Name: Kormoua Khongmeng
Neptun Code: I3MLPQ

# Basic of Programming 2

Laboratory report 7

**Task 1.**

b. Examine the *operator=* and answer the following questions:

- Under what constraints can we call *operator=* from a copy constructor?

  **Ans:** we can call *operator=* at the copy constructor when we have a prototype of this operator overloading "*operator=*" at the header file, so the complier knows the existing of it. and later this operator overloading "*operator=*" for this vector must be implemented completely in the C++ source file too.

- Why are both parameter and returning value passed by reference?

  **Ans:** For the parameter, we passed a reference because it will make more sense to pass a reference of the other object that we want to copy, instead of passing the whole other object itself. Since passing by reference will not make any copy of the other object at all, so it will make the code more efficient as well.

  For the returning value, we have a reference return type because: first of all, this is an assignment operation, so it would be better if we can do a chaining assignment like v1 = v2 = v3. And this is possible when we return a reference instead of a value.

- What happens with self-assignment like *v=v*?

  **Ans:** In my code, I already added a condition to check for self-assignment, what I did there is when I got a self-assignment, I just simply return this vector back. So, in my solution, when we have a self-assignment like v = v, Vector v will remain unchanged. I already added a test case in the test file as well.

c. What modifications need to be done at *Vector* class to be able to manipulate user-defined data types instead of a built-in *int* type (e.g. to have a vector of *String* objects)?

**Ans:** If we want to manipulate user-defined data type instead of a built-in int type. We actually need to modify the type of "pData". So, it really depends on what kind of data type that we want:

if we want just a small modification like we want a vector of double instead of int, then we don't have so many things to modify. The main thing is to modify "int* pData" to "double* pData".

but if we want to have a vector of an object like Triangle (assume that we already have class Triangle) then what we should do in the vector class, first is to modify from "int* pData" to "Triangle* pData". In this way we can have a vector of a Triangle. But we still have to add some modification for its methods to be able to work with these triangles as well like: insert(), empty(), at(), etc. Which we may have a really big modification, sometimes we may have to reimplement a lot of methods again. But this really depends on what do we want our vector to be and how do we want it to work.

**Homework. (optional)**

Add a sorting function to the *Vector* class. Call it from *VectorSample.cpp* and make sure that everything works properly.

**Ans:** for this problem I am not sure if I need to implement everything from scratch or not. But I found that std::sort of C++ is considerably very fast and very simple, so I just used it in my code for simplicity.