Assignment 5/25/2021

Daniel Way

C042021 – C#/.NET

Summary of .NET Architecture & History

.NET originally shipped in 2002 but had been created around 1997. The .NET architecture didn’t really start out as just that, it kind of evolved from a collaboration of needed patches. People began to find each other if you will to solve various problems. With each new version of Windows brought about the Common Runtime Language (CLR). Prior to C#, Microsoft didn’t really have their own language. They just used other languages in their software that they distributed

Patrick Dussud was an advocate for Garbage Collection for the CLR. This allowed developers to not worry about managing memory and focus on the business aspect of what they were writing. .NET basically replaced .COM objects that programs had to have to run on their infrastructure. And with each new version of Windows, this framework grew to what it is today, and growing.

When .NET, as a tool, has developed 22 different software languages to include the likes of COBOL.NET, J++, F# etc… Today many of those languages have gone away and yet others have been created based the .NET architecture.

Unsafe code and pointers

Safe code doesn’t access memory or allocate raw memory. Instead it creates managed objects. Unsafe code is basically code that the .NET framework cannot verify as safe. To be considered “*unsafe code”* it must have the following properties:

Methods, types, and code blocks can be defined as unsafe.

I may increase the performance of an application by removing by removing array bound checks.

It is required when you call native functions that require pointers.

It introduces stability and security risks.

Unsafe code must be compiled with the AllowUnsafeBlocks compiler object.

Pointers are variables that point at other variables. It is the starting address (memory) of another variable. A pointer allows indirect access to a variable thus enabling the passing of variables between functions efficiently. The values of the variable are then copied to the parameters of the function. The \* is used to declare pointer if used with type, and the same symbol is also used to de-reference the pointer. Pointers can be added or subtracted, provided declared type is defined. Pointers allow creating dynamic arrays. The -> operator is used to access type member: field or method. A Null Pointer does not point to anything, and a Dangling pointer stores an invalid address.

Below are some examples of the use of pointers:

Example 1.

int a = 2;

int \*pointerToA = &a;

Example 2.

int \*pointerToB;

int b = 3;

pointerToB = &b;

Example 3.

Int \*pointerToArray;

Int array[] = {1, 2, 3};

poitnerToArray = array;