**Cheatsheet OOP244**

**Namespaces** is a declarative region that provides a scope to the identifiers.

a::x++ b:x-- OR using a:xx X++ OR using namespace a x++

**OOP Pillars**: *Encapsulation*: Principal concept of OOP, placing data and functions that work with data on the same place

*Inheritance*:One class inherit from another class, reduce code size

*Polymorphism*: Single interface provides multiple implementations, operator that works in many different ways – *Overloading* is a branch of it

**Stages of compilation**: Pre-processing (put together all #include and #define), Compiling (compiles separetly) and Linkage(put togethter everything in .exe)

**Order**: #include <>, #include “…”, namespace, directives

**Guards** -> #ifndef \_name\_ and #define\_name\_and #endif run until the endif one time

**Safeguard** -> #pragma once makes sure the directive is include only 1 time ->less code + speed ,a void name problems

**Dynamic Memory**: using the “stack” memory, overwritten it as functions run, allows to save memory at the beginning of the program and clear it after

**Safe State**: Initialize the vars of an object inside a constructor so is well defined.

*Function overloading and constructor overloading*

#include <cstdio> use printf()

#include <iostream> use (cin) or (cout,cerr(unbuffered),clog(buffered)) >> <<

*Block scope and global scope*

**By Reference -> url, alias of the variable, for operator overloads and copy constr**

Void swap (char&a, char &b){

char c; c=a; a=b; b=c;} ->Compiler coverts everything to a pointer with fix address

**By address ->**

**BY value -> paper**

Void Student::display() const{} -> possible to call another function inside display() or global functiobn using ::display()

Class -> private Struct->public auto + 4 , set int and start at 4

Set(int n, const char\*g)

If-validate else-runs code(for(i=0;g[i]!0)(if-accepts client data and else initiliaze))

and Display()

Empty State – validate data if not valid declare empty state – data is either data or empty state

Member functions -> Width(push to the right),fill,setf,unsetf,precision

<iomanip> for manipulators

Cout<<fixed<<left<setw(5)<<setprecision(1)<<12.376=12.4 setw pushes to the right

<istream> supports ignore(),ignore(200,\n”),get(u,8,\t) – 3 ways ,getline(as get but takes the delimiter too no need to use cin.ignore())

Skipws, noskipws, setw(int)(1 more)

State -> good() fail() eof() bad() clear()

**Dynamic Memory** load time -> static memory, during execution -> dynamic, lifetime ends when the pointers goes out of scope

**Dynamic Allocation Deallocation**

Pointer=new Type[size] delete[]pointer

Int n; delete[]cpa

Student \*cpa =nullptr cpa=nullptr;

//code

Cpa = new Student[n]

**Memory Issues:**

**Memory Leak: 1.** Pointer goes before out of scope before deallocating **2**:changes it’s values before the app has deallocated the memory starting at that value \*No halt execution

**Insufficient Memory**> System resources

**+**(destination,source,size) last size = ‘\0’

**Constructor**

Fundamental type -> int, double, char, etc

Compound type -> class, pointer, reference

Student::Student(){} - default

Student::Student(){no=0; grade[0] = \0“} -initialize

**Destructor ~Type() Student(); ~Student();**

\*this -> refer to current object

This => has no meaning outside a member function