

ENCM Fall 2005: Handout for L02 (Dr. Norman's lecture) Fri., Oct. 28 and Mon., Oct. 31

Author: Dr. S. A. Norman. Electronic copies of handouts for L02 and T02 can be found at <http://www.enel.ucalgary.ca/People/Norman/encm339fall2005/>

Memory management policy for `StringOne` class objects:

The `chars` belonging to a `StringOne` class object are stored in a built-in string in a dynamically allocated array. (Built-in string: sequence of `chars` terminated with `'\0'`.)

The array is *exactly* the right size to hold the `chars` up to and including the `'\0'`.

File with `StringOne` class definition:

```
// StringOne.h
#ifndef STRING_ONE_H
#define STRING_ONE_H
// WARNING: The StringOne class has a major defect. If you
// try to copy a StringOne object, bad things will happen.

class StringOne {
public:
    StringOne();
    // PROMISES: Empty string object is created.

    StringOne(const char *s);
    // PROMISES: s points to first char of a built-in string.
    // REQUIRES: String object is created by copying chars from s.

    ~StringOne();

    int length() const;
    // PROMISES: Return value is number of chars in string.

    char get_char(int pos) const;
    // REQUIRES: pos >= 0 && pos < length()
    // PROMISES:
    // Return value is char at position pos.
    // (The first char in the string is at position 0.)

    const char * c_str() const;
    // PROMISES:
    // Return value points to first char in built-in string
    // containing the chars of the string object.

    void set_char(int pos, char c);
    // REQUIRES: pos >= 0 && pos < length(), c != '\0'
    // PROMISES: Character at position pos is set equal to c.

    void append(const StringOne& tail);
    // PROMISES: chars are copied from tail to the end of the
    // string object.

    void truncate(int new_length);
    // REQUIRES: new_length >= 0 && new_length <= length()
    // PROMISES: Length of string is reduced to new_length.

private:
    int lengthM;
    char *storageM;
};
#endif
```

File with most of the StringOne member function definitions:

```
// StringOne.cpp
#include <cassert>
#include <cstring>
using namespace::std;

#include "StringOne.h"

StringOne::StringOne()
: lengthM(0), storageM(new char[1])
{
    storageM[0] = '\0';
}

StringOne::StringOne(const char *s)
: lengthM(strlen(s))
{
    storageM = new char[lengthM + 1];
    strcpy(storageM, s);
}

StringOne::~StringOne()
{
    delete [] storageM;
}

int StringOne::length() const
{
    return lengthM;
}

char StringOne::get_char(int pos) const
{
    assert(pos >= 0 && pos < length());
    return storageM[pos];
}

const char * StringOne::c_str() const
{
    return storageM;
}

void StringOne::set_char(int pos, char c)
{
    assert(pos >= 0 && pos < length());
    assert(c != '\0');
    storageM[pos] = c;
}

// Definitions for append and truncate omitted.
```