

Hi-

I think this SimpleVector class is moveable because:

- 1) No references or pointers are stored to the object itself or to subobjects in the methods of the type, into variables that exist after the method finishes execution.
- 2) The types that can be moved in memory using memcpy are called moveable.

I think that is good enough. It was very hard for me to try to break the assertion but it is good practice to try out memcpy.

I never did end up breaking that assertion. I took out the Vector from the members.

thanks - jim

```
#include <Core/Core.h>
#include<iostream>
#include<cstring>
```

```
using namespace Upp;
using namespace std;
```

```
class SimpleVector: Moveable<SimpleVector>{
    int len;
    char * iptr;
    int a;
    int * ptr;

public:
    SimpleVector(const char *iptr2);

    SimpleVector& operator=( const SimpleVector& rhs );
    SimpleVector(const SimpleVector & s1);
    ~SimpleVector();
    char* GetString();
} sv1("");
```

```
SimpleVector::SimpleVector(const char *iptr2){
    std::cout<<"Entering constructor";
    std::cout<<endl;
```

```
    len = strlen(iptr2) + 1;
    iptr = new char[len + 1];
    strncpy(iptr, iptr2, len);
    iptr[len-1]='\0';
```

```

a = 10;
ptr = new int(2);

std::cout<<"leaving constructor";
}
SimpleVector& SimpleVector::operator=( const SimpleVector& rhs ){

    Cout()<<"entering assignment";
    std::cout<<endl;

    if (this == &rhs)
        return *this;

    std::cout<<"debug test"<<endl;

    delete [] iptr;
    iptr=0;
    len = rhs.len;

    delete ptr;
    ptr=0;

    if(rhs.iptr){
        iptr = new char[len + 1];
        strncpy(iptr,rhs.iptr,len);
        a = rhs.a;
        Cout()<<*iptr;
        std::cout<<endl;
    }else
        iptr=0;

    if(rhs.ptr){
        ptr = new int(2);
        memcpy(ptr,rhs.ptr,sizeof(int));
    }else
        ptr=0;

    return *this;
}

SimpleVector::~SimpleVector(){
    Cout()<< "the new soundgarden album is out!";
    std::cout<<endl;
    delete[] iptr;
    iptr = 0;

    delete ptr;

```

```
ptr = 0;
std::cout<<"debug test"<<endl;
AssertMoveable<SimpleVector>();
}
```

```
CONSOLE_APP_MAIN
{
    std::cout<<endl;
    std::cout<<"program begin!";
    cout<<endl;
    SimpleVector s("foo1");
    SimpleVector s2("foo2");
    // SimpleVector s2 = s; copy constructor
    s2=s;
}
```
