Multiple-Precision Unsigned Integer Arithmetic

Title:
: Multiple-Precision Unsigned Integer Arithmetic

Language:
: C++

Author:
: Philip J. Erdelsky

- pje@efgh.com
- http://www.alumni.caltech.edu/~pje/

Date:
: October 25, 2001

Usage:
: Public domain; no restrictions on use

Portability:
: Any C++ environment

Keywords:
: unsigned arithmetic, multiple precision arithmetic

Abstract:
: A C++ package to perform multiple-precision unsigned integer arithmetic

This package performs the standard arithmetic operations on arbitrarily long unsigned integers. It resides in the header file MPUINT.H and the source file MPUINT.CPP.

When numeric overflow occurs, the package calls the function numeric_overflow(), which must be defined by the caller. It must take some kind of suitable action and then abort the program.

The package represents an unsigned integer internally as an array of 16-bit unsigned short integers. Hence the number of bits must be a multiple of 16. The package will perform arithmetic operations on unsigned integers of different lengths.

A multiple-precision unsigned integer is a variable of the class mpuint:

```
mpuint x(n);
```

Here n is the number of 16-bit unsigned integers used to represent x.

The package also has a copy constructor.

The integers used to represent x and its length are accessible as members:

```
x.value[0] (least significant)
x.value[1]
x.value[2]
  ***
x.value[x.length-1] (most significant)
x.length
```

All the "value" members are of type "unsigned short".

The package implements the following operations, in which x is of type mpuint and y is of type mpuint or unsigned short:

```
x = y
x += y
x -= y
x *= y
x /= y
```

```
x %= y
x == y
x != y
x > y
x >= y
x < y
x <= y
```

The following member function edits a number of type mpuint into its ASCII decimal representation:

```
x.edit(s);

mpuint x;

char *s;    buffer to hold edited result
```

Leading zeros are suppressed. The buffer must be large enough to hold the edited result; buffer overflow and catastrophic failure may occur otherwise.

The following member function scans a string for an unsigned ASCII decimal integer and puts its value into the mpuint number:

```
found = x.scan(s);

mpuint x;        variable to receive result

const char *&s;  pointer to first character to be scanned

bool found;      true if an unsigned integer was found
```

The function scans away leading spaces and tabs. It calls numeric_overflow() if the result overflows. The pointer s is left pointing to the first character following the number.

A special function is used to return the quotient and remainder:

```
mpuint::Divide(dividend, divisor, quotient, remainder);

const mpuint &dividend;
const mpuint &divisor;
mpuint &quotient;
mpuint &remainder;
```

Another special function is used to evaluate the power function, done in modular arithmetic:

```
mpuint::Power(base, exponent, modulus, result);

const mpuint &base;
const mpuint &exponent;
const mpuint &modulus;
mpuint &result;
```

Special comparison functions available for three-way comparisons with only a single call on the package:

```
n = x.Compare(y);

mpuint x;
const mpuint &y;
int n;           negative if x < y, positive if x > y, 0 if x = y

n = x.Compare(y);

const mpuint &x;
unsigned short y;
int n;           negative if x < y, positive if x > y, 0 if x = y
```

A simple member function determines whether a number is zero:

```
r = x.IsZero();

mpuint x;
bool r;          true if x is equal to zero
```