My Project

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 BINT Struct Reference

`#include <config.h>`

Public Attributes

- bool sign
- int wordlen
- WORD * val

3.1.1 Detailed Description

Definition at line 33 of file config.h.

3.1.2 Member Data Documentation

3.1.2.1 sign

`bool BINT::sign`

Definition at line 34 of file config.h.

3.1.2.2 val

`WORD* BINT::val`

Definition at line 36 of file config.h.

3.1.2.3 wordlen

`int BINT::wordlen`

Definition at line 35 of file config.h.

The documentation for this struct was generated from the following file:

- config.h

Chapter 4

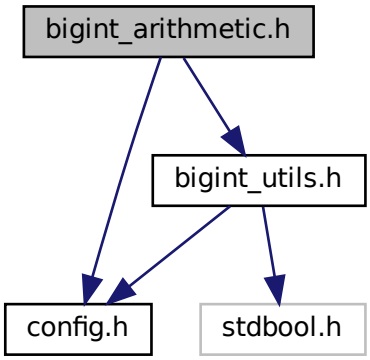
File Documentation

4.1 bigint_arithmetic.c File Reference

4.2 bigint_arithmetic.h File Reference

```
#include "config.h"
#include "bigint_utils.h"
```

Include dependency graph for bigint_arithmetic.h:



Functions

- void NOT_BINT (BINT **ptrbint_dst, BINT **pptrBint_src)

  Performs a bitwise NOT operation on a BINT object.
- void AND_BINT (BINT *ptrX, BINT *ptrY, BINT **pptrZ)

  Performs a bitwise AND operation between two BINT objects.
- void OR_BINT (BINT *ptrX, BINT *ptrY, BINT **pptrZ)

  Performs a bitwise OR operation between two BINT objects.

- void XOR_BINT (BINT *ptrX, BINT *ptrY, BINT **pptrZ)

  Performs a bitwise XOR operation between two BINT objects.
- void add_carry (WORD x, WORD y, WORD k, WORD *ptrQ, WORD *ptrR)

  Performs addition with carry of two WORDs and stores the result and carry.
- void add_core_xyz (BINT **pptrX, BINT **pptrY, BINT **pptrZ)

  Adds two multi-word integers (BINTs) and stores the result in a third BINT, managing memory as needed.
- void ADD (BINT **pptrX, BINT **pptrY, BINT **pptrZ)

  Public API function for adding two BINTs, wrapping the core addition functionality with user-friendly access.
- void sub_borrow (WORD x, WORD y, WORD *ptrQ, WORD *ptrR)

  Subtracts two WORD-sized integers, taking into account a borrow, and outputs both the difference and the resulting borrow.
- void sub_core_xyz (BINT **pptrX, BINT **pptrY, BINT **pptrZ)

  Subtracts two multi-word integers (BINTs) and stores the result in a third BINT, managing memory as needed.
- void SUB (BINT **pptrX, BINT **pptrY, BINT **pptrZ)

  Public API function for subtracting two BINTs, encapsulating the core subtraction functionality for ease of use.
- void mul_xyz (WORD valX, WORD valY, BINT **pptrZ)

  Multiplies two WORD-sized integers and stores the result in a BINT object.
- void mul_core_TxtBk_xyz (BINT **pptrX, BINT **pptrY, BINT **pptrZ)

  Core textbook algorithm for multiplying two BINT objects and storing the result in a third BINT object.
- void mul_core_ImpTxtBk_test (BINT **pptrX, BINT **pptrY, BINT **pptrZ)

  Tests the improved textbook multiplication algorithm on two BINT objects, storing the result in a third BINT object.
- void MUL_Core_ImpTxtBk_xyz (BINT **pptrX, BINT **pptrY, BINT **pptrZ)

  Core function for the improved textbook multiplication algorithm of two BINT objects, storing the result in a third BINT object.
- void mul_core_Krtsb_test (BINT **pptrX, BINT **pptrY, BINT **pptrZ)

  Tests the Karatsuba multiplication algorithm on two BINT objects, storing the result in a third BINT object, with a flag for testing purposes.
- void Krtsb_FLAG_Test (BINT **pptrX, BINT **pptrY, BINT **pptrZ, int flag)

  Karatsuba multiplication test function with a flag parameter to control the behavior of the multiplication algorithm for two BINT objects.
- void MUL_Core_Krtsb_xyz (BINT **pptrX, BINT **pptrY, BINT **pptrZ)

  Core Karatsuba multiplication algorithm for two BINT objects, storing the result in a third BINT object.
- void squ_core (WORD valX, BINT **pptrZ)

  Performs squaring of a WORD-sized integer and stores the result in a BINT object.
- void SQU_Txtbk_xz (BINT **pptrX, BINT **pptrZ)

  Textbook squaring algorithm for a BINT object, storing the result in another BINT object.
- void SQU_Krtsb_xz (BINT **pptrX, BINT **pptrZ)

  Karatsuba algorithm for squaring a BINT object, storing the result in another BINT object.
- void DIV_Bianry_Long_Test (BINT **pptrDividend, BINT **pptrDivisor, BINT **pptrQ, BINT **pptrR)

  Tests the binary long division algorithm using BINT objects for the dividend and divisor, and stores the quotient and remainder.
- void DIV_Bianry_Long (BINT **pptrDividend, BINT **pptrDivisor, BINT **pptrQ, BINT **pptrR)

  Performs binary long division on BINT objects for the dividend and divisor, and stores the quotient and remainder.
- void DIV_General_Long (BINT **pptrDividend, BINT **pptrDivisor, BINT **pptrQ, BINT **pptrR)

  Performs general long division on BINT objects for the dividend and divisor, outputting the quotient and remainder.
- void mul_LeftToRight (BINT **ptrX, BINT **ptrY, BINT **ptrZ)

  Multiplies two BINT objects using a left-to-right binary method, storing the result in a third BINT object.
- void add_LeftToRight (BINT **ptrX, BINT **ptrY, BINT **ptrZ)

  Adds two BINT objects using a left-to-right binary method, storing the result in a third BINT object.
- void mul_RightToLeft (BINT **ptrX, BINT **ptrY, BINT **ptrZ)

  Multiplies two BINT objects using a right-to-left binary method, storing the result in a third BINT object.
- void add_RightToLeft (BINT **ptrX, BINT **ptrY, BINT **ptrZ)

Adds two BINT objects using a right-to-left binary method, storing the result in a third BINT object.

- void exp_Mongomery (BINT **ptrX, BINT **ptrY, BINT **ptrZ)

  Calculates the Montgomery exponentiation of two BINT objects and stores the result in a third BINT object.

- void Mod_Exp_Mongo (BINT **ptrX, BINT **ptrY, BINT **ptrM, BINT **ptrZ)

  Performs modular exponentiation using the Montgomery method on three BINT objects and stores the result in a fourth BINT object.

Variables

- Here are the professional annotations for the division
- Here are the professional annotations for the multiplication
- Here are the professional annotations for the addition
- Here are the professional annotations for the and exponentiation functions in your C header file
- Here are the professional annotations for the and exponentiation functions in your C header BINT ** pptrDivisor
- Here are the professional annotations for the and exponentiation functions in your C header BINT BINT ** pptrQ
- Here are the professional annotations for the and exponentiation functions in your C header BINT BINT BINT ** pptrR

4.2.1 Function Documentation

4.2.1.1 ADD()

```
void ADD (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ )
```

Public API function for adding two BINTs, wrapping the core addition functionality with user-friendly access.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	Pointer to the pointer of the first BINT operand.
pptrY	Pointer to the pointer of the second BINT operand.
pptrZ	Pointer to the pointer where the addition result will be stored.

Note

Utilizes add_core_xyz internally; users should call this function for adding BINTs.

4.2.1.2 add_carry()

```
void add_carry (
            WORD x,
            WORD y,
            WORD k,
            WORD * ptrQ,
            WORD * ptrR )
```

Performs addition with carry of two WORDs and stores the result and carry.

Author

Your Name

Date

Date of creation or last update

Parameters

x	The first WORD operand.
y	The second WORD operand.
k	The carry-in value.
ptrQ	A pointer to the WORD where the sum will be stored.
ptrR	A pointer to the WORD where the carry-out will be stored.

Note

This is a low-level operation used in multi-word arithmetic.

4.2.1.3 add_core_xyz()

```
void add_core_xyz (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ )
```

Adds two multi-word integers (BINTs) and stores the result in a third BINT, managing memory as needed.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	Pointer to the pointer of the first BINT operand.
pptrY	Pointer to the pointer of the second BINT operand.
pptrZ	Pointer to the pointer where the resulting BINT will be stored after the addition.

Note

This function handles the intricacies of BINT addition, including memory reallocation for the result, if necessary.

4.2.1.4 add_LeftToRight()

```
void add_LeftToRight (
            BINT ** ptrX,
            BINT ** ptrY,
            BINT ** ptrZ )
```

Adds two BINT objects using a left-to-right binary method, storing the result in a third BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrX	A pointer to the pointer of the first BINT addend.
ptrY	A pointer to the pointer of the second BINT addend.
ptrZ	A pointer to the pointer where the addition result will be stored.

Note

This method adds the BINT objects starting from the most significant bit and moving to the least significant bit.

4.2.1.5 add_RightToLeft()

```
void add_RightToLeft (
            BINT ** ptrX,
            BINT ** ptrY,
            BINT ** ptrZ )
```

Adds two BINT objects using a right-to-left binary method, storing the result in a third BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrX	A pointer to the pointer of the first BINT addend.
ptrY	A pointer to the pointer of the second BINT addend.
ptrZ	A pointer to the pointer where the addition result will be stored.

Note

This method adds the BINT objects starting from the least significant bit and moving to the most significant bit.

4.2.1.6 AND_BINT()

```
void AND_BINT (
            BINT * ptrX,
            BINT * ptrY,
            BINT ** pptrZ )
```

Performs a bitwise AND operation between two BINT objects.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrX	A pointer to the first BINT object operand.
ptrY	A pointer to the second BINT object operand.
pptrZ	A pointer to the BINT object pointer where the result will be stored.

Note

The function will allocate memory for the result if necessary.

4.2.1.7 DIV_Bianry_Long()

```
void DIV_Bianry_Long (
            BINT ** pptrDividend,
            BINT ** pptrDivisor,
            BINT ** pptrQ,
            BINT ** pptrR )
```

Performs binary long division on BINT objects for the dividend and divisor, and stores the quotient and remainder.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrDividend	A pointer to the pointer of the BINT object representing the dividend.
pptrDivisor	A pointer to the pointer of the BINT object representing the divisor.
pptrQ	A pointer to the pointer where the quotient BINT object will be stored.
pptrR	A pointer to the pointer where the remainder BINT object will be stored.

Note

Implements the binary long division algorithm which is efficient for large BINT objects.

4.2.1.8 DIV_Bianry_Long_Test()

```
void DIV_Bianry_Long_Test (
            BINT ** pptrDividend,
            BINT ** pptrDivisor,
            BINT ** pptrQ,
            BINT ** pptrR )
```

Tests the binary long division algorithm using BINT objects for the dividend and divisor, and stores the quotient and remainder.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrDividend	A pointer to the pointer of the BINT object representing the dividend.
pptrDivisor	A pointer to the pointer of the BINT object representing the divisor.
pptrQ	A pointer to the pointer where the quotient BINT object will be stored.
pptrR	A pointer to the pointer where the remainder BINT object will be stored.

Note

This function is intended for testing the binary long division algorithm.

4.2.1.9 DIV_General_Long()

```
void DIV_General_Long (
            BINT ** pptrDividend,
            BINT ** pptrDivisor,
            BINT ** pptrQ,
            BINT ** pptrR )
```

Performs general long division on BINT objects for the dividend and divisor, outputting the quotient and remainder.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrDividend	A pointer to the pointer of the BINT object representing the dividend.
pptrDivisor	A pointer to the pointer of the BINT object representing the divisor.
pptrQ	A pointer to the pointer where the quotient BINT object will be stored.
pptrR	A pointer to the pointer where the remainder BINT object will be stored.

Note

This version of long division is optimized for a general case and can handle different forms of BINT objects.

4.2.1.10 exp_Mongomery()

```
void exp_Mongomery (
            BINT ** ptrX,
```

```
            BINT ** ptrY,
            BINT ** ptrZ )
```

Calculates the Montgomery exponentiation of two BINT objects and stores the result in a third BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrX	A pointer to the pointer of the base BINT operand.
ptrY	A pointer to the pointer of the exponent BINT operand.
ptrZ	A pointer to the pointer where the exponentiation result will be stored.

Note

This method is used for efficient modular exponentiation, especially in cryptographic applications.

4.2.1.11 Krtsb_FLAG_Test()

```
void Krtsb_FLAG_Test (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ,
            int flag )
```

Karatsuba multiplication test function with a flag parameter to control the behavior of the multiplication algorithm for two BINT objects.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	A pointer to the pointer of the first BINT operand.
pptrY	A pointer to the pointer of the second BINT operand.
pptrZ	A pointer to the pointer where the multiplication result will be stored.
flag	An integer to toggle specific behaviors or optimizations during the multiplication process.

Note

This function allows for controlled experimentation with the Karatsuba algorithm.

4.2.1.12 Mod_Exp_Mongo()

```
void Mod_Exp_Mongo (
            BINT ** ptrX,
            BINT ** ptrY,
            BINT ** ptrM,
            BINT ** ptrZ )
```

Performs modular exponentiation using the Montgomery method on three BINT objects and stores the result in a fourth BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrX	A pointer to the pointer of the base BINT operand.
ptrY	A pointer to the pointer of the exponent BINT operand.
ptrM	A pointer to the pointer of the modulus BINT operand.
ptrZ	A pointer to the pointer where the modular exponentiation result will be stored.

Note

This function is suitable for high-precision arithmetic, such as cryptographic operations involving large numbers.

4.2.1.13 mul_core_ImpTxtBk_test()

```
void mul_core_ImpTxtBk_test (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ )
```

Tests the improved textbook multiplication algorithm on two BINT objects, storing the result in a third BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	A pointer to the pointer of the first BINT operand.
pptrY	A pointer to the pointer of the second BINT operand.
pptrZ	A pointer to the pointer where the multiplication result will be stored.

Note

This function is used for testing optimizations in the textbook multiplication algorithm.

4.2.1.14 MUL_Core_ImpTxtBk_xyz()

```
void MUL_Core_ImpTxtBk_xyz (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ )
```

Core function for the improved textbook multiplication algorithm of two BINT objects, storing the result in a third BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	A pointer to the pointer of the first BINT operand.
pptrY	A pointer to the pointer of the second BINT operand.
pptrZ	A pointer to the pointer where the multiplication result will be stored.

Note

This function is a more efficient version of the textbook multiplication algorithm, optimized for larger BINT objects.

4.2.1.15 mul_core_Krtsb_test()

```
void mul_core_Krtsb_test (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ )
```

Tests the Karatsuba multiplication algorithm on two BINT objects, storing the result in a third BINT object, with a flag for testing purposes.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	A pointer to the pointer of the first BINT operand.
pptrY	A pointer to the pointer of the second BINT operand.
pptrZ	A pointer to the pointer where the multiplication result will be stored.
flag	An integer flag used to control test behavior or algorithm variants.

Note

This function is used to validate the Karatsuba algorithm's correctness and performance.

4.2.1.16 MUL_Core_Krtsb_xyz()

```
void MUL_Core_Krtsb_xyz (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ )
```

Core Karatsuba multiplication algorithm for two BINT objects, storing the result in a third BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	A pointer to the pointer of the first BINT operand.
pptrY	A pointer to the pointer of the second BINT operand.
pptrZ	A pointer to the pointer where the multiplication result will be stored.

Note

This function implements the Karatsuba algorithm for fast multiplication of large BINT objects.

4.2.1.17 mul_core_TxtBk_xyz()

```
void mul_core_TxtBk_xyz (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ )
```

Core textbook algorithm for multiplying two BINT objects and storing the result in a third BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	A pointer to the pointer of the first BINT operand.
pptrY	A pointer to the pointer of the second BINT operand.
pptrZ	A pointer to the pointer where the multiplication result will be stored.

Note

This function implements the standard long multiplication algorithm taught in textbooks.

4.2.1.18 mul_LeftToRight()

```
void mul_LeftToRight (
            BINT ** ptrX,
            BINT ** ptrY,
            BINT ** ptrZ )
```

Multiplies two BINT objects using a left-to-right binary method, storing the result in a third BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrX	A pointer to the pointer of the first BINT operand.
ptrY	A pointer to the pointer of the second BINT operand.
ptrZ	A pointer to the pointer where the multiplication result will be stored.

Note

This method multiplies from the most significant bit to the least significant bit of the multiplier.

4.2.1.19 mul_RightToLeft()

```
void mul_RightToLeft (
            BINT ** ptrX,
            BINT ** ptrY,
            BINT ** ptrZ )
```

Multiplies two BINT objects using a right-to-left binary method, storing the result in a third BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrX	A pointer to the pointer of the first BINT operand.
ptrY	A pointer to the pointer of the second BINT operand.
ptrZ	A pointer to the pointer where the multiplication result will be stored.

Note

This method multiplies from the least significant bit to the most significant bit of the multiplier.

4.2.1.20 mul_xyz()

```
void mul_xyz (
            WORD valX,
            WORD valY,
            BINT ** pptrZ )
```

Multiplies two WORD-sized integers and stores the result in a BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

valX	The first WORD-sized integer operand.
valY	The second WORD-sized integer operand.
pptrZ	A pointer to the pointer of the BINT object where the result will be stored.

Note

This function is utilized for single word multiplication that needs to be stored in a multi-word BINT object.

4.2.1.21 NOT_BINT()

```
void NOT_BINT (
            BINT ** ptrbint_dst,
            BINT ** pptrBint_src )
```

Performs a bitwise NOT operation on a BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrbint_dst	A pointer to the BINT object pointer where the result will be stored.
pptrBint_src	A pointer to the BINT object pointer to be negated.

Note

The result is stored in ptrbint_dst, and the original BINT object is not modified.

4.2.1.22 OR_BINT()

```
void OR_BINT (
            BINT * ptrX,
            BINT * ptrY,
            BINT ** pptrZ )
```

Performs a bitwise OR operation between two BINT objects.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrX	A pointer to the first BINT object operand.
ptrY	A pointer to the second BINT object operand.
pptrZ	A pointer to the BINT object pointer where the result will be stored.

Note

The function will allocate memory for the result if necessary.

4.2.1.23 squ_core()

```
void squ_core (
            WORD valX,
            BINT ** pptrZ )
```

Performs squaring of a WORD-sized integer and stores the result in a BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

valX	The WORD-sized integer to be squared.
pptrZ	A pointer to the pointer of the BINT object where the result will be stored.

Note

This function is optimized for squaring a single WORD-sized integer within a multi-word BINT context.

4.2.1.24 SQU_Krtsb_xz()

```
void SQU_Krtsb_xz (
            BINT ** pptrX,
            BINT ** pptrZ )
```

Karatsuba algorithm for squaring a BINT object, storing the result in another BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	A pointer to the pointer of the BINT object to be squared.
pptrZ	A pointer to the pointer where the squaring result will be stored.

Note

Utilizes the efficient Karatsuba multiplication algorithm adapted for the squaring operation.

4.2.1.25 SQU_Txtbk_xz()

```
void SQU_Txtbk_xz (
            BINT ** pptrX,
            BINT ** pptrZ )
```

Textbook squaring algorithm for a BINT object, storing the result in another BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	A pointer to the pointer of the BINT object to be squared.
pptrZ	A pointer to the pointer where the squaring result will be stored.

Note

Implements the standard algorithm for squaring, similar to the textbook multiplication algorithm.

4.2.1.26 SUB()

```
void SUB (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ )
```

Public API function for subtracting two BINTs, encapsulating the core subtraction functionality for ease of use.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	Pointer to the pointer of the first BINT operand.
pptrY	Pointer to the pointer of the second BINT operand.
pptrZ	Pointer to the pointer where the subtraction result will be stored.

Note

Utilizes sub_core_xyz internally; intended to be the function users call for subtracting BINTs.

4.2.1.27 sub_borrow()

```
void sub_borrow (
            WORD x,
            WORD y,
            WORD * ptrQ,
            WORD * ptrR )
```

Subtracts two WORD-sized integers, taking into account a borrow, and outputs both the difference and the resulting borrow.

Author

Your Name

Date

Date of creation or last update

Parameters

x	The WORD from which y is to be subtracted.
y	The WORD to be subtracted from x.
ptrQ	Pointer to the variable where the difference will be stored.
ptrR	Pointer to the variable where the resulting borrow will be stored; it will be non-zero if the subtraction underflows.

Note

Essential for multi-word subtraction, this function ensures that borrows are correctly propagated through the operation.

4.2.1.28 sub_core_xyz()

```
void sub_core_xyz (
            BINT ** pptrX,
            BINT ** pptrY,
            BINT ** pptrZ )
```

Subtracts two multi-word integers (BINTs) and stores the result in a third BINT, managing memory as needed.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrX	Pointer to the pointer of the first BINT operand.
pptrY	Pointer to the pointer of the second BINT operand.
pptrZ	Pointer to the pointer where the resulting BINT will be stored after the subtraction.

Note

This function handles the complexities of BINT subtraction, including memory reallocation for the result, if necessary.

4.2.1.29 XOR_BINT()

```
void XOR_BINT (
            BINT * ptrX,
            BINT * ptrY,
            BINT ** pptrZ )
```

Performs a bitwise XOR operation between two BINT objects.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrX	A pointer to the first BINT object operand.
ptrY	A pointer to the second BINT object operand.
pptrZ	A pointer to the BINT object pointer where the result will be stored.

Note

The function will allocate memory for the result if necessary.

4.2.2 Variable Documentation

4.2.2.1 addition

```
Here are the professional annotations for the addition
```

Definition at line 291 of file bigint_arithmetic.h.

4.2.2.2 division

```
Here are the professional annotations for the division
```

Definition at line 291 of file bigint_arithmetic.h.

4.2.2.3 file

```
Here are the professional annotations for the and exponentiation functions in your C header
file
```

Definition at line 291 of file bigint_arithmetic.h.

4.2.2.4 multiplication

```
Here are the professional annotations for the multiplication
```

Definition at line 291 of file bigint_arithmetic.h.

4.2.2.5 pptrDivisor

```
Here are the professional annotations for the and exponentiation functions in your C header
BINT** pptrDivisor
```

Definition at line 305 of file bigint_arithmetic.h.

4.2.2.6 pptrQ

```
Here are the professional annotations for the and exponentiation functions in your C header
BINT BINT** pptrQ
```

Definition at line 305 of file bigint_arithmetic.h.

4.2.2.7 pptrR

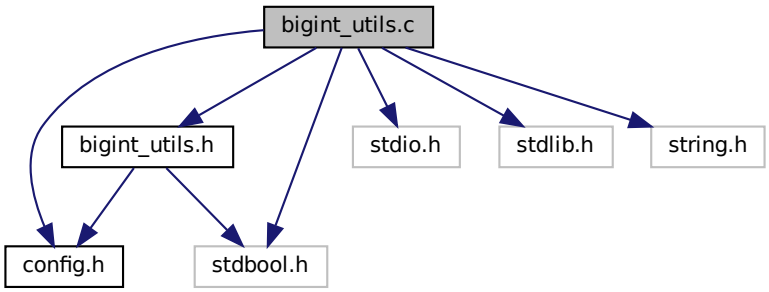
```
Here are the professional annotations for the and exponentiation functions in your C header
BINT BINT BINT** pptrR
```

Definition at line 305 of file bigint_arithmetic.h.

4.3 bigint_utils.c File Reference

```
#include "config.h"
#include "bigint_utils.h"
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for bigint_utils.c:



Macros

- #define CHECK_PTR_AND_DEREF(ptr, name, func)

Functions

- void exit_on_null_error (const void *ptr, const char *ptr_name, const char *function_name)

  Exits program if pointer is NULL, indicating an allocation error.

4.3.1 Macro Definition Documentation

4.3.1.1 CHECK_PTR_AND_DEREF

```
#define CHECK_PTR_AND_DEREF(
                ptr,
                name,
                func )
```

Value:

```
    do { \
        exit_on_null_error(ptr, name, func); \
        exit_on_null_error(*ptr, "*" name, func); \
    } while(0)
```

Definition at line 15 of file bigint_utils.c.

4.3.2 Function Documentation

4.3.2.1 exit_on_null_error()

```
void exit_on_null_error (
            const void * ptr,
            const char * ptr_name,
            const char * function_name )
```

Exits program if pointer is NULL, indicating an allocation error.

Author

Ji Yong-Hyeon

Date

2023-11-26

Parameters

ptr	- The pointer to check.
ptr_name	- The name of the pointer variable.
function_name	- The name of the function calling this check.

Note

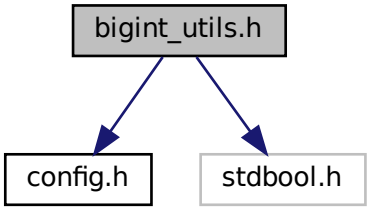
This function will terminate the program on a null pointer error.

Definition at line 9 of file bigint_utils.c.

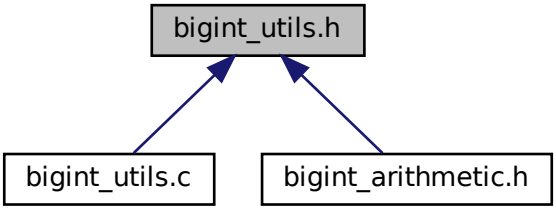
4.4 bigint_utils.h File Reference

```
#include "config.h"
#include <stdbool.h>
```

Include dependency graph for bigint_utils.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define BIGINT_UTILS_H

Functions

- void exit_on_null_error (const void *ptr, const char *ptr_name, const char *function_name)

  Exits program if pointer is NULL, indicating an allocation error.
- void delete_bint (BINT **pptrBint)

  Frees the memory allocated for a BINT object and sets the pointer to NULL.
- void init_bint (BINT **pptrBint, int wordlen)

  Initializes a BINT object with the specified word length.
- void SET_BINT_ZERO (BINT **pptrBint)

  Sets the value of a BINT object to zero.
- void SET_BINT_ONE (BINT **pptrBint)

  Sets the value of a BINT object to one (1).
- void copyBINT (BINT **pptrBint_dst, const BINT *ptrBint_src)

  Copies the value of one BINT object to another.
- void swapBINT (BINT **pptrBint1, BINT **pptrBint2)

Swaps the values of two BINT objects.

- void makeEven (BINT *ptrBint)

  Modifies a BINT object to be an even number by clearing the least significant bit.
- void matchSize (BINT *ptrBint1, BINT *ptrBint2)

  Ensures that two BINT objects have the same size by padding the smaller one with zeros.
- void resetBINT (BINT *ptrBint)

  Resets a BINT object to its default state with a value of zero and minimal size.
- void refineBINT (BINT *ptrBint)

  Refines the size of a BINT object to remove any leading zero words.
- bool isZero (BINT *ptrBint)

  Checks if a BINT object represents the zero value.
- bool isOne (BINT *ptrBint)

  Checks if a BINT object represents the value one (1).
- bool GET_BIT (BINT **pptrBint, int i_th)

  Retrieves the value of a specific bit in a BINT object.
- void RANDOM_ARRARY (WORD *dst, int wordlen)

  Fills an array with random WORD values.
- void RANDOM_BINT (BINT **pptrBint, bool sign, int wordlen)

  Initializes a BINT object with random values.
- bool compare_bint (BINT *pptrBint1, BINT *pptrBint2)

  Compares two BINT objects for equality.
- bool compare_abs_bint (BINT *pptrBint1, BINT *pptrBint2)

  Compares the absolute values of two BINT objects for equality.
- int BIT_LENGTH (BINT *pptrBint)

  Calculates the bit length of a BINT object, including leading zeros.
- int BIT_LENGTH_NONZERO (BINT *pptrBint)

  Calculates the bit length of a BINT object, excluding leading zeros.
- void left_shift_word (BINT **pptrBint, int shift_amount)

  Performs a left shift operation on a BINT object by a specified number of words.
- void right_shift_word (BINT **pptrBint, int shift_amount)

  Performs a right shift operation on a BINT object by a specified number of words.
- void left_shift_bit (BINT **pptrBint, int shift_amount)

  Performs a left shift operation on a BINT object by a specified number of bits.
- void left_right_bit (BINT **pptrBint, int shift_amount)

  Performs a right shift operation on a BINT object by a specified number of bits (possibly a typo for right_shift_bit).
- void reduction (BINT **pptrBint, int pwOf2)

  Reduces a BINT object modulo a power of 2 specified by pwOf2.
- WORD hexSubstringToWord (const char *str, int start, int length)

  Converts a hexadecimal substring to a WORD value.
- void strToBINT (BINT **pptrBint, const char *hexString)

  Converts a hexadecimal string to a BINT object.
- void HexDigitToBinary (WORD hex_digit, bool *binary, int start_index, int bits)

  Converts a hexadecimal digit to a binary representation and stores it in an array.
- bool * HexToBinary (BINT *hex)

  Converts a BINT object's hexadecimal value to a binary array representation.
- void PrintBinary (bool *binary, int length)

  Prints the binary representation of an array to standard output.
- WORD BinaryToHexDigit (bool *binary, int start_index, int bits)

  Converts a binary array to a hexadecimal digit starting from a specified index.
- BINT * BinaryToHex (bool *binary, int length)

  Converts a binary array to a BINT object's hexadecimal representation.

- void print_bint_bin (const BINT *ptrBint)

  Prints the binary representation of a BINT object to standard output.
- void print_bint_hex (const BINT *ptrBint)

  Prints the hexadecimal representation of a BINT object to standard output.
- void print_bint_bin_split (const BINT *ptrBint)

  Prints the binary representation of a BINT object to standard output with a split for readability.
- void print_bint_hex_split (const BINT *ptrBint)

  Prints the hexadecimal representation of a BINT object to standard output with a split for readability.
- void print_bint_bin_py (const BINT *ptrBint)

  Prints the binary representation of a BINT object to standard output in a format compatible with Python lists.
- void print_bint_hex_py (const BINT *ptrBint)

  Prints the hexadecimal representation of a BINT object to standard output in a format compatible with Python lists.

Variables

- Continuing with the professional annotations for the remaining functions in your C header file
- Continuing with the professional annotations for the remaining functions in your C header int shift_amount

4.4.1 Macro Definition Documentation

4.4.1.1 BIGINT_UTILS_H

```
#define BIGINT_UTILS_H
```

Definition at line 6 of file bigint_utils.h.

4.4.2 Function Documentation

4.4.2.1 BinaryToHex()

```
BINT* BinaryToHex (
            bool * binary,
            int length )
```

Converts a binary array to a BINT object's hexadecimal representation.

Author

Your Name

Date

Date of creation or last update

Parameters

binary	A pointer to the binary array to convert.
length	The length of the binary array.

Returns

A pointer to the BINT object representing the binary array's value.

Note

The caller is responsible for managing the memory of the returned BINT object.

4.4.2.2 BinaryToHexDigit()

```
WORD BinaryToHexDigit (
            bool * binary,
            int start_index,
            int bits )
```

Converts a binary array to a hexadecimal digit starting from a specified index.

Author

Your Name

Date

Date of creation or last update

Parameters

binary	A pointer to the binary array.
start_index	The starting index in the array where the binary digit begins.
bits	The number of bits to use for the conversion.

Returns

The WORD value of the hexadecimal digit.

Note

Assumes the binary array has at least 'bits' elements starting from 'start_index'.

4.4.2.3 BIT_LENGTH()

```
int BIT_LENGTH (
            BINT * pptrBint )
```

Calculates the bit length of a BINT object, including leading zeros.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the BINT object.

Returns

The bit length of the BINT object.

Note

This function includes leading zeros in the bit length calculation.

4.4.2.4 BIT_LENGTH_NONZERO()

```
int BIT_LENGTH_NONZERO (
            BINT * pptrBint )
```

Calculates the bit length of a BINT object, excluding leading zeros.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the BINT object.

Returns

The bit length of the BINT object, excluding leading zeros.

Note

Useful for determining the actual number of significant bits.

4.4.2.5 compare_abs_bint()

```
bool compare_abs_bint (
            BINT * pptrBint1,
            BINT * pptrBint2 )
```

Compares the absolute values of two BINT objects for equality.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint1	A pointer to the first BINT object for comparison.
pptrBint2	A pointer to the second BINT object for comparison.

Returns

True if the absolute values of the BINT objects are equal; otherwise, false.

Note

This function ignores the signs of the BINT objects.

4.4.2.6 compare_bint()

```
bool compare_bint (
            BINT * pptrBint1,
            BINT * pptrBint2 )
```

Compares two BINT objects for equality.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint1	A pointer to the first BINT object for comparison.
pptrBint2	A pointer to the second BINT object for comparison.

Returns

True if the BINT objects are equal; otherwise, false.

Note

This function compares both value and sign.

4.4.2.7 copyBINT()

```
void copyBINT (
            BINT ** pptrBint_dst,
            const BINT * ptrBint_src )
```

Copies the value of one BINT object to another.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint_dst	A pointer to the point of the destination BINT object pointer.
ptrBint_src	A pointer to the point of the source BINT object to copy from.

Note

Assumes that both BINT objects have been properly initialized.

4.4.2.8 delete_bint()

```
void delete_bint (
            BINT ** pptrBint )
```

Frees the memory allocated for a BINT object and sets the pointer to NULL.

Author

Ji Yong-Hyeon

Date

2023-11-26

Parameters

pptrBint	A pointer to the pointer of the BINT object to be deleted.

Note

This function should be called to avoid memory leaks.

4.4.2.9 exit_on_null_error()

```
void exit_on_null_error (
            const void * ptr,
            const char * ptr_name,
            const char * function_name )
```

Exits program if pointer is NULL, indicating an allocation error.

Author

Ji Yong-Hyeon

Date

2023-11-26

Parameters

ptr	- The pointer to check.
ptr_name	- The name of the pointer variable.
function_name	- The name of the function calling this check.

Note

This function will terminate the program on a null pointer error.

Definition at line 9 of file bigint_utils.c.

4.4.2.10 GET_BIT()

```
bool GET_BIT (
            BINT ** pptrBint,
            int i_th )
```

Retrieves the value of a specific bit in a BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the pointer of the BINT object.
i_th	The index of the bit to retrieve.

Returns

True if the bit is set; otherwise, false.

Note

Indexing starts at 0 and refers to the least significant bit.

4.4.2.11 HexDigitToBinary()

```
void HexDigitToBinary (
            WORD hex_digit,
            bool * binary,
            int start_index,
            int bits )
```

Converts a hexadecimal digit to a binary representation and stores it in an array.

Author

Your Name

Date

Date of creation or last update

Parameters

hex_digit	The hexadecimal digit to convert.
binary	A pointer to the array where the binary representation will be stored.
start_index	The starting index in the array where the binary representation begins.
bits	The number of bits to represent the hexadecimal digit.

Note

Assumes the binary array has enough space to store the representation.

4.4.2.12 hexSubstringToWord()

```
WORD hexSubstringToWord (
            const char * str,
            int start,
            int length )
```

Converts a hexadecimal substring to a WORD value.

Author

Your Name

Date

Date of creation or last update

Parameters

str	The string containing the hexadecimal characters.
start	The starting index of the substring.
length	The length of the substring.

Returns

The WORD value of the hexadecimal substring.

Note

If the substring is longer than what a WORD can represent, the behavior is undefined.

4.4.2.13 HexToBinary()

```
bool* HexToBinary (
            BINT * hex )
```

Converts a BINT object's hexadecimal value to a binary array representation.

Author

Your Name

Date

Date of creation or last update

Parameters

hex	A pointer to the BINT object to convert.

Returns

A pointer to the binary array representing the BINT object's value.

Note

The caller is responsible for freeing the memory allocated for the binary array.

4.4.2.14 init_bint()

```
void init_bint (
            BINT ** pptrBint,
            int wordlen )
```

Initializes a BINT object with the specified word length.

Author

Ji Yong-Hyeon

Date

2023-11-26

Parameters

pptrBint	A pointer to the pointer of the BINT object to be initialized.
wordlen	The word length for the BINT object.

Note

The BINT object's memory will be dynamically allocated.

4.4.2.15 isOne()

```
bool isOne (
            BINT * ptrBint )
```

Checks if a BINT object represents the value one (1).

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to check.

Returns

True if the BINT object is one; otherwise, false.

Note

This is a quick check and does not modify the BINT object.

4.4.2.16 isZero()

```
bool isZero (
            BINT * ptrBint )
```

Checks if a BINT object represents the zero value.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to check.

Returns

True if the BINT object is zero; otherwise, false.

Note

This is a quick check and does not modify the BINT object.

4.4.2.17 left_right_bit()

```
void left_right_bit (
            BINT ** pptrBint,
            int shift_amount )
```

Performs a right shift operation on a BINT object by a specified number of bits (possibly a typo for right_shift_bit).

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the BINT object pointer to be shifted.
shift_amount	The number of bits to shift the BINT object by.

Note

This operation can decrease the size of the BINT object if the shift goes beyond the current least significant bit.

4.4.2.18 left_shift_bit()

```
void left_shift_bit (
            BINT ** pptrBint,
            int shift_amount )
```

Performs a left shift operation on a BINT object by a specified number of bits.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the BINT object pointer to be shifted.
shift_amount	The number of bits to shift the BINT object by.

Note

This operation can increase the size of the BINT object if the shift goes beyond the current most significant bit.

4.4.2.19 left_shift_word()

```
void left_shift_word (
            BINT ** pptrBint,
            int shift_amount )
```

Performs a left shift operation on a BINT object by a specified number of words.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the BINT object pointer to be shifted.
shift_amount	The number of words to shift the BINT object by.

Note

This operation is equivalent to multiplying the BINT object by 2^(WORD_SIZE * shift_amount).

4.4.2.20 makeEven()

```
void makeEven (
            BINT * ptrBint )
```

Modifies a BINT object to be an even number by clearing the least significant bit.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to be modified.

Note

If the BINT object is already even, the function will have no effect.

4.4.2.21 matchSize()

```
void matchSize (
            BINT * ptrBint1,
            BINT * ptrBint2 )
```

Ensures that two BINT objects have the same size by padding the smaller one with zeros.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint1	A pointer to the first BINT object.
ptrBint2	A pointer to the second BINT object.

Note

This can be important for operations that require BINT objects of the same size.

4.4.2.22 print_bint_bin()

```
void print_bint_bin (
            const BINT * ptrBint )
```

Prints the binary representation of a BINT object to standard output.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to print.

Note

Useful for debugging or displaying the binary form of a BINT object.

4.4.2.23 print_bint_bin_py()

```
void print_bint_bin_py (
            const BINT * ptrBint )
```

Prints the binary representation of a BINT object to standard output in a format compatible with Python lists.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to print.

Note

The output format is designed to be directly usable in Python code.

4.4.2.24 print_bint_bin_split()

```
void print_bint_bin_split (
            const BINT * ptrBint )
```

Prints the binary representation of a BINT object to standard output with a split for readability.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to print.

Note

The binary output is split into sections for easier reading.

4.4.2.25 print_bint_hex()

```
void print_bint_hex (
            const BINT * ptrBint )
```

Prints the hexadecimal representation of a BINT object to standard output.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to print.

Note

Useful for debugging or displaying the hexadecimal form of a BINT object.

4.4.2.26 print_bint_hex_py()

```
void print_bint_hex_py (
            const BINT * ptrBint )
```

Prints the hexadecimal representation of a BINT object to standard output in a format compatible with Python lists.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to print.

Note

The output format is designed to be directly usable in Python code.

4.4.2.27 print_bint_hex_split()

```
void print_bint_hex_split (
            const BINT * ptrBint )
```

Prints the hexadecimal representation of a BINT object to standard output with a split for readability.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to print.

Note

The hexadecimal output is split into sections for easier reading.

4.4.2.28 PrintBinary()

```
void PrintBinary (
            bool * binary,
            int length )
```

Prints the binary representation of an array to standard output.

Author

Your Name

Date

Date of creation or last update

Parameters

binary	A pointer to the binary array to print.
length	The length of the binary array.

Note

Useful for debugging or displaying the binary form of data.

4.4.2.29 RANDOM_ARRARY()

```
void RANDOM_ARRARY (
            WORD * dst,
            int wordlen )
```

Fills an array with random WORD values.

Author

Your Name

Date

Date of creation or last update

Parameters

dst	A pointer to the WORD array to fill.
wordlen	The length of the WORD array.

Note

The randomness depends on the underlying random number generator.

4.4.2.30 RANDOM_BINT()

```
void RANDOM_BINT (
            BINT ** pptrBint,
            bool sign,
            int wordlen )
```

Initializes a BINT object with random values.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the pointer of the BINT object to be initialized.
sign	The sign of the BINT object, where true indicates a negative number.
wordlen	The word length for the BINT object.

Note

This function is useful for generating random BINT objects for testing.

4.4.2.31 reduction()

```
void reduction (
            BINT ** pptrBint,
            int pwOf2 )
```

Reduces a BINT object modulo a power of 2 specified by pwOf2.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the BINT object pointer to be reduced.
pwOf2	The power of 2 to use as the modulus for the reduction.

Note

The reduction is done in-place and affects the original BINT object.

4.4.2.32 refineBINT()

```
void refineBINT (
            BINT * ptrBint )
```

Refines the size of a BINT object to remove any leading zero words.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to refine.

Note

This function optimizes memory usage by a BINT object.

4.4.2.33 resetBINT()

```
void resetBINT (
            BINT * ptrBint )
```

Resets a BINT object to its default state with a value of zero and minimal size.

Author

Your Name

Date

Date of creation or last update

Parameters

ptrBint	A pointer to the BINT object to reset.

Note

This function can be used to reuse a BINT object without reallocating memory.

4.4.2.34 right_shift_word()

```
void right_shift_word (
            BINT ** pptrBint,
            int shift_amount )
```

Performs a right shift operation on a BINT object by a specified number of words.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the BINT object pointer to be shifted.
shift_amount	The number of words to shift the BINT object by.

Note

This operation is equivalent to dividing the BINT object by 2^(WORD_SIZE * shift_amount), without a remainder.

4.4.2.35 SET_BINT_ONE()

```
void SET_BINT_ONE (
            BINT ** pptrBint )
```

Sets the value of a BINT object to one (1).

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the pointer of the BINT object to be set to one.

Note

Assumes the BINT object has been properly initialized.

4.4.2.36 SET_BINT_ZERO()

```
void SET_BINT_ZERO (
            BINT ** pptrBint )
```

Sets the value of a BINT object to zero.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the pointer of the BINT object to be set to zero.

Note

Assumes the BINT object has been properly initialized.

4.4.2.37 strToBINT()

```
void strToBINT (
            BINT ** pptrBint,
            const char * hexString )
```

Converts a hexadecimal string to a BINT object.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint	A pointer to the pointer of the BINT object to be initialized.
hexString	The hexadecimal string to convert.

Note

The function initializes the BINT object with the value represented by the hexadecimal string.

4.4.2.38 swapBINT()

```
void swapBINT (
            BINT ** pptrBint1,
            BINT ** pptrBint2 )
```

Swaps the values of two BINT objects.

Author

Your Name

Date

Date of creation or last update

Parameters

pptrBint1	A pointer to the point of the first BINT object pointer.
pptrBint2	A pointer to the point of the second BINT object pointer.

Note

Assumes that both BINT objects have been properly initialized.

4.4.3 Variable Documentation

4.4.3.1 file

```
Continuing with the professional annotations for the remaining functions in your C header file
```

Definition at line 278 of file bigint_utils.h.

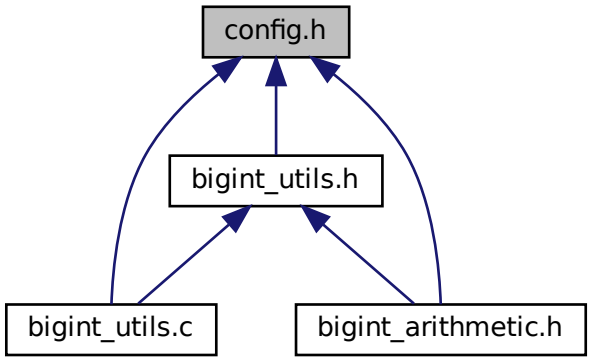
4.4.3.2 shift_amount

```
Continuing with the professional annotations for the remaining functions in your C header int
shift_amount
```

Definition at line 290 of file bigint_utils.h.

4.5 config.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct BINT

Macros

- #define false 0
- #define true !false
- #define FLAG 8
- #define MAXIMUM(x1, x2) (x1 > x2 ? x1 : x2)
- #define MINIMUM(x1, x2) (x1 < x2 ? x1 : x2)
- #define WORD_BITLEN 32

Typedefs

- typedef unsigned char u8
- typedef unsigned int u32
- typedef unsigned long u64
- typedef u32 WORD

4.5.1 Macro Definition Documentation

4.5.1.1 false

```
#define false 0
```

Definition at line 10 of file config.h.

4.5.1.2 FLAG

```
#define FLAG 8
```

Definition at line 14 of file config.h.

4.5.1.3 MAXIMUM

```
#define MAXIMUM(
              x1,
              x2 ) (x1 > x2 ?  x1 :  x2)
```

Definition at line 17 of file config.h.

4.5.1.4 MINIMUM

```
#define MINIMUM(
              x1,
              x2 ) (x1 < x2 ?  x1 :  x2)
```

Definition at line 18 of file config.h.

4.5.1.5 true

```
#define true !false
```

Definition at line 11 of file config.h.

4.5.1.6 WORD_BITLEN

```
#define WORD_BITLEN 32
```

Definition at line 21 of file config.h.

4.5.2 Typedef Documentation

4.5.2.1 u32

```
typedef unsigned int u32
```

Definition at line 6 of file config.h.

4.5.2.2 u64

```
typedef unsigned long u64
```

Definition at line 7 of file config.h.

4.5.2.3 u8

```
typedef unsigned char u8
```

Definition at line 5 of file config.h.

4.5.2.4 WORD

```
typedef u32 WORD
```

Definition at line 29 of file config.h.

Index