The *unsafe* keyword in C# is required whenever operations are done with pointers. The scope of *unsafe* is until the end of the method.

Also, to compile “unsafe” code, you must include the */unsafe* option. If not included, the unsafe code cannot be read by a regular language runtime.

An example of using unsafe for a linked list to have a sorted list, and being able to insert values in a sorted list more easily:

unsafe public static void Main()

{

Node\* myList = new Node();

myList->value = 10;

addNode(myList, 5);

addNode(myList, 3);

addNode(myList, 12);

}

unsafe struct Node

{

int value;

Node\* next;

Node\* prev;

}

unsafe void addNode(Node\* a, int b)

{

// if b is lower, then add the new node

if (b < a->value){

Node\* temp = new Node();

temp->value = b;

temp->next = a;

temp->prev = a->prev;

if (a->prev != null)

a->prev->next = temp;

a->prev = temp;

} else if ((b > a->value) && (a->next != null)){

addNode(a->next, b);

} else {

Node\* temp = new Node();

temp->value = b;

temp->next = null;

a->next = temp;

temp->prev = a;

}

}

While adding more integers to this linked list, we do not have to shift values (as if they were in an array). This saves time while we add more and more elements to the list.

An advantage of this facet is that it will prevent anyone from accidently modifying pointers and possibly changing memory.

A disadvantage of this is that this makes it difficult to easily work with pointers like in C or C++. Also, this also means that whenever a programmer wants to use pointers in his/her code, he/she must make sure it is compiled correctly and all the methods have the *unsafe* keyword.

Something like this would be difficult to include into Java or ML because the programmer has no access to the pointers. In Java, all of the pointers are done below what the programmer can access.

Personally, I like this form of syntax. As stated before, by needing explicitly declaring something as *unsafe*, this makes it impossible to accidentally modify pointers and memory addresses. I’m unsure of how this facet could be improved, because it’s much like a *cast*, where the coder is telling the compiler that he/she knows what he/she is doing, and thus allowing for “unsafe code”.

http://msdn.microsoft.com/en-us/library/chfa2zb8(v=vs.71).aspx

http://www.codeproject.com/KB/cs/unsafe.aspx