Aerobus

v1.2

Generated by Doxygen 1.9.1

1 Class Index	1
1.1 Class List	1
2 Class Documentation	3
2.1 aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, E > Struct Template Reference	3
$ 2.2 \ aerobus::polynomial < Ring, \ variable_name >::val < coeffN >::coeff_at < index, \ std::enable_if_ \leftrightarrow t < (index < 0 index > 0) > > Struct \ Template \ Reference$	3
2.3 aerobus::polynomial < Ring, variable_name >::val < coeffN >::coeff_at < index, std::enable_if_ \leftarrow t < (index==0) > > Struct Template Reference	3
2.4 aerobus::ContinuedFraction< values > Struct Template Reference	4
2.4.1 Detailed Description	4
2.5 aerobus::ContinuedFraction < a0 > Struct Template Reference	4
2.6 aerobus::ContinuedFraction < a0, rest > Struct Template Reference	4
2.7 aerobus::i32 Struct Reference	5
2.7.1 Detailed Description	6
2.8 aerobus::i64 Struct Reference	6
2.8.1 Detailed Description	8
2.9 aerobus::polynomial< Ring, variable_name >::eval_helper< valueRing, P >::inner< index, stop > Struct Template Reference	8
2.10 aerobus::polynomial< Ring, variable_name >::eval_helper< valueRing, P >::inner< stop, stop > Struct Template Reference	8
2.11 aerobus::is_prime< n > Struct Template Reference	8
2.11.1 Detailed Description	8
2.12 aerobus::polynomial < Ring, variable_name > Struct Template Reference	9
2.12.1 Detailed Description	10
2.12.2 Member Typedef Documentation	10
2.12.2.1 add_t	10
2.12.2.2 derive_t	11
2.12.2.3 div_t	11
2.12.2.4 eq_t	11
2.12.2.5 gcd_t	12
2.12.2.6 gt_t	12
2.12.2.7 lt_t	12
2.12.2.8 mod_t	13
2.12.2.9 monomial_t	13
2.12.2.10 mul_t	13
2.12.2.11 pos_t	14
2.12.2.12 simplify_t	14
2.12.2.13 sub_t	14
2.13 aerobus::type_list< Ts >::pop_front Struct Reference	15
2.14 aerobus::type_list< Ts >::split< index > Struct Template Reference	15
	15
2.15.1 Detailed Description	16

2.16 aerobus::type_list<> Struct Reference	16
2.17 aerobus::i32::val < x > Struct Template Reference	17
2.17.1 Detailed Description	17
2.17.2 Member Function Documentation	17
2.17.2.1 eval()	17
2.17.2.2 get()	18
2.18 aerobus::i64::val < x > Struct Template Reference	18
2.18.1 Detailed Description	19
2.18.2 Member Function Documentation	19
2.18.2.1 eval()	19
2.18.2.2 get()	19
$ 2.19 \ aerobus::polynomial < Ring, variable_name > ::val < coeffN, coeffs > Struct \ Template \ Reference \ . \ . \\$	20
2.19.1 Member Typedef Documentation	20
2.19.1.1 coeff_at_t	20
2.19.2 Member Function Documentation	21
2.19.2.1 eval()	21
2.19.2.2 to_string()	21
2.20 aerobus::zpz::val< x > Struct Template Reference	22
2.21 aerobus::polynomial < Ring, variable_name >::val < coeffN > Struct Template Reference	22
2.22 aerobus::zpz Struct Template Reference	23
2.22.1 Detailed Description	24
3 Example Documentation	25
3.1 i32::template	25
3.2 i64::template	25
3.3 polynomial	25
3.4 PI fraction::val	25
3.5 E_fraction::val	26
Index	27

Chapter 1

Class Index

1.1 Class List

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

aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, E >	3
aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, std::enable_if_t<(index< 0.3)	0 index > 0)>
aerobus::polynomial < Ring, variable_name >::val < coeffN >::coeff_at < index, std::enable_if_t < (index==0.0000000000000000000000000000000000	0)>>
3	
aerobus::ContinuedFraction< values >	
Continued fraction a0 + 1/(a1 + 1/())	4
aerobus::ContinuedFraction < a0 >	4
aerobus::ContinuedFraction < a0, rest >	4
aerobus::i32	
32 bits signed integers, seen as a algebraic ring with related operations	5
aerobus::i64	
64 bits signed integers, seen as a algebraic ring with related operations	6
aerobus::polynomial < Ring, variable_name >::eval_helper < valueRing, P >::inner < index, stop >	8
aerobus::polynomial < Ring, variable_name >::eval_helper < valueRing, P >::inner < stop, stop >	8
aerobus::is_prime< n >	
Checks if n is prime	8
aerobus::polynomial < Ring, variable_name >	9
aerobus::type_list< Ts >::pop_front	15
aerobus::type_list< Ts >::split< index >	15
aerobus::type_list< Ts >	
Empty pure template struct to handle type list	15
aerobus::type_list<>	16
aerobus::i32::val< x >	
Values in i32	17
aerobus::i64::val< x >	
Values in i64	18
aerobus::polynomial < Ring, variable_name >::val < coeffN, coeffs >	20
aerobus::zpz::val< x >	22
aerobus::polynomial< Ring, variable_name >::val< coeffN >	22
aerobus "znz< n >	23

2 Class Index

Chapter 2

Class Documentation

2.1 aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, E > Struct Template Reference

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

- src/lib.h
- 2.2 aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, std::enable_if_t<(index< 0||index>0> > Struct Template Reference

Public Types

• using type = typename Ring::zero

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

- · src/lib.h
- 2.3 aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, std::enable_if_t<(index==0)> > Struct Template Reference

Public Types

• using type = aN

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

• src/lib.h

2.4 aerobus::ContinuedFraction< values > Struct Template Reference

represents a continued fraction a0 + 1/(a1 + 1/(...))
#include <lib.h>

2.4.1 Detailed Description

template < int64_t... values > struct aerobus::ContinuedFraction < values > represents a continued fraction a0 + 1/(a1 + 1/(...))

Template Parameters

...values

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

· src/lib.h

2.5 aerobus::ContinuedFraction < a0 > Struct Template Reference

Public Types

using type = typename q64::template inject_constant_t< a0 >

Static Public Attributes

• static constexpr double **val** = type::template get<double>()

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

• src/lib.h

2.6 aerobus::ContinuedFraction< a0, rest... > Struct Template Reference

Public Types

• using **type** = q64::template add_t< typename q64::template inject_constant_t< a0 >, typename q64::template div_t< typename q64::one, typename ContinuedFraction< rest... >::type > >

Static Public Attributes

• static constexpr double **val** = type::template get<double>()

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

• src/lib.h

2.7 aerobus::i32 Struct Reference

32 bits signed integers, seen as a algebraic ring with related operations

```
#include <lib.h>
```

Classes

• struct val values in i32

Public Types

```
• using inner_type = int32_t
• using zero = val< 0 >
     constant zero

    using one = val< 1 >

     constant one

    template<auto x>

 using inject_constant_t = val< static_cast< int32_t >(x)>
• template<typename v >
 using inject_ring_t = v
• template<typename v1 , typename v2 >
 using add_t = typename add< v1, v2 >::type
     addition operator

    template<typename v1 , typename v2 >

 using sub_t = typename sub< v1, v2 >::type
     substraction operator

    template<typename v1 , typename v2 >

  using mul_t = typename mul < v1, v2 >::type
     multiplication operator

    template<typename v1 , typename v2 >

 using div_t = typename div < v1, v2 >::type
     division operator
• template<typename v1 , typename v2 >
  using mod_t = typename remainder < v1, v2 >::type
     modulus operator
• template<typename v1 , typename v2 >
  using gt_t = typename gt < v1, v2 >::type
     strictly greater operator (v1 > v2)
```

Static Public Attributes

```
    static constexpr bool is_field = false
        integers are not a field
    static constexpr bool is_euclidean_domain = true
        integers are an euclidean domain
```

2.7.1 Detailed Description

32 bits signed integers, seen as a algebraic ring with related operations

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

• src/lib.h

2.8 aerobus::i64 Struct Reference

64 bits signed integers, seen as a algebraic ring with related operations

```
#include <lib.h>
```

Classes

• struct val values in i64

Public Types

```
• using inner_type = int64_t

    template<auto x>

 using inject_constant_t = val< static_cast< int64_t >(x)>
• template<typename v >
 using inject_ring_t = v
using zero = val< 0 >
     constant zero
• using one = val< 1 >
     constant one

    template<typename v1 , typename v2 >

  using add_t = typename add< v1, v2 >::type
     addition operator

    template<typename v1 , typename v2 >

  using sub_t = typename sub< v1, v2 >::type
     substraction operator
• template<typename v1 , typename v2 >
  using mul_t = typename mul < v1, v2 >::type
     multiplication operator
• template<typename v1 , typename v2 >
  using div_t = typename div < v1, v2 >::type
     division operator
• template<typename v1 , typename v2 >
  using mod t = typename remainder < v1, v2 >::type
     modulus operator
• template<typename v1 , typename v2 >
  using gt_t = typename gt < v1, v2 >::type
     strictly greater operator (v1 > v2)

    template<typename v1 , typename v2 >

  using It_t = typename It< v1, v2 >::type
     strict less operator (v1 < v2)
• template<typename v1 , typename v2 >
  using eq_t = typename eq< v1, v2 >::type
     equality operator
• template<typename v1 , typename v2 >
  using gcd_t = gcd_t < i64, v1, v2 >
     greatest common divisor
• template<typename v >
  using pos_t = typename pos< v >::type
     is v posititive
```

Static Public Attributes

```
    static constexpr bool is_field = false
        integers are not a field
    static constexpr bool is_euclidean_domain = true
```

integers are an euclidean domain

2.8.1 Detailed Description

64 bits signed integers, seen as a algebraic ring with related operations

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

• src/lib.h

2.9 aerobus::polynomial < Ring, variable_name >::eval_helper < valueRing, P >::inner < index, stop > Struct Template Reference

Static Public Member Functions

• static constexpr valueRing func (const valueRing &accum, const valueRing &x)

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

· src/lib.h

2.10 aerobus::polynomial < Ring, variable_name >::eval_helper < valueRing, P >::inner < stop, stop > Struct Template Reference

Static Public Member Functions

• static constexpr valueRing func (const valueRing &accum, const valueRing &x)

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

• src/lib.h

2.11 aerobus::is_prime< n > Struct Template Reference

```
checks if n is prime
```

```
#include <lib.h>
```

Static Public Attributes

static constexpr bool value = internal::_is_prime<n, 5>::value
 true iff n is prime

2.11.1 Detailed Description

```
template<int32_t n>
struct aerobus::is_prime< n>
```

checks if n is prime

Template Parameters



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

• src/lib.h

2.12 aerobus::polynomial< Ring, variable_name > Struct Template Reference

```
#include <lib.h>
```

Classes

- struct val
- struct val< coeffN >

Public Types

```
    using zero = val< typename Ring::zero >

     constant zero
• using one = val< typename Ring::one >
     constant one
• using X = val< typename Ring::one, typename Ring::zero >
     generator

    template<typename P >

 using simplify_t = typename simplify< P >::type
     simplifies a polynomial (deletes highest degree if null, do nothing otherwise)
• template<typename v1 , typename v2 >
  using add_t = typename add< v1, v2 >::type
     adds two polynomials
• template<typename v1 , typename v2 >
  using sub_t = typename sub< v1, v2 >::type
     substraction of two polynomials
• template<typename v1 , typename v2 >
  using mul_t = typename mul < v1, v2 >::type
     multiplication of two polynomials
• template<typename v1 , typename v2 >
  using eq_t = typename eq_helper< v1, v2 >::type
     equality operator
• template<typename v1 , typename v2 >
  using lt_t = typename lt_helper< v1, v2 >::type
     strict less operator
• template<typename v1 , typename v2 >
  using gt_t = typename gt_helper< v1, v2 >::type
     strict greater operator
```

```
• template<typename v1 , typename v2 >
  using div_t = typename div < v1, v2 >::q_type
     division operator

    template<typename v1 , typename v2 >

  using mod_t = typename div_helper< v1, v2, zero, v1 >::mod_type
     modulo operator
• template<typename coeff , size_t deg>
 using monomial_t = typename monomial < coeff, deg >::type
     monomial : coeff X^{\wedge} deg
• template<typename v >
  using derive_t = typename derive_helper< v >::type
     derivation operator
• template<typename v >
  using pos_t = typename Ring::template pos_t < typename v::aN >
     checks for positivity (an > 0)
• template<typename v1, typename v2 >
  using gcd_t = std::conditional_t < Ring::is_euclidean_domain, typename make_unit < gcd_t < polynomial <
  Ring, variable_name >, v1, v2 > >::type, void >
     greatest common divisor of two polynomials

    template<auto x>

  using inject_constant_t = val< typename Ring::template inject_constant_t < x > >
• template<typename v >
  using inject_ring_t = val< v >
```

Static Public Attributes

- static constexpr bool is_field = false
- static constexpr bool is_euclidean_domain = Ring::is_euclidean_domain

2.12.1 Detailed Description

```
template<typename Ring, char variable_name = 'x'> struct aerobus::polynomial< Ring, variable_name >
```

polynomial with coefficients in Ring Ring must be an integral domain

2.12.2 Member Typedef Documentation

2.12.2.1 add t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::add_t = typename add<v1, v2>::type
```

adds two polynomials

Template Parameters

v1	
v2	

2.12.2.2 derive t

```
template<typename Ring , char variable_name = 'x'>
template<typename v >
using aerobus::polynomial< Ring, variable_name >::derive_t = typename derive_helper<v>::type
```

derivation operator

Template Parameters

V	

2.12.2.3 div_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::div_t = typename div<v1, v2>::q_type
```

division operator

Template Parameters

v1	
v2	

2.12.2.4 eq_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::eq_t = typename eq_helper<v1, v2>::type
```

equality operator

Template Parameters

v1	
v2	

2.12.2.5 gcd_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::gcd_t = std::conditional_t< Ring::is_←
euclidean_domain, typename make_unit<gcd_t<polynomial<Ring, variable_name>, v1, v2> >::type, void>
```

greatest common divisor of two polynomials

Template Parameters

v1	
v2	

2.12.2.6 gt_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::gt_t = typename gt_helper<v1, v2>::type
```

strict greater operator

Template Parameters

v1	
v2	

2.12.2.7 lt_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::lt_t = typename lt_helper<v1, v2>::type
```

strict less operator

Template Parameters

v1	
v2	

2.12.2.8 mod t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::mod_t = typename div_helper<v1, v2, zero,
v1>::mod_type
```

modulo operator

Template Parameters

v1	
v2	

2.12.2.9 monomial_t

```
template<typename Ring , char variable_name = 'x'>
template<typename coeff , size_t deg>
using aerobus::polynomial< Ring, variable_name >::monomial_t = typename monomial<coeff, deg>
::type
```

monomial : coeff X^deg

Template Parameters

coeff	
deg	

2.12.2.10 mul_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::mul_t = typename mul<v1, v2>::type
```

multiplication of two polynomials

Template Parameters

v1	
v2	

2.12.2.11 pos t

```
template<typename Ring , char variable_name = 'x'>
template<typename v >
using aerobus::polynomial< Ring, variable_name >::pos_t = typename Ring::template pos_t<typename v::aN>
```

checks for positivity (an > 0)

Template Parameters



2.12.2.12 simplify_t

```
template<typename Ring , char variable_name = 'x'>
template<typename P >
using aerobus::polynomial< Ring, variable_name >::simplify_t = typename simplify<P>::type
```

simplifies a polynomial (deletes highest degree if null, do nothing otherwise)

Template Parameters



2.12.2.13 sub_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::sub_t = typename sub<v1, v2>::type
```

substraction of two polynomials

Template Parameters

v1	
v2	

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

• src/lib.h

2.13 aerobus::type_list< Ts >::pop_front Struct Reference

Public Types

- using **type** = typename internal::pop_front_h< Ts... >::head
- using **tail** = typename internal::pop_front_h< Ts... >::tail

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

· src/lib.h

2.14 aerobus::type_list< Ts >::split< index > Struct Template Reference

Public Types

- using **head** = typename inner::head
- using tail = typename inner::tail

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

· src/lib.h

2.15 aerobus::type_list< Ts > Struct Template Reference

Empty pure template struct to handle type list.

Classes

- struct pop_front
- struct split

Public Types

```
template<typename T > using push_front = type_list< T, Ts... >
template<uint64_t index> using at = internal::type_at_t< index, Ts... >
template<typename T > using push_back = type_list< Ts..., T >
template<typename U > using concat = typename concat_h< U >::type
template<uint64_t index, typename T > using insert = typename internal::insert_h< index, type_list< Ts... >, T >::type
template<uint64_t index> using remove = typename internal::remove_h< index, type_list< Ts... >>::type
```

Static Public Attributes

• static constexpr size_t length = sizeof...(Ts)

2.15.1 Detailed Description

```
template < typename... Ts> struct aerobus::type_list < Ts>
```

Empty pure template struct to handle type list.

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

• src/lib.h

2.16 aerobus::type_list<> Struct Reference

Public Types

```
    template < typename T > using push_front = type_list < T >
    template < typename T > using push_back = type_list < T >
    template < typename U > using concat = U
    template < uint64_t index, typename T > using insert = type_list < T >
```

Static Public Attributes

• static constexpr size_t length = 0

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

• src/lib.h

2.17 aerobus::i32::val < x > Struct Template Reference

```
#include <lib.h>
```

Public Types

values in i32

```
    using is_zero_t = std::bool_constant< x==0 >
        is value zero
```

Static Public Member Functions

```
    template < typename valueType > static constexpr valueType get ()
        cast x into valueType
    static std::string to_string ()
        string representation of value
    template < typename valueRing > static constexpr valueRing eval (const valueRing &v)
        cast x into valueRing
```

Static Public Attributes

• static constexpr int32_t v = x

2.17.1 Detailed Description

```
template < int32_t x>
struct aerobus::i32::val < x >

values in i32

Template Parameters
```

an actual integer

2.17.2 Member Function Documentation

2.17.2.1 eval()

```
template<int32_t x>
template<typename valueRing >
```

cast x into valueRing

Template Parameters

```
valueRing | double for example
```

2.17.2.2 get()

```
template<iint32_t x>
template<typename valueType >
static constexpr valueType aerobus::i32::val< x >::get ( ) [inline], [static], [constexpr]
```

cast x into valueType

Template Parameters

```
valueType | double for example
```

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

• src/lib.h

2.18 aerobus::i64::val < x > Struct Template Reference

```
values in i64
```

```
#include <lib.h>
```

Public Types

```
    using is_zero_t = std::bool_constant< x==0 >
        is value zero
```

Static Public Member Functions

```
    template<typename valueType > static constexpr valueType get ()
        cast value in valueType
    static std::string to_string ()
        string representation
    template<typename valueRing > static constexpr valueRing eval (const valueRing &v)
        cast value in valueRing
```

Static Public Attributes

• static constexpr int64_t **v** = x

2.18.1 Detailed Description

```
template < int64_t x> struct aerobus::i64::val < x> values in i64
```

Template Parameters

```
x an actual integer
```

2.18.2 Member Function Documentation

2.18.2.1 eval()

cast value in valueRing

Template Parameters

```
valueRing (double for example)
```

2.18.2.2 get()

```
template<int64_t x>
template<typename valueType >
static constexpr valueType aerobus::i64::val< x >::get ( ) [inline], [static], [constexpr]
```

cast value in valueType

Template Parameters

```
valueType (double for example)
```

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

· src/lib.h

2.19 aerobus::polynomial< Ring, variable_name >::val< coeffN, coeffs > Struct Template Reference

Public Types

```
    using aN = coeffN
        heavy weight coefficient (non zero)
    using strip = val < coeffs... >
        using is_zero_t = std::bool_constant < (degree==0) &&(aN::is_zero_t::value) >
            true if polynomial is constant zero
    template < size_t index >
        using coeff_at_t = typename coeff_at < index >::type
            coefficient at index
```

Static Public Member Functions

```
    static std::string to_string ()
        get a string representation of polynomial
    template<typename valueRing >
        static constexpr valueRing eval (const valueRing &x)
        evaluates polynomial seen as a function operating on ValueRing
```

Static Public Attributes

static constexpr size_t degree = sizeof...(coeffs)
 degree of the polynomial

2.19.1 Member Typedef Documentation

2.19.1.1 coeff_at_t

```
template<typename Ring , char variable_name = 'x'>
template<typename coeffN , typename... coeffs>
template<size_t index>
using aerobus::polynomial< Ring, variable_name >::val< coeffN, coeffs >::coeff_at_t = typename
coeff_at<index>::type
```

coefficient at index

Template Parameters

index	

2.19.2 Member Function Documentation

2.19.2.1 eval()

evaluates polynomial seen as a function operating on ValueRing

Template Parameters

valueRing	usually float or double
-----------	-------------------------

Parameters

x value

Returns

P(x)

2.19.2.2 to_string()

```
template<typename Ring , char variable_name = 'x'> template<typename coeffN , typename... coeffs> static std::string aerobus::polynomial< Ring, variable_name >::val< coeffN, coeffs >::to_\leftrightarrow string ( ) [inline], [static]
```

get a string representation of polynomial

Returns

```
something like a_n X^n + ... + a_1 X + a_0
```

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

• src/lib.h

2.20 aerobus::zpz::val< x > Struct Template Reference

Public Types

• using is zero t = std::bool constant< x% p==0 >

Static Public Member Functions

- template<typename valueType >
 static constexpr valueType get ()
- static std::string to_string ()
- template<typename valueRing >
 static constexpr valueRing eval (const valueRing &v)

Static Public Attributes

• static constexpr int32_t v = x % p

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

• src/lib.h

2.21 aerobus::polynomial< Ring, variable_name >::val< coeffN > Struct Template Reference

Classes

- struct coeff_at
- struct coeff_at < index, std::enable_if_t < (index < 0||index > 0) > >
- struct coeff_at< index, std::enable_if_t<(index==0)>>

Public Types

- using **aN** = coeffN
- using strip = val< coeffN >
- using is_zero_t = std::bool_constant< aN::is_zero_t::value >
- template < size_t index > using coeff_at_t = typename coeff_at < index >::type

Static Public Member Functions

- static std::string to_string ()
- template<typename valueRing >
 static constexpr valueRing eval (const valueRing &x)

Static Public Attributes

• static constexpr size_t degree = 0

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

• src/lib.h

2.22 aerobus::zpz Struct Template Reference

```
#include <lib.h>
```

Classes

struct val

Public Types

```
• using inner_type = int32_t

    template<auto x>

  using inject_constant_t = val< static_cast< int32_t >(x)>
• using zero = val < 0 >
• using one = val< 1 >
• template<typename v1 , typename v2 >
  using add_t = typename add< v1, v2 >::type
• template<typename v1 , typename v2 >
  using sub_t = typename sub< v1, v2 >::type
• template<typename v1 , typename v2 >
  using mul_t = typename mul < v1, v2 >::type
• template<typename v1 , typename v2 >
  using div_t = typename div< v1, v2 >::type

    template<typename v1 , typename v2 >

  using mod_t = typename remainder< v1, v2 >::type

    template<typename v1 , typename v2 >

  using gt_t = typename gt < v1, v2 >::type

    template<typename v1 , typename v2 >

  using It_t = typename It< v1, v2 >::type
• template<typename v1 , typename v2 >
  using eq_t = typename eq< v1, v2 >::type
• template<typename v1 , typename v2 >
  using gcd_t = gcd_t < i32, v1, v2 >
• template<typename v1 >
  using pos_t = typename pos< v1 >::type
```

Static Public Attributes

- static constexpr bool is_field = is_prime::value
- static constexpr bool is_euclidean_domain = true

2.22.1 Detailed Description

$$\label{eq:control_to_p} \begin{split} \text{template} &< \text{int32_t p} > \\ \text{struct aerobus::zpz} &< \text{p} > \end{split}$$

congruence classes of integers for a modulus if p is prime, zpz is a field, otherwise an integral domain with all related operations

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

• src/lib.h

Chapter 3

Example Documentation

3.1 i32::template

inject a native constant

Template Parameters

x inject_constant_2<2> -> i32::template val<2>

3.2 i64::template

injects constant as an i64 value

Template Parameters

x inject_constant_t<2>

3.3 polynomial

makes the constant (native type) polynomial a_0

Template Parameters

x <i32>::template inject_constant_t<2>

3.4 PI_fraction::val

representation of PI as a continued fraction -> 3.14...

3.5 E_fraction::val

approximation of e -> 2.718...

Index

```
add t
                                                         aerobus::type_list< Ts >::split< index >, 15
     aerobus::polynomial < Ring, variable_name >, 10
                                                         aerobus::type_list<>, 16
aerobus::ContinuedFraction < a0 >, 4
                                                         aerobus::zpz , 23
aerobus::ContinuedFraction < a0, rest... >, 4
                                                         aerobus::zpz<p>::val<math><x>, 22
aerobus::ContinuedFraction < values >, 4
                                                         coeff at t
aerobus::i32, 5
                                                                                                 variable_name
                                                              aerobus::polynomial<
                                                                                       Ring,
aerobus::i32::val< x >, 17
                                                                   >::val< coeffN, coeffs >, 20
    eval, 17
    get, 18
                                                         derive t
aerobus::i64, 6
                                                              aerobus::polynomial < Ring, variable_name >, 11
aerobus::i64::val < x > 18
                                                         div t
    eval, 19
                                                              aerobus::polynomial < Ring, variable_name >, 11
    get, 19
aerobus::is_prime< n >, 8
                                                         eq_t
aerobus::polynomial < Ring, variable name >, 9
                                                              aerobus::polynomial < Ring, variable name >, 11
    add_t, 10
                                                         eval
    derive_t, 11
                                                              aerobus::i32::val< x >, 17
    div_t, 11
                                                              aerobus::i64::val < x > 19
    eq_t, 11
                                                              aerobus::polynomial<
                                                                                                 variable name
                                                                                       Ring,
    gcd_t, 12
                                                                   >::val< coeffN, coeffs >, 21
    gt_t, 12
    It t, 12
                                                         gcd_t
    mod t, 13
                                                              aerobus::polynomial < Ring, variable_name >, 12
    monomial_t, 13
                                                         get
    mul t, 13
                                                              aerobus::i32::val < x >, 18
    pos t, 14
                                                              aerobus::i64::val < x >, 19
    simplify_t, 14
                                                         gt_t
    sub_t, 14
                                                              aerobus::polynomial < Ring, variable name >, 12
aerobus::polynomial < Ring, variable_name >::eval_helper <
         valueRing, P >::inner< index, stop >, 8
aerobus::polynomial < Ring, variable name >::eval helper <
                                                              aerobus::polynomial < Ring, variable name >, 12
         valueRing, P >::inner< stop, stop >, 8
aerobus::polynomial < Ring, variable_name >::val < co-
                                                       mod t
         effN >, 22
                                                              aerobus::polynomial < Ring, variable name >, 13
aerobus::polynomial < Ring, variable_name >::val < co-
                                                        monomial t
         effN >::coeff_at< index, E >, 3
                                                              aerobus::polynomial < Ring, variable_name >, 13
aerobus::polynomial < Ring, variable_name >::val < co-
         effN >::coeff_at< index, std::enable_if_t<(index<
                                                              aerobus::polynomial < Ring, variable_name >, 13
         0 \mid | index > 0) > 3
aerobus::polynomial < Ring, variable_name >::val < co- pos_t
         effN>::coeff\_at<index,std::enable\_if\_t<(index==0)\\ \geqslant aerobus::polynomial<Ring,variable\_name>,14
         >, 3
                                                         simplify_t
aerobus::polynomial < Ring, variable name >::val < co-
                                                              aerobus::polynomial < Ring, variable_name >, 14
         effN, coeffs >, 20
                                                         sub t
    coeff at t, 20
                                                              aerobus::polynomial < Ring, variable_name >, 14
    eval, 21
    to_string, 21
                                                         to string
aerobus::type_list< Ts >, 15
                                                              aerobus::polynomial<
                                                                                       Ring,
                                                                                                 variable name
aerobus::type_list< Ts >::pop_front, 15
                                                                   >::val< coeffN, coeffs >, 21
```