

## General Reminders

<code>#include "myfile.h"</code>	Include file.
<code>rand()</code>	Random int, <code>#include&lt;cstdlib&gt;</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&lt;t&gt;(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>sizeof(var)</code>	return the number of bytes used by the variable. <code>sizeof</code> runs at compile-time.
<code>ternary Operator</code>	condition ? ifTrue : ifFalse

## Strings

<code>str[i]</code>	Get or set the char at the index i.
<code>str.length()</code>	Return the number of characters.
<code>str.substr(a,b)</code>	Returns the substring from a to b.
<code>str.find(subStr)</code>	Return the start index of the substring
<code>str.replace(i,1,str)</code>	Replace substring from i to 1 with str
<code>stoi(str)</code>	Convert a string to int, <code>#include&lt;string&gt;</code> .
<code>to_string(var)</code>	Convert var to a string, <code>#include&lt;string&gt;</code> .

## 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.

<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.

## Vectors

<code>#include&lt;vector&gt;</code>	Include vector library.
<code>vector&lt;type&gt; V;</code>	Instantiate a vector.
<code>vector&lt;type&gt; V(size);</code>	Instantiate a vector from Array obj.
<code>vector&lt;type&gt; V{6,3,3};</code>	Instantiate a vector from Array.
<code>V=vector&lt;type&gt;();</code>	Re-instantiate V.
<code>V.at(i)</code>	Returns the element at index i.
<code>V.size()</code>	Return the number of elements.
<code>V.push_back(Value)</code>	Add the new element at the end.
<code>V.pop_back()</code>	Remove the last element.
<code>V.clear()</code>	Empty the vector.
<code>V.insert(i, Value)</code>	Insert Value at 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&lt;fstream&gt;</code>	Include stream library.
<code>#include&lt;sstream&gt;</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 with the stream.
<code>myS.close()</code>	Close the stream file.
<code>getline(fin, line)</code>	Get the next line from fin.
<code>fout&lt;&lt;"hello"</code>	Output in stream "helloWorld".
<code>fin&gt;&gt;var</code>	Input from stream to var.
<code>&lt;&lt;setprecision(n)&lt;&lt;</code>	Set decimal points, <code>#include&lt;iomanip&gt;</code> .
<code>&lt;&lt;setw(n)&lt;&lt;</code>	Establishes a print field of n spaces.
<code>&lt;&lt;fixed&lt;&lt;</code>	Display floating point numbers in fixed point notation.
<code>&lt;&lt;showpoint&lt;&lt;</code>	Enables or disables the unconditional inclusion of the decimal point character in floating-point output.
<code>&lt;&lt;noshowpoint&lt;&lt;</code>	
<code>&lt;&lt;left&lt;&lt;</code>	output the string on the left.
<code>&lt;&lt;right&lt;&lt;</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) {
    cout << "cin failed state";
    cin.clear();
    cin.ignore(1000, '\n');
}
```

## cmath

<code>#include&lt;cmath&gt;</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&lt;cassert&gt;</code>	Include assert library.
<code>#include&lt;stdexcept&gt;</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&amp; 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){
        // Body of constructor
    }

    ~myClasses(){
        // Body of destructor
    }
};
```

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.

## 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:
    MyClass(); //default constructor
    MyClass(p1, p2); //parameterized constructor
    int publicAttribute;
    void myFunction() const;
private:
    int privAttribute;
};
#endif
```

### Class file(.cpp)

```
#include <iostream>
#include "myHeader.h"

MyClass::MyClass(){
    publicAttribute = 0;
    privAttribute = 0;
}

MyClass::MyClass(int p1, int p2){
    publicAttribute = p1;
    privAttribute = p2;
}
```

```
}

void MyClass::myFunction() const{
    // my code
}
```

## Switch case

```
switch (x){
    case 0:
        /*Code in case 0*/
        break;
    :
    case n:
        /*Code in case 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 referens 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.

## Important ASCII Conversions

ASCII	int	ASCII	int	ASCII	int	ASCII	int	ASCII	int
A	65	a	97	N	78	n	110	0	48
B	66	b	98	O	79	o	111	1	49
C	67	c	99	P	80	p	112	2	50
D	68	d	100	Q	81	q	113	3	51
E	69	e	101	R	82	r	114	4	52
F	70	f	102	S	83	s	115	5	53
G	71	g	103	T	84	t	116	6	54
H	72	h	104	U	85	u	117	7	55
I	73	i	105	V	86	v	118	8	56
J	74	j	106	W	87	w	119	9	57
K	75	k	107	X	88	x	120		
L	76	l	108	Y	89	y	121		
M	77	m	109	Z	90	z	123		