Subject: Re: Basic questions about u++
Posted by mrjt on Wed, 07 Oct 2009 16:14:33 GMT
View Forum Message <> Reply to Message

irtech wrote on Wed, 07 October 2009 15:18I'm confused
Mirek said
If the stack frame goes out of scope, object is destryoed via destructor.

Implying it is done automatically.

but mr_ped said
Resources works like in ordinary C++,...If you use some "new", then it's up to you to "delete" of course.

The idea of U++ is to utilize the standard behaviour of C++ to avoid most of the memory management problems that normally plague C++ code. This is done in two main ways:
- Structuring the library so that every object is 'owned' by something. This means declaring objects either on the stack (as local variables) or as member variables. This allows the compiler to clean up by itself when either the owning object is destroyed or the variable goes out of scope.
- Providing the tools (mainly the NTL) necessary to control dynamic memory in a way that fits with the 'everything is owned' style. This means that although new and delete are used internally in the library you rarely, if ever, need them in your application code.

A simple example:
'ordinary' C++ code:
```
int CalcSomething {
    int *array = new int[256]

    // Do a load of stuff

    delete[] array;
};
```

U++:
```
int CalcSomething {
    Buffer<int> array(256); // Buffer is closest to a C array

    // Do a load of stuff
};
```

By wrapping the new/delete calls in an object (one that has already been thoughly tested) you are able to utilise C++ inbuilt destruction mechanics and avoid the error-prone call to delete. There isn't any 'magic' going on outside of using templates in a clever way

As far as I'm concerned the Upp memory management philosophy is to never use new/delete. There is almost always a better way .