
Subject: Re: Initialization for Buffer<T>

Posted by [mirek](#) on Sun, 30 Aug 2015 05:38:19 GMT

[View Forum Message](#) <> [Reply to Message](#)

Mindtraveller wrote on Sat, 29 August 2015 18:41 Buffer<T> is known as replacement for C-style arrays. So I guess it needs some way to fill it as easy as possible (like we initialize arrays in C). So here is my proposal with little additions:

```
template <class T>
```

```
class Buffer : Moveable< Buffer<T> > {
```

```
    size_t size;
```

```
    mutable T *ptr;
```

```
    int insertl;
```

```
public:
```

```
    operator T*()          { return ptr; }
```

```
    operator const T*() const { return ptr; }
```

```
    T *operator~()         { return ptr; }
```

```
    const T *operator~() const { return ptr; }
```

```
    Buffer<T> & Alloc(size_t _size) { Clear(); ptr = new T[_size]; size = _size; return *this; }
```

```
    Buffer<T> & Alloc(size_t _size, const T& in) { Clear(); ptr = new T[_size]; size = _size; Fill(ptr, ptr + size, in); return *this; }
```

```
    void Clear() { if(ptr) delete[] ptr; ptr = NULL; insertl = 0; size = 0; }
```

```
    size_t GetCount() { return size; }
```

```
    Buffer() { Init(); ptr = NULL; }
```

```
    Buffer(size_t size) { Init(); ptr = new T[size]; }
```

```
    Buffer(size_t size, const T& init) { Init(); ptr = new T[size]; Fill(ptr, ptr + size, init); }
```

```
    ~Buffer() { if(ptr) delete[] ptr; }
```

```
    void Init() { size = 0; insertl = 0; }
```

```
    void operator=(Buffer rval_v) { if(ptr) delete[] ptr; ptr = v.ptr; v.ptr = NULL; }
```

```
    Buffer<T> & operator<< (const T &in){ ptr[insertl++] = in; return *this; }
```

```
    Buffer(Buffer rval_v) { ptr = v.ptr; v.ptr = NULL; size = v.size; v.size=0; insertl = 0; }  
};
```

```
// Usage:
```

```
// buffer.Alloc(3) << v1 << v2 << v3;
```

Adding 2 new member variable seems quite wasteful.

However, I am now in process of adding C++11 `std::initializer_list` to all containers. Without your post, I would probably forgot about Buffer. With C++11, we can now have

```
Buffer<int> x{ v1, v2, v3 };
```

```
x = { v4, v5, v6 };
```
