

## -- C++ CHEATSHEET --

### Headers

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

### Compilation/Execution

```
$ g++ test.cc -o test -Wall -std=c++11
$ ./test
```

### Commentaires

```
// ...      tout ce qui suit // sur cette ligne est un commentaire
/* ... */    tout ce qu'il y a entre /* et */ est un commentaire
```

### Fonctions

```
typeout funcName(typein_1 attr1, typein_2 attr2) {
    // ...
    return value;
}
```

### Types

char	un seul caractère (1 byte)
bool	valeur booléenne : <i>true</i> ou <i>false</i> (1 byte)
int	nombre entier positif ou négatif (4 bytes)
float	nombre réel (4 bytes)
double	nombre réel (8 bytes)
intX_t	nombre entier positif ou négatif (X bits, X={8,16,32,64})
uintX_t	nombre entier sans signe (X bits, X={8,16,32,64})

### Variables

int i(42); ou int i = 42;	déclaration
i = 5;	assignation
i	valeur
&i	adresse mémoire

### Pointeurs

int *p(0);	déclaration
p = &i;	assignation
int *p = new int(42);	déclaration + assignation
delete p;	destruction
p	valeur (adresse de la variable pointée)
&p	adresse mémoire
*p	valeur de la variable pointée

### Conditions

a==b	vrai si a égal b
a!=b	vrai si a différent de b
a>b	vrai si a strictement supérieur à b
a>=b	vrai si a supérieur ou égal à b
a % b	vrai si a proportionnel à b

### Opérateurs logiques

A    B	vrai si A ou B
A && B	vrai si A et B
A ^ B	vrai si A ou B mais pas les deux (xor)

### Entrée/Sortie

std::cout<< "a" ;	affiche a sur la sortie standard
std::cout<< a ;	affiche la valeur de a sur la sortie standard
<<std::endl ;	affiche un retour à la ligne sur la sortie standard

std::cin>>a;            enregistre l'entrée standard dans la variable a

## Structures de contrôle

<pre>if (cond1) {     // ... } else if(cond2) {     // ... } else {     // ... }</pre>	<pre>for (i=0; i&lt;size; i++) {     // ... }</pre>	<pre>while (condition) {     // ... }</pre>
--	---	---

## Opérateurs de boucle

**break;**            arrête la boucle courante (for ou while)  
**continue;**        saute à l'itération de boucle suivante (for ou while)

## Smart pointers

<code>#include &lt;memory&gt;</code>	include
<code>unique_ptr&lt;int&gt;ptr(new int(5)) ;</code>	déclaration (unique)
<code>shared_ptr&lt;int&gt;ptr1=make_shared&lt;int&gt;(5);</code>	déclaration (shared)
<code>shared_ptr&lt;int&gt;ptr2(ptr1) ;</code>	déclaration (shared)

## Tableaux

<code>int tab[5];</code>	tableau statique
<code>int *tab;</code>	tableau "dynamique", déclaration
<code>tab = new int[5];</code>	tableau "dynamique", allocation
<code>delete[] tab;</code>	tableau "dynamique", désallocation

## Vecteurs

<code>#include &lt;vector&gt;</code>	include
<code>vector&lt;int&gt; vec(5);</code>	déclaration
<code>vec.push_back(val);</code>	ajoute l'élément <code>val</code> au vecteur
<code>vec.pop_back();</code>	supprime le dernier élément du vecteur
<code>vec.size();</code>	taille du vecteur

## Itérateurs

<code>vector&lt;int&gt;::iterator it;</code>	déclaration d'un itérateur
<code>vec.begin();</code>	itérateur sur le premier élément de <code>vec</code>
<code>vec.end();</code>	itérateur sur le dernier élément de <code>vec</code>

## Itérations

<code>for(int *it=tab; it!=tab+5; it++)</code>	tableaux
<code>for(it=vec.begin(); it!=vec.end(); ++it)</code>	vecteurs (it prédéclaré)
<code>for(auto it=vec.begin(); it!=vec.end(); ++it)</code>	vecteurs (déclaration à la volée)

## Classes

<code>class ClassName</code>	définition
<code>{</code>	
<code>private:</code>	
<code>type1 attrib;</code>	attribut
<code>public:</code>	
<code>classname(...);</code>	constructeur
<code>~classname(...);</code>	destructeur
<code>type2 method(...);</code>	méthode
<code>};</code>	
<code>ClassName::ClassName(...) {}</code>	implémentation constructeur
<code>type2 ClassName::method(...) {}</code>	implémentation méthode

## Surcharge d'opérateur

`<typeout> operatorX(<type> const&, <type> const&);` surcharge de X pour les objets <type>