sum= 21+7= 28	k= 7+1= 8≤10	sum= 28+8= 36	k= 8+1= 9≤10	sum= 36+9= 45	k= 9+1= 10≤10	sum= 45+10= 55	k= 10+1= 11 ≤10	أصبح الشرط غير محقق
k= 1	sum= 0																								
k= 1≤10	sum= 0+1= 1																								
k= 1+1= 2≤10	sum= 1+2= 3																								
k= 2+1= 3≤10	sum= 3+3= 6																								
k= 3+1= 4≤10	sum= 6+4= 10																								
k= 4+1= 5≤10	sum= 10+5= 15																								
k= 5+1= 6≤10	sum= 15+6= 21																								
k= 6+1= 7≤10	sum= 21+7= 28																								
k= 7+1= 8≤10	sum= 28+8= 36																								
k= 8+1= 9≤10	sum= 36+9= 45																								
k= 9+1= 10≤10	sum= 45+10= 55																								
k= 10+1= 11 ≤10	أصبح الشرط غير محقق																								

Boucles while imbriquées

En C++, les "boucles imbriquées" désignent une boucle placée à l'intérieur d'une autre boucle.

```
while (condition)
{
    while (condition)
    {
        // Instructions;
    }
}
```

Exemple 1: Ecrire un programme en C++ qui calcul la valeur de Z à partir de la formule suivante en utilisant la boucle while:

$$Z = \sum_{i=0}^5 \sum_{j=0}^4 i \times j$$

Solution:

<pre># include <iostream> using namespace std; int main (){ int i=0, Z=0; while(i<=5) { int j=0; while(j<=4){ Z = Z + (i*j); j = j + 1; } i = i + 1; } cout<<"Z = "<<Z; }</pre>	<p><i>Résultat :</i></p> <p>Z = 150</p>
---	--

Exercices sur les boucles while :

Exercice 1 : Ecrire un programme qui reçoit 5 entiers et trouve la somme et la moyenne.

Solution:

<pre># include <iostream> using namespace std; int main (){ int i=1, x ; float sum=0, m; while (i<=5) { cout<< "donnez la valeur no "<< i<<endl; cin>>x; sum = sum + x; i=i+1; } m = sum/5; cout<<"sum= "<<sum<<endl; cout<<"moyenne= "<<m; }</pre>	<p><i>Résultat :</i></p> <p>donnez la valeur no 1 5 donnez la valeur no 2 4 donnez la valeur no 3 8 donnez la valeur no 4 6 donnez la valeur no 5 3 sum= 26 moyenne= 5.2</p>
--	--

Exercice 2 : Ecrire un programme qui calcul la somme suivante :

$$somme = 1^1 + 2^2 + 3^3 + 4^4 + \dots + n^n$$

<pre>#include <iostream> #include <cmath> using namespace std; int main (){ int i = 1,n, s=0; cout<< "donnez la valeur de n: "; cin>>n; while (i<=n) { s = s+pow(i,i); i = i + 1; } cout<<"somme= "<<s<<endl; }</pre>	<p><i>Résultat :</i></p> <p>donnez la valeur de n: 3 somme= 32</p>
--	--

Exercice 3 : Ecrire un programme qui calcul la somme suivante en utilisant la boucle **while**:

$$somme = \sum_{k=1}^{50} \frac{k^3}{(k+1)^2}$$

<pre>#include <iostream> #include <cmath> using namespace std; int main (){ float s = 0; int k = 1; while (k<=50) { s = s + pow(k,3)/pow((k+1),2); k = k+1; } cout<<"somme= "<<s<<endl; }</pre>	<p><u>Résultat :</u></p> <p>somme= 1184.93</p>
--	--

Exercice 4 : Ecrire un programme qui calcul la somme suivante en utilisant la boucle **while**:

$$somme = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$$

<pre>#include <iostream> using namespace std; int main (){ int n, j, i=1; float f, s=0; cout<< "donnez la valeur de n: "; cin>>n; while (i<=n) { j = 1; f = 1; while (j<=i) { f = f * j; j = j + 1; } s = s + 1/f; i = i + 1; } cout<<"somme= "<<s<<endl; }</pre>	<p><u>Résultat :</u></p> <p>donnez la valeur de n: 4 somme= 1.70833</p>
---	---

Exercice 5 : Ecrire qui imprime la forme suivante en utilisant la boucle **while**:

```
*
***
*****
*****
*****
```

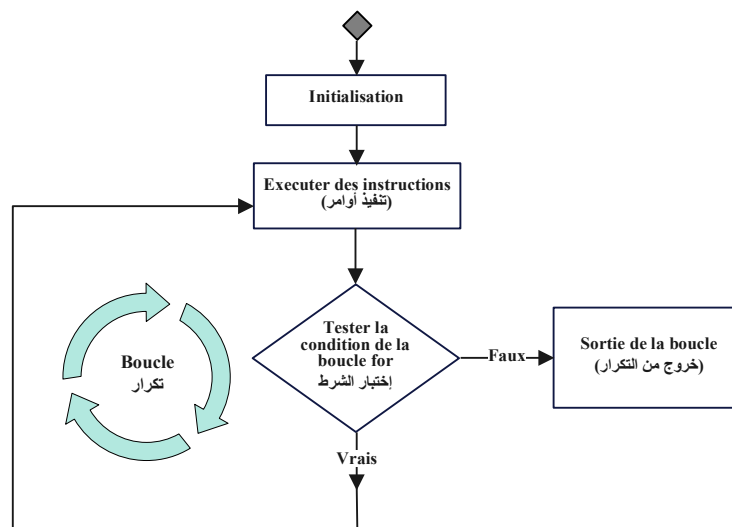
<pre>#include <iostream> using namespace std; int main() { int n = 5, j, i = 1; while (i <= n) { j = 1; while (j <= i) { cout << "*"; j = j + 1; } cout << endl; i = i + 1; } return 0; }</pre>	<p><u>Résultat :</u></p> <pre>* ** *** **** *****</pre>
---	---

c) La boucle `do...while`

La déclaration `do...while` est presque identique à la déclaration `while`. Sa syntaxe est :

```
do
{
    // Instructions (أوامر)
} while(condition)
        طشرا
```

La seule différence entre `while` et `do...while` réside dans le fait que la déclaration `do...while` exécute d'abord les instructions, puis teste la condition. Ces deux étapes sont répétées jusqu'à ce que la condition devienne fausse. Une boucle `do...while` s'effectue toujours au moins une fois, indépendamment de la valeur de la condition, car l'instruction s'exécute avant l'évaluation de la condition.



Exemple : Ecrire un programme qui calcul le factoriel d'un nombre en utilisant `do...while`.

```
#include <iostream>
using namespace std;
```

```
int main() {
    int n;
    int f = 1;

    cout << "Entrez un nombre positif: ";
    cin >> n;

    if (n < 0) {
```

Résultat :

```
Entrez un nombre
positif: 4
Factoriel de 4 =
24
```

```

        cout << "Factoriel n'existe pas";
    } else {
        int i = 1;
        do {
            f = f*i;
            i = i+1;
        } while (i <= n);

        cout << "Factoriel de " << n << " = " <<
f;
    }

    return 0;
}

```

d) L'instruction break

L'instruction break, souvent utilisée dans diverses boucles telles que while, do...while et for. Lorsqu'elle est invoquée, cette instruction met fin immédiatement à la boucle en cours. Pour expliquer sa fonctionnalité, plongeons dans un exemple :

```

#include <iostream>
using namespace std;
int main()
{
    int i = 1;
    while (i<=100)
    {
        if (i > 5)
        {
            break; // fin de la boucle lorsque
i>5.
        }
        cout<< i<<endl;
        i = i+1;
    }
    return 0;
}

```

Résultat :

1
2
3
4
5

e) L'instruction continue

L'instruction `continue` est une commande utilisée en programmation pour influencer le comportement d'une boucle (comme une boucle `while`, `do...while` ou `for`). Contrairement à l'instruction `break`, qui permet de sortir complètement de la boucle, `continue` permet de passer à l'itération suivante de la boucle en cours sans exécuter le reste du code qui se trouve à l'intérieur de cette itération particulière.

<pre>#include <iostream> using namespace std; int main() { int i = 1; while (i<=5) { if (i == 4) { i=i+1; continue; // Si i=4, l'itération sera sautée } cout<< i<<endl; i=i+1; } return 0; }</pre>	<p><u>Résultat :</u></p> <p>1 2 3 5</p>
--	---

Chapitre 4 :

Entrées/Sorties

Chapitre 4: Entrées/Sorties

4.1 Introduction

Dans le monde du développement logiciel, la capacité d'un programme à communiquer efficacement avec l'extérieur est essentielle. Que ce soit pour recevoir des instructions de l'utilisateur, afficher des résultats, ou manipuler des données stockées dans des fichiers, les entrées et sorties constituent la pierre angulaire de l'interaction entre le programme et son environnement. Ce chapitre se consacre à l'étude des entrées et sorties en C++, un aspect fondamental qui permet aux programmes d'être dynamiques et réactifs.

4.2 Flux de Sortie pour Affichage

Dans cette partie, nous nous concentrons sur les flux de sortie en C++, un élément crucial pour l'affichage de données. Le flux de sortie standard en C++ est représenté par l'objet **cout**, issu de la bibliothèque **iostream**. Il est utilisé pour envoyer des données vers la sortie standard, qui est généralement la console ou l'écran.

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World" << endl;
    return 0;
}
```

Affichage :

Hello World

Dans cet exemple, **cout** est utilisé pour afficher la chaîne de caractères "Hello World" à l'écran. L'opérateur **<<** est utilisé pour envoyer des données vers **cout**, et **endl** est utilisé pour ajouter un saut de ligne à la fin de l'affichage.

Affichage de différents types de données :

Dans le prochain exemple, nous verrons comment **cout** peut être utilisé pour afficher différents types de données, tels que des entiers, des flottants, et des caractères.

<pre>#include <iostream> using namespace std; int main() { char genre = 'm'; // char est entre guillemets '' bool estMarie = true; // true ou false unsigned short nbEnfants = 8; // [0, 255] short anneeNaissance = 1945; // [-32767, 32768] unsigned int salaire = 88000; // [0, 4294967295] double poids = 88.88; // Avec partie fractionnaire // cout << imprimer la valeur de n'importe quel type cout << "Le genre : " << genre << endl; cout << "Est marie : " << estMarie << endl; cout << "Nombre d'enfants : " << nbEnfants << endl; cout << "Ann naissance : " << anneeNaissance << endl; cout << "Le salaire est " << salaire << endl; cout << "Le poids : " << poids << endl; return 0; }</pre>	<p><u>Affichage :</u></p> <pre>Le genre : m Est marie : 1 Nombre d'enfants : 8 Ann naissance :1945 Le salaire est 88000 Le poids : 88.88</pre>
--	--

Concaténation de flux :

<pre>#include <iostream> using namespace std; int main() { int age = 22; cout << "J'ai " << age << " ans." << endl; return 0; }</pre>	<p><u>Affichage :</u></p> <pre>J'ai 22 ans.</pre>
--	---

Dans l'exemple précédent, nous avons combiné une chaîne de caractères avec une variable (ici age) pour former une phrase complète. Cela illustre la flexibilité de **cout** pour construire des sorties dynamiques.

Manipulation de la sortie :

Il est également possible de manipuler la sortie pour un affichage formaté. Par exemple, pour contrôler le nombre de décimales d'un nombre à virgule flottante, on peut utiliser la fonction **setprecision** de la bibliothèque **<iomanip>**.

<pre>#include <iostream> #include <iomanip> using namespace std; int main() { double nombre = 3.14159265; // Affichage avec une précision de deux décimales cout << fixed << setprecision(2); cout << nombre << endl; return 0; }</pre>	<p><u>Affichage :</u></p> <pre>3.14</pre>
---	---

4.3 Flux d'Entrée Clavier

Après avoir examiné comment afficher des informations à l'utilisateur, nous allons maintenant explorer comment un programme peut recevoir des données de l'utilisateur via le clavier. En C++, le flux d'entrée standard est représenté par `cin`, qui est utilisé pour lire des données saisies au clavier.

<pre>#include <iostream> using namespace std; int main() { int nombre; cout << "Entrez un nombre : "; cin >> nombre; cout << "Vous avez entre : " << nombre; return 0; }</pre>	<p><u>Affichage :</u></p> <pre>Entrez un nombre : 17 Vous avez entre : 17</pre>
--	---

Dans cet exemple, `cin` est utilisé pour lire un entier saisi par l'utilisateur. L'opérateur `>>` est utilisé pour diriger les données saisies dans la variable `nombre`.

Lecture de Chaînes de Caractères :

<pre>#include <iostream> #include <string> // pour utiliser le type string using namespace std; int main() { string mot; cout << " Entrez un mot : "; cin >> mot; cout << "Vous avez entre : " << mot << endl; return 0; }</pre>	<p><u>Affichage :</u></p> <pre>Entrez un mot : mohammedi Vous avez entre : mohammedi</pre>
--	--

Gestion des Entrées Multiples :

<pre>#include <iostream> using namespace std; int main() { int age; string nom; cout << "Entrez votre nom : "; cin >> nom; cout << "Entrez votre age : "; cin >> age; cout << nom << "Vous avez " << age << " ans."; return 0; }</pre>	<p><u>Affichage :</u></p> <pre>Entrez votre nom : samir Entrez votre age : 22 samir Vous avez 22 ans.</pre>
--	---

4.4 Gestion des Chaînes de Caractères

Les chaînes de caractères (strings) sont un aspect fondamental de la programmation en C++. Cette section explique comment gérer les chaînes de caractères, depuis leur création jusqu'à leur manipulation.

Déclaration et Initialisation des Chaînes :

<pre>#include <iostream> #include <string> // le type string using namespace std; int main() { string salutation = "Bonjour"; string nom; cout << "Entrez votre nom : "; cin >> nom; string message = salutation + " " + nom; cout << message << endl; return 0; }</pre>	<p><u>Affichage :</u></p> <p>Entrez votre nom : mohammedi Bonjour mohammedi</p>
---	---

Dans l'exemple précédent, nous avons initialisé une chaîne de caractères `salutation` et liée le nom de l'utilisateur pour former un message complet. L'opérateur `+` est utilisé pour concaténer les chaînes.

Manipulation des Chaînes :

Ici, nous explorons quelques opérations communes sur les chaînes, telles que la détermination de leur longueur, l'accès à des caractères spécifiques et la transformation de la chaîne en majuscules.

<pre>#include <iostream> using namespace std; int main() { string phrase = "C++ est amusant"; cout << "Longueur de la phrase : " << phrase.length() << endl; cout << "Caractere a l'indice 0 : " << phrase[0] << endl; cout << "Phrase en majuscules : "; for (int i = 0; i < phrase.length(); i++) { cout << char(toupper(phrase[i])); } cout << endl; return 0; }</pre>	<p><u>Affichage :</u></p> <p>Longueur de la phrase : 15 Caractere a l'indice 0 : C Phrase en majuscules : C++ EST AMUSANT</p>
---	---

Utilisation des Chaînes avec les Flux d'Entrée :

<pre>#include <iostream> using namespace std; int main() { string ligne; cout << "Entrez une phrase : "; getline(cin, ligne); cout << "Vous avez ecrit : " << ligne << endl; return 0; }</pre>	<p><u>Affichage :</u></p> <p>Entrez une phrase : univ de djelfa Vous avez ecrit : univ de djelfa</p>
--	--

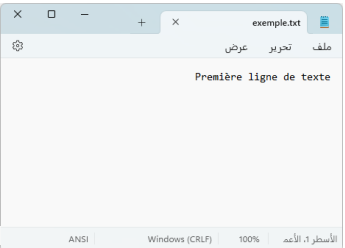
Cet exemple montre comment utiliser `getline` pour lire une ligne entière de texte, ce qui est utile lorsque l'entrée contient des espaces.

4.5 Gestion des Fichiers

La gestion des fichiers est un élément essentiel en C++, permettant aux programmes de lire et d'écrire des données dans des fichiers persistants. Cette section couvre les bases de l'interaction avec les fichiers.

a) Ouverture et Fermeture de Fichiers

Dans cet exemple, nous utilisons `ofstream` pour créer et écrire dans un fichier. La méthode `open` est utilisée pour ouvrir le fichier, et `close` pour le fermer après l'écriture.

<pre>#include <iostream> #include <fstream> // utiliser ifstream & ofstream using namespace std; int main() { ofstream monFichier; monFichier.open("exemple.txt"); if (monFichier.is_open()) { monFichier << "Premiere ligne de texte\n"; monFichier.close(); } else { cout << "Impossible d'ouvrir le fichier"; } return 0; }</pre>	<p><u>Affichage :</u></p> 
--	---

Lecture de Fichiers :

Ici, `ifstream` est utilisé pour lire le contenu d'un fichier ligne par ligne. La fonction `getline` est utilisée pour lire chaque ligne du fichier.

<pre>#include <iostream> #include <fstream> using namespace std; int main() { string ligne; ifstream monFichier("exemple.txt"); if (monFichier.is_open()) { while (getline(monFichier, ligne)) { cout << ligne << '\n'; } monFichier.close(); } else { cout << "Impossible d'ouvrir le fichier"; } return 0; }</pre>	<p><u>Affichage :</u></p> <p>Premiere ligne de texte</p>
--	--

Chapitre 5 :

Pointeurs et Tableaux

Chapitre 5: Pointeurs et Tableaux

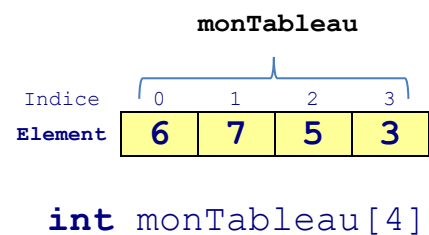
5.1 Les tableaux

Les tableaux, essentiels en programmation, permettent de stocker plusieurs valeurs sous un seul nom de variable. Cette configuration facilite la manipulation efficace des données. Dans les ordinateurs, les éléments des tableaux occupent des emplacements mémoire adjacents, garantissant un accès et un traitement rationalisés. Les tableaux se présentent sous deux formes principales :

1. Les vecteurs
2. Les matrices

a) Les vecteurs

- Un tableau est une suite de variables du même type.
- Le premier élément se trouve à l'indice 0.
- Les boucles **for** et **while** sont couramment employées pour parcourir, lire et afficher les éléments des tableaux en C++.



Les tableaux en C++ sont des collections d'éléments du même type, stockés en mémoire de manière adjacente. Pour déclarer un tableau, il faut spécifier le type des éléments et la taille du tableau :

```
int monTableau[4]; // tableau de 4 nombres entiers
float monTableau[4]; // tableau de 4 nombres réelles
char monTableau[4]; // tableau de 4 caractères
```

Cette ligne réserve un bloc de mémoire capable de stocker 4 entiers. Les éléments du tableau sont indexés de 0 à 3. Donc pour accéder aux éléments du tableau, on fait

```
monTableau[0] = 10; // met la valeur 10 au premier élément
int x = monTableau[3]; // lit la valeur du 4ème élément
```

On peut initialiser un tableau lors de sa déclaration :

```
int monTableau[5] = {6, 7, 5, 3};
```

On peut parcourir un tableau avec une boucle for :

```
for(int i=0; i < monTableau.size(); i++) {
    cout << monTableau[i] << endl;
}
```

Ceci affiche chaque élément du tableau. Les tableaux sont des structures de données très utiles

```
for(int i=0; i < 4; i++) {
    cin >> monTableau[i] ;
}
```

en C++ pour stocker et manipuler des collections de données.

Pour saisir les éléments du tableau :

Exemples 1:

<pre>#include <iostream> using namespace std; int main() { int i; int T[5] = {1,-2,4,-5,3}; for (i=0; i<=4;i++){ cout<<T[i]<<endl; } return 0; }</pre>	<p><u>Affichage :</u></p> <pre>1 -2 4 -5 3</pre>
---	--

Exemples 2:

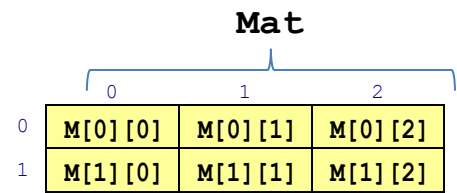
<pre>#include <iostream> using namespace std; int main() { int i, T[5], sum = 0; for (i=0; i<=4;i++){ cout<<"Donner " <<"T[" <<i+1<<" : "; cin>>T[i]; } for (i=0; i<=4;i++){ sum=sum+T[i]; } cout<<"La somme = " <<sum; return 0; }</pre>	<p><u>Affichage :</u></p> <pre>Donner T[1]:1 Donner T[2]:2 Donner T[3]:5 Donner T[4]:6 Donner T[5]:4 La somme = 18</pre>
---	--

b) Les matrices

Pour déclarer un tableau 2D (matrice) en C++, on utilise :

```
type nom[nbr des lignes][ nbr des colonnes];
```

Par exemple, `int Mat[2][3];` crée un tableau 2D d'entiers avec 2 lignes et 3 colonnes.



```
int Mat[2][3]
```

Exemple 1:

برنامج يقوم بتعريف المصفوفة M ثم عرض عناصرها.

```
#include <iostream>
using namespace std;
int main(){
    int i,j;
    int M[2][3] = {{1,-2,4},{5,3,-6}};
    for (i=0; i<=1;i++){
        for (j=0; j<=2;j++){
            cout<<M[i][j]<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

Affichage :

```
1 -2 4
5 3 -6
```

Exemple 2:

برنامج يقوم بقراءة المصفوفة M ثم جمع عناصرها.

```
#include <iostream>
using namespace std;
int main(){
    int M[2][3],i,j,sum=0;
    for (i=0; i<=1;i++){
        for (j=0; j<=2;j++){
            cout<<"donnez M["<<i+1<<" , "<<j+1<<"] : ";
            cin>>M[i][j];
        }
    }
    for (i=0; i<=1;i++){
        for (j=0; j<=2;j++){
            sum=sum+M[i][j];
        }
    }
    cout<<"La somme= "<<sum;
    return 0;
}
```

Affichage :

```
donnez M[1,1]: 1
donnez M[1,2]: 2
donnez M[1,3]: 1
donnez M[2,1]: 0
donnez M[2,2]: 3
donnez M[2,3]: 2
La somme= 9
```

c) Exercices sur les tableaux

Exercice 1

Écrire un programme qui lit un tableau de 5 nombres réels. Le programme calcule la somme, la moyenne et l'élément maximum du tableau.

Solution :

<pre>#include<iostream> using namespace std; int main() { float A[5]; int i; for (i=0; i<5; i++){ cout<<"Donner " <<"A[" <<i+1<<" : "; cin>>A[i]; } float max=A[0], sum=0; for (i=0; i<5; i++) { sum+=A[i]; if (A[i]>max) max=A[i]; } float m=sum/5; cout<<"somme = " << sum<<endl; cout<<"moyenne = " << m<<endl; cout<<"max = " <<max<<endl; }</pre>	<p><u>Affichage :</u></p> <pre>Donner A[1]:1 Donner A[2]:2 Donner A[3]:3 Donner A[4]:4 Donner A[5]:6 somme = 16 moyenne = 3.2 max = 6</pre>
--	---

Exercice 2

Écrire un programme pour trier un tableau de 5 entiers dans l'ordre croissant.

<pre>#include<iostream> using namespace std; int main() { float A[5]; int i,j; for (i=0; i<5; i++){ cout<<"Donner " <<"A[" <<i+1<<" :"; cin>>A[i]; } int t; for (i=0; i<5; i++) { for (j=0; j<5; j++) { if (A[i]<A[j]){ t = A[i]; A[i] = A[j]; A[j] = t; } } } for (i=0; i<=4;i++){ cout<<A[i]<<endl; } }</pre>	<p><u>Affichage :</u></p> <p>Donner A[1]:5</p> <p>Donner A[2]:6</p> <p>Donner A[3]:2</p> <p>Donner A[4]:3</p> <p>Donner A[5]:1</p> <p>1</p> <p>2</p> <p>3</p> <p>5</p> <p>6</p>
--	---

Exercice 3

Créer un programme calculant la quantité de nombres pairs et impairs dans un tableau contenant cinq éléments de type entier.

Solution :

<pre>#include<iostream> using namespace std; int main() { int A[5]; int i; for (i=0; i<5; i++){ cout<<"Donner " <<"A[" <<i+1<<" :"; cin>>A[i]; } int nbr_paires=0, nbr_impaires=0; for (i=0; i<5; i++) { if (A[i]%2 ==0) { nbr_paires+=1; }else{ nbr_impaires+=1; } } }</pre>	<p><u>Affichage :</u></p> <p>Donner A[1]:1</p> <p>Donner A[2]:2</p> <p>Donner A[3]:5</p> <p>Donner A[4]:6</p> <p>Donner A[5]:3</p> <p>nbr_paires = 2</p> <p>nbr_impaires = 3</p> <p><u>NB:</u></p>
---	--

<pre> } cout<<"nbr_paires = "<<nbr_paires<<endl; cout<<"nbr_impaires = "<<nbr_impaires; } </pre>	<p>ne marche pas avec A float x%2 x doit être entier</p>
--	--

Exercice 4

Écrivez un programme pour lire deux matrices A[2][2] et B[2][2], puis calcule leur somme ainsi que leur différence.

<pre> #include <iostream> using namespace std; int main() { int A[2][2], B[2][2], C[2][2], D[2][2]; int i, j; for (i=0; i<2; i++){ for (j=0; j<2; j++){ cout<<"donnez A["<<i+1<<","<<j+1<<"]: "; cin>>A[i][j]; } } for (i=0; i<2; i++){ for (j=0; j<2; j++){ cout<<"donnez B["<<i+1<<","<<j+1<<"]: "; cin>>B[i][j]; } } for (i=0; i<2; i++){ for (j=0; j<2; j++){ C[i][j] = A[i][j]+B[i][j]; D[i][j] = A[i][j]-B[i][j]; } } cout<<"Somme = "<<endl; for (i=0; i<2; i++){ for (j=0; j<2; j++){ cout<<C[i][j]<<" "; } cout<<endl; } cout<<"Difference = "<<endl; for (i=0; i<2; i++){ for (j=0; j<2; j++){ cout<<D[i][j]<<" "; } cout<<endl; } return 0; } </pre>	<p><u>Affichage :</u></p> <pre> donnez A[1,1]: 1 donnez A[1,2]: 2 donnez A[2,1]: 3 donnez A[2,2]: 4 donnez B[1,1]: 1 donnez B[1,2]: 3 donnez B[2,1]: 2 donnez B[2,2]: 4 Somme = 2 5 5 8 Difference = 0 -1 1 0 </pre>
--	--

Exercice 5

Écrivez un programme pour lire deux matrices A[2][2], puis calculez, puis calculez son minimum et son maximum.

<pre>#include <iostream> using namespace std; int main() { int A[2][2], B[2][2], C[2][2], D[2][2]; int i, j; for (i=0; i<2; i++){ for (j=0; j<2; j++){ cout<<"donnez A["<<i+1<<" , "<<j+1<<"] : "; cin>>A[i][j]; } } int max = A[0][0], min = A[0][0]; for (i=0; i<2; i++){ for (j=0; j<2; j++){ if (A[i][j]>max) { max = A[i][j]; } if (A[i][j]<min) { min = A[i][j]; } } } cout<<"max = "<<max<<endl; cout<<"min = "<<min<<endl; return 0; }</pre>	<p><i>Affichage :</i></p> <p>donnez A[1,1]: 1 donnez A[1,2]: 2 donnez A[2,1]: 3 donnez A[2,2]: 4 max = 4 min = 1</p>
--	---

Exercice 6

Écrivez un programme pour lire deux matrices A[2][2] et B[2][2], puis calculez leur produit.

<pre>#include <iostream> using namespace std; int main() { int A[2][2], B[2][2], P[2][2]; int i, j, k; for (i=0; i<2; i++){ for (j=0; j<2; j++){ cout<<"donnez A["<<i+1<<" , "<<j+1<<"] : "; cin>>A[i][j]; } } for (i=0; i<2; i++){ for (j=0; j<2; j++){ cout<<"donnez B["<<i+1<<" , "<<j+1<<"] : "; cin>>B[i][j]; } } for (i=0; i<2; i++){ for (j=0; j<2; j++){ P[i][j] = 0; for (k = 0; k<2; k++)</pre>	<p><i>Affichage :</i></p> <p>donnez A[1,1]: 1 donnez A[1,2]: 2 donnez A[2,1]: 3 donnez A[2,2]: 4 donnez B[1,1]: 1 donnez B[1,2]: 0 donnez B[2,1]: 2 donnez B[2,2]: 1 5 2 11 4</p>
--	--

<pre> { P[i][j] += A[i][k] * B[k][j] ; } } } cout<<"Le produit = "<<endl; for (i=0; i<2;i++){ for (j=0; j<2;j++){ cout<<P[i][j]<<" "; } } cout<<endl; } return 0; } </pre>	
--	--

Exercice 7

Écrivez un programme en C++ pour inverser les éléments d'un tableau. Par exemple, si le tableau original est [1, 2, 3, 4, 5], le tableau inversé devrait être [5, 4, 3, 2, 1].

<pre> #include<iostream> using namespace std; int main() { float A[5]; int i,j; for (i=0; i<5; i++){ cout<<"Donner " <<"A[" <<i+1<<"]:"; cin>>A[i];} int t; float B[5]; for (i=0; i<5; i++) { B[i] = A[4-i];} for (i=0; i<=4;i++){ cout<<B[i]<<endl;} } </pre>	<p><i>Affichage :</i></p> <pre> Donner A[1]:1 Donner A[2]:2 Donner A[3]:3 Donner A[4]:4 Donner A[5]:5 5 4 3 2 1 </pre>
--	--

Chapitre 6 :

Fonctions

Chapitre 6: Fonctions

6.1 Introduction

Les fonctions sont un outil essentiel pour tout programmeur C++. Elles permettent d'organiser le code de manière modulaire et d'éviter les répétitions inutiles. Maîtriser les fonctions est indispensable pour programmer efficacement en C++. Ce chapitre vous fournira toutes les bases nécessaires pour utiliser les fonctions de manière optimale dans vos programmes.

Les fonctions en C++ sont de deux types :

- Les fonctions retournant une valeur au programme principal se terminent par une instruction **return**.
- Les fonctions ne renvoyant pas de valeur sont définies par le mot **void**.

6.2 Fonctions retournant une valeur

La définition de la fonction est illustrée ci-dessous :

```
type nom_de_la_fonction (type paramètre1, type paramètre2, ..... )
{
    //instructions ;
    return valeur ;
}
```

Dans la définition ci-dessus, le premier mot est le type de la fonction, il correspond au type de données qu'elle renvoie. Le deuxième élément est le nom de la fonction. (type paramètre1, type paramètre2,) sont appelés les arguments de la fonction. Par exemple :

```
int somme(int x, int y)
{
    int somme = x + y;
    return somme;
}
```

L'appel de la fonction : استدعاء الدالة

Le programme principal appelle la fonction en déclarant une variable suivie du nom de la fonction et de ses arguments, comme suit :

```
nom_de_variable = nom_de_la_fonction(arguments)
```

Par exemple : `s1 = somme(a, b) ;`

Le programme complet sera le suivant :

<pre>#include <iostream> using namespace std; int somme(int x, int y) { int somme = x + y; return somme; } int main() { int a = 2, b = 3; int s1 = somme(a, b) ; ← استدعاء الدالة cout<<"s1 = "<<s1; }</pre>	<p><u>Affichage :</u> s1 = 5</p>
--	--------------------------------------

Exemple 1 :

Écrivez une fonction qui trouve la somme de deux nombres, le programme principal appelle cette fonction pour trouver la somme de quatre nombres.

<pre>#include<iostream> using namespace std; float sum(float x, float y) { return x + y; } int main() { float a, b, c, d; cout<<"donnez les 4 nombres : "<<endl; cin >> a >> b >> c >> d; float s = sum(sum(a, b), sum(c, d)); cout << "sum = " << s; return 0; }</pre>	<p><u>Affichage :</u> donnez les 4 nombres : 2 4 5 6 sum = 17</p>
---	---

Exemple 2 :

Écrivez un programme qui inclut fonction nommée `puissance`, reçoit deux nombres et renvoie la puissance `puissance(a, b)=ab`.

<pre>#include<iostream> using namespace std; int puissance(int x, int y) { int p = 1; if (y==0){ return p; } else { for(int i=1; i<=y; i++) { p=p*x; } return p; } }</pre>	<p><u>Affichage :</u> donnez a et b pour calculer a^b : 2 3 a^b = 8</p>
---	---

```

int main(){
    float a,b,c;
    cout<<"donnez a et b pour calculer a^b : "<<endl;
    cin >> a >> b ;
    c = puissance(a,b);
    cout << "a^b = " << c;
    return 0;
}

```

Exemple 3 :

Écrivez un programme qui inclut fonction nommée `factoriel`, reçoit un nombre et renvoie le factoriel `factoriel(n)=n!`.

```

#include<iostream>
using namespace std;
int factoriel(int x)
{
    int f = 1;
    for(int i=1; i<=x; i++) {
        f=f*i;
    }
    return f;
}

int main(){
    float n,f;
    cout<<"donnez n pour calculer n! : ";
    cin >> n;
    f = factoriel(n);
    cout << "n! = " << f;
    return 0;
}

```

Affichage :

```

donnez n pour
calculer n! : 3
n! = 6

```

Exemple 4 :

Écrivez une fonction qui trouve le factoriel d'un entier, le programme principal appelle cette fonction pour calculer "y" à partir de la formule suivante. $y = \frac{k! \times m!}{(k-m)!}$ avec $y = k > m$

```

#include<iostream>
using namespace std;
int factoriel(int x)
{
    int f = 1;
    for(int i=1; i<=x; i++) {
        f=f*i;
    }
    return f;
}

int main(){
    float y, k, m;
    cout<<"donnez k et m: ";
    cin >> k>>m;
    y = factoriel(k)* factoriel(m)/ factoriel(k-m);
    cout << "y = " << y;
    return 0;
}

```

Affichage :

```

donnez k et m: 3 2
y = 12

```


6.3 Fonctions void

Une fonction qui ne renvoie pas de valeur au programme principal est connue sous le nom de *procédure*. En C++, une telle fonction est simplement identifiée en plaçant le mot **void** avant le nom de la fonction, comme montré ci-dessous :

```
void somme(int x, int y)
{
    int sum; sum = x+y;
    cout<<sum;
}
```

Ce type de fonctions est appelé *تُسندعى* par son nom directement depuis le programme principal, comme illustré :

```
somme(a, b) ;
```

Le programme complet sera le suivant :

<pre>#include <iostream> using namespace std; void somme(int x, int y) { int sum; sum = x+y; cout<<sum; } int main(){ int a = 2, b = 3; somme(a, b) ;</pre> <p style="text-align: right;">← استدعاء الدالة</p>	<p><u>Affichage :</u> 5</p>
--	---------------------------------

Exemple :

Écrivez une fonction qui trouve et imprime le carré d'un entier, le programme principal appelle cette fonction pour trouver les carrés de 0 à 10.

<pre>#include<iostream> using namespace std; int caree(int x) { int f; f=x*x; cout<<f<<endl; } int main(){ for (int i=1;i<=10;i++) caree(i); return 0; }</pre>	<p><u>Affichage :</u> 1 4 9 16 25 36 49 64 81 100</p>
--	---

Chapitre 7 :
Programmation orientée objet en
C++

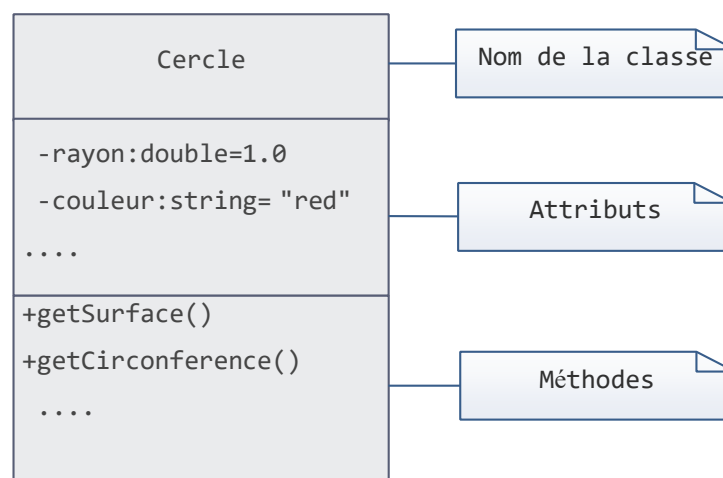
Chapitre 7: Programmation orientée objet en C++

7.1 Introduction

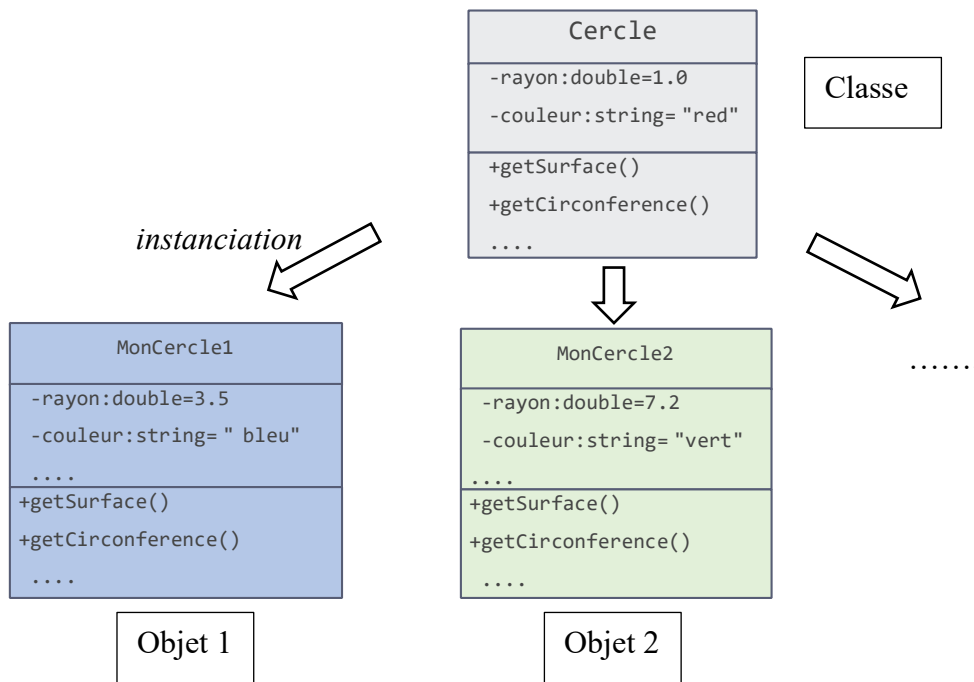
La programmation orientée objet (POO) est un style de programmation qui structure les programmes informatiques autour de données appelées *objets* plutôt que des fonctions. La POO présente de nombreux avantages. Elle permet de décomposer logiquement un programme en entités modulaires, de le rendre plus facile à maintenir et faire évoluer. Le code peut être réutilisé via l'héritage. C++ est un langage qui intègre nativement et efficacement la POO. Nous allons à présent illustrer ses concepts clés à travers des exemples concrets de code.

7.2 Notions de classe et d'objet

On définit « l'objet » comme une entité atomique du monde réel composée d'un état et d'un comportement. L'objet est donc une capsule qui contient ce que fait l'objet; les noms employés à l'intérieur n'interfèrent pas avec l'extérieur. Une classe d'objets décrit un groupe d'objets ayant des attributs similaires, des comportements identiques, des relations communes avec les autres objets. Il s'agit d'une description abstraite en termes de données et de comportements d'une famille d'objets. Les généralités sont contenues dans la classe et les particularités sont contenues dans les objets. Les objets informatiques sont construits à partir de la classe, par un processus appelé : *instanciation*.



Exemple de représentation graphique d'une classe



Déclaration en C++

```

#include <iostream>
using namespace std;

class Cercle {
public:
    double rayon = 1.0;
    string couleur="rouge";

    double getSurface() {
        return 3.14159 * rayon * rayon;
    }

    double getCirconference() {
        return 2 * 3.14159 * rayon;
    }
};

int main() {
    // Objet N 01
    Cercle monCercle1;
    monCercle1.rayon = 3.5; monCercle1.couleur = "bleu";
    cout << "Rayon du cercle 1 : " << monCercle1.rayon << endl;
    cout << "Couleur du cercle 1 : " << monCercle1.couleur << endl;
    cout << "Surface du cercle 1: " << monCercle1.getSurface() << endl;
    cout << "Circonference du cercle 1 : " << monCercle1.getCirconference() << endl;

    // Objet N 02
    Cercle monCercle2;
    monCercle2.rayon = 7.2; monCercle2.couleur = "vert";
    cout << "Rayon du cercle 2 : " << monCercle2.rayon << endl;
    cout << "Couleur du cercle 2 : " << monCercle2.couleur << endl;
    cout << "Surface du cercle 2: " << monCercle2.getSurface() << endl;
    cout << "Circonference du cercle 2 : " << monCercle2.getCirconference() << endl;
    return 0;
}
    
```

Affichage :

```

Rayon du cercle 1 : 3.5
Couleur du cercle 1 : bleu
Surface du cercle 1: 38.4845
Circonference du cercle 1 : 21.9911
Rayon du cercle 2 : 7.2
Couleur du cercle 2 : vert
Surface du cercle 2: 162.86
Circonference du cercle 2 : 45.2389
    
```

Dans l'exemple précédent, **Cercle** est une classe avec deux attributs (rayon, couleur) et deux méthodes **getSurface()** et **getCirconference()**. Dans main, nous avons créé deux objets : **monCercle1** et **monCercle2** à partir de cette classe.

7.2.1 Principes de la programmation orientée objet

Dans cette section, nous présentons certaines propriétés essentielles de l'approche orientée objet. En effet, seules les caractéristiques présentant un intérêt incontestable durant la phase d'analyse et de conception du programme, sont exposées ci-dessous.

a) Concept d'encapsulation

L'encapsulation consiste à masquer des attributs et des méthodes de l'objet vis-à-vis de accès extérieurs non autorisés. En effet, certains attributs et méthodes ont pour seul objectif des traitements internes à l'objet et ne doivent pas être exposés à l'accès d'extérieur. *Encapsulés*, ils sont appelés les attributs et méthodes privés de l'objet. La définition de l'encapsulation se fait au niveau de la classe. Le C++ implémente l'encapsulation en utilisant les mots réservés : *public*, *private* ou *protected*.

- *public* : Les membres sont accessibles de n'importe où dans le programme.
- *private* : Les membres sont accessibles uniquement à l'intérieur de la classe elle-même.
- *protected* : Les membres sont accessibles à l'intérieur de la classe et par les classes qui en héritent.

Exemple :

```
#include <iostream>
using namespace std;

class Cercle {
private:
    double rayon;
    string couleur;
public:
    double getSurface() {
        return 3.14159 * rayon * rayon;
    }

    double getCirconference() {
        return 2 * 3.14159 * rayon;
    }

    void setRayon(double r) {
        rayon = r;
    }

    void setCouleur(string c) {
        rayon = c;
    }
};
```

} private

```

int main() {
    Cercle monCercle1;
    monCercle1.setRayon(3.5);
    monCercle1.setCouleur("bleu");

    cout << "Surface du cercle: " << monCercle1.getSurface() << endl;
    cout << "Circonference du cercle: " << monCercle1.getCirconference() << endl;

    return 0;
}

```

Affichage :

```

Surface du cercle: 38.4845
Circonference du cercle: 21.9911

```

Dans l'exemple précédent, les deux attributs **rayon** et **couleur** de la classe **cercle** sont déclarés comme **private**. Cela signifie qu'ils sont protégés et ne peuvent pas être accédés directement de l'extérieur de la classe.

b) Constructeur

En C++, un constructeur est une fonction membre spéciale d'une classe qui est automatiquement appelée lorsqu'un objet de cette classe est créé.

Exemple :

```

#include <iostream>
using namespace std;

class Cercle {
private:
    double rayon;
    string couleur;

public:
    Cercle(double r, string c) : rayon(r), couleur(c) {}
    double getSurface() {
        return 3.14159 * rayon * rayon;
    }

    double getCirconference() {
        return 2 * 3.14159 * rayon;
    }
};

int main() {
    Cercle monCercle1(3.5,"bleu");
    cout << "Surface du cercle: " << monCercle1.getSurface() << endl;
    cout << "Circonference du cercle: " << monCercle1.getCirconference() << endl;

    return 0;
}

```

← Constructeur

Dans cet exemple, le constructeur de la classe **Cercle** est défini pour initialiser l'objet **monCercle1** avec deux paramètres spécifiques : **rayon** (3.5) et **couleur** ("bleu").

Voici une explication détaillée du rôle du constructeur dans cet exemple :

1. **Initialisation des Attributs** : Le constructeur `Cercle(double r, string c)` prend deux paramètres - un `double` pour le rayon (`r`) et une `string` pour la couleur (`c`). Il utilise l'initialisation de liste des membres (member initializer list) pour assigner ces valeurs aux attributs privés `rayon` et `couleur` de l'objet. Cette méthode d'initialisation est plus efficace que l'assignation directe dans le corps du constructeur.
2. **Syntaxe d'Initialisation** : La syntaxe `: rayon(r), couleur(c) {}` après la signature du constructeur est la liste d'initialisation des membres. Ici, `radius(r)` initialise l'attribut `rayon` avec la valeur du paramètre `r`, et `couleur(c)` fait de même avec l'attribut `couleur` et le paramètre `c`.
3. **Création d'Objets** : Dans la fonction `main`, un objet `monCercle` de la classe `Cercle` est créé en utilisant ce constructeur : `Cercle monCercle(10.0, "rouge");`. Cela crée un cercle de rayon `10.0` et de couleur `rouge`.
4. **Utilisation de Méthodes** : Après la création de l'objet `monCercle`, les méthodes `getSurface` et `getCirconference` sont appelées pour calculer et afficher la surface et la circonférence du cercle.

En résumé, le constructeur dans cet exemple est utilisé pour initialiser les objets `Cercle` avec des valeurs spécifiques pour le rayon et la couleur, permettant une utilisation directe et pratique de l'objet avec ses méthodes associées.

c) Destructeur

Le destructeur est une méthode spéciale qui est appelée automatiquement lorsqu'un objet est détruit. Voici un exemple :

```
#include <iostream>
using namespace std;


class Cercle {
private:
    double rayon;
    string couleur;

public:
    Cercle(double r, string c) : rayon(r), couleur(c) {}

    ~Cercle() {
        cout << "Destruction du cercle de couleur " << couleur << endl;
    }

    double getSurface() {
        return 3.14159 * rayon * rayon;
    }

    double getCirconference() {
        return 2 * 3.14159 * rayon;
    }
};
```

 Destructeur

```
};  
  
int main() {  
    Cercle monCercle1(3.5, "bleu");  
    cout << "Surface du cercle: " << monCercle1.getSurface() << endl;  
    cout << "Circonférence du cercle: " << monCercle1.getCirconférence() << endl;  
  
    // Le destructeur sera appelé automatiquement ici, à la fin de main  
    return 0;  
}
```

Affichage :

```
Surface du cercle: 38.4845  
Circonférence du cercle: 21.9911  
Destruction du cercle de couleur bleu
```

Dans cet exemple, le destructeur de la classe **Cercle** est défini pour effectuer des actions lors de la destruction d'un objet de cette classe. Voici une explication détaillée du rôle du destructeur dans cet exemple :

1. **Définition du Destructeur** : Le destructeur `~Cercle()` est déclaré dans la classe **Cercle**. Comme pour tous les destructeurs en C++, il commence par un tilde (`~`) suivi du nom de la classe. Ce destructeur est conçu pour s'exécuter automatiquement lorsqu'un objet `monCercle1` est détruit.
2. **Action du Destructeur** : À l'intérieur du destructeur, un message est affiché indiquant que l'objet `monCercle1` est en train d'être détruit. Le message inclut également la **couleur** de l'objet, illustrant comment on peut accéder aux attributs de l'objet dans le destructeur.
3. **Appel Automatique du Destructeur** : Dans la fonction `main`, lorsque l'exécution atteint la fin du bloc (c'est-à-dire la fin de la fonction `main`), l'objet `monCercle1` sort de portée. À ce moment-là, le destructeur de l'objet `monCercle1` est automatiquement appelé. Il affiche le message défini dans le destructeur, indiquant que l'objet est en train d'être détruit.
4. **Gestion des Ressources** : Bien que dans cet exemple simple, le destructeur ne fasse qu'afficher un message, dans des cas plus complexes, il serait utilisé pour libérer des ressources allouées par l'objet, comme la mémoire ou les connexions de fichiers, pour éviter les fuites de ressources.

En résumé, le destructeur dans cet exemple est principalement utilisé à des fins de démonstration pour montrer quand et comment il est appelé. Il fournit un mécanisme pour effectuer un nettoyage ou d'autres actions lorsque l'objet est sur le point d'être détruit.

d) Concept d'héritage

L'héritage est une technique offerte par les langages de programmation pour construire une classe à partir d'une ou plusieurs autres classes. Il met en œuvre les principes de généralisation et de spécialisation en partageant explicitement les attributs et méthodes communs au moyen d'une hiérarchie de classes. La classe fille (sous-classe) hérite des caractéristiques (attributs et méthodes) de sa classe mère (super-classe), mais elle se distingue par ses caractéristiques propres.

Pour démontrer l'héritage en utilisant l'exemple de la classe `Cercle`, nous pouvons créer une classe de base, disons `Forme`, et ensuite faire hériter la classe `Cercle` de cette classe de base. Voici comment cela pourrait être fait :

Étape 1 : Créer une Classe de Base `Forme`

```
class Forme {
    private:
        string couleur; // Attribut privé

    public:
        Forme(string c) : couleur(c) {}

        string getCouleur() {
            return couleur;
        }
};
```

Ici, la classe `Forme` a un attribut privé `couleur` et un constructeur pour initialiser cet attribut.

La méthode `getCouleur` permet d'accéder à la `couleur`.

Étape 2 : Faire Hériter la Classe `Cercle` de `Forme` وراثة

```
class Cercle : public Forme {
    private:
        double rayon;

    public:
        Cercle(double r, string c) : Forme(c), rayon(r) {}

        double getSurface() {
            return 3.14159 * rayon * rayon;
        }

        double getCirconference() {
            return 2 * 3.14159 * rayon;
        }
};
```

Dans cette version, `Cercle` hérite toujours de `Forme`. Le constructeur de `Cercle` initialise le `rayon` et appelle le constructeur de `Forme` pour initialiser la `couleur`.

Étape 3: Utiliser la Classe **Cercle** dans **main**

```

int main() {
    Cercle monCercle1(3.5, "bleu");
    cout << "Surface du cercle: " << monCercle1.getSurface() << endl;
    cout << "Circonference du cercle: " << monCercle1.getCirconference() << endl;
    cout << "Couleur du cercle: " << monCercle1.getCouleur() << endl;

    return 0;
}

```

Dans ce code, **main** crée un objet **Cercle** et utilise ses méthodes, y compris **getCouleur** héritée de **Forme**.

Cette approche simplifiée de l'héritage en C++ montre comment une classe de base peut transmettre ses attributs et méthodes à une classe dérivée. Cela convient aux scénarios où l'héritage n'implique pas la nécessité de redéfinir des comportements (méthodes) dans les classes dérivées.

Programme complet :

```

#include <iostream>
using namespace std;

// Classe de base Forme
class Forme {
private:
    string couleur; // Attribut privé

public:
    // Constructeur
    Forme(string c) : couleur(c) {}

    // Getter pour couleur
    string getCouleur() {
        return couleur;
    }
};

// Classe dérivée Cercle
class Cercle : public Forme {
private:
    double rayon;

public:
    // Constructeur
    Cercle(double r, string c) : Forme(c), rayon(r) {}

    // Méthode pour calculer la surface
    double getSurface() {
        return 3.14159 * rayon * rayon;
    }

    // Méthode pour calculer la circonférence
    double getCirconference() {
        return 2 * 3.14159 * rayon;
    }
};

// Fonction principale
int main() {
    // Création d'un objet Cercle
    Cercle monCercle1(3.5, "bleu");
}

```

```
// Affichage de la surface et de la circonférence
cout << "Surface du cercle: " << monCercle1.getSurface() << endl;
cout << "Circonference du cercle: " << monCercle1.getCirconference() << endl;
cout << "Couleur du cercle: " << monCercle1.getCouleur() << endl;

return 0;
}
```

Affichage :

```
Surface du cercle: 38.4845
Circonference du cercle: 21.9911
Couleur du cercle: bleu
```

Travaux Pratiques

TP N° 01 Introduction à la programmation en C++

Objectif : Cette séance d'introduction vise à se familiariser avec le langage de programmation C++. On apprendra à configurer un environnement de développement C++ et à écrire quelques programmes simples.

Matériel nécessaire : Un ordinateur avec un environnement de développement C++ installé (par exemple, Code::Blocks, Dev-C++, ou Visual Studio Code avec l'extension C++).

Lorsque vous lancez Dev C++, vous voyez apparaître l'écran ci-contre.



```

1 #include <iostream>
2 using namespace std;
3 int main () {
4     cout<<"Hello World";
5
6
7     return 0;
8 }
9
                    
```

include <iostream>

هذا السطر معناه تضمين مكتبة الإدخال والإخراج (Input/Output) داخل البرنامج، هذه المكتبة تحتوي جميع الأوامر التي تسمح بالقراءة والطباعة مثل `cin` و `cout`.

using namespace std;

هذا السطر يستخدم لجعل جميع الأوامر التي تأتي من مكتبة الإدخال والإخراج في C++ معروفة داخل البرنامج بدون الحاجة إلى كتابة std:: قبل كل استخدام. وبالتالي فهذا السطر يُبسط كتابة البرنامج ويجعله أكثر

```

int main() {
    // programme
    return 0;
}
                    
```

البرنامج يُكتب بين الحاضنتين.

Affichage d'un texte	<code>cout<<"Hello World";</code>	طباعة نص
endl: Aller à la ligne suivante	<code>cout<<"Hello World"<<endl;</code>	معناها اذهب الى السطر الموالي
Lecture d'une variable	<code>cin>>x;</code>	قراءة متغير
Affichage de la valeur d'une variable	<code>cout<<x;</code>	طباعة قيمة المتغير
Commentaire sur une seule ligne	<code>// voici un commentaire.</code>	كتابة تعليق في سطر واحد
Commentaire sur plusieurs lignes	<code>/* voici un commentaire. */</code>	كتابة تعليق في عدة سطور

- كل الأوامر في C++ تنتهي بنقطة بفاصلة.
- كل الأوامر في لغة C++ تُكتب بأحرف صغيرة، مثلا `cout` وليس `Cout`
- الـ C++ يُفرق ما بين الأحرف الصغيرة والأحرف الكبيرة يعني مثلا اذا كتبنا `x=10 ; X=14 ;` فإن الـ C++ يعتبرهما متغيران منفصلان وليسا نفس المتغير.
- أسماء المتغيرات دائما تبدأ بحرف أو _ ومنموع احتواءها على رموز خاصة مثل: `! @ $ % & ' () * + , - . / : ;` إلخ

```

1 # include <iostream>
2 using namespace std;
3 int main() {
4     // calcul de la somme de deux nombres
5     int a,b,r;
6     cout<<"donner la valeur de a ";
7     cin>>a;
8     cout<<"donner la valeur de b ";
9     cin>>b;
10    r = a+b;
11    cout<<a<< " + "<<b << " = "<<r ;
12    return 0;
13 }

```

Opération	Symbole	Exemple	العملية
Addition	+	r = a+b	الجمع
Soustraction	-	r = a-b	الطرح
Multiplication	*	r = a*b	الضرب
Division	/	r = a/b	القسمة
Modulo	%	r = a%b	باقي القسمة

Type de variable	Explication	نوع المتغير
const	const folat k=0.28; const folat k(0.28);	ثابت
int	Entier (100)	عدد صحيح
float/double	Réel (1.253)	عدد حقيقي
char	Caractère ("A")	حرف واحد
string	Chaine de caractère ("mot")	نص
bool	True ou False	صح أو خطأ

Nom et prénom		Poste	
Spécialité			

Application 1

Écrire un programme en C++ permettant **d'afficher** votre nom et prénom, spécialité et matricule.

<i>nom et prénom</i> <i>spécialité</i> <i>matricule</i>

Application 2

Écrire un programme en C++ qui permet d'additionner, de soustraire, de multiplier et de diviser **deux nombres réels**.

Application 3

Créer un programme en C++ qui **demande le rayon** d'un cercle à l'utilisateur. Ensuite, le programme doit calculer et **afficher le périmètre et la surface** du cercle.

Application 4

Écrire un programme qui permet de convertir une température **de Celsius en Kelvin et Fahrenheit** en utilisant les formules suivantes :

$$K = C + 273.15 \quad F = C \times \frac{9}{5} + 32$$



TP N° 02 : Les conditions en C++

La structure générale : الصيغة العامة

<pre>if (condition){ // ... }</pre>	<pre>if (condition){ // ... } else { // ... }</pre>	<pre>if (condition){ // ... } else if (condition){ // ... } else { // ... }</pre>
---	---	---

Exemples :

أكتب كل مثال من الأمثلة التالية
في صفحة منفصلة

<pre># include <iostream> using namespace std; int main (){ float n; cout<<"donner n: "; cin>> n; if (n!=0) { cout<<"pas nul"; } return 0; }</pre>	<pre># include <iostream> using namespace std; int main (){ float m; cout<<"Entrez votre moyenne: "; cin>> m; if (m>=10) { cout<<"Admis"; } else{ cout<<"Ajourne"; } return 0; }</pre>
--	---

<pre># include <iostream> using namespace std; int main (){ float x; cout<<"donner x: "; cin>> x; if (x>0) { cout<<"x positif"; } else if (x<0){ cout<<"x negatif"; } else { cout<<"x nul"; } return 0; }</pre>	<pre>#include <iostream> using namespace std; int main() { int n; cout << "Entrez un chiffre entre 1 et 3: "; cin >> n; switch (n) { case 1: cout << "Vous avez choisi 1"; break; case 2: cout << "Vous avez choisi 2"; break; case 3: cout << "Vous avez choisi 3"; break; default: cout << "Erreur!."; } return 0; }</pre>
--	--

`#include <cmath>` inclut des fonctions mathématiques en C++. كتابة الدوال الرياضية

La fonction	La signification	Exemple
<code>sin(x)</code>	Le sinus de x (en radian) حساب الجيب بالراديان	<code>sin(2) → 0.909297</code>
<code>cos(x)</code>	Le cosinus de x (en radian) حساب جيب التمام بالراديان	<code>cos(2) → -0.416147</code>
<code>tan(x)</code>	Le tangent de x (en radian) حساب الظل بالراديان	<code>tan(2) → -2.18504</code>
<code>asin(x)</code>	L'arc sinus de x (en radian) الدالة العكسية للجيب	<code>asin(0.2) → 0.201358</code>
<code>acos(x)</code>	L'arc cosinus de x (en radian) الدالة العكسية لجيب التمام	<code>acos(0.2) → 1.36944</code>
<code>atan(x)</code>	L'arc tangent de x (en radian) الدالة العكسية للظل	<code>atan(0.2) → 0.197396</code>
<code>sinh(x), cosh(x), tanh(x)</code>	Le sinus, cosinus et tangent hyperbolique de x	<code>sinh(2) → 3.62686</code>
<code>pow(a,b)</code>	a puissance b (a^b) حساب الأس	<code>pow(2,3) → 8</code>
<code>sqrt(x)</code>	La racine carrée de x → \sqrt{x} الجذر التربيعي	<code>sqrt(2) → 1.41421</code>
<code>abs(x)</code>	La valeur absolue de x → $ x $	<code>abs(-2) → 2</code>
<code>exp(x)</code>	= الدالة الأسية e^x	<code>exp(1) → 2.71828</code>
<code>log(x)</code>	Logarithme naturel de x → $\ln(x)$ اللوغاريتم النيبيري	<code>log(2) → 0.693147</code>
<code>log10(x)</code>	Logarithme à base 10 (أساس 10) اللوغاريتم العشري	<code>log10(2) → 0.30103</code>
<code>round(x)</code>	Arrondir (أقرب عدد صحيح) التدوير	<code>round(1.61) → 2</code>

Opérations logiques :

Opérateur	Utilisation	Exemple	Explication
<	Inférieur à	A<B	أقل من
>	Supérieur à	A>B	أكبر من
<=	Inférieur ou égal à	A<=B	أصغر من أو يُساوي
>=	Supérieur ou égal à	A>=B	أكبر من أو يُساوي
==	Égalité	A == B	يُساوي
!=	Différent de	A != B	لا يُساوي
& (and)	ET	شرط 1 & شرط 2	و
(or)	OU	شرط 1 شرط 2	أو

Exemple : quel sera le résultat de ce programme ? ماهي نتيجة هذا المثال ؟

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
    float x, y ;
    x = 9;
    if (x > 0 & x <= 5) {
        y = sqrt(x);
    } else {
        y = pow(x,2);
    }
    cout << "y = " << y;
    return 0;
}
```



TP N° 03 : Les boucles en C++

La structure générale de la boucle « for » :

```

for ( initialisation ; condition ; incrémentation ) {
    // ...
}
```

قوي ادبيل
طرش
قو طخل

Exemples :

أكتب الأمثلة 1-3 في صفحات منفصلة

<pre># include <iostream> using namespace std; int main (){ int s,i; s = 0; for (i=0; i<=10; i++){ s = s+i; } cout<< "s = " << s; return 0; }</pre>	1	<pre># include <iostream> using namespace std; int main (){ int p,i; p = 1; for (i=5; i>=1; i--){ p = p*i; } cout<< "p = " << p; return 0; }</pre>	2
s= 0+1+2+3+4+5+6+7+8+9+10= 55		p= 5×4×3×2×1= 120	

إختصارات :

Opération العملية	Expression العبارة	Equivalent المكافئ
++	i++	i = i+1
--	i--	i = i-1
+=	i+=3	i = i+3
-=	i-=3	i = i-3
=	i=2	i = i*2
/=	i/=2	i = i/2

Exemple : *quel sera le résultat de ces programmes ? ماهي نتيجة هذه البرامج ؟*

<pre># include <iostream> using namespace std; int main (){ int s,i; s = 0; for (i=2; i<6; i++){ s = s+i; } cout<< "s = " << s; return 0; }</pre>	<pre># include <iostream> using namespace std; int main (){ int p,i; p = 1; for (i=5; i>2; i--){ p = p*i; } cout<< "p = " << p; return 0; }</pre>
---	---

La structure générale de la boucle « while » :

<pre>while (<u>condition</u>) { <i>طرشا</i> <i>مادام أن الشرط مُحقق.</i> // ... }</pre>	يبقى التكرار مستمر
---	--------------------

Exemples :

<pre># include <iostream> # include <cmath> using namespace std; int main (){ int k = 1; float r = 0; while (k<=7){ r = r + pow(k,2); k=k+1; } cout<< "r = " << r; return 0; }</pre>	3	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">k= 1</td> <td style="width: 35%;">r= 0</td> </tr> <tr> <td>k= 1≤5</td> <td>r= 0+1²= 1</td> </tr> <tr> <td>k= 1+1= 2≤5</td> <td>r= 1+2²= 5</td> </tr> <tr> <td>k= 2+1= 3≤5</td> <td>r= 5+3²= 14</td> </tr> <tr> <td>k= 3+1= 4≤5</td> <td>r= 14+4²= 30</td> </tr> <tr> <td>k= 4+1= 5≤5</td> <td>r= 30+5²= 55</td> </tr> <tr> <td>k= 5+1= 6≤5</td> <td>r= 55+6²= 91</td> </tr> <tr> <td>k= 6+1= 7≤7</td> <td>r= 91+7²= 140</td> </tr> <tr> <td>k= 7+1= 8≤7</td> <td>أصبح الشرط غير محقق</td> </tr> </table>	k= 1	r= 0	k= 1≤5	r= 0+1 ² = 1	k= 1+1= 2≤5	r= 1+2 ² = 5	k= 2+1= 3≤5	r= 5+3 ² = 14	k= 3+1= 4≤5	r= 14+4 ² = 30	k= 4+1= 5≤5	r= 30+5 ² = 55	k= 5+1= 6≤5	r= 55+6 ² = 91	k= 6+1= 7≤7	r= 91+7 ² = 140	k= 7+1= 8≤7	أصبح الشرط غير محقق
k= 1	r= 0																			
k= 1≤5	r= 0+1 ² = 1																			
k= 1+1= 2≤5	r= 1+2 ² = 5																			
k= 2+1= 3≤5	r= 5+3 ² = 14																			
k= 3+1= 4≤5	r= 14+4 ² = 30																			
k= 4+1= 5≤5	r= 30+5 ² = 55																			
k= 5+1= 6≤5	r= 55+6 ² = 91																			
k= 6+1= 7≤7	r= 91+7 ² = 140																			
k= 7+1= 8≤7	أصبح الشرط غير محقق																			

الفرق بين for و while : `for` تُستخدم عادةً عندما نعرف مسبقًا عدد المرات التي يُنفذ فيها التكرار، أما `while` تُستخدم عندما لا نعرف مسبقًا عدد المرات التي قد يُنفذ فيها التكرار (يعني مادام أن الشرط مُحقق فالتكرار مُستمر).

Exemple : *quel sera le résultat de ce programme ?*

<pre># include <iostream> # include <cmath> using namespace std; int main (){ int k = 1; float r = 0; while (k<4){ r = r + pow(k,2); k=k+1; } cout<< "r = " << r; return 0; }</pre>	<pre># include <iostream> # include <cmath> using namespace std; int main (){ int k = 2; float r = 0; while (k<=3){ r = r + pow(k,3); k=k+1; } cout<< "r = " << r; return 0; }</pre>
--	---



Nom et prénom		Poste	
Spécialité			

Application 1

Ecrire un programme qui affiche tous les entiers de 8 jusqu'à 23 en utilisant un **for** puis la boucle **while**.

Application 2

Ecrire un programme en C++ pour calculer le **factoriel** d'un nombre. Le nombre doit être saisi par l'utilisateur (utiliser la boucle **for**).

- AN : $n = 9$

Application 3

En utilisant la boucle **for** écrire un programme en C++ qui calcule et affiche les valeurs suivantes :

$$S = \sum_{k=1}^{10} \frac{k^2}{2k+1} \quad P_1 = \prod_{k=1}^{10} \frac{\sqrt{k^2+1}}{2k+1}$$

$$P_2 = \prod_{k=3}^{10} (-1)^k \frac{1}{k+1}$$

Application 4

Ecrire un programme en C++ qui calcule et affiche les valeurs suivantes :

$$z = \sum_{i=0}^5 \sum_{j=0}^4 \sqrt{i \times j}$$

Solution des exemples

<pre># include <iostream> using namespace std; int main (){ int s,i; s = 0; for (i=2; i<6; i++){ s = s+i; } cout<< "s = " << s; return 0; }</pre>	<pre># include <iostream> using namespace std; int main (){ int p,i; p = 1; for (i=5; i>2; i--){ p = p*i; } cout<< "p = " << p; return 0; }</pre>
$s = 0+2+3+4+5 = \boxed{14}$	$p = 1 \times 5 \times 4 \times 3 = \boxed{60}$

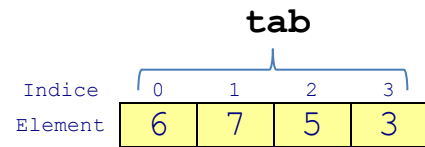
<pre># include <iostream> # include <cmath> using namespace std; int main (){ int k = 1; float r = 0; while (k<4){ r = r + pow(k,2); k=k+2; } cout<< "r = " << r; return 0; }</pre>	<pre># include <iostream> # include <cmath> using namespace std; int main (){ int k = 2; float r = 0; while (k<=3){ r = r + pow(k,3); k=k+1; } cout<< "r = " << r; return 0; }</pre>
$r = 0+1^2+3^2 = \boxed{10}$	$r = 0+2^3+3^3 = \boxed{35}$

```
# include <iostream>
# include <cmath>
using namespace std;
int main(){
    float s=1 ;
    for (float k=3;k<=10;k++){
        s = s+pow(-1,k)*(k/|k+1|);
    }
    cout<<s;
}
```

TP N° 04 : Les Tableaux (vecteurs et matrices) en C++

La structure générale :

- Un tableau est une suite de variables du même type.
- Le premier élément se trouve à l'indice 0.
- Les boucles for et while sont couramment employées pour parcourir, lire et afficher les éléments des tableaux en C++.



`int tab[4]`

Déclaration :

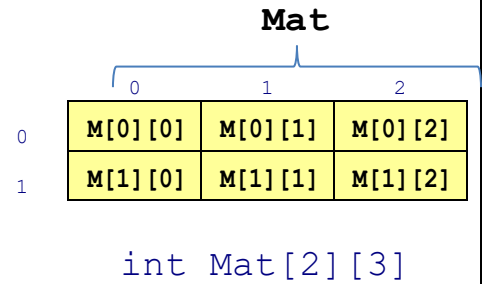
```
int A[10]; // Le tableau A contient 10 éléments de type entier. أعداد صحيحة
float B[20]; // Le tableau B contient 20 éléments de type flottant. أعداد حقيقية
char name[20]; // Le tableau name contient 20 éléments de type caractère. حروف
```

Exemples :

اكتب الأمثلة 1-2 في صفحات منفصلة

<pre>#include <iostream> using namespace std; int main(){ int i; int T[5] = {1,-2,4,-5,3}; for (i=0; i<=4;i++){ cout<<T[i]<<endl; } return 0; }</pre>	1	<pre>#include <iostream> using namespace std; int main(){ int i, T[5], sum = 0; for (i=0; i<=4;i++){ cout<<"Donner " <<"T["<<i+1<<"]:"; cin>>T[i]; } for (i=0; i<=4;i++){ sum=sum+T[i]; } cout<<"La somme = "<<sum; return 0; }</pre>	2
برنامج يقوم بتعريف الشعاع T ثم عرض عناصره.		برنامج يقوم بقراءة الشعاع T ثم جمع عناصره.	

- Pour déclarer un tableau 2D (matrice) en C++, on utilise :
type nom[nbr des lignes][nbr des colonnes];
Par exemple, `int Mat[2][3];` crée un tableau 2D d'entiers avec 2 lignes et 3 colonnes.



Exemples :

أكتب الأمثلة 1-2 في صفحات منفصلة

```
#include <iostream>
using namespace std;
int main(){
    int i,j;
    int M[2][3] = {{1,-2,4},{5,3,-6}};
    for (i=0; i<=1;i++){
        for (j=0; j<=2;j++){
            cout<<M[i][j]<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

3

برنامج يقوم بتعريف المصفوفة M ثم عرض عناصرها.

```
#include <iostream>
using namespace std;
int main(){
    int M[2][3],i,j,sum=0;
    for (i=0; i<=1;i++){
        for (j=0; j<=2;j++){
            cout<<"donner M["<<i+1<<","<<j+1<<"] : ";
            cin>>M[i][j];
        }
    }
    for (i=0; i<=1;i++){
        for (j=0; j<=2;j++){
            sum=sum+M[i][j];
        }
    }
    cout<<"La somme= "<<sum;
    return 0;
}
```

4

برنامج يقوم بقراءة المصفوفة M ثم جمع عناصرها.

Nom et prénom		Poste	
Spécialité			

Application 1

Écrire un programme qui calcul **le produit scalaire** de deux vecteurs dont la taille et les éléments sont saisis par l'utilisateur. برنامج لحساب الجداء السلمي لشعاعين

5	2	10	
8	1	8	
2	5	10	
-1	2	-2	
4	-5	-20	
8	1	8	
A	X	B	14

Application 2

Écrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau. Le programme doit afficher **le nombre d'entiers supérieurs ou égaux à 10**. عدد العناصر التي أكبر أو تساوي 10.

Application 3

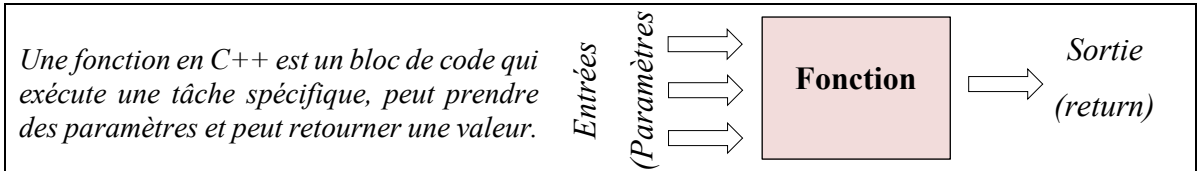
Écrire un programme qui recherche une valeur dans un tableau d'entiers (saisie par l'utilisateur), et affiche un message pour dire si **oui** ou **non** la valeur a été retrouvée. برنامج يبحث عن قيمة معينة داخل جدول

Application 4

Écrire un programme qui affiche les valeurs **Minimale**, **Maximale**, et la **Moyenne** des éléments d'un tableau d'entiers dont les éléments sont saisis par l'utilisateur. أكتب برنامجا يقوم بعرض القيمة الصغرى، الكبرى، والمتوسطة لجدول من القيم الصحيحة يقوم بإدخالها المستخدم.

TP N° 05 : Les Fonctions en C++

La structure générale :

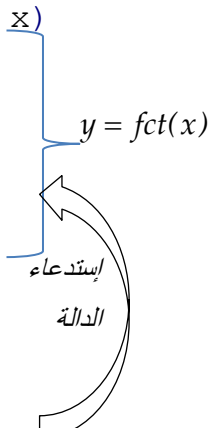
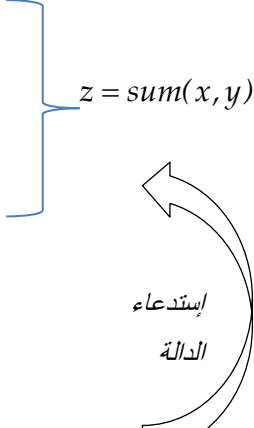


Déclaration :

```
type nom(type entrée 1, type entrée 2,.....)
{
    .....
    return sortie;
}
```

Exemples :

أكتب الأمثلة 1-2 في صفحات منفصلة

برنامج يقوم بحساب جذر تربيعي لعدد باستخدام دالة خارجية.	برنامج يقوم بجمع عددين باستخدام دالة خارجية.
<pre>#include<iostream> #include<cmath> using namespace std; float fct(float x) { float y; y = sqrt(x); return y; } int main() { float a,b; a = 9; b = fct(a); cout<<"La racine = "<<b; return 0; }</pre> <div style="text-align: right; margin-top: 20px;"> $y = fct(x)$  </div>	<pre>#include<iostream> using namespace std; int sum(int x, int y) { int z; z = x+y; return z; } int main() { int a,b,c; a = 3; b = 4; c = sum(a,b); cout<<"somme= "<<c; return 0; }</pre> <div style="text-align: right; margin-top: 20px;"> $z = sum(x, y)$  </div>
1	2

Nom et prénom		Poste	
Spécialité			

Application 1

Écrire une fonction **distance** ayant comme paramètres 4 réelles x_a , y_a et x_b , y_b qui représentent les coordonnées de deux points A et B et qui renvoie la distance AB . برنامج لحساب المسافة بين نقطتين باستخدام دالة

- AN : (1,2), (-2,1).

Application 2

Écrire un programme en C++ qui demande à l'utilisateur de saisir le rayon d'une sphère, puis calcule son périmètre ($P = 2\pi r$), sa surface ($S = 4\pi r^2$) et son volume ($V = 4/3\pi r^3$), où r représente le rayon. Le programme doit définir trois fonctions : "**perimetre**", "**surface**" et "**volume**". برنامج لحساب محيط، مساحة وحجم كرة.

- AN : $r = 2.36$

Application 3

*Ecrire un programme en C++ pour calculer le factoriel d'un nombre en utilisant la fonction **fact**.*

برنامج يقوم بحساب العاملية

- AN : $n = 6$

Application 4

*Ecrire un programme en C++ qui demande à l'utilisateur de saisir **les trois coefficients** a , b et c d'une équation de second degré de la forme $ax^2+bx+c = 0$; Le programme doit ensuite **déterminer et afficher les solutions**. Le programme doit définir la fonction **delta** pour le calcul du discriminant Δ .*

- AN : $a = 1$, $b = 2$ et $c = -2$.

Exercice 3 :

Ecrire le programme **Prog3.cpp** qui résout l'équation $f(x) = e^{-2x} - \cos(x) - 3$ par la méthode de Newton avec $x_i = -2$ et $\varepsilon = 0.001$.

Ecrire le programme **Prog4.cpp** qui résout l'équation $f(x) = x - \sin(x) + \frac{\pi}{6} - \frac{\sqrt{3}}{2}$ par la méthode de Newton avec $x_i = \pi$ et $\varepsilon = 0.001$.

Algorithme de la méthode de Newton :

Données : f, f', x_i, ε ;

Tan que $|f(x_i)| > \varepsilon$

$$x_f = x_i - \frac{f(x_i)}{f'(x_i)};$$

$$x_i = x_f;$$

Fin Tanque

Afficher (x_f)

Exercice 4 :

Ecrire le programme **Prog5.cpp** qui résout l'équation $x + e^x + 1 = 0$ par la méthode de point fixe avec $x_i = 1$ et $\varepsilon = 0.001$.

Ecrire le programme **Prog6.cpp** qui résout l'équation $f(x) = x - \sin(x) + \frac{\pi}{6} - \frac{\sqrt{3}}{2}$ par la méthode de point fixe avec $x_i = \pi$ et $\varepsilon = 0.001$.

Algorithme de la méthode de Point Fixe :

Données : f, g, x_i, ε ;

Tan que $|f(x_i)| > \varepsilon$

$$x_f = g(x_i);$$

$$x_i = x_f;$$

Fin Tanque

Afficher (x_f);

Nom et prénom		Poste	
Spécialité			

Application 1

```

# include <iostream>
# include <cmath>
using namespace std;
float f(float x){
    float y;
    y = x-exp(x);
    return y;
}
int main(){
    float x = 2;
    cout<<"f(x) = "<<f(x);
}

```

} $y = f(x)$

Application 2

```

# include <iostream>
# include <cmath>
using namespace std;
float f(float x){
    .....
}
int main(){
    float a,b,.....;
    a= .....; b=.....; eps = .....;
    xm=.....;
    while (abs(f(xm))>eps){
        if (.....){
            .....;
        } else {
            .....;
        }
        xm = (a+b)/2;
    }
    cout<<"la solution =
"<<.....;
}

```

Annexe

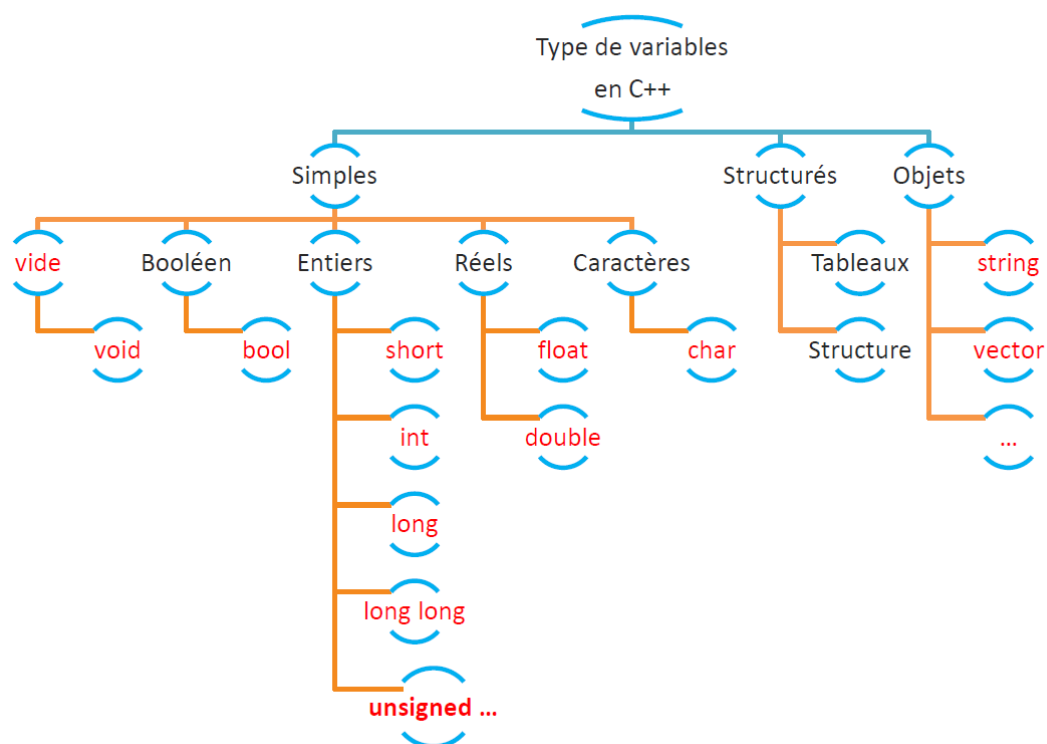
1. Quelques fichiers d'entêtes du C++

Nom du fichier d'entête	Rôle	
<code><iostream></code>	Fournit les capacités centrales d'entrée/sortie du langage C++	cin, cout, cerr, clog
<code><cstdio></code>	Fournit les capacités centrales d'entrée/sortie du langage C	printf, scanf
<code><cmath></code>	Fournit les fonctions mathématiques courantes	fabs, fmax, log, sin, cos, sqrt, pow,...
<code><cstring></code>	Permet de faire des opérations sur les chaînes de caractères.	strcat, strchr, strcmp, strepy,...
<code><cfloat></code>	Spécifie les propriétés des nombres en virgule flottante	
<code><ctime></code>	Permet de manipuler les formats de date et d'heure	
<code><string></code>	Permet de manipuler les chaînes de caractères.	
<code><vector></code>	Permet de manipuler des tableaux dynamiques d'éléments contigus	

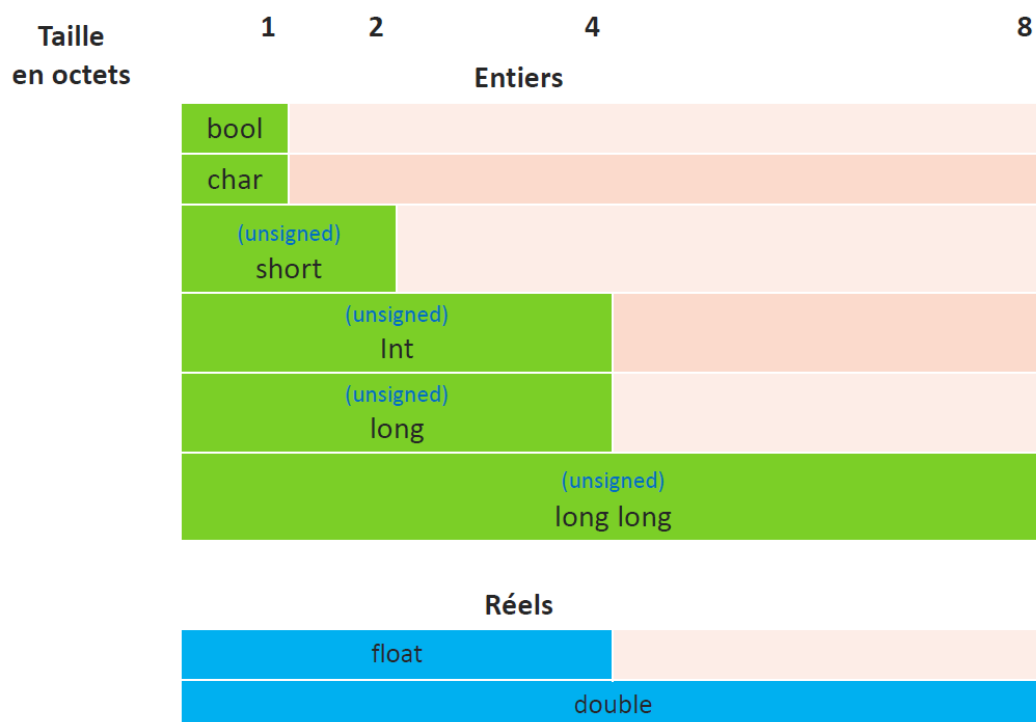
2. Mots clés réservés de C++

<code>alignas</code>	<code>enum explicit</code>	<code>return short</code>
<code>alignof</code>	<code>export extern</code>	<code>signed</code>
<code>and</code>	<code>false float</code>	<code>sizeof</code>
<code>and_eq asm</code>	<code>for friend</code>	<code>static</code>
<code>auto bitand</code>	<code>goto</code>	<code>static_assert</code>
<code>bitor bool</code>	<code>if inline int</code>	<code>static_cast</code>
<code>break case</code>	<code>long mutable</code>	<code>struct switch</code>
<code>catch char</code>	<code>namespace new</code>	<code>template this</code>
<code>char16_t</code>	<code>noexcept</code>	<code>thread_local</code>
<code>char32_t</code>	<code>not</code>	<code>throw true</code>
<code>class</code>	<code>not_eq</code>	<code>try typedef</code>
<code>compl const</code>	<code>nullptr</code>	<code>typeid</code>
<code>constexpr</code>	<code>operator or</code>	<code>typename union</code>
<code>const_cast</code>	<code>or_eq private</code>	<code>unsigned using</code>
<code>continue</code>	<code>protected public</code>	<code>virtual void</code>
<code>decltype</code>	<code>register</code>	<code>volatile</code>
<code>default</code>	<code>reinterpret_cast</code>	<code>wchar_t while</code>
<code>delete do</code>		<code>xor xor_eq</code>
<code>double</code>		
<code>dynamic_cast</code>		
<code>else</code>		

3. Types de variables



4. Tailles des types de variables



Bibliographie

- [1] Bjarne Stroustrup, Marie-Cécile Baland, Emmanuelle Burr, Christine Eberhardt, « Programmation: Principes et pratique avec C++ », Edition Pearson, 2012.
- [2] Jean-Cédric Chappelier, Florian Seydoux, « C++ par la pratique. Recueil d'exercices corrigés et aide- mémoire », PPUR Édition : 3e édition, 2012.
- [3] Jean-Michel Léry, Frédéric Jacquenot, « Algorithmique, applications aux langages C, C++ en Java », Edition Pearson, 2013.
- [4] Frédéric DROUILLON, « Du C au C++ - De la programmation procédurale à l'objet », Eni; Édition : 2e édition, 2014.
- [5] Claude Delannoy, « Programmer en langage C++ », Edition Eyrolles, 2000.