

General Reminders

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

Strings

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

Vectors

<code>#include<vector></code>	Include vector library.
<code>vector<type> V;</code>	Instantiate a vector.
<code>vector<type> V(s);</code>	Instantiate a vector of size s.
<code>vector<type> V{6,3,3};</code>	Instantiate a vector from Array.
<code>vector<type> V(s, var);</code>	Instantiate a vector of size s with all elements initialized to var.
<code>.at(i)</code>	Returns the element at index i.
<code>.size()</code>	Return the number of elements.
<code>.push_back(Value)</code>	Add the new element at the end.
<code>.pop_back()</code>	Remove the last element.
<code>.clear()</code>	Empty the vector.
<code>.insert(i, Value)</code>	Insert Value at i.
<code>.reserve(int)</code>	Pre-allocate memory for perf. when the max size is known.

Arrays

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

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

Structures

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

<code>myStruct Obj;</code>	instantiate structure object.
<code>Obj.param1</code>	Access param1 of Obj.

Streams

<code>#include<fstream></code>	Include stream library.
<code>#include<sstream></code>	Include string stream library.
<code>ifstream fin;</code>	Instantiate a input stream.
<code>ofstream fout;</code>	Instantiate a output stream.
<code>stringstream s(str);</code>	Instantiate a string stream.
<code>myS.open("file.txt")</code>	Open txt file whith the stream.
<code>myS.close()</code>	Close the stream file.
<code>getline(fin, line)</code>	Get the next line from fin.
<code>fout<<"hello"</code>	Output in stream "helloWorld".
<code>fin>>var</code>	Input from stream to var.
<code><<setprecision(n)<<</code>	Set decimal points, <code>#include<iomanip></code> .
<code><<setw(n)<<</code>	Establishes a print field of n spaces.
<code><<fixed<<</code>	Display floating point numbers in fixed.
	point notation.
<code><<showpoint<<</code>	Enables or disables the unconditional
<code><<noshowpoint<<</code>	inclusion of the decimal point character
	in floating-point output.
<code><<left<<</code>	output the string on the left.
<code><<right<<</code>	output the string on the 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

<code>#include<cmath></code>	Include cmath library.
<code>sqrt(x)</code>	Square root of x.
<code>pow(x, y)</code>	x raised to the power y.
<code>abs(x)</code>	Absolute value overloads.
<code>floor(x)</code>	Greatest integer $\leq x$.
<code>ceil(x)</code>	Smallest integer $\geq x$.
<code>fmod(x, y)</code>	Floating-point remainder of x/y.

Error Handling

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

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

Object Oriented Programing(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);
    }
};
```

<code>myClasses myObj(3,5);</code>	Instantiate an myClasses type obj.
<code>myClasses myObj;</code>	Call the default constructor.
<code>protected:</code>	similar to private, but it can also be accessed in the inherited class.
<code>virtual</code>	Specify that a method can be overridden in a derived class.

OOP With header file

If you use a header the file wich 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

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

Bitwise Operators

<code>&</code>	Bitwise AND.	<code>~</code>	Bitwise NOT.	<code>^</code>	Bitwise XOR.
<code> </code>	Bitwise OR.	<code><<</code>	Left shift.	<code>>></code>	Right shift.

Namespaces

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

Lambda Expression

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

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

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	Maximum-width integer type.
intptr_t & uintptr_t	Integer types capable of storing a pointer value.
size_t	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 a macro.
#define F(x) x*x	Define a function-like macro.
#ifdef NAME	If the macro NAME is defined.
#ifndef NAME	If the macro NAME is not defined.
#else	Alternative case for ifdef/ifndef.
#endif	End conditional directive.
#include	Include a file.
__FILE__	Current file name.
__LINE__	Current line number.
__DATE__	Compilation date.
__TIME__	Compilation time.
#pragma	Implementation-specific instruction.
• once	— simple include guard for header files.
• pack(push, n)	

/ #pragma pack(pop) — set and restore struct packing/alignment to n bytes.

- pack(n) — set struct member alignment to n.
- GCC optimize("...") — enable compiler-specific optimizations (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.

Compiler Commends

clang++ fileName	commend to compile c++ code with clang , clang is a LLVM compiler.
-o name	define the name of the compiled object.
-v	Makes the compiler print detailed information. "v" stands for "Verbose".
-E	Prints the preprocessor output.
-Wall	activates all warnings
-Wextra	Enable extra warnings beyond -Wall.
-c fileName	generate an object file. To add .o filse to the compilation simply add those like a regular file.
-O0, -O1, -O2, -O3, -Ofast	Optimizations levels, where -O0 is not optimization