Cheatography

C++ Cheatsheet(inc. C++2011) Cheat Sheet

by Leupi via cheatography.com/5907/cs/1197/

Standard Sytemtypes and their range

int8_t	-128 to 127
uint8_t	0 to 255
int16_t	-32768 to 32767
uint16_t	0 to 65535
int32_t	-2147483648 to 2147483647
uint32_t	0 to 4294967295
int64_t	$-9.2 * 10^{18}$ to $9.2 * 10^{18}$
uint64_t	0 to $1.8 * 10^{19}$

Literals & Co.

255	Integer
0xaf	Hexadecimal Integer
1234.0	double
234.243f	float
true	bool
"Hello World"	string/c-string
'a'	char

Preprocessor

#include <LIB>	Includes standard header file LIB
#include "Header.h"	Includes Header.h file
#define PI 3.14159265359	Defines Easy Macro
#define f(a,b) a+b	More Complex Macro
#undef PI	Remove Definition
#ifdef PI	Checks for definition, if true, code will be compiled
#else	Else part of the above if-statement
#endif	Ends the if-statement

Nullpointer

```
int* pInt;
//old, deprecated:
if(pInt == NULL)
//new:
if(pInt == nullptr)
//still valid:
if(!pInt)
```

Bitwise Operators

&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Shift left
>>	Shift right

Boolean Logic

==	Test of equality
!=	Test of non-equality
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
!	NOT
&&	AND
||	OR

Boolean expressions in C++ are evaluated left-to-right!

Basic Operators

+	addition
-	subtraction
*	multiplication
/	division
%	modulo
++var	increase before evaluation
var++	increase after evaluation
condition ? result : alternative;	short form of if-like structure
::	Scope Operator
->	pointer-to-member
.	Access member
<< >>	Bitshift(with streams: input/output)

Pointers: Quick and dirty

&	Gets RAM adress of Variable(to save into pointer)
*	Dereferences pointer(returns it´s content) or defines a variable to be a pointer
->	Access pointer class member. same as (*pointer).member()
new	Create new object on heap, returns pointer to object
delete	Remove object at the pointer on heap

Pointers allocate space on heap, normal variables on stack!

using replaces typedef

```
//typedef is deprecated
typedef uint32_t uint32;
//now: using directive!
//using identifier = type_name;
using uint32 = uint32_t;
```

Auto Datatype

```
//auto is an automatic datatype:
int x = 4; //equals:
auto y = 4;
//works for most cases, esp. STL:
for(std::vector<int>::Iterator it = v.begin().....)
//with auto:
for(auto it = v.begin; it != v.end(); ++it)
```

Compile-time assertion check

```
static assert(bool_constexpr, string)
```

Multithreading

```
//Thread Creation & Management:
void thread_func(int a)
{
std::cout << "My Number is: " << a;
}
main()
{
std::thread t1(thread_func(1));
t1.join();
```

By **Leupi**
cheatography.com/leupi/

Published 18th August, 2013.
Last updated 18th August, 2013.

Multithreading (cont)

```
}
//Synchronization via:
std::mutex.lock();
std::mutex.unlock();
```

Range based loops

```
//Easy range based loop:
std::vector<int> vec = .....;
for(int i : vec) //foreach i in vec
//Same with auto:
const auto vi = ....;
for(auto i : vi)
```

Works with all Containers with begin() and end()

By **Leupi**
cheatography.com/leupi/

Published 18th August, 2013.
Last updated 18th August, 2013.