Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Programmation procédurale en C++

Laurent Debize

isitech
L'ÉCOLE DU NUMÉRIQUE

BTS SIO

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

1 Outils

2 Entrées/sorties

3 Variables

4 Opérateurs

- Affectation
- Opérateurs arithmétiques
- Opérateurs relationnels

5 Tests

- if
- if else
- switch

6 Boucles

- while
- do ... while
- for

7 Les chaînes de caractères

8 Tableaux 1D

- Déclaration
- Accès
- Initialisation

9 Les fonctions

- Définition
- Utilisation
- Retour
- Variables

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

IDE

IDE = Integrated Development Environment

Un environnement de développement intégré est composé de 3 éléments :

- Editeur de texte
- Compilateur
- Debogueur

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Editeur

- Coloration syntaxique
- Aide à la saisie

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Compilateur

Fonctionnement

1. Analyse lexicale
2. Analyse syntaxique
3. Génération de code binaire

Remarques

- Le fichier exécutable n'est produit que si le code source est valide.

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Debogueur

- Utile quand le code est exécutable, mais ne se comporte pas comme prévu.
- Exécution du code ligne par ligne
- Visualisation de l'état des variables
- Ne fonctionne que sur un code compilé, donc sans erreurs syntaxiques.

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Trame de base

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

int main()
{
   ...
   return 0;
}
```

Inclure la bibliothèque d'entrées/sorties
Espace de travail std (raccourci)

programme principal (point d'entrée)

Corps du programme
Renvoyer l'entier 0 (= « tout est OK ! »)

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

1 Outils

2 Entrées/sorties

3 Variables

4 Opérateurs

- Affectation
- Opérateurs arithmétiques
- Opérateurs relationnels

5 Tests

- if
- if else
- switch

6 Boucles

- while
- do ... while
- for

7 Les chaînes de caractères

8 Tableaux 1D

- Déclaration
- Accès
- Initialisation

9 Les fonctions

- Définition
- Utilisation
- Retour
- Variables

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Entrées/sorties

L'ordinateur est égocentrique :

- Entrée = **son** entrée : clavier
- Sortie = **sa** sortie : écran

En C++

Flux d'entrée : `cin` (console in)
Flux de sortie : `cout` (console out)

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Sortie

Voir ça comme un flux qui se dirige vers la sortie :

```
cout << "Hello world!" << endl;
```

On enchaîne tout :

- Les chevrons << indiquent le sens comme des flèches. Ici lire : « envoyer dans `cout` »
- Affichage de "Hello world !"
- Puis retour à la ligne (`endl`)

Remarques

- Toutes les instructions terminent par un point-virgule !
- Les chaînes de caractères doivent être entre guillemets doubles (" ").

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Entrée

Pour lire un caractère au clavier, voir ça comme un flux qui vient de l'entrée :

```
cin >> n;
```

- Les chevrons >> indiquent : « router le clavier vers la variable `n` »

Attention

- Il va falloir définir qui est `n` !

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

1 Outils
2 Entrées/sorties
3 Variables
4 Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels
5 Tests
if
if else
switch
6 Boucles
while
do ... while
for
7 Les chaînes de caractères
8 Tableaux 1D
Déclaration
Accès
Initialisation
9 Les fonctions
Définition
Utilisation
Retour
Variables

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Variables

En C++, les types de base sont :

- `bool` : booléen, peut valoir `true` ou `false`,
- `char` : caractère
- `string` : chaîne de caractères
- `int` : entier
- `float` : flottant (nombre à virgule)
- `void` : ensemble vide de valeurs.

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Variables

Important !

Toutes les variables doivent être déclarées avant utilisation pour spécifier au compilateur combien de cases mémoire il doit réserver.

Exemples :

`int n;`	entier `n`
`int a, b;`	entiers `a` et `b`
`float x;`	réel `x`

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Variables

Exemple pour demander un nombre :

```
//Déclaration de n :
int n;

cout << "Saisir un nombre" << endl;

//Enregistrement de la valeur dans la variable n
cin >> n;

//affichage du contenu de n
//(on remarque le chaînage des chevrons)
cout << "Vous avez saisi : " << n << endl;
```

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

Opérateurs

Affectation

L'affectation est l'action de **mettre une valeur dans une variable**.
Cela s'écrit avec l'opérateur $=$

Exemple

`n = 3;` signifie « `n` prend la valeur 3 ».

On peut voir ça comme une flèche où 3 va dans `n` :
`n <- 3`

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

Opérateurs

Opérateurs arithmétiques

- $+$: addition
- - : soustraction
- * : multiplication
- / : division (entière ou réelle)
- % : modulo = reste de la division entière

Outils
Entrées/sorties
Variables
Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels
Tests
if
if else
switch
Boucles
while
do ... while
for
Les chaînes de caractères
Tableaux 1D
Déclaration
Accès
Initialisation
Les fonctions
Définition
Utilisation
Retour
Variables

Opérateurs

Opérateurs relationnels

- comparaisons : `>` `>=` `<=` `<`
- égalité et inégalité : `==` `!=`
- négation (opérateur unaire) : `!`
- ET relationnel : `&&`
- OU relationnel : `||`

Remarques :

- `>=` se lit « supérieur ou égal ».
- Ne pas confondre `==` (comparaison) avec `=` (affectation) :
  a = b ne vérifie pas si a vaut b, mais met la valeur de b dans a.
- `!=` : remarquer que le « ! » barre le « = » pour faire un « $\neq$ ».

1 Outils
2 Entrées/sorties
3 Variables
4 Opérateurs
- Affectation
- Opérateurs arithmétiques
- Opérateurs relationnels

5 Tests
- if
- if else
- switch

6 Boucles
- while
- do ... while
- for

7 Les chaînes de caractères
8 Tableaux 1D
- Déclaration
- Accès
- Initialisation

9 Les fonctions
- Définition
- Utilisation
- Retour
- Variables

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

Tests : if

Structure :

```
if (expression)
{
   instructions;
}
```

Exemple

```
if (n >= 0)
{
   cout << "Nombre positif" << endl;
}
```

Outils
Entrées/sorties
Variables
Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels
Tests
if
if else
switch
Boucles
while
do ... while
for
Les chaînes de caractères
Tableaux 1D
Déclaration
Accès
Initialisation
Les fonctions
Définition
Utilisation
Retour
Variables

Tests : if else

Structure :

```
if (expression)
{
   instructions1;
}
else
{
   instructions2;
}
```

Exemple

```
if (n == 0)
{
   cout << "Nombre nul" << endl;
}
else
{
   cout << "Nombre non nul" << endl;
}
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Tests : if else

Remarque : on peut enchaîner les if else

```
if (n == 0)
{
   cout << "Nombre nul" << endl;
}
else if (n > 0)
{
   cout << "Nombre strictement positif" << endl;
}
else
{
   cout << "Nombre strictement négatif" << endl;
}
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Tests : switch

Structure :

```
switch (expression)
{
   case constante1:
      instruction1;
      break;
   case constante2:
      instruction2;
      break;
   default:
      instructions;
}
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Tests : switch

Exemple avec un caractère :

```
char c;
cin >> c;
switch (c)
{
   case 's':
      cout << "Enregistrement du fichier..." << endl;
      break;
   case 'o':
      cout << "Ouverture du fichier..." << endl;
      break;
   case 'n':
      cout << "Nouveau fichier..." << endl;
      break;
   default:
       cout << "Raccourci inconnu !" << endl;
}
```

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

1 Outils

2 Entrées/sorties

3 Variables

4 Opérateurs

- Affectation
- Opérateurs arithmétiques
- Opérateurs relationnels

5 Tests

- if
- if else
- switch

6 Boucles

- while
- do ... while
- for

7 Les chaînes de caractères

8 Tableaux 1D

- Déclaration
- Accès
- Initialisation

9 Les fonctions

- Définition
- Utilisation
- Retour
- Variables

Outils
Entrées/sorties
Variables
Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels
Tests
if
if else
switch
Boucles
while
do ... while
for
Les chaînes de caractères
Tableaux 1D
Déclaration
Accès
Initialisation
Les fonctions
Définition
Utilisation
Retour
Variables

Boucle while

Structure

Faire `instructions` tant que `expression` est vrai.

```
while (expression)
{
    instructions;
}
```

Exemple

```
int i = 0;
while (i < 10)
{
    cout << "i = " << i << endl;
    i++;
}
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Boucle do ... while

Structure

Identique à `while`, sauf qu'on fait au moins une fois les instructions dans le `do` même si `expression` est faux.

```
do
{
   instructions;
} while (expression); //ne pas oublier ce ';'
```

Exemple

Le `do ... while` est très utile pour refaire saisir : en effet il faut bien exécuter les instructions au moins une fois pour avoir des valeurs à tester !

```
double x;
do
{
   cout << "Entrez un nombre positif" << endl;
   cin >> x;
}while(x < 0);
```

Outils
Entrées/sorties
Variables
Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels
Tests
if
if else
switch
Boucles
while
do ... while
for
Les chaînes de caractères
Tableaux 1D
Déclaration
Accès
Initialisation
Les fonctions
Définition
Utilisation
Retour
Variables

Boucle for

Structure

Là où avec un **while** on écrit :

```
expression1;
while(expression2)
{
    instructions;
    expression3;
}
```

Avec un **for** on écrit :

```
for (expression1; expression2; expression3)
{
    instructions;
}
```

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

Boucle for

Exemple :

```
int i = 0;
while(i < 10)
{
    cout << "i = " << i << endl;
    i++;
}
```

devient :

```
for (int i = 0; i < 10;  i++)
{
    cout << "i = " << i << endl;
}
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Boucle for

Remarques :

- `for (int i = 0; i < 10;  i++)` peut se lire :
  « Pour i allant de 0 à 9 par pas de 1 »
- On peut même déclarer la variable de boucle `i` dans le for :
  `for (int i = 0; etc.)`
  La variable de boucle est ainsi connue uniquement dans la boucle : c'est ce qu'on appelle une **variable locale**.

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

1 Outils
2 Entrées/sorties
3 Variables
4 Opérateurs
- Affectation
- Opérateurs arithmétiques
- Opérateurs relationnels

5 Tests
- if
- if else
- switch

6 Boucles
- while
- do ... while
- for

7 Les chaînes de caractères
8 Tableaux 1D
- Déclaration
- Accès
- Initialisation

9 Les fonctions
- Définition
- Utilisation
- Retour
- Variables

Caractères

- Type : `char`
- Codé sur 1 octet
- Se note entre simples guillemets : `'a'`

Un caractère fait partie de la table ASCII :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STH	ETH	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	CD2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	Ž	□
9	□	'	'	"	"	•	–	—	˜	™	š	›	œ	□	ž	Ÿ
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Exemple

```
char mon_caractere = 'a';
cout << mon caractere: //affiche a.
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Chaînes de caractères

- Type : `string`
- Suite de caractères
- Se note entre doubles guillemets : `"Hello world !"`

Exemple

```
char ma_chaine = "Hello world !";
cout << ma_chaine; //affiche Hello world !
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Chaînes de caractères

Concaténation

La concaténation est l'action de coller bout à bout deux chaînes de caractères. Cela se fait avec l'opérateur $+$.

Exemple :

```
string chaine1 = "Bonjour ";
string chaine2 = "Toto";
string chaine3 = chaine1 + chaine2;
cout << chaine3; //affiche "Bonjour Toto".
```

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

Tableaux 1D

Définition

- Ensemble d'éléments tous de même type
- Regroupés sous un même nom : le nom du tableau
- Chaque élément est repéré par un indice (entier)

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Tableaux 1D

Déclaration

Structure de la déclaration d'un tableau :

TYPE NOM [TAILLE];

Exemple : réserver l'emplacement pour un tableau de 10 éléments de type `int`, nommé `t` :

```
int t[10];
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Tableaux 1D

Remarques :

- **L'emplacement est fixé** à la compilation, définitif
- La taille doit être **connue** et **constante** : pas de variable !
- **Un tableau ne peut pas changer de taille**

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Tableaux 1D

Accès

On accède à l'élément numéro `i` d'un tableau `t` de cette façon :

```
t[i]
```

Cela permet de lire la valeur de l'élément ou d'y affecter une valeur comme on le ferait avec une variable :

```
t[3] = 42;
\\L'élément numéro 3 prend la valeur 42
cout << t[3] << endl;
\\Affichage de la valeur de l'élément numéro 3 de t
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Tableaux 1D

Attention ! ! !

- Les indices commencent à **ZERO** :

  `t[0], t[1],t[2], ..., t[9]`

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Tableaux 1D

Opérations sur un tableau

- Aucune opération possible sur la totalité du tableau
- Les opérations se font élément par élément, avec une boucle

En particulier :

- `cout << t` n'affiche pas le tableau, mais uniquement son adresse en mémoire
- `t1=t2` ne recopie pas le tableau `t2` dans le tableau `t1`
- `t1==t2` ne compare pas le tableau `t2` et le tableau `t1`, mais compare l'emplacement de `t1` et de `t2`

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Tableaux 1D

Initialisation

L'instruction suivante n'affecte aucune valeur particulière dans les cases du tableau : elles contiendront ce qui sera présent à cet instant dans la mémoire.

```
int t[10];
```

Outils
Entrées/sorties
Variables
Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels
Tests
if
if else
switch
Boucles
while
do ... while
for
Les chaînes de caractères
Tableaux 1D
Déclaration
Accès
Initialisation
Les fonctions
Définition
Utilisation
Retour
Variables

Tableaux 1D

Initialisation

Il est donc **nécessaire** d'initialiser les tableaux. Il existe plusieurs possibilités :

- Initialisation **totale** lors de la déclaration :

```
int t1[5] = {10, 20, 5, 0, 3};
\\place 10, 20, 5, 0, 3 dans les cinq éléments de t1
```

- Initialisation **partielle** lors de la déclaration :

```
int t2[5] = {10, 20}
\\place les valeurs 10 et 20 dans les deux premiers
\\éléments de t2
```

- Initialisation totale à l'aide d'une **boucle** (utile pour de grands tableaux) :

```
for(int i=0; i<5; i++){
   t[i] = 0;
}
```

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Pourquoi des fonctions ?

Objectifs

- Diviser le problème initial en plusieurs sous-problèmes plus simples
- Réutiliser le code existant
- Ne pas écrire plusieurs fois la même chose

Exemple :

Calcul du salaire moyen de chaque employé d'une société :

- Trouver ce que chacun gagne individuellement
- Compter le nombre de personnes
- Faire le total de tous les salaires
- Diviser ce total par le nombre de personnes.

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Les fonctions

Définition

Une fonction :

- est un **bloc d'instructions**
- peut éventuellement accepter des paramètres d'entrée appelés **arguments**
- peut fournir un résultat appelé **valeur de retour**.

Outils
Entrées/sorties
Variables
Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels
Tests
if
if else
switch
Boucles
while
do ... while
for
Les chaînes de caractères
Tableaux 1D
Déclaration
Accès
Initialisation
Les fonctions
Définition
Utilisation
Retour
Variables

Les fonctions

Une fonction est définie comme dans cet exemple :

```
float fexemple (float x, int b, int c) \\ en-tête
{
\\ corps de la fonction
}
```

Avec dans l'ordre :

- le type de la valeur de retour : ici `float`
- le nom de la fonction : ici `fexemple`
- le type et le nom des arguments : ici `float x, int b, int c`
- le corps contient les instructions que doit effectuer la fonction

Outils

Entrées/sorties

Variables

Opérateurs

Affectation

Opérateurs arithmétiques

Opérateurs relationnels

Tests

if

if else

switch

Boucles

while

do ... while

for

Les chaînes de caractères

Tableaux 1D

Déclaration

Accès

Initialisation

Les fonctions

Définition

Utilisation

Retour

Variables

Utilisation

```
fexemple (z, n, p);

fexemple (2.5, 4, 3);
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Utilisation

Remarque :

Si la fonction a été définie plus haut dans le fichier, la déclaration peut être omise car le compilateur connaît déjà cette fonction.

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Retour

Renvoyer une valeur

L'instruction `return` sert à :

- fournir une valeur de retour
- mettre fin à l'exécution de la fonction

Exemple :

```
return b; \\renvoie la valeur b et met fin à la fonction

return; \\ne renvoie rien mais met fin à la fonction
```

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Retour

Remarques

- `return` peut apparaître plusieurs fois dans une même fonction
- `return` peut mentionner une expression
- Si aucune instruction `return` n'est rencontrée, le retour est mis en place à la fin de la fonction
- Si la fonction ne renvoie rien, le type de la valeur de retour est `void`. Exemple :
  `void fSansValeurRetour(...)`

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Variables

Variable globales et locales

- **Variable globale :** variable déclarée en dehors de toute fonction (y compris du `main`)
- **Variable locale :** variable déclarée au sein d'une fonction.

Remarques :

- Une variable **globale** est connue dans **toute** la partie du programme **suivant** sa déclaration.
- Une variable **locale** est connue uniquement **dans la fonction** dans laquelle elle est déclarée.
- Il est possible de déclarer des variables **locales à un bloc** (une variable de boucle `for` par ex).

Outils

Entrées/sorties

Variables

Opérateurs
Affectation
Opérateurs arithmétiques
Opérateurs relationnels

Tests
if
if else
switch

Boucles
while
do ... while
for

Les chaînes de caractères

Tableaux 1D
Déclaration
Accès
Initialisation

Les fonctions
Définition
Utilisation
Retour
Variables

Variables

Attention !

- Eviter à tout prix d'utiliser des variables globales : il est très difficile de prédire l'état de cette variable étant donné qu'elle est accessible partout.