
HugeWord
C++ Class Library of Unsigned Multi-Precision Integer
User's Manual



Machine Learning Laboratory, Inc.

18 Dec 2002

1. Usage

In the C++ class library of unsigned multi-precision integer (HugeWord), the following types, class and its friend functions are defined.

| <i>Type or Class</i> | <i>Definition</i> |
|----------------------|---------------------------------------|
| BYTE | Unsigned 8-bit integer |
| WORD | Unsigned 16-bit integer |
| DWORD | Unsigned 32-bit integer |
| HugeWord | Unsigned multi-precision integer |
| HugeWord::Algorithm | Enumeration type to specify algorithm |

The C++ class library of unsigned multi-precision integer (HugeWord) is composed of the following files.

| <i>File</i> | <i>Description</i> |
|--------------|--|
| HugeWord.h | Declaration Parts of HugeWord and external ASM Functions |
| HugeWord.cpp | Description parts of HugeWord |
| Primes.cpp | Table of prime numbers |
| HW_*.ASM | External ASM Functions |

The C++ class library of unsigned multi-precision integer (HugeWord) requires the following:

| | |
|----------|--------------------------|
| CPU | i80386 or later |
| OS | Windows 2000, Windows XP |
| Compiler | C++ Builder 6 |

2. Reference

In this section, the function of the C++ class library of unsigned multi-precision integer (HugeWord) is described in detail. In the following, not only object of HugeWord but also the integer corresponding to HugeWord is called HugeWord as long as there is no confusion.

2.1. Data Structure

HugeWord records an unsigned integer as DWORD array. The array size is variable, and HugeWord reallocates the array automatically if necessary.

2.2. Constructors and Related Functions

HugeWord(): This default constructor initializes HugeWord object and sets its value to 0.

HugeWord(DWORD x, int s=32): This constructor initializes HugeWord objects of s bits and sets its value to x . Note that s need not be specified, because HugeWord automatically allocate and reallocate necessary memory.

HugeWord(const char* x): This constructor initializes HugeWord object and sets its value specified by the string x . The expression of the string x follows the expression of "unsigned int".

HugeWord(const HugeWord& x): This copy constructor initializes HugeWord objects and sets its value to x .

Member Function: void Build(DWORD x, int s = 32) : This function initializes HugeWord object. The function is similar to the constructor with the same arguments. If "Build" is executed on the already initialized object, the object is reinitialized.

Member Function: void Build(const char* x): This function initializes HugeWord object. The function is similar to the constructor with the same argument. If "Build" is executed on the already initialized object, the object is reinitialized.

Member Function: void Build(const HugeWord& x): This function initializes HugeWord object. The function is similar to the constructor with the same argument. If "Build" is executed on the already initialized object, the object is reinitialized.

Member Function: void ReAlloc(int s): This function changes the memory size allocated to s bits. If s is smaller than necessary memory size, exception (runtime_error) is sent.

Member Function: void Alloc(int s): This function is equivalent to Build((DWORD) 0, int).

2.3. Operators

Each of the following operators has the same function as the unsigned int's operator of the same name, respectively. Mixture operations with DWORD are permitted about these operators. HugeWord automatically reallocates necessary memory, therefore the overflow is never caused as long as memory can be allocated. However, exception (runtime_error) is sent if the subtraction result becomes negative, because the negative value is not permitted. In addition, operator~ is not defined, because the number of digits is variable.

| <i>Classification</i> | <i>Operators</i> |
|-------------------------|---|
| Comparison operators | <, >, ==, !=, <=, >= |
| Assignment operators | =, +=, -=, *=, /=, %=, >>=, <<=, &=, ^=, = |
| Increment and Decrement | ++, -- |
| Binary operators | +, -, *, /, %, &, ^, |
| Shift operators | <<, >> |

2.4. Stream I/O

The text mode stream I/O of HugeWord's object follows that of "unsigned int". Because number of digits of HugeWord is variable, the binary mode stream I/O is not supported. Directly access DWORD's array where HugeWord's value is stored, by using member function "GetValue", if the binary mode I/O is necessary.

2.5. Power

Function: HugeWord Pow(x,y): This function calculates the x^y , and returns the result in HugeWord; where x and y are HugeWord or DWORD, and at least one of them must be HugeWord. If $y=0$, this function returns 1.

Function: HugeWord Pow(x,y,z): This function calculates the $x^y \bmod z$, and return the result in HugeWord; where x , y and z are HugeWord or DWORD, and at least one of them must be HugeWord. If $y=0$, this function returns 1.

2.6. Jacobi Symbol

Function: int Jacobi(x,y): This function evaluates the Jacobi symbol (x/y) , and return the result in int; where x and y are HugeWord or DWORD, and at least one of them must be HugeWord, and y must be odd. Note that this function returns 0, if x and y are not relatively prime.

2.7. Square

Function: HugeWord Sqr(x): This function calculates the square x^2 , and return the result in HugeWord; where x is HugeWord.

Function: HugeWord Sqr(x,y): This function calculates the quadratic residue $x^2 \pmod{y}$, and return the result in HugeWord; where x and y are HugeWord or DWORD, and at least one of them must be HugeWord.

2.8. Square Root

Function: HugeWord Sqrt(a): This function calculates $\lceil a^{1/2} \rceil$, and return the result in HugeWord, where a is HugeWord.

Function: HugeWord Sqrt(a,y): This function solves the congruence equation $x^2 \equiv a \pmod{y}$, and return the solution in HugeWord; where a is HugeWord or DWORD, y is HugeWord, y is a prime such that $y \equiv 3 \pmod{4}$, and a is a quadratic residue of y .

2.9. G.C.D. and L.C.M.

Function: HugeWord GCD(x,y): This function calculates the G.C.D. of x , y , and returns the result in HugeWord; where x and y are HugeWord or DWORD, and at least one of them must be HugeWord.

Function: HugeWord LCM(x,y): This function calculates the L.C.M. of x , y , and returns the result in HugeWord; where x and y are HugeWord or DWORD, and at least one of them must be HugeWord.

2.10. Inverse

Function: HugeWord Inv(x,y): This function calculates $1/x \pmod{y}$, and returns the results in HugeWord; where x is HugeWord or DWORD, and y is HugeWord.

2.11. Primality Test

Member Function: int IsPrime(int m, Algorithm algo): This function returns 1 if HugeWord is decided to be prime, or 0 otherwise; where “*algo*” specifies the primality test algorithm to be used, and “*m*” specifies the iterations of the the test. There are three kinds of primality test algorithms (the Fermat method, the Solovay-Strassen method, and the Rabin method), and it is specified respectively with enumeration type constant HugeWord::Fermat, HugeWord::SolovayStrassen, and HugeWord::Rabin.

2.12. Combinational Function:

Function: HugeWord Fact(int n): This function calculates the factorial $n!$, and returns the result in HugeWord; where n is “int”.

Function: HugeWord Comb(int n, int k): This function calculates number of combinations ${}_nC_k$, and returns the result in HugeWord; where n and k are “int”.

2.13. Hamming Weight and Parity

Member Function: int GetWeight(): This function calculates the Hamming weight of HugeWord as an binary vector, and return the result in “int”.

Member Function: int GetParity(): This function returns 1 if HugeWord has odd Hamming weight, or 0 otherwise.

2.14. Pseudo-Random Number

Member Function: void Rand(int m): This function generates a random HugeWord with m bits.

Member Function: void Random(HugeWord& x): This function generates a random

HugeWord in the range 0 to $x-1$ inclusive.

2.15. Number of Digits

Member Function: int GetLength(): This function calculates the HugeWord's length in bits, and returns the result in int. Note that this function returns 1, when the object takes the value 0. The member function "int Len()" is an alias of the member function "int GetLength()".

Member Function: int GetLength16(): This function calculates the HugeWord's length in WORD, and return the result in int. Note that this function returns 1, when the object takes the value 0. The member function "int Len16()" is an alias of the member function "int GetLength16()".

Member Function: int GetLength32():This function calculates the HugeWord's length in DWORD, and return the result in int. Note that this function returns 1, when the object takes the value 0. The member function "int Len32()" is an alias of the member function "int GetLength32()".

2.16. Access Digits

Member Function: int GetBit(int j): This member function returns HugeWord's j th bit.

Member Function: WORD GetWord16(int j): This member function returns HugeWord's j th WORD.

Member Function: DWORD GetWord32(int j): This member function returns HugeWord's j th DWORD.

Member Function: void SetBit(int x, int j): This function assigns x to HugeWord's j th bit. This function automatically allocates or reallocates memory if necessary.

Member Function: void SetWord16(WORD x, int j): This function assigns x to HugeWord's j th WORD. This function automatically allocates or reallocates memory if necessary.

Member Function: void SetWord32(DWORD x, int j): This function assigns x to HugeWord's j th DWORD. This function automatically allocates or reallocates memory if necessary.

2.17. Access Data Members

Member Function: int GetSize(): This function returns the length(in bits) of the array that stored HugeWord's value.

Member Function: int GetSize16(): This function returns the length (in WORDs) of the array that stored HugeWord's value.

Member Function: int GetSize32(): This function returns the length (in DWORDs) of the array that stored HugeWord's value.

Member Function: DWORD* GetValue(): This function returns the pointer to the array that stored HugeWord's value.

Revision History

Version 1.0 – 20 Aug 2000

Version 2.0 – 31 Oct 2000 – Ported to Borland C++ 5

Version 2.1 – 18 Dec 2002 – Ported to Borland C++ 6