

C++ Cheat Sheet

Class: CSCI 1300

©Maximilien Notz 2025

General Reminders

#include "myfile.h"	Include file.
rand()	Random int, #include<cstdlib>.
int(var)	Convert a data type var to int.
float(var)	Convert a data type var to float.
double(var)	Convert a data type var to double.
static_cast<t>(var)	Convert var to the type t.
void myF() const	read only fonction.
inline	the whole code of the inline function is inserted or substituted at the point of its call during the compilation.
constexpr	that specify that an expression must be evaluated at compile time.
sizeof(var)	return the number of bytes used by the variable. sizeof runs at compile-time.
move(obj)	During assignation move the already existing object instead of creating a copy of it (memory optimizations).
ternary Operator	condition ? ifTrue : ifFalse

Strings

str[i]	Get or set the char at the index i.
.length()	Return the number of characters.
.substr(a,b)	Returns the substring starting at index a with length b.
.find(subStr)	Return the start index of the substring
.replace(i,l,str)	Replace l characters starting at index i with str
#include <string>	all the above must include string.
stoi(str)	Convert a string to int.
stof(str)	Convert a string to float,
stod(str)	Convert a string to double,
to_string(var)	Convert var to a string,

Arrays

0	1	...	n
"Max"	"Tom"	...	arr[n]

This table illustrate the structure of an array of strings. Considering that n is equal to the number of element minus one. Arrays are a static data type.

int arr[4];	Create a array of int and with 4 element.
int arr[4]={6,3};	
arr[i]	Get or set the element at the index i.

Vectors

#include<vector>	Include vector library.
vector<type> V;	Instantiate a vector.
vector<type> V(size);	Instantiate a vector from Array obj.
vector<type> V{6,3,3};	Instantiate a vector from Array.
V=vector<type>();	Re-instantiate V.
.at(i)	Returns the element at index i.
.size()	Return the number of elements.
.push_back(Value)	Add the new element at the end.
.pop_back()	Remove the last element.
.clear()	Empty the vector.
.insert(i, Value)	Insert Value at i.

Structures

```
struct myStruct {  
    string param1; // attribute 1  
    double param2; // attribute 2  
}s1, s2; // myStruct instances
```

myStruct Obj; instantiate structure object.
Obj.param1 Access param1 of Obj.

Streams

```
#include<fstream>  
#include<sstream>  
ifstream fin;  
ofstream fout;  
stringstream s(str);  
myS.open("file.txt")  
myS.close()  
getline(fin, line)  
fout<<"hello"  
fin>>var  
<<setprecision(n)<<  
<<setw(n)<<  
<<fixed<<  
  
<<showpoint<<  
<<noshowpoint<<  
  
<<left<<  
<<right<<
```

clear buffer

The buffer must be cleared after after getting an input from a stream if you input and output in the same file at the same time.

```
if(cin.fail() == true) {  
    cin.clear();  
    cin.ignore(1000, '\n');  
}
```

cmath

```
#include<cmath>  
sqrt(x)  
pow(x, y)  
abs(x)  
floor(x)  
ceil(x)  
fmod(x, y)
```

Include cmath library.
Square root of x.
x raised to the power y.
Absolute value overloads.
Greatest integer \leq x.
Smallest integer \geq x.
Floating-point remainder of x/y.

Error Handling

```
try {  
    // risky operation  
} catch (exceptions) {  
    // runs if an exception of type Ex is thrown  
}
```

```

#include<cassert>      Include assert library.
#include<stdexcept>    Common standard exceptions.
throw myException       Throw an error of type myException.
exception::what()      Retrieve diagnostic message.
catch (const auto& e)   Catch exceptions by const reference.
catch(...)              Fallback handler; rethrow if unsure.
exception               Parent of all exceptions class.

```

Object Oriented Programming(OOP)

```

class myClasses :public parentClass{
    private:
        // private methods and variables
    public:
        // public methods and variables

    myClasses(int p1, int p2){...} // Constructor

    ~myClasses(){...} // Destructor

    // Override the inherited method parentMethod()
    void parentMethod() override { ... }

    //Example Operator Overloading
    Number operator+(const Number &n){
        return Number(value + n.value);
    }
};

myClasses myObj(3,5);  Instantiate an myClasses type obj.
myClasses myObj;       Call the default constructor.
protected:             similar to private, but it can also be
                        accessed in the inherited class.

virtual               Specify that a method can be overridden
                        in a derived class.

```

OOP With header file

If you use a header the file which contain the main function must include the header file.

Header file(myHeader.h)

```

#ifndef MYCLASS_H //if no def for MyClass
#define MYCLASS_H //else

using namespace std;

class MyClass{
    public:
        :
    private:
        :
};

#endif

```

Class file(.cpp)

```

#include "myHeader.h"

MyClass::MyClass(int p1, ...){
    publicAttribute = p1;
    :
}

```

Genericity

```

template <typename T_1, ..., typename T_n>
class myClasse{
    :
myClasse<T_1, ..., T_n>(...);

```

Switch case

```

switch (x){
    case 0:
        /*Code in case x = 0*/
        break;
    :
    case n:
        /*Code in case x = n*/
        break;
    default:
        /*Code if no case match*/
}

```

Pointer & References

int* myInt;	* means myInt work form a pointer.
new	dynamically allocate a block of memory.
delete	release dynamically allocated memory.
NULL	Macro that refersens to null pointer.
*var	Get var value, where var is a pointer.
&var	Get memory addresse of var.
void* var	Pointer with no associated data type.

Lambda Expression

```

... = [captureClause] (parameters) -> returnType {
    // definition
}

```

[&]	capture all external variables by reference.
[=]	capture all external variables by value.
[a, &b]	capture 'a' by value and 'b' by reference.

Namespaces

namespace NS { ... }	Define a namespace.
NS::func()	Access member of namespace.
using namespace NS;	Import all names from namespace.
using NS::func;	Import specific name from NS.
namespace {...}	Anonymous namespace: limits scope to current translation unit (file).
inline namespace NS { ... }	Members are accessible without qualification by default.
namespace alias = NS;	Create an alias for a namespace.
::globalVar	Access global namespace explicitly.

Special Ints

signed fixed width integer types

int8_t	int16_t	int32_t	int64_t
int_fast8_t	int_fast16_t	int_fast32_t	int_fast64_t
int_least8_t	int_least16_t	int_least32_t	int_least64_t

unsigned fixed width integer types

uint8_t	uint16_t	uint32_t	uint64_t
uint_fast8_t	uint_fast16_t	uint_fast32_t	uint_fast64_t
uint_least8_t	uint_least16_t	uint_least32_t	uint_least64_t

other integer types

`intmax_t & uintmax_t`

`intptr_t & uintptr_t`

`size_t`

Maximum-width integer type.

Integer types capable of storing a pointer value.

An unsigned integer data type to represent the size of objects in bytes; commonly used for array indexing and loop counters.

Preprocessing

`#define NAME value`

`#define F(x) x*x`

`#ifdef NAME`

`#ifndef NAME`

`#else`

`#endif`

`#include`

`_FILE_`

`_LINE_`

`_DATE_`

`_TIME_`

`#pragma`

- `once` — simple include guard for header files.
- `pack(push, n)` — set and restore struct packing/alignment to n bytes.
- `pack(n)` — set struct member alignment to n.
- `GCC optimize("...")` — enable compiler-specific opti-

Define a macro.

Define a function-like macro.

If the macro NAME is defined.

If the macro NAME is not defined.

Alternative case for ifdef/ifndef.

End conditional directive.

Include a file.

Current file name.

Current line number.

Compilation date.

Compilation time.

Implementation-specific instruction.

- `push/pop` — control MSVC warnings.
- `#pragma message("text")` — emit a compile-time message.
- `#pragma comment(lib, "name.lib")` — instruct MSVC linker to link a library.
- `#pragma omp parallel` — OpenMP parallel region (requires OpenMP enabled by compiler flags).
- `#pragma intrinsic(func)` — request compiler intrinsic (MSVC).

mizations (GCC/Clang). • `#pragma warning(push) / #pragma warning(pop) / #pragma warning(disable:NNNN)` — control MSVC warnings.

- `#pragma message("text")` — emit a compile-time message.
- `#pragma comment(lib, "name.lib")` — instruct MSVC linker to link a library.
- `#pragma omp parallel` — OpenMP parallel region (requires OpenMP enabled by compiler flags).
- `#pragma intrinsic(func)` — request compiler intrinsic (MSVC).

Compiler Commands

`clang++ fileName`

`-o name`

`-v`

`-E`

`-Wall`

`-Wextra`

`-c fileName`

`-g`

`-fline-tables-only`

`-O0, -O1,`

`-O2, -O3,`

`-Ofast`

command to compile c++ code with **clang**, clang is a LLVM compiler.

define the name of the compiled object.

Makes the compiler print detailed information. "v" stands for "Verbose".

Prints the preprocessor output.

activates all warnings

Enable extra warnings beyond -Wall.

generate an object file. To add .o file to the compilation simply add those like a regular file.

Include debug symbols for gdb/lldb.

Minimal debug info for profilers.

Optimizations levels, where -O0 is no optimization