Doc. No.: X3J16/95-0167
WG21/ N0767
Date: September 20, 1995
Project: Programming Language C++
Reply To: Uwe Steinmüller
Siemens AG
Uwe.Steinmueller@zfe.siemens.de

Cleaning up signatures for the string class

The Problem

The current signatures have been revised several times for many different reasons (DynArray, STL, ...). Especially with the use of default arguments there are now signatures which can lead to confusion by the users. Code which looks very similar may behave very different.

Here are examples how these signatures may confuse the users:

basic_string& append(const basic_string& str, size_type pos = 0, size_type n = npos);
basic_string& append(const charT* s, size_type n);

string s =”123“;

s.append(string(”abc“), 2); // 2 is a **position** s = ”123c“
s.append(”abc“, 2); // 2 is a **length** s = ”123ab“

basic_string& assign(const basic_string& str, size_type pos = 0, size_type n = npos);
basic_string& assign(const charT* s, size_type n);

string s =”123“;

s.assign(string(”abc“), 2); // s = “c“
s.assign(”abc“, 2); // s = “ab“

basic_string& insert(size_type pos1, const basic_string& str, size_type pos2 = 0, size_type n = npos);
basic_string& insert(size_type pos, const charT* s, size_type n);

string s =”123“;

s.insert(0, string(”abc“), 2); // s = “c123“
s.insert(0, ”abc“, 2); // s = “ab123“

basic_string& replace(size_type pos1, size_type n1, const basic_string& str, size_type pos2 = 0, size_type n = npos);
basic_string& replace(size_type pos, size_type n1, const charT* s, size_type n2);

string s =”123“;

s.replace(0, 1, string(”abc“), 2); // s = “c23“
s.replace(0, 1, ”abc“, 2); // s = “ab23“

int compare(size_type pos1, size_type n1, const basic_string& str, size_type pos2 = 0, size_type n2 = npos) const;
int compare(size_type pos, size_type n, charT* s, size_type n = npos) const;

string s =”123“;

```
int         i;

i = s.compare(string(”1234“), 3);  // i != 0
i = s.compare(“123“, 3);           // i == 0
```

The reason for this confusion is the use of the default arguments in the signatures which have the (string, pos, length) tripple of arguments.

Possible solutions:

a) no change. Consequences: Confusion remains in the standard.
b) Change the order of the (string, pos, length) tripple to (string, length, pos). Consequences: many changes in the signatures necessary and this is non intuitive.
c) see proposed solution

Proposed solution

Add to all above mentioned functions one signature just taking a single string argument and remove the default arguments from the other signature taking a string argument.

basic_string& append(const basic_string& str);
basic_string& append(const basic_string& str, size_type pos, size_type n);
basic_string& append(const charT* s, size_type n);

```
string s =”123“;

s.append(string(”abc“), 2);          // now an error
s.append(”abc“, 2);                  // s = ”123ab“
```

basic_string& assign(const basic_string& str);
basic_string& assign(const basic_string& str, size_type pos, size_type n);
basic_string& assign(const charT* s, size_type n);

```
string s =”123“;

s.assign(string(”abc“), 2);          // now an error
s.assign(”abc“, 2);                  // s = “ab“
```

basic_string& insert(size_type pos1, const basic_string& str);
basic_string& insert(size_type pos1, const basic_string& str, size_type pos2, size_type n);
basic_string& insert(size_type pos, const charT* s, size_type n);

```
string s =”123“;

s.insert(0, string(”abc“), 2);       // now an error
s.insert(0, ”abc“, 2);               // s = “ab123“
```

basic_string& replace(size_type pos1, size_type n1, const basic_string& str);
basic_string& replace(size_type pos1, size_type n1, const basic_string& str, size_type pos2, size_type n);
basic_string& replace(size_type pos, size_type n1, const charT* s, size_type n2);

```
string s =”123“;

s.replace(0, 1, string(”abc“), 2);   // now an error
s.replace(0, 1, ”abc“, 2);           // s = “ab23“
```

int compare(size_type pos1, size_type n1, const basic_string& str) const;

```
int compare(size_type pos1, size_type n1, const basic_string& str, size_type pos2, size_type n2) const;
int compare(size_type pos, size_type n, charT* s) const;
int compare(size_type pos, size_type n, charT* s, size_type n) const;
```

```
string    s =”123“;
int       i;

i = s.compare(string(”1234“), 3);  // now an error
i = s.compare(“123“, 3);           // i == 0
```

Consequences

There are 6 new signatures added, but these have been virtually already there (covered by the default arguments). There is an additional benefit: All overloads can be now described in terms of the simple case appending(assigning, ...) just a complete string.