Contents

Standard Library

1 auxiliary

1.1 Benchmarking.sac

```
/*-----------------------------------------------*/

module Benchmarking;

export all;
import C99Benchmarking : all;

/*-----------------------------------------------*/
```

1.2 C99Benchmarking.sac

```
/*-----------------------------------------------*/

class C99Benchmarking;

import Interval : all;
use String : { string, sprintf, +};
use Array : all except { +};
use StdIO : all;
export { Interval, start, end, benchThis, returnResultUnit, printResult,
   ↪ getInterval, destroyInterval};


/*
 *  type definitions
 */

external classtype C99Benchmarking::C99Benchmarking;


/*
 *  prototypes for externals (FUNDECS)
 */

external C99Benchmarking create_TheBenchmarkObject();
#pragma linkname "benchCreate"
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect World::TheWorld

external Interval _getInterval_u( string interval_name, int interval_number, int
   ↪  unit_time);
#pragma linkname "benchGetInterval_siu"
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject

external Interval _getInterval( string interval_name, int interval_number);
#pragma linkname "benchGetInterval_si"
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject

external Interval _getInterval( string interval_name);
```

```
#pragma linkname "benchGetInterval_s"
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject

external Interval _getInterval( int interval_number);
#pragma linkname "benchGetInterval_i"
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject

external string benchUnitType( Interval &interval);
#pragma linkname "benchUnitName"
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject

external void  benchStart( Interval &interval, double timestamp);
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma effect C99Benchmarking::TheBenchmarkObject

external void  benchEnd( Interval &interval, double timestamp);
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma effect C99Benchmarking::TheBenchmarkObject

external int benchUnit( Interval &interval);
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject

external double benchRes( Interval &interval);
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject

external int benchNum( Interval &interval);
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject

external string benchName( Interval &interval);
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject

external void  benchThis();
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma effect C99Benchmarking::TheBenchmarkObject

external void  benchDestroyInterval( Interval interval);
#pragma linkname "benchDestroyInterval"
#pragma linkobj "src/C99Benchmarking/bench.o"
#pragma linksign []
#pragma effect C99Benchmarking::TheBenchmarkObject



/*
 *  global objects
 */

C99Benchmarking TheBenchmarkObject = create_TheBenchmarkObject() ;
```

```
/*
 *  function definitions (FUNDEFS)
 */


/*****************************************************************************
 * getInterval(...) [ body ]
 *****************************************************************************/
Interval getInterval( string interval_name, int interval_number, int unit_time)


/*****************************************************************************
 * getInterval(...) [ body ]
 *****************************************************************************/
Interval getInterval( string interval_name, int interval_number)


/*****************************************************************************
 * getInterval(...) [ body ]
 *****************************************************************************/
Interval getInterval( string interval_name)


/*****************************************************************************
 * getInterval(...) [ body ]
 *****************************************************************************/
Interval getInterval( int interval_number)


/*****************************************************************************
 * benchTime(...) [ body ]
 *****************************************************************************/
double benchTime( int unit_time)


/*****************************************************************************
 * start(...) [ body ]
 *****************************************************************************/
void  start( Interval &interval)


/*****************************************************************************
 * end(...) [ body ]
 *****************************************************************************/
void  end( Interval &interval)
```

```
/*****************************************************************************
 * returnResultUnit(...) [ body ]
 *****************************************************************************/
double, string returnResultUnit( Interval &int1)



/*****************************************************************************
 * printResult(...) [ body ]
 *****************************************************************************/
void  printResult( Interval &int1)



/*****************************************************************************
 * printResult(...) [ body ]
 *****************************************************************************/
void  printResult( Interval &int1, Interval &int2)



/*****************************************************************************
 * printResult(...) [ body ]
 *****************************************************************************/
void  printResult( Interval &int1, Interval &int2, Interval &int3)



/*****************************************************************************
 * printResult(...) [ body ]
 *****************************************************************************/
void  printResult( Interval &int1, Interval &int2, Interval &int3, Interval &
   ↪ int4)



/*****************************************************************************
 * printResult(...) [ body ]
 *****************************************************************************/
void  printResult( Interval &int1, Interval &int2, Interval &int3, Interval &
   ↪ int4, Interval &int5)



/*****************************************************************************
 * printResult(...) [ body ]
 *****************************************************************************/
void  printResult( Interval &int1, Interval &int2, Interval &int3, Interval &
   ↪ int4, Interval &int5, Interval &int6)
```

```
/******************************************************************************
 * printResult(...) [ body ]
 ******************************************************************************/
void  printResult( Interval &int1, Interval &int2, Interval &int3, Interval &
   ↪ int4, Interval &int5, Interval &int6, Interval &int7)



/******************************************************************************
 * printResults(...) [ body ]
 ******************************************************************************/
void  printResults( StringArray::stringArray names, double[.] results)



/******************************************************************************
 * destroyInterval(...) [ body ]
 ******************************************************************************/
void  destroyInterval( Interval interval)


/*--------------------------------------------*/
```

1.3 Hiding.sac

```
/*--------------------------------------------*/

module Hiding;

export all;
use Array : all;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * hideValue(...) [ body ]
 ******************************************************************************/
inline
int[*] hideValue( int[*] in)



/******************************************************************************
 * hideValue(...) [ body ]
 ******************************************************************************/
inline
int[*] hideValue( int i, int[*] in)



/******************************************************************************
 * hideValue(...) [ body ]
```

```
 ******************************************************************************/
inline
char[*] hideValue( char[*] in)



/******************************************************************************
 * hideValue(...) [ body ]
 ******************************************************************************/
inline
char[*] hideValue( int i, char[*] in)



/******************************************************************************
 * hideValue(...) [ body ]
 ******************************************************************************/
inline
float[*] hideValue( float[*] in)



/******************************************************************************
 * hideValue(...) [ body ]
 ******************************************************************************/
inline
float[*] hideValue( int i, float[*] in)



/******************************************************************************
 * hideValue(...) [ body ]
 ******************************************************************************/
inline
double[*] hideValue( double[*] in)



/******************************************************************************
 * hideValue(...) [ body ]
 ******************************************************************************/
inline
double[*] hideValue( int i, double[*] in)



/******************************************************************************
 * hideValue(...) [ body ]
 ******************************************************************************/
inline
bool[*] hideValue( bool[*] in)



/******************************************************************************
```

```
 * hideValue(...) [ body ]
 ******************************************************************************/
inline
bool[*] hideValue( int i, bool[*] in)



/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
int[*] hideShape( int[*] in)



/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
int[*] hideShape( int i, int[*] in)



/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
char[*] hideShape( char[*] in)



/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
char[*] hideShape( int i, char[*] in)



/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
float[*] hideShape( float[*] in)



/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
float[*] hideShape( int i, float[*] in)
```

```
/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
double[*] hideShape( double[*] in)



/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
double[*] hideShape( int i, double[*] in)



/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
bool[*] hideShape( bool[*] in)



/******************************************************************************
 * hideShape(...) [ body ]
 ******************************************************************************/
inline
bool[*] hideShape( int i, bool[*] in)



/******************************************************************************
 * hideDimensionality(...) [ body ]
 ******************************************************************************/
inline
int[*] hideDimensionality( int[*] in)



/******************************************************************************
 * hideDimensionality(...) [ body ]
 ******************************************************************************/
inline
int[*] hideDimensionality( int i, int[*] in)



/******************************************************************************
 * hideDimensionality(...) [ body ]
 ******************************************************************************/
inline
char[*] hideDimensionality( char[*] in)
```

```
/******************************************************************************
 * hideDimensionality(...) [ body ]
 *****************************************************************************/
inline
char[*] hideDimensionality( int i, char[*] in)



/******************************************************************************
 * hideDimensionality(...) [ body ]
 *****************************************************************************/
inline
float[*] hideDimensionality( float[*] in)



/******************************************************************************
 * hideDimensionality(...) [ body ]
 *****************************************************************************/
inline
float[*] hideDimensionality( int i, float[*] in)



/******************************************************************************
 * hideDimensionality(...) [ body ]
 *****************************************************************************/
inline
double[*] hideDimensionality( double[*] in)



/******************************************************************************
 * hideDimensionality(...) [ body ]
 *****************************************************************************/
inline
double[*] hideDimensionality( int i, double[*] in)



/******************************************************************************
 * hideDimensionality(...) [ body ]
 *****************************************************************************/
inline
bool[*] hideDimensionality( bool[*] in)



/******************************************************************************
 * hideDimensionality(...) [ body ]
 *****************************************************************************/
inline
bool[*] hideDimensionality( int i, bool[*] in)
```

```
/*-----------------------------------------------*/
```

1.4 Interval.sac

```
/*-----------------------------------------------*/

class Interval;

export { Interval};


/*
 *  type definitions
 */

external classtype Interval::Interval;

/*-----------------------------------------------*/
```

2 classes

2.1 auxiliary

2.1.1 Counter.sac

```
/*-----------------------------------------------*/

class Counter;

use ScalarArith : { +};
export all;


/*
 *  type definitions
 */

classtype int Counter::Counter;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * newCounter(...) [ body ]
 *****************************************************************************/
Counter newCounter()



/******************************************************************************
 * newCounter(...) [ body ]
 *****************************************************************************/
Counter newCounter( int v)
```

```
/*****************************************************************************
 * valueOf(...) [ body ]
 *****************************************************************************/
int valueOf( Counter &cnt)



/*****************************************************************************
 * increment(...) [ body ]
 *****************************************************************************/
void  increment( Counter &cnt)



/*****************************************************************************
 * incrementBy(...) [ body ]
 *****************************************************************************/
void  incrementBy( Counter &cnt, int x)



/*****************************************************************************
 * next(...) [ body ]
 *****************************************************************************/
int next( Counter &cnt)


/*-----------------------------------------------*/
```

2.2 random

2.2.1 Rand.sac

```
/*-----------------------------------------------*/

class Rand;

export all;


/*
 *  type definitions
 */

external classtype Rand::Rand;


/*
 *  prototypes for externals (FUNDECS)
 */

external Rand randGen();
#pragma linkname "create_RandGen"
#pragma linkobj "src/Rand/Rand.o"
#pragma linksign []

external void  srandom( uint SEED);
#pragma linkname "srand"
```

```
#pragma linkobj "src/Rand/Rand.o"
#pragma effect Rand::RandomGen

external int random( int MIN, int MAX);
#pragma linkname "SACrand"
#pragma linkobj "src/Rand/Rand.o"
#pragma linksign []
#pragma effect Rand::RandomGen

external int rand();
#pragma linkname "SACcrand"
#pragma linkobj "src/Rand/Rand.o"
#pragma linksign []
#pragma effect Rand::RandomGen

external double random( double MIN, double MAX);
#pragma linkname "SACdrand"
#pragma linkobj "src/Rand/Rand.o"
#pragma linksign []
#pragma effect Rand::RandomGen



/*
 *  global objects
 */

Rand RandomGen = randGen() ;



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * random(...) [ body ]
 ******************************************************************************/
int[*] random( int[.] shp, int MIN, int MAX)



/******************************************************************************
 * random(...) [ body ]
 ******************************************************************************/
double[*] random( int[.] shp, double MIN, double MAX)



/*--------------------------------------------*/
```

2.2.2 Rand48.sac

```
/*--------------------------------------------*/

class Rand48;

export all;
```

```
/*
 *  type definitions
 */

external classtype Rand48::Rand48;


/*
 *  prototypes for externals (FUNDECS)
 */

external Rand48 randGen();
#pragma linkname "create_Rand48Gen"
#pragma linkobj "src/Rand48/Rand48.o"
#pragma linksign []

external void  srandom( long SEED);
#pragma linkname "srand48"
#pragma linkobj "src/Rand48/Rand48.o"
#pragma effect Rand48::RandomGen

external int random( int MIN, int MAX);
#pragma linkname "SACrand48"
#pragma linkobj "src/Rand48/Rand48.o"
#pragma linksign []
#pragma effect Rand48::RandomGen

external double random( double MIN, double MAX);
#pragma linkname "SACdrand48"
#pragma linkobj "src/Rand48/Rand48.o"
#pragma linksign []
#pragma effect Rand48::RandomGen



/*
 *  global objects
 */

Rand48 RandomGen = randGen() ;



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * random(...) [ body ]
 *****************************************************************************/
int[*] random( int[.] shp, int MIN, int MAX)



/******************************************************************************
 * random(...) [ body ]
 *****************************************************************************/
double[*] random( int[.] shp, double MIN, double MAX)
```

```
/*--------------------------------------------*/
```

2.2.3 RandLC.sac

```
/*--------------------------------------------*/

class RandLC;

use ScalarArith : all;
use ArrayBasics : all;
export all;


/*
 *  type definitions
 */

classtype double[6] RandLC::RandLC;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * create_randlc(...) [ body ]
 ******************************************************************************/
RandLC create_randlc( double x, double a)




/******************************************************************************
 * delete_randlc(...) [ body ]
 ******************************************************************************/
void  delete_randlc( RandLC randlc)




/******************************************************************************
 * randlc(...) [ body ]
 ******************************************************************************/
double randlc( RandLC &randlc)



/*--------------------------------------------*/
```

2.2.4 Random.sac

```
/*--------------------------------------------*/

class Random;

export all;
```

```
/*
 *  type definitions
 */

external classtype Random::Random;


/*
 *  prototypes for externals (FUNDECS)
 */

external void  srandom( uint SEED);
#pragma linkobj "src/Random/Random.o"
#pragma effect Random::RandomGen

external int random( int MIN, int MAX);
#pragma linkname "SACrandom"
#pragma linkobj "src/Random/Random.o"
#pragma linksign []
#pragma effect Random::RandomGen

external double random( double MIN, double MAX);
#pragma linkname "SACdrandom"
#pragma linkobj "src/Random/Random.o"
#pragma linksign []
#pragma effect Random::RandomGen



/*
 *  global objects
 */

Random RandomGen;



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * random(...) [ body ]
 *****************************************************************************/
int[*] random( int[.] shp, int MIN, int MAX)



/******************************************************************************
 * random(...) [ body ]
 *****************************************************************************/
double[*] random( int[.] shp, double MIN, double MAX)



/*--------------------------------------------*/
```

3 numerical

3.1 ComplexMath.sac

```
/*-----------------------------------------------*/

module ComplexMath;

export all;
use ComplexBasics : all;
use ComplexScalarArith : all;
use ArrayBasics : { sel};


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * sin(...) [ body ]
 ******************************************************************************/
inline
complex sin( complex X)




/******************************************************************************
 * cos(...) [ body ]
 ******************************************************************************/
inline
complex cos( complex X)




/******************************************************************************
 * tan(...) [ body ]
 ******************************************************************************/
inline
complex tan( complex X)




/******************************************************************************
 * asin(...) [ body ]
 ******************************************************************************/
inline
complex asin( complex X)




/******************************************************************************
 * acos(...) [ body ]
 ******************************************************************************/
inline
complex acos( complex X)
```

```
/*****************************************************************************
 * atan(...) [ body ]
 *****************************************************************************/
inline
complex atan( complex X)



/*****************************************************************************
 * sinh(...) [ body ]
 *****************************************************************************/
inline
complex sinh( complex X)



/*****************************************************************************
 * cosh(...) [ body ]
 *****************************************************************************/
inline
complex cosh( complex X)



/*****************************************************************************
 * tanh(...) [ body ]
 *****************************************************************************/
inline
complex tanh( complex X)



/*****************************************************************************
 * asinh(...) [ body ]
 *****************************************************************************/
inline
complex asinh( complex X)



/*****************************************************************************
 * acosh(...) [ body ]
 *****************************************************************************/
inline
complex acosh( complex X)



/*****************************************************************************
 * atanh(...) [ body ]
 *****************************************************************************/
inline
complex atanh( complex X)
```

```
/*****************************************************************************
 * exp(...) [ body ]
 *****************************************************************************/
inline
complex exp( complex X)



/*****************************************************************************
 * log(...) [ body ]
 *****************************************************************************/
inline
complex log( complex X)



/*****************************************************************************
 * pow(...) [ body ]
 *****************************************************************************/
inline
complex pow( complex BASE, complex EXPON)



/*****************************************************************************
 * pow(...) [ body ]
 *****************************************************************************/
inline
complex pow( complex BASE, double EXPON)



/*****************************************************************************
 * pow(...) [ body ]
 *****************************************************************************/
inline
complex pow( double BASE, complex EXPON)



/*****************************************************************************
 * sqrt(...) [ body ]
 *****************************************************************************/
inline
complex sqrt( complex X)



/*-----------------------------------------------*/
```

3.2 FixedPoint.sac

```
/*-----------------------------------------------*/
```

```
module FixedPoint;

use SaCMath : all;
export all;


/*
 *  type definitions
 */

typedef int fixedpoint_32;
typedef int fixedpoint_24;
typedef int fixedpoint_16;
typedef int fixedpoint_8;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8 +( fixedpoint_8 a, fixedpoint_8 b)




/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] +( fixedpoint_8[+] a, fixedpoint_8[+] b)




/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] +( fixedpoint_8[+] a, fixedpoint_8 b)




/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] +( fixedpoint_8 a, fixedpoint_8[+] b)




/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
```

```
inline
fixedpoint_16 +( fixedpoint_16 a, fixedpoint_16 b)



/*****************************************************************************
 * +(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] +( fixedpoint_16[+] a, fixedpoint_16[+] b)



/*****************************************************************************
 * +(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] +( fixedpoint_16[+] a, fixedpoint_16 b)



/*****************************************************************************
 * +(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] +( fixedpoint_16 a, fixedpoint_16[+] b)



/*****************************************************************************
 * +(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24 +( fixedpoint_24 a, fixedpoint_24 b)



/*****************************************************************************
 * +(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] +( fixedpoint_24[+] a, fixedpoint_24[+] b)



/*****************************************************************************
 * +(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] +( fixedpoint_24[+] a, fixedpoint_24 b)



/*****************************************************************************
 * +(...) [ body ]
```

```
 ******************************************************************************/
inline
fixedpoint_24[+] +( fixedpoint_24 a, fixedpoint_24[+] b)



/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32 +( fixedpoint_32 a, fixedpoint_32 b)



/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] +( fixedpoint_32[+] a, fixedpoint_32[+] b)



/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] +( fixedpoint_32[+] a, fixedpoint_32 b)



/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] +( fixedpoint_32 a, fixedpoint_32[+] b)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8 -( fixedpoint_8 a, fixedpoint_8 b)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] -( fixedpoint_8[+] a, fixedpoint_8[+] b)



/******************************************************************************
```

```
 * -(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] -( fixedpoint_8[+] a, fixedpoint_8 b)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] -( fixedpoint_8 a, fixedpoint_8[+] b)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_16 -( fixedpoint_16 a, fixedpoint_16 b)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_16[+] -( fixedpoint_16[+] a, fixedpoint_16[+] b)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_16[+] -( fixedpoint_16[+] a, fixedpoint_16 b)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_16[+] -( fixedpoint_16 a, fixedpoint_16[+] b)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_24 -( fixedpoint_24 a, fixedpoint_24 b)
```

```
/*****************************************************************************
 * -(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] -( fixedpoint_24[+] a, fixedpoint_24[+] b)



/*****************************************************************************
 * -(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] -( fixedpoint_24[+] a, fixedpoint_24 b)



/*****************************************************************************
 * -(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] -( fixedpoint_24 a, fixedpoint_24[+] b)



/*****************************************************************************
 * -(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32 -( fixedpoint_32 a, fixedpoint_32 b)



/*****************************************************************************
 * -(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32[+] -( fixedpoint_32[+] a, fixedpoint_32[+] b)



/*****************************************************************************
 * -(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32[+] -( fixedpoint_32[+] a, fixedpoint_32 b)



/*****************************************************************************
 * -(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32[+] -( fixedpoint_32 a, fixedpoint_32[+] b)
```

```
/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8 /( fixedpoint_8 a, fixedpoint_8 b)
```

```
/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8[+] /( fixedpoint_8[+] a, fixedpoint_8[+] b)
```

```
/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8[+] /( fixedpoint_8[+] a, fixedpoint_8 b)
```

```
/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8[+] /( fixedpoint_8 a, fixedpoint_8[+] b)
```

```
/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16 /( fixedpoint_16 a, fixedpoint_16 b)
```

```
/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] /( fixedpoint_16[+] a, fixedpoint_16[+] b)
```

```
/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] /( fixedpoint_16[+] a, fixedpoint_16 b)
```

```
/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] /( fixedpoint_16 a, fixedpoint_16[+] b)



/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24 /( fixedpoint_24 a, fixedpoint_24 b)



/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] /( fixedpoint_24[+] a, fixedpoint_24[+] b)



/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] /( fixedpoint_24[+] a, fixedpoint_24 b)



/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] /( fixedpoint_24 a, fixedpoint_24[+] b)



/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32 /( fixedpoint_32 a, fixedpoint_32 b)



/*****************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32[+] /( fixedpoint_32[+] a, fixedpoint_32[+] b)
```

```
/******************************************************************************
 * /(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] /( fixedpoint_32[+] a, fixedpoint_32 b)



/******************************************************************************
 * /(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] /( fixedpoint_32 a, fixedpoint_32[+] b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8 *( fixedpoint_8 a, fixedpoint_8 b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] *( fixedpoint_8[+] a, fixedpoint_8[+] b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] *( fixedpoint_8[+] a, fixedpoint_8 b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] *( fixedpoint_8 a, fixedpoint_8[+] b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_16 *( fixedpoint_16 a, fixedpoint_16 b)
```

```
/*****************************************************************************
 * *(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] *( fixedpoint_16[+] a, fixedpoint_16[+] b)



/*****************************************************************************
 * *(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] *( fixedpoint_16[+] a, fixedpoint_16 b)



/*****************************************************************************
 * *(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] *( fixedpoint_16 a, fixedpoint_16[+] b)



/*****************************************************************************
 * *(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24 *( fixedpoint_24 a, fixedpoint_24 b)



/*****************************************************************************
 * *(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] *( fixedpoint_24[+] a, fixedpoint_24[+] b)



/*****************************************************************************
 * *(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] *( fixedpoint_24[+] a, fixedpoint_24 b)



/*****************************************************************************
 * *(...) [ body ]
 *****************************************************************************/
inline
```

```
fixedpoint_24[+] *( fixedpoint_24 a, fixedpoint_24[+] b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32 *( fixedpoint_32 a, fixedpoint_32 b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] *( fixedpoint_32[+] a, fixedpoint_32[+] b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] *( fixedpoint_32[+] a, fixedpoint_32 b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] *( fixedpoint_32 a, fixedpoint_32[+] b)



/******************************************************************************
 * sqrt(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8 sqrt( fixedpoint_8 arg)



/******************************************************************************
 * sqrt(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] sqrt( fixedpoint_8[+] a)



/******************************************************************************
 * sqrt(...) [ body ]
 ******************************************************************************/
```

```
inline
fixedpoint_16 sqrt( fixedpoint_16 arg)



/*****************************************************************************
 * sqrt(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] sqrt( fixedpoint_16[+] a)



/*****************************************************************************
 * sqrt(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24 sqrt( fixedpoint_24 arg)



/*****************************************************************************
 * sqrt(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] sqrt( fixedpoint_24[+] a)



/*****************************************************************************
 * sqrt(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32 sqrt( fixedpoint_32 arg)



/*****************************************************************************
 * sqrt(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32[+] sqrt( fixedpoint_32[+] a)



/*****************************************************************************
 * cbrt(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8 cbrt( fixedpoint_8 arg)



/*****************************************************************************
 * cbrt(...) [ body ]
```

```
 ******************************************************************************/
inline
fixedpoint_8[+] cbrt( fixedpoint_8[+] a)



/******************************************************************************
 * cbrt(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_16 cbrt( fixedpoint_16 arg)



/******************************************************************************
 * cbrt(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_16[+] cbrt( fixedpoint_16[+] a)



/******************************************************************************
 * cbrt(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_24 cbrt( fixedpoint_24 arg)



/******************************************************************************
 * cbrt(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_24[+] cbrt( fixedpoint_24[+] a)



/******************************************************************************
 * cbrt(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32 cbrt( fixedpoint_32 arg)



/******************************************************************************
 * cbrt(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] cbrt( fixedpoint_32[+] a)



/******************************************************************************
```

```
 * trunc(...) [ body ]
 ****************************************************************************/
inline
int trunc( fixedpoint_8 a)



/****************************************************************************
 * trunc(...) [ body ]
 ****************************************************************************/
inline
int[+] trunc( fixedpoint_8[+] a)



/****************************************************************************
 * trunc(...) [ body ]
 ****************************************************************************/
inline
int trunc( fixedpoint_16 a)



/****************************************************************************
 * trunc(...) [ body ]
 ****************************************************************************/
inline
int[+] trunc( fixedpoint_16[+] a)



/****************************************************************************
 * trunc(...) [ body ]
 ****************************************************************************/
inline
int trunc( fixedpoint_24 a)



/****************************************************************************
 * trunc(...) [ body ]
 ****************************************************************************/
inline
int[+] trunc( fixedpoint_24[+] a)



/****************************************************************************
 * trunc(...) [ body ]
 ****************************************************************************/
inline
int trunc( fixedpoint_32 a)
```

```
/*****************************************************************************
 * trunc(...) [ body ]
 *****************************************************************************/
inline
int[+] trunc( fixedpoint_32[+] a)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
int floor( fixedpoint_8 a)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
int[+] floor( fixedpoint_8[+] a)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
int floor( fixedpoint_16 a)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
int[+] floor( fixedpoint_16[+] a)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
int floor( fixedpoint_24 a)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
int[+] floor( fixedpoint_24[+] a)
```

```
/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
int floor( fixedpoint_32 a)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
int[+] floor( fixedpoint_32[+] a)



/*****************************************************************************
 * ceil(...) [ body ]
 *****************************************************************************/
inline
int ceil( fixedpoint_8 a)



/*****************************************************************************
 * ceil(...) [ body ]
 *****************************************************************************/
inline
int[+] ceil( fixedpoint_8[+] a)



/*****************************************************************************
 * ceil(...) [ body ]
 *****************************************************************************/
inline
int ceil( fixedpoint_16 a)



/*****************************************************************************
 * ceil(...) [ body ]
 *****************************************************************************/
inline
int[+] ceil( fixedpoint_16[+] a)



/*****************************************************************************
 * ceil(...) [ body ]
 *****************************************************************************/
inline
int ceil( fixedpoint_24 a)
```

```
/**************************************************************************
 * ceil(...) [ body ]
 **************************************************************************/
inline
int[+] ceil( fixedpoint_24[+] a)



/**************************************************************************
 * ceil(...) [ body ]
 **************************************************************************/
inline
int ceil( fixedpoint_32 a)



/**************************************************************************
 * ceil(...) [ body ]
 **************************************************************************/
inline
int[+] ceil( fixedpoint_32[+] a)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool <( fixedpoint_8 a, fixedpoint_8 b)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool[+] <( fixedpoint_8[+] a, fixedpoint_8[+] b)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool[+] <( fixedpoint_8[+] a, fixedpoint_8 b)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool[+] <( fixedpoint_8 a, fixedpoint_8[+] b)
```

```
/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool <( fixedpoint_16 a, fixedpoint_16 b)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool[+] <( fixedpoint_16[+] a, fixedpoint_16[+] b)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool[+] <( fixedpoint_16[+] a, fixedpoint_16 b)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool[+] <( fixedpoint_16 a, fixedpoint_16[+] b)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool <( fixedpoint_24 a, fixedpoint_24 b)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool[+] <( fixedpoint_24[+] a, fixedpoint_24[+] b)



/**************************************************************************
 * <(...) [ body ]
 **************************************************************************/
inline
bool[+] <( fixedpoint_24[+] a, fixedpoint_24 b)
```

```
/*****************************************************************************
 * <(...) [ body ]
 *****************************************************************************/
inline
bool[+] <( fixedpoint_24 a, fixedpoint_24[+] b)



/*****************************************************************************
 * <(...) [ body ]
 *****************************************************************************/
inline
bool <( fixedpoint_32 a, fixedpoint_32 b)



/*****************************************************************************
 * <(...) [ body ]
 *****************************************************************************/
inline
bool[+] <( fixedpoint_32[+] a, fixedpoint_32[+] b)



/*****************************************************************************
 * <(...) [ body ]
 *****************************************************************************/
inline
bool[+] <( fixedpoint_32[+] a, fixedpoint_32 b)



/*****************************************************************************
 * <(...) [ body ]
 *****************************************************************************/
inline
bool[+] <( fixedpoint_32 a, fixedpoint_32[+] b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool >( fixedpoint_8 a, fixedpoint_8 b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
```

```
bool[+] >( fixedpoint_8[+] a, fixedpoint_8[+] b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_8[+] a, fixedpoint_8 b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_8 a, fixedpoint_8[+] b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool >( fixedpoint_16 a, fixedpoint_16 b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_16[+] a, fixedpoint_16[+] b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_16[+] a, fixedpoint_16 b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_16 a, fixedpoint_16[+] b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
```

```
inline
bool >( fixedpoint_24 a, fixedpoint_24 b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_24[+] a, fixedpoint_24[+] b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_24[+] a, fixedpoint_24 b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_24 a, fixedpoint_24[+] b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool >( fixedpoint_32 a, fixedpoint_32 b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_32[+] a, fixedpoint_32[+] b)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool[+] >( fixedpoint_32[+] a, fixedpoint_32 b)



/*****************************************************************************
 * >(...) [ body ]
```

```
 ****************************************************************************/
inline
bool[+] >( fixedpoint_32 a, fixedpoint_32[+] b)



/****************************************************************************
 * ==(...) [ body ]
 ****************************************************************************/
inline
bool ==( fixedpoint_8 a, fixedpoint_8 b)



/****************************************************************************
 * ==(...) [ body ]
 ****************************************************************************/
inline
bool[+] ==( fixedpoint_8[+] a, fixedpoint_8[+] b)



/****************************************************************************
 * ==(...) [ body ]
 ****************************************************************************/
inline
bool[+] ==( fixedpoint_8[+] a, fixedpoint_8 b)



/****************************************************************************
 * ==(...) [ body ]
 ****************************************************************************/
inline
bool[+] ==( fixedpoint_8 a, fixedpoint_8[+] b)



/****************************************************************************
 * ==(...) [ body ]
 ****************************************************************************/
inline
bool ==( fixedpoint_16 a, fixedpoint_16 b)



/****************************************************************************
 * ==(...) [ body ]
 ****************************************************************************/
inline
bool[+] ==( fixedpoint_16[+] a, fixedpoint_16[+] b)



/****************************************************************************
```

```
 * ==(...) [ body ]
 ******************************************************************************/
inline
bool[+] ==( fixedpoint_16[+] a, fixedpoint_16 b)



/******************************************************************************
 * ==(...) [ body ]
 ******************************************************************************/
inline
bool[+] ==( fixedpoint_16 a, fixedpoint_16[+] b)



/******************************************************************************
 * ==(...) [ body ]
 ******************************************************************************/
inline
bool ==( fixedpoint_24 a, fixedpoint_24 b)



/******************************************************************************
 * ==(...) [ body ]
 ******************************************************************************/
inline
bool[+] ==( fixedpoint_24[+] a, fixedpoint_24[+] b)



/******************************************************************************
 * ==(...) [ body ]
 ******************************************************************************/
inline
bool[+] ==( fixedpoint_24[+] a, fixedpoint_24 b)



/******************************************************************************
 * ==(...) [ body ]
 ******************************************************************************/
inline
bool[+] ==( fixedpoint_24 a, fixedpoint_24[+] b)



/******************************************************************************
 * ==(...) [ body ]
 ******************************************************************************/
inline
bool ==( fixedpoint_32 a, fixedpoint_32 b)
```

```
/*****************************************************************************
 * ==(...) [ body ]
 *****************************************************************************/
inline
bool[+] ==( fixedpoint_32[+] a, fixedpoint_32[+] b)



/*****************************************************************************
 * ==(...) [ body ]
 *****************************************************************************/
inline
bool[+] ==( fixedpoint_32[+] a, fixedpoint_32 b)



/*****************************************************************************
 * ==(...) [ body ]
 *****************************************************************************/
inline
bool[+] ==( fixedpoint_32 a, fixedpoint_32[+] b)



/*****************************************************************************
 * <=(...) [ body ]
 *****************************************************************************/
inline
bool <=( fixedpoint_8 a, fixedpoint_8 b)



/*****************************************************************************
 * <=(...) [ body ]
 *****************************************************************************/
inline
bool[+] <=( fixedpoint_8[+] a, fixedpoint_8[+] b)



/*****************************************************************************
 * <=(...) [ body ]
 *****************************************************************************/
inline
bool[+] <=( fixedpoint_8[+] a, fixedpoint_8 b)



/*****************************************************************************
 * <=(...) [ body ]
 *****************************************************************************/
inline
bool[+] <=( fixedpoint_8 a, fixedpoint_8[+] b)
```

```
/**************************************************************************
 * <=(...) [ body ]
 **************************************************************************/
inline
bool <=( fixedpoint_16 a, fixedpoint_16 b)



/**************************************************************************
 * <=(...) [ body ]
 **************************************************************************/
inline
bool[+] <=( fixedpoint_16[+] a, fixedpoint_16[+] b)



/**************************************************************************
 * <=(...) [ body ]
 **************************************************************************/
inline
bool[+] <=( fixedpoint_16[+] a, fixedpoint_16 b)



/**************************************************************************
 * <=(...) [ body ]
 **************************************************************************/
inline
bool[+] <=( fixedpoint_16 a, fixedpoint_16[+] b)



/**************************************************************************
 * <=(...) [ body ]
 **************************************************************************/
inline
bool <=( fixedpoint_24 a, fixedpoint_24 b)



/**************************************************************************
 * <=(...) [ body ]
 **************************************************************************/
inline
bool[+] <=( fixedpoint_24[+] a, fixedpoint_24[+] b)



/**************************************************************************
 * <=(...) [ body ]
 **************************************************************************/
inline
bool[+] <=( fixedpoint_24[+] a, fixedpoint_24 b)
```

```
/******************************************************************************
 * <=(...) [ body ]
 ******************************************************************************/
inline
bool[+] <=( fixedpoint_24 a, fixedpoint_24[+] b)



/******************************************************************************
 * <=(...) [ body ]
 ******************************************************************************/
inline
bool <=( fixedpoint_32 a, fixedpoint_32 b)



/******************************************************************************
 * <=(...) [ body ]
 ******************************************************************************/
inline
bool[+] <=( fixedpoint_32[+] a, fixedpoint_32[+] b)



/******************************************************************************
 * <=(...) [ body ]
 ******************************************************************************/
inline
bool[+] <=( fixedpoint_32[+] a, fixedpoint_32 b)



/******************************************************************************
 * <=(...) [ body ]
 ******************************************************************************/
inline
bool[+] <=( fixedpoint_32 a, fixedpoint_32[+] b)



/******************************************************************************
 * >=(...) [ body ]
 ******************************************************************************/
inline
bool >=( fixedpoint_8 a, fixedpoint_8 b)



/******************************************************************************
 * >=(...) [ body ]
 ******************************************************************************/
inline
bool[+] >=( fixedpoint_8[+] a, fixedpoint_8[+] b)
```

```
/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_8[+] a, fixedpoint_8 b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_8 a, fixedpoint_8[+] b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool >=( fixedpoint_16 a, fixedpoint_16 b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_16[+] a, fixedpoint_16[+] b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_16[+] a, fixedpoint_16 b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_16 a, fixedpoint_16[+] b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool >=( fixedpoint_24 a, fixedpoint_24 b)
```

```
/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_24[+] a, fixedpoint_24[+] b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_24[+] a, fixedpoint_24 b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_24 a, fixedpoint_24[+] b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool >=( fixedpoint_32 a, fixedpoint_32 b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_32[+] a, fixedpoint_32[+] b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool[+] >=( fixedpoint_32[+] a, fixedpoint_32 b)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
```

```
bool[+] >=( fixedpoint_32 a, fixedpoint_32[+] b)



/*****************************************************************************
 * to_int(...) [ body ]
 *****************************************************************************/
inline
int to_int( fixedpoint_8 a)



/*****************************************************************************
 * to_int(...) [ body ]
 *****************************************************************************/
inline
int[+] to_int( fixedpoint_8[+] a)



/*****************************************************************************
 * to_int(...) [ body ]
 *****************************************************************************/
inline
int to_int( fixedpoint_16 a)



/*****************************************************************************
 * to_int(...) [ body ]
 *****************************************************************************/
inline
int[+] to_int( fixedpoint_16[+] a)



/*****************************************************************************
 * to_int(...) [ body ]
 *****************************************************************************/
inline
int to_int( fixedpoint_24 a)



/*****************************************************************************
 * to_int(...) [ body ]
 *****************************************************************************/
inline
int[+] to_int( fixedpoint_24[+] a)



/*****************************************************************************
 * to_int(...) [ body ]
 *****************************************************************************/
```

```
inline
int to_int( fixedpoint_32 a)



/*****************************************************************************
 * to_int(...) [ body ]
 *****************************************************************************/
inline
int[+] to_int( fixedpoint_32[+] a)



/*****************************************************************************
 * to_raw(...) [ body ]
 *****************************************************************************/
inline
int to_raw( fixedpoint_8 a)



/*****************************************************************************
 * to_raw(...) [ body ]
 *****************************************************************************/
inline
int[+] to_raw( fixedpoint_8[+] a)



/*****************************************************************************
 * to_raw(...) [ body ]
 *****************************************************************************/
inline
int to_raw( fixedpoint_16 a)



/*****************************************************************************
 * to_raw(...) [ body ]
 *****************************************************************************/
inline
int[+] to_raw( fixedpoint_16[+] a)



/*****************************************************************************
 * to_raw(...) [ body ]
 *****************************************************************************/
inline
int to_raw( fixedpoint_24 a)



/*****************************************************************************
 * to_raw(...) [ body ]
```

```
 ****************************************************************************/
inline
int[+] to_raw( fixedpoint_24[+] a)



/****************************************************************************
 * to_raw(...) [ body ]
 ****************************************************************************/
inline
int to_raw( fixedpoint_32 a)



/****************************************************************************
 * to_raw(...) [ body ]
 ****************************************************************************/
inline
int[+] to_raw( fixedpoint_32[+] a)



/****************************************************************************
 * to_fixedpoint_8(...) [ body ]
 ****************************************************************************/
inline
fixedpoint_8 to_fixedpoint_8( int a)



/****************************************************************************
 * to_fixedpoint_8(...) [ body ]
 ****************************************************************************/
inline
fixedpoint_8[+] to_fixedpoint_8( int[+] a)



/****************************************************************************
 * to_fixedpoint_16(...) [ body ]
 ****************************************************************************/
inline
fixedpoint_16 to_fixedpoint_16( int a)



/****************************************************************************
 * to_fixedpoint_16(...) [ body ]
 ****************************************************************************/
inline
fixedpoint_16[+] to_fixedpoint_16( int[+] a)



/****************************************************************************
```

```
 * to_fixedpoint_24(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_24 to_fixedpoint_24( int a)



/******************************************************************************
 * to_fixedpoint_24(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_24[+] to_fixedpoint_24( int[+] a)



/******************************************************************************
 * to_fixedpoint_32(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32 to_fixedpoint_32( int a)



/******************************************************************************
 * to_fixedpoint_32(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_32[+] to_fixedpoint_32( int[+] a)



/******************************************************************************
 * to_fixedpoint_8(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8 to_fixedpoint_8( fixedpoint_8 a)



/******************************************************************************
 * to_fixedpoint_8(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_8[+] to_fixedpoint_8( fixedpoint_8[+] a)



/******************************************************************************
 * to_fixedpoint_16(...) [ body ]
 ******************************************************************************/
inline
fixedpoint_16 to_fixedpoint_16( fixedpoint_8 a)
```

```
/*****************************************************************************
 * to_fixedpoint_16(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] to_fixedpoint_16( fixedpoint_8[+] a)



/*****************************************************************************
 * to_fixedpoint_24(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24 to_fixedpoint_24( fixedpoint_8 a)



/*****************************************************************************
 * to_fixedpoint_24(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] to_fixedpoint_24( fixedpoint_8[+] a)



/*****************************************************************************
 * to_fixedpoint_32(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32 to_fixedpoint_32( fixedpoint_8 a)



/*****************************************************************************
 * to_fixedpoint_32(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32[+] to_fixedpoint_32( fixedpoint_8[+] a)



/*****************************************************************************
 * to_fixedpoint_8(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8 to_fixedpoint_8( fixedpoint_16 a)



/*****************************************************************************
 * to_fixedpoint_8(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8[+] to_fixedpoint_8( fixedpoint_16[+] a)
```

```
/*****************************************************************************
 * to_fixedpoint_16(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16 to_fixedpoint_16( fixedpoint_16 a)



/*****************************************************************************
 * to_fixedpoint_16(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] to_fixedpoint_16( fixedpoint_16[+] a)



/*****************************************************************************
 * to_fixedpoint_24(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24 to_fixedpoint_24( fixedpoint_16 a)



/*****************************************************************************
 * to_fixedpoint_24(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] to_fixedpoint_24( fixedpoint_16[+] a)



/*****************************************************************************
 * to_fixedpoint_32(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32 to_fixedpoint_32( fixedpoint_16 a)



/*****************************************************************************
 * to_fixedpoint_32(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32[+] to_fixedpoint_32( fixedpoint_16[+] a)



/*****************************************************************************
 * to_fixedpoint_8(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8 to_fixedpoint_8( fixedpoint_24 a)
```

```
/*****************************************************************************
 * to_fixedpoint_8(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8[+] to_fixedpoint_8( fixedpoint_24[+] a)



/*****************************************************************************
 * to_fixedpoint_16(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16 to_fixedpoint_16( fixedpoint_24 a)



/*****************************************************************************
 * to_fixedpoint_16(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] to_fixedpoint_16( fixedpoint_24[+] a)



/*****************************************************************************
 * to_fixedpoint_24(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24 to_fixedpoint_24( fixedpoint_24 a)



/*****************************************************************************
 * to_fixedpoint_24(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] to_fixedpoint_24( fixedpoint_24[+] a)



/*****************************************************************************
 * to_fixedpoint_32(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32 to_fixedpoint_32( fixedpoint_24 a)



/*****************************************************************************
 * to_fixedpoint_32(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32[+] to_fixedpoint_32( fixedpoint_24[+] a)
```

```
/*****************************************************************************
 * to_fixedpoint_8(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8 to_fixedpoint_8( fixedpoint_32 a)



/*****************************************************************************
 * to_fixedpoint_8(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_8[+] to_fixedpoint_8( fixedpoint_32[+] a)



/*****************************************************************************
 * to_fixedpoint_16(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16 to_fixedpoint_16( fixedpoint_32 a)



/*****************************************************************************
 * to_fixedpoint_16(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_16[+] to_fixedpoint_16( fixedpoint_32[+] a)



/*****************************************************************************
 * to_fixedpoint_24(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24 to_fixedpoint_24( fixedpoint_32 a)



/*****************************************************************************
 * to_fixedpoint_24(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_24[+] to_fixedpoint_24( fixedpoint_32[+] a)



/*****************************************************************************
 * to_fixedpoint_32(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32 to_fixedpoint_32( fixedpoint_32 a)
```

```
/*****************************************************************************
 * to_fixedpoint_32(...) [ body ]
 *****************************************************************************/
inline
fixedpoint_32[+] to_fixedpoint_32( fixedpoint_32[+] a)



/*****************************************************************************
 * to_int2(...) [ body ]
 *****************************************************************************/
inline
int[2] to_int2( fixedpoint_8 a)



/*****************************************************************************
 * to_int2(...) [ body ]
 *****************************************************************************/
inline
int[+] to_int2( fixedpoint_8[+] a)



/*****************************************************************************
 * to_int2(...) [ body ]
 *****************************************************************************/
inline
int[2] to_int2( fixedpoint_16 a)



/*****************************************************************************
 * to_int2(...) [ body ]
 *****************************************************************************/
inline
int[+] to_int2( fixedpoint_16[+] a)



/*****************************************************************************
 * to_int2(...) [ body ]
 *****************************************************************************/
inline
int[2] to_int2( fixedpoint_24 a)



/*****************************************************************************
 * to_int2(...) [ body ]
 *****************************************************************************/
inline
```

```
int[+] to_int2( fixedpoint_24[+] a)



/*****************************************************************************
 * to_int2(...) [ body ]
 *****************************************************************************/
inline
int[2] to_int2( fixedpoint_32 a)



/*****************************************************************************
 * to_int2(...) [ body ]
 *****************************************************************************/
inline
int[+] to_int2( fixedpoint_32[+] a)



/*****************************************************************************
 * fraction(...) [ body ]
 *****************************************************************************/
inline
int fraction( fixedpoint_8 a)



/*****************************************************************************
 * fraction(...) [ body ]
 *****************************************************************************/
inline
int fraction( fixedpoint_16 a)



/*****************************************************************************
 * fraction(...) [ body ]
 *****************************************************************************/
inline
int fraction( fixedpoint_24 a)



/*****************************************************************************
 * fraction(...) [ body ]
 *****************************************************************************/
inline
int fraction( fixedpoint_32 a)



/*****************************************************************************
 * to_int(...) [ body ]
 *****************************************************************************/
```

```
inline
int to_int( int a)


/*-----------------------------------------------*/
```

3.3 Math.sac

```
/*-----------------------------------------------*/

module Math;

use ScalarArith : all;
export all;


/*
 *  prototypes for externals (FUNDECS)
 */

external double cos( double X);
#pragma linksign []

external double sin( double X);
#pragma linksign []

external double tan( double X);
#pragma linksign []

external double acos( double X);
#pragma linksign []

external double asin( double X);
#pragma linksign []

external double atan( double X);
#pragma linksign []

external double atan2( double X, double Y);
#pragma linksign []

external double cosh( double X);
#pragma linksign []

external double sinh( double X);
#pragma linksign []

external double tanh( double X);
#pragma linksign []

external double acosh( double X);
#pragma linksign []

external double asinh( double X);
#pragma linksign []

external double atanh( double X);
#pragma linksign []

external double exp( double X);
#pragma linksign []
```

```
external float expf( float X);
#pragma linksign []

external double, int frexp( double X);
#pragma linksign []

external double ldexp( double X, int EXP);
#pragma linksign []

external double log( double X);
#pragma linksign []

external double log10( double X);
#pragma linksign []

external double log2( double X);
#pragma linksign []

external double expm1( double X);
#pragma linksign []

external double log1p( double X);
#pragma linksign []

external double, double modf( double X);
#pragma linksign []

external double pow( double X, double Y);
#pragma linksign []

external float powf( float X, float Y);
#pragma linksign []

external double sqrt( double X);
#pragma linksign []

external float sqrtf( float X);
#pragma linksign []

external double cbrt( double X);
#pragma linksign []

external double ceil( double X);
#pragma linksign []

external double fabs( double X);
#pragma linksign []

external double floor( double X);
#pragma linksign []

external double fmod( double X, double Y);
#pragma linksign []

external bool isinf( double X);
#pragma linkname "SAC_MATH_isinf"
#pragma linkobj "src/Math/isinf.o"
#pragma linksign []

external bool isnan( double X);
#pragma linkname "SAC_MATH_isnan"
```

```
#pragma linkobj "src/Math/isnan.o"
#pragma linksign []

external bool finite( double X);
#pragma linkname "SAC_MATH_isfinite"
#pragma linkobj "src/Math/isfinite.o"
#pragma linksign []

external double copysign( double X, double Y);
#pragma linksign []

external double rint( double X);
#pragma linksign []

external double hypot( double X, double Y);
#pragma linksign []



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * cos(...) [ body ]
 *****************************************************************************/
inline
float cos( float X)




/******************************************************************************
 * sin(...) [ body ]
 *****************************************************************************/
inline
float sin( float X)




/******************************************************************************
 * sincos(...) [ body ]
 *****************************************************************************/
inline
float, float sincos( float x)




/******************************************************************************
 * sincos(...) [ body ]
 *****************************************************************************/
inline
double, double sincos( double x)



/******************************************************************************
```

```
 * tan(...) [ body ]
 ******************************************************************************/
inline
float tan( float X)



/******************************************************************************
 * acos(...) [ body ]
 ******************************************************************************/
inline
float acos( float X)



/******************************************************************************
 * asin(...) [ body ]
 ******************************************************************************/
inline
float asin( float X)



/******************************************************************************
 * atan(...) [ body ]
 ******************************************************************************/
inline
float atan( float X)



/******************************************************************************
 * atan2(...) [ body ]
 ******************************************************************************/
inline
float atan2( float X, float Y)



/******************************************************************************
 * cosh(...) [ body ]
 ******************************************************************************/
inline
float cosh( float X)



/******************************************************************************
 * sinh(...) [ body ]
 ******************************************************************************/
inline
float sinh( float X)
```

```
/*****************************************************************************
 * tanh(...) [ body ]
 *****************************************************************************/
inline
float tanh( float X)



/*****************************************************************************
 * acosh(...) [ body ]
 *****************************************************************************/
inline
float acosh( float X)



/*****************************************************************************
 * asinh(...) [ body ]
 *****************************************************************************/
inline
float asinh( float X)



/*****************************************************************************
 * atanh(...) [ body ]
 *****************************************************************************/
inline
float atanh( float X)



/*****************************************************************************
 * exp(...) [ body ]
 *****************************************************************************/
inline
float exp( float X)



/*****************************************************************************
 * frexp(...) [ body ]
 *****************************************************************************/
inline
float, int frexp( float X)



/*****************************************************************************
 * ldexp(...) [ body ]
 *****************************************************************************/
inline
float ldexp( float X, int EXP)
```

```
/*****************************************************************************
 * log(...) [ body ]
 *****************************************************************************/
inline
float log( float X)



/*****************************************************************************
 * log10(...) [ body ]
 *****************************************************************************/
inline
float log10( float X)



/*****************************************************************************
 * log2(...) [ body ]
 *****************************************************************************/
inline
float log2( float X)



/*****************************************************************************
 * expm1(...) [ body ]
 *****************************************************************************/
inline
float expm1( float X)



/*****************************************************************************
 * log1p(...) [ body ]
 *****************************************************************************/
inline
float log1p( float X)



/*****************************************************************************
 * modf(...) [ body ]
 *****************************************************************************/
inline
float, float modf( float X)



/*****************************************************************************
 * pow(...) [ body ]
 *****************************************************************************/
inline
float pow( float X, float Y)
```

```
/*****************************************************************************
 * sqrt(...) [ body ]
 *****************************************************************************/
inline
float sqrt( float X)



/*****************************************************************************
 * cbrt(...) [ body ]
 *****************************************************************************/
inline
float cbrt( float X)



/*****************************************************************************
 * ceil(...) [ body ]
 *****************************************************************************/
inline
float ceil( float X)



/*****************************************************************************
 * fabs(...) [ body ]
 *****************************************************************************/
inline
float fabs( float X)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
float floor( float X)



/*****************************************************************************
 * fmod(...) [ body ]
 *****************************************************************************/
inline
float fmod( float X, float Y)



/*****************************************************************************
 * copysign(...) [ body ]
 *****************************************************************************/
inline
float copysign( float X, float Y)
```

```
/*****************************************************************************
 * rint(...) [ body ]
 *****************************************************************************/
inline
float rint( float X)



/*****************************************************************************
 * hypot(...) [ body ]
 *****************************************************************************/
inline
float hypot( float X, float Y)



/*****************************************************************************
 * sign(...) [ body ]
 *****************************************************************************/
inline
int sign( float x)



/*****************************************************************************
 * sign(...) [ body ]
 *****************************************************************************/
inline
int sign( double x)



/*****************************************************************************
 * fl_e(...) [ body ]
 *****************************************************************************/
inline
float fl_e()



/*****************************************************************************
 * fl_log2e(...) [ body ]
 *****************************************************************************/
inline
float fl_log2e()



/*****************************************************************************
 * fl_log10e(...) [ body ]
 *****************************************************************************/
inline
float fl_log10e()
```

```
/*****************************************************************************
 * fl_ln2(...) [ body ]
 *****************************************************************************/
inline
float fl_ln2()



/*****************************************************************************
 * fl_ln10(...) [ body ]
 *****************************************************************************/
inline
float fl_ln10()



/*****************************************************************************
 * fl_pi(...) [ body ]
 *****************************************************************************/
inline
float fl_pi()



/*****************************************************************************
 * fl_pi_2(...) [ body ]
 *****************************************************************************/
inline
float fl_pi_2()



/*****************************************************************************
 * fl_pi_rec(...) [ body ]
 *****************************************************************************/
inline
float fl_pi_rec()



/*****************************************************************************
 * fl_pi_4(...) [ body ]
 *****************************************************************************/
inline
float fl_pi_4()



/*****************************************************************************
 * fl_pi_rec_2(...) [ body ]
 *****************************************************************************/
inline
```

```
float fl_pi_rec_2()



/*****************************************************************************
 * fl_sqrtpi_rec_2(...) [ body ]
 *****************************************************************************/
inline
float fl_sqrtpi_rec_2()



/*****************************************************************************
 * fl_sqrt2(...) [ body ]
 *****************************************************************************/
inline
float fl_sqrt2()



/*****************************************************************************
 * fl_sqrt2_rec(...) [ body ]
 *****************************************************************************/
inline
float fl_sqrt2_rec()



/*****************************************************************************
 * e(...) [ body ]
 *****************************************************************************/
inline
double e()



/*****************************************************************************
 * log2e(...) [ body ]
 *****************************************************************************/
inline
double log2e()



/*****************************************************************************
 * log10e(...) [ body ]
 *****************************************************************************/
inline
double log10e()



/*****************************************************************************
 * ln2(...) [ body ]
 *****************************************************************************/
```

```
inline
double ln2()



/*****************************************************************************
 * ln10(...) [ body ]
 *****************************************************************************/
inline
double ln10()



/*****************************************************************************
 * pi(...) [ body ]
 *****************************************************************************/
inline
double pi()



/*****************************************************************************
 * pi_2(...) [ body ]
 *****************************************************************************/
inline
double pi_2()



/*****************************************************************************
 * pi_rec(...) [ body ]
 *****************************************************************************/
inline
double pi_rec()



/*****************************************************************************
 * pi_4(...) [ body ]
 *****************************************************************************/
inline
double pi_4()



/*****************************************************************************
 * pi_rec_2(...) [ body ]
 *****************************************************************************/
inline
double pi_rec_2()



/*****************************************************************************
 * sqrtpi_rec_2(...) [ body ]
```

```
 ******************************************************************************/
inline
double sqrtpi_rec_2()



/******************************************************************************
 * sqrt2(...) [ body ]
 ******************************************************************************/
inline
double sqrt2()



/******************************************************************************
 * sqrt2_rec(...) [ body ]
 ******************************************************************************/
inline
double sqrt2_rec()



/*--------------------------------------------*/
```

3.4 MathArray.sac

```
/*--------------------------------------------*/

module MathArray;

use ScalarArith : { zero};
use ArrayBasics : all;
import Math : all;
export all;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * log(...) [ body ]
 ******************************************************************************/
inline
float[+] log( float[+] A)



/******************************************************************************
 * log(...) [ body ]
 ******************************************************************************/
inline
double[+] log( double[+] A)
```

```
/*****************************************************************************
 * log2(...) [ body ]
 *****************************************************************************/
inline
float[+] log2( float[+] A)



/*****************************************************************************
 * log2(...) [ body ]
 *****************************************************************************/
inline
double[+] log2( double[+] A)



/*****************************************************************************
 * log10(...) [ body ]
 *****************************************************************************/
inline
float[+] log10( float[+] A)



/*****************************************************************************
 * log10(...) [ body ]
 *****************************************************************************/
inline
double[+] log10( double[+] A)



/*****************************************************************************
 * exp(...) [ body ]
 *****************************************************************************/
inline
float[+] exp( float[+] A)



/*****************************************************************************
 * exp(...) [ body ]
 *****************************************************************************/
inline
double[+] exp( double[+] A)



/*****************************************************************************
 * fabs(...) [ body ]
 *****************************************************************************/
inline
float[+] fabs( float[+] A)
```

```
/*****************************************************************************
 * fabs(...) [ body ]
 *****************************************************************************/
inline
double[+] fabs( double[+] A)



/*****************************************************************************
 * sqrt(...) [ body ]
 *****************************************************************************/
inline
float[+] sqrt( float[+] A)



/*****************************************************************************
 * sqrt(...) [ body ]
 *****************************************************************************/
inline
double[+] sqrt( double[+] A)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
float[+] floor( float[+] A)



/*****************************************************************************
 * floor(...) [ body ]
 *****************************************************************************/
inline
double[+] floor( double[+] A)



/*****************************************************************************
 * pow(...) [ body ]
 *****************************************************************************/
inline
float[+] pow( float[+] A, float S)



/*****************************************************************************
 * pow(...) [ body ]
 *****************************************************************************/
inline
double[+] pow( double[+] A, double S)
```

```
/*-----------------------------------------------*/
```

3.5 Numerical.sac

```
/*-----------------------------------------------*/

module Numerical;

import Math : all;
import MathArray : all;
export all;

/*-----------------------------------------------*/
```

3.6 SaCMath.sac

```
/*-----------------------------------------------*/

module SaCMath;

export all;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * pow2(...) [ body ]
 *****************************************************************************/
inline
int pow2( int a)




/******************************************************************************
 * log2(...) [ body ]
 *****************************************************************************/
inline
int log2( int a)



/*-----------------------------------------------*/
```

4 stdio

4.1 BinFile.sac

```
/*-----------------------------------------------*/

class BinFile;

use World : { TheWorld};
use FileSystem : { TheFileSystem};
use String : { string};
```

```
use SysErr : { syserr};
export all;


/*
 *  type definitions
 */

external classtype BinFile::BinFile;


/*
 *  prototypes for externals (FUNDECS)
 */

external syserr, BinFile binfopen( string NAME, int FLAGS);
#pragma linkname "SACbinfopen"
#pragma linkobj "src/BinFile/binfopen.o"
#pragma linksign []
#pragma effect TheFileSystem

external syserr binfclose( BinFile BINSTREAM);
#pragma linkname "SACbinfclose"
#pragma linkobj "src/BinFile/binfclose.o"
#pragma linksign []
#pragma effect TheFileSystem

external int O_RDONLY();
#pragma linkname "SACbinf_O_RDONLY"
#pragma linkobj "src/BinFile/binfflags.o"
#pragma linksign []

external int O_WRONLY();
#pragma linkname "SACbinf_O_WRONLY"
#pragma linkobj "src/BinFile/binfflags.o"
#pragma linksign []

external int O_RDWR();
#pragma linkname "SACbinf_O_RDWR"
#pragma linkobj "src/BinFile/binfflags.o"
#pragma linksign []

external int O_CREAT();
#pragma linkname "SACbinf_O_CREAT"
#pragma linkobj "src/BinFile/binfflags.o"
#pragma linksign []

external int O_TRUNC();
#pragma linkname "SACbinf_O_TRUNC"
#pragma linkobj "src/BinFile/binfflags.o"
#pragma linksign []

external double[+] binfReadDoubleArray( BinFile &binstream, int DIM, int[.]
   ↪ SHAPE);
#pragma linkname "SACbinfReadDoubleArray"
#pragma linkobj "src/BinFile/binfReadDblArr.o"
#pragma linksign []

external void  binfWriteDoubleArray( BinFile &binstream, int DIM, int[+] SHAPE,
   ↪ double[+] ARRAY);
#pragma linkname "SACbinfWriteDoubleArray"
#pragma linkobj "src/BinFile/binfWriteDblArr.o"
```

```
/*-----------------------------------------------*/
```

4.2 Color8IO.sac

```
/*-----------------------------------------------*/

module Color8IO;

use IOresources : all;
use Color8 : { color};
import ArrayIO : { print};
export all;


/*
 *  function definitions (FUNDEFS)
 */



/*****************************************************************************
 * print(...) [ body ]
 *****************************************************************************/
inline
void  print( color[*] c)



/*-----------------------------------------------*/
```

4.3 ComplexIO.sac

```
/*-----------------------------------------------*/

module ComplexIO;

use IOresources : all;
use String : { string, to_string, sprintf};
use Complex : { complex, imag, real, toc, shape, dim};
use ScalarArith : { ==};
use TermFile : { TermFile};
export { print, fprint};


/*
 *  prototypes for externals (FUNDECS)
 */

external void  printarray( File &stream, int d, int[.] s, complex[*] a);
#pragma linkname "COMPLEXIO__PrintComplexArray"
#pragma linkobj "src/ComplexIO/PrintComplexArray.o"

external void  printarray( TermFile &stream, int d, int[.] s, complex[*] a);
#pragma linkname "COMPLEXIO__PrintComplexArray"
#pragma linkobj "src/ComplexIO/PrintComplexArray.o"

external void  printarray( File &stream, string format, int d, int[.] s, complex
   ↪ [*] a);
#pragma linkname "COMPLEXIO__PrintComplexArrayFormat"
```

```
#pragma linkobj "src/ComplexIO/PrintComplexArray.o"

external void  printarray( TermFile &stream, string format, int d, int[.] s,
   ↪ complex[*] a);
#pragma linkname "COMPLEXIO__PrintComplexArrayFormat"
#pragma linkobj "src/ComplexIO/PrintComplexArray.o"



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
inline
void  fprint( File &stream, complex c)



/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
inline
void  fprint( File &stream, complex c, int mode)



/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
inline
void  fprint( File &stream, complex c, int mode, int prec)



/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
inline
void  fprint( TermFile &stream, complex c)



/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
inline
void  fprint( TermFile &stream, complex c, int mode)



/******************************************************************************
 * fprint(...) [ body ]
```

```
 ******************************************************************************/
inline
void  fprint( TermFile &stream, complex c, int mode, int prec)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
inline
void  print( complex c)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
inline
void  print( complex c, int mode)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
inline
void  print( complex c, int mode, int prec)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
inline
void  print( complex[+] arr)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
inline
void  print( complex[+] arr, int mode)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
inline
void  print( complex[+] arr, int mode, int prec)



/******************************************************************************
```

```
 * fscancomplex(...) [ body ]
 ******************************************************************************/
inline
bool, complex fscancomplex( File &stream)



/******************************************************************************
 * fscancomplex(...) [ body ]
 ******************************************************************************/
inline
bool, complex fscancomplex( File &stream, int mode)



/******************************************************************************
 * fscancomplex(...) [ body ]
 ******************************************************************************/
inline
bool, complex fscancomplex( TermFile &stream)



/******************************************************************************
 * fscancomplex(...) [ body ]
 ******************************************************************************/
inline
bool, complex fscancomplex( TermFile &stream, int mode)



/******************************************************************************
 * scancomplex(...) [ body ]
 ******************************************************************************/
inline
bool, complex scancomplex()



/******************************************************************************
 * scancomplex(...) [ body ]
 ******************************************************************************/
inline
bool, complex scancomplex( int mode)


/*-----------------------------------------------*/
```

4.4 FibreIO.sac

```
/*-----------------------------------------------*/

module FibreIO;

use IOresources : all;
use Array : { shape, sel, dim};
```

```
use String : { string};
use StringArray : { stringArray};
use TermFile : { TermFile};
export all;


/*
 *  prototypes for externals (FUNDECS)
 */

external int FibreScanInt( File &stream);
#pragma linkobj "src/FibreIO/FibreScan.tab.o" "src/FibreIO/lex.FibreScan.o" "src
   ↪ /FibreIO/ScanInt.o"
#pragma linksign []

external float FibreScanFloat( File &stream);
#pragma linkobj "src/FibreIO/ScanFlt.o"
#pragma linksign []

external double FibreScanDouble( File &stream);
#pragma linkobj "src/FibreIO/ScanDbl.o"
#pragma linksign []

external int[*] FibreScanIntArray( File &stream);
#pragma linkobj "src/FibreIO/ScanIntArr.o"
#pragma linksign []

external float[*] FibreScanFloatArray( File &stream);
#pragma linkobj "src/FibreIO/ScanFltArr.o"
#pragma linksign []

external double[*] FibreScanDoubleArray( File &stream);
#pragma linkobj "src/FibreIO/ScanDblArr.o"
#pragma linksign []

external stringArray FibreScanStringArray( File &stream);
#pragma linkobj "src/FibreIO/ScanStringArr.o"
#pragma linksign []

external int FibreScanInt( TermFile &stream);
#pragma linkobj "src/FibreIO/FibreScan.tab.o" "src/FibreIO/lex.FibreScan.o" "src
   ↪ /FibreIO/ScanInt.o"
#pragma linksign []

external float FibreScanFloat( TermFile &stream);
#pragma linkobj "src/FibreIO/ScanFlt.o"
#pragma linksign []

external double FibreScanDouble( TermFile &stream);
#pragma linkobj "src/FibreIO/ScanDbl.o"
#pragma linksign []

external int[*] FibreScanIntArray( TermFile &stream);
#pragma linkobj "src/FibreIO/ScanIntArr.o"
#pragma linksign []

external float[*] FibreScanFloatArray( TermFile &stream);
#pragma linkobj "src/FibreIO/ScanFltArr.o"
#pragma linksign []

external double[*] FibreScanDoubleArray( TermFile &stream);
#pragma linkobj "src/FibreIO/ScanDblArr.o"
```

```
#pragma linksign []

external stringArray FibreScanStringArray( TermFile &stream);
#pragma linkobj "src/FibreIO/ScanStringArr.o"
#pragma linksign []

external int[*] FibreScanIntArrayStr( string stream);
#pragma linkobj "src/FibreIO/ScanIntArr.o"
#pragma linksign []

external float[*] FibreScanFloatArrayStr( string stream);
#pragma linkobj "src/FibreIO/ScanFltArr.o"
#pragma linksign []

external double[*] FibreScanDoubleArrayStr( string stream);
#pragma linkobj "src/FibreIO/ScanDblArr.o"
#pragma linksign []

external stringArray FibreScanStringArrayStr( string stream);
#pragma linkobj "src/FibreIO/ScanStringArr.o"
#pragma linksign []

external void  FibrePrint( File &stream, int DIM, int[+] SHAPE, int[+] ARRAY);
#pragma linkname "FibrePrintIntArray"
#pragma linkobj "src/FibreIO/FibrePrint.o"

external void  FibrePrint( File &stream, int DIM, int[+] SHAPE, float[+] ARRAY);
#pragma linkname "FibrePrintFloatArray"
#pragma linkobj "src/FibreIO/FibrePrint.o"

external void  FibrePrint( File &stream, int DIM, int[+] SHAPE, double[+] ARRAY)
   ↪ ;
#pragma linkname "FibrePrintDoubleArray"
#pragma linkobj "src/FibreIO/FibrePrint.o"

external void  FibrePrint( File &stream, int DIM, int[+] SHAPE, stringArray
   ↪ ARRAY);
#pragma linkname "FibrePrintStringArray"
#pragma linkobj "src/FibreIO/FibrePrint.o"

external void  FibrePrint( TermFile &stream, int DIM, int[+] SHAPE, int[+] ARRAY
   ↪ );
#pragma linkname "FibrePrintIntArray"
#pragma linkobj "src/FibreIO/FibrePrint.o"

external void  FibrePrint( TermFile &stream, int DIM, int[+] SHAPE, float[+]
   ↪ ARRAY);
#pragma linkname "FibrePrintFloatArray"
#pragma linkobj "src/FibreIO/FibrePrint.o"

external void  FibrePrint( TermFile &stream, int DIM, int[+] SHAPE, double[+]
   ↪ ARRAY);
#pragma linkname "FibrePrintDoubleArray"
#pragma linkobj "src/FibreIO/FibrePrint.o"

external void  FibrePrint( TermFile &stream, int DIM, int[+] SHAPE, stringArray
   ↪ ARRAY);
#pragma linkname "FibrePrintStringArray"
#pragma linkobj "src/FibreIO/FibrePrint.o"
```

```
/*
 *  function definitions (FUNDEFS)
 */


/******************************************************************************
 * FibreScanIntArray(...) [ body ]
 ******************************************************************************/
int[*] FibreScanIntArray()



/******************************************************************************
 * FibreScanFloatArray(...) [ body ]
 ******************************************************************************/
float[*] FibreScanFloatArray()



/******************************************************************************
 * FibreScanDoubleArray(...) [ body ]
 ******************************************************************************/
double[*] FibreScanDoubleArray()



/******************************************************************************
 * FibreScanStringArray(...) [ body ]
 ******************************************************************************/
stringArray FibreScanStringArray()



/******************************************************************************
 * FibrePrint(...) [ body ]
 ******************************************************************************/
inline
void  FibrePrint( File &stream, int[+] arr)



/******************************************************************************
 * FibrePrint(...) [ body ]
 ******************************************************************************/
inline
void  FibrePrint( File &stream, int arr)



/******************************************************************************
 * FibrePrint(...) [ body ]
 ******************************************************************************/
inline
void  FibrePrint( File &stream, float[+] arr)
```

```
/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( File &stream, float arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( File &stream, double[+] arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( File &stream, double arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( File &stream, stringArray arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( TermFile &stream, int[+] arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( TermFile &stream, int arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( TermFile &stream, float[+] arr)
```

```
/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( TermFile &stream, float arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( TermFile &stream, double[+] arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( TermFile &stream, double arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( TermFile &stream, stringArray arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( int[+] arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( int arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
```

```
void  FibrePrint( float[+] arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( float arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( double[+] arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( double arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( StringArray::stringArray arr)



/*****************************************************************************
 * FibrePrint(...) [ body ]
 *****************************************************************************/
inline
void  FibrePrint( string scaler)



/*-----------------------------------------------*/
```

4.5 File.sac

```
/*-----------------------------------------------*/

class File;

use FileSystem : { TheFileSystem};
use String : { string};
use SysErr : { syserr};
export all;
```

```
/*
 *  type definitions
 */

external classtype File::File;


/*
 *  prototypes for externals (FUNDECS)
 */

external syserr, File fopen( string NAME, string MODE);
#pragma linkname "SACfopen"
#pragma linkobj "src/File/fopen.o"
#pragma linksign []
#pragma effect TheFileSystem

external syserr, File mkstemp( string template);
#pragma linkname "SACmkstemp"
#pragma linkobj "src/File/mkstemp.o"
#pragma linksign []
#pragma effect TheFileSystem

external void  fclose( File STREAM);
#pragma linkname "SACfclose"
#pragma linkobj "src/File/fclose.o"
#pragma effect TheFileSystem

external void  fremove( string fname);
#pragma linkobj "src/File/rm.o"
#pragma effect TheFileSystem

external void  fputc( char C, File &STREAM);
#pragma linkname "SACfputc"
#pragma linkobj "src/File/fputc.o"

external char fgetc( File &STREAM);
#pragma linkname "SACfgetc"
#pragma linkobj "src/File/fgetc.o"
#pragma linksign []

external void  ungetc( char C, File &STREAM);
#pragma linkname "SACungetc"
#pragma linkobj "src/File/ungetc.o"

external void  fputs( string S, File &STREAM);
#pragma linkname "SACfputs"
#pragma linkobj "src/File/fputs.o"

external void  fprintf( File &STREAM, string FORMAT, ...);
#pragma linkname "SACfprintf"
#pragma linkobj "src/File/fprintf.o"

external int, ... fscanf( File &STREAM, string FORMAT);
#pragma linkname "SACfscanf"
#pragma linkobj "src/File/fscanf.o"
#pragma linksign []

external string fscans( File &STREAM, int MAX);
#pragma linkobj "src/File/fscans.o"
#pragma linksign []
```

```
external string fscanl( File &STREAM, int MAX);
#pragma linkobj "src/File/fscanl.o"
#pragma linksign []

external bool feof( File &STREAM);
#pragma linkname "SACfeof"
#pragma linkobj "src/File/feof.o"
#pragma linksign []

external void  fflush( File &STREAM);
#pragma linkname "SACfflush"
#pragma linkobj "src/File/fflush.o"

external void  fseek( File &STREAM, int OFFSET, int BASE);
#pragma linkname "SACfseek"
#pragma linkobj "src/File/fseek.o"

external int ftell( File &STREAM);
#pragma linkname "SACftell"
#pragma linkobj "src/File/ftell.o"
#pragma linksign []

external void  rewind( File &STREAM);
#pragma linkname "SACrewind"
#pragma linkobj "src/File/rewind.o"



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * tmpfile(...) [ body ]
 *****************************************************************************/
syserr, File tmpfile()



/*-----------------------------------------------*/
```

4.6 GreyIO.sac

```
/*-----------------------------------------------*/

module GreyIO;

use IOresources : all;
use Grey : { grey};
import ArrayIO : { print};
import ScalarIO : { print};
export all;


/*
 *  function definitions (FUNDEFS)
 */
```

```
/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
inline
void  print( grey[*] c)



/*-----------------------------------------------*/
```

4.7 IOresources.sac

```
/*-----------------------------------------------*/

module IOresources;

import File : { File};
import TermFile : { TermFile, stdin, stdout, stderr};
import Terminal : { TheTerminal};
import FileSystem : { TheFileSystem};
import World : { TheWorld};
export all;

/*-----------------------------------------------*/
```

4.8 ListIO.sac

```
/*-----------------------------------------------*/

module ListIO;

use IOresources : all;
use List : all;
use String : { string, to_string};
use ScalarIO : { scanint};
use ScalarArith : all;
use TermFile : { TermFile};
export all;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
void  fprint( File &stream, list L)




/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
void  fprint( File &stream, list L, int ElemsPerLine)
```

```
/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
void  fprint( File &stream, list L, int ElemsPerLine, int ColWidth)



/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
void  fprint( TermFile &stream, list L)



/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
void  fprint( TermFile &stream, list L, int ElemsPerLine)



/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
void  fprint( TermFile &stream, list L, int ElemsPerLine, int ColWidth)



/*****************************************************************************
 * print(...) [ body ]
 *****************************************************************************/
void  print( list L)



/*****************************************************************************
 * print(...) [ body ]
 *****************************************************************************/
void  print( list L, int ElemsPerLine)



/*****************************************************************************
 * print(...) [ body ]
 *****************************************************************************/
void  print( list L, int ElemsPerLine, int ColWidth)



/*-----------------------------------------------*/
```

4.9 PGM.sac

```
/*-----------------------------------------------*/
```

```
class PGM;

use String : { string};
use ArrayBasics : { shape};
use File : { File, fopen, fclose};
use SysErr : { fail};
use RuntimeError : { error};
use ScalarArith : { ==, >};
use ArrayTransform : { maxval};
export all;


/*
 *  type definitions
 */

external classtype PGM::PGM;


/*
 *  prototypes for externals (FUNDECS)
 */

external PGM parsePGM( File &fp);
#pragma linkname "SAC_PGM_parse"
#pragma linkobj "src/PGM/pgm2array.o"
#pragma linksign []
#pragma effect FileSystem::TheFileSystem

external void  freePGM( PGM &pgm);
#pragma linkname "SAC_PGM_free"
#pragma linkobj "src/PGM/pgm2array.o"
#pragma effect FileSystem::TheFileSystem

external int PGMwidth( PGM &pgm);
#pragma linkname "SAC_PGM_width"
#pragma linkobj "src/PGM/pgm2array.o"
#pragma linksign []
#pragma effect FileSystem::TheFileSystem

external int PGMheight( PGM &pgm);
#pragma linkname "SAC_PGM_height"
#pragma linkobj "src/PGM/pgm2array.o"
#pragma linksign []
#pragma effect FileSystem::TheFileSystem

external int PGMmaxval( PGM &pgm);
#pragma linkname "SAC_PGM_maxval"
#pragma linkobj "src/PGM/pgm2array.o"
#pragma linksign []
#pragma effect FileSystem::TheFileSystem

external int[.,.] readPGMdata( PGM &pgm);
#pragma linkname "SAC_PGM_read_data"
#pragma linkobj "src/PGM/pgm2array.o"
#pragma effect FileSystem::TheFileSystem

external File PGMstream( PGM &pgm);
#pragma linkname "SAC_PGM_stream"
#pragma linkobj "src/PGM/pgm2array.o"
#pragma linksign []
```

```
#pragma effect FileSystem::TheFileSystem

external PGM newPGM( int[2] shp, int mval, bool binary, File &fp);
#pragma linkname "SAC_PGM_new"
#pragma linkobj "src/PGM/array2pgm.o"
#pragma linksign []
#pragma effect FileSystem::TheFileSystem

external void  writePGMheader( PGM &pgm);
#pragma linkname "SAC_PGM_write_header"
#pragma linkobj "src/PGM/array2pgm.o"
#pragma effect FileSystem::TheFileSystem

external void  writePGMdata( int[.,.] data, PGM &pgm);
#pragma linkname "SAC_PGM_write_data"
#pragma linkobj "src/PGM/array2pgm.o"
#pragma linksign []
#pragma effect FileSystem::TheFileSystem



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * readPGM(...) [ body ]
 ******************************************************************************/
int[.,.], int readPGM( string filename)



/******************************************************************************
 * readPGM(...) [ body ]
 ******************************************************************************/
int[.,.], int readPGM( File stream)



/******************************************************************************
 * openPGM(...) [ body ]
 ******************************************************************************/
PGM openPGM( string name)



/******************************************************************************
 * writePGM(...) [ body ]
 ******************************************************************************/
void  writePGM( int[.,.] image, int[2] shp, int mval, bool binary, string name)



/******************************************************************************
 * writePGM(...) [ body ]
 ******************************************************************************/
```

```
void  writePGM( int[.,.] image, int[2] shp, int mval, bool binary, File stream)



/*****************************************************************************
 * writePGM(...) [ body ]
 *****************************************************************************/
inline
void  writePGM( int[.,.] img, int mval, bool binary, string filename)



/*****************************************************************************
 * writePGM(...) [ body ]
 *****************************************************************************/
inline
void  writePGM( int[.,.] img, int mval, string filename)



/*****************************************************************************
 * writePGM(...) [ body ]
 *****************************************************************************/
inline
void  writePGM( int[.,.] img, bool binary, string filename)



/*****************************************************************************
 * writePGM(...) [ body ]
 *****************************************************************************/
inline
void  writePGM( int[.,.] img, string filename)


/*-----------------------------------------------*/
```

4.10 PPM.sac

```
/*-----------------------------------------------*/

module PPM;

use Color8 : { color, shape};
use File : { File, fopen, fclose};
use RuntimeError : { error};
use ScalarArith : { ==};
use String : { string};
use TermFile : { TermFile, stdin, stdout};
export { readPPM, printPPM};


/*
 *  prototypes for externals (FUNDECS)
 */

external color[.,.] readStream( TermFile &stream);
```

```
#pragma linkname "SAC_PPM_ppm2array"
#pragma linkobj "src/PPM/ppm2array.o"
#pragma effect Terminal::TheTerminal

external color[.,.] readStream( File &stream);
#pragma linkname "SAC_PPM_ppm2array"
#pragma linkobj "src/PPM/ppm2array.o"
#pragma effect FileSystem::TheFileSystem

external void  writeStream( TermFile &stream, color[.,.] image, int[2] shp, bool
   ↪  binary);
#pragma linkname "SAC_PPM_array2ppm"
#pragma linkobj "src/PPM/array2ppm.o"
#pragma effect Terminal::TheTerminal

external void  writeStream( File &stream, color[.,.] image, int[2] shp, bool
   ↪ binary);
#pragma linkname "SAC_PPM_array2ppm"
#pragma linkobj "src/PPM/array2ppm.o"
#pragma effect FileSystem::TheFileSystem



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * readPPM(...) [ body ]
 ******************************************************************************/
inline
color[.,.] readPPM()



/******************************************************************************
 * readPPM(...) [ body ]
 ******************************************************************************/
inline
color[.,.] readPPM( string name)



/******************************************************************************
 * printPPM(...) [ body ]
 ******************************************************************************/
inline
void  printPPM( color[.,.] img)



/******************************************************************************
 * printPPM(...) [ body ]
 ******************************************************************************/
inline
void  printPPM( color[.,.] img, string name, bool binary)
```

```
/*--------------------------------------------*/
```

4.11 ScalarIO.sac

```
/*--------------------------------------------*/

module ScalarIO;

use IOresources : all;
use String : { string, to_string, strlen};
use ScalarArith : all;
use ArrayFormat : all;
use TermFile : { TermFile};
export all;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
inline
void  fprint( File &stream, int n)




/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
inline
void  fprint( File &stream, float n)




/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
inline
void  fprint( File &stream, float n, int prec)




/******************************************************************************
 * fprint(...) [ body ]
 ******************************************************************************/
inline
void  fprint( File &stream, double n)




/******************************************************************************
 * fprint(...) [ body ]
```

```
 ***************************************************************************/
inline
void  fprint( File &stream, double n, int prec)



/***************************************************************************
 * fprint(...) [ body ]
 ***************************************************************************/
void  fprint( File &stream, bool n)



/***************************************************************************
 * fprint(...) [ body ]
 ***************************************************************************/
void  fprint( File &stream, bool n, int mode)



/***************************************************************************
 * fprint(...) [ body ]
 ***************************************************************************/
inline
void  fprint( File &stream, char c)



/***************************************************************************
 * fprint(...) [ body ]
 ***************************************************************************/
inline
void  fprint( File &stream, string s)



/***************************************************************************
 * fprint(...) [ body ]
 ***************************************************************************/
inline
void  fprint( TermFile &stream, int n)



/***************************************************************************
 * fprint(...) [ body ]
 ***************************************************************************/
inline
void  fprint( TermFile &stream, float n)



/***************************************************************************
 * fprint(...) [ body ]
 ***************************************************************************/
```

```
inline
void  fprint( TermFile &stream, float n, int prec)



/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
inline
void  fprint( TermFile &stream, double n)



/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
inline
void  fprint( TermFile &stream, double n, int prec)



/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
void  fprint( TermFile &stream, bool n)



/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
void  fprint( TermFile &stream, bool n, int mode)



/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
inline
void  fprint( TermFile &stream, char c)



/*****************************************************************************
 * fprint(...) [ body ]
 *****************************************************************************/
inline
void  fprint( TermFile &stream, string s)



/*****************************************************************************
 * show(...) [ body ]
 *****************************************************************************/
void  show( bool c)
```

```
/*****************************************************************************
 * show(...) [ body ]
 *****************************************************************************/
void  show( int c)


/*****************************************************************************
 * show(...) [ body ]
 *****************************************************************************/
void  show( float c)


/*****************************************************************************
 * show(...) [ body ]
 *****************************************************************************/
void  show( double c)


/*****************************************************************************
 * show(...) [ body ]
 *****************************************************************************/
void  show( char c)


/*****************************************************************************
 * show(...) [ body ]
 *****************************************************************************/
void  show( ulonglong c)


/*****************************************************************************
 * print(...) [ body ]
 *****************************************************************************/
void  print( int n)


/*****************************************************************************
 * print(...) [ body ]
 *****************************************************************************/
void  print( float n)


/*****************************************************************************
 * print(...) [ body ]
```

```
 ******************************************************************************/
void  print( float n, int prec)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
void  print( double n)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
void  print( double n, int prec)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
void  print( bool n)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
void  print( bool n, int mode)



/******************************************************************************
 * print(...) [ body ]
 ******************************************************************************/
void  print( char c)



/******************************************************************************
 * fscanint(...) [ body ]
 ******************************************************************************/
inline
bool, int fscanint( File &stream)



/******************************************************************************
 * fscanfloat(...) [ body ]
 ******************************************************************************/
inline
bool, float fscanfloat( File &stream)
```

```
/*****************************************************************************
 * fscandouble(...) [ body ]
 *****************************************************************************/
inline
bool, double fscandouble( File &stream)



/*****************************************************************************
 * fscanbool(...) [ body ]
 *****************************************************************************/
bool, bool fscanbool( File &stream)



/*****************************************************************************
 * fscanchar(...) [ body ]
 *****************************************************************************/
inline
bool, char fscanchar( File &stream)



/*****************************************************************************
 * fscanstring(...) [ body ]
 *****************************************************************************/
inline
bool, string fscanstring( File &stream, int length)



/*****************************************************************************
 * fscanint(...) [ body ]
 *****************************************************************************/
inline
bool, int fscanint( TermFile &stream)



/*****************************************************************************
 * fscanfloat(...) [ body ]
 *****************************************************************************/
inline
bool, float fscanfloat( TermFile &stream)



/*****************************************************************************
 * fscandouble(...) [ body ]
 *****************************************************************************/
inline
bool, double fscandouble( TermFile &stream)
```

```
/*****************************************************************************
 * fscanbool(...) [ body ]
 *****************************************************************************/
bool, bool fscanbool( TermFile &stream)



/*****************************************************************************
 * fscanchar(...) [ body ]
 *****************************************************************************/
inline
bool, char fscanchar( TermFile &stream)



/*****************************************************************************
 * fscanstring(...) [ body ]
 *****************************************************************************/
inline
bool, string fscanstring( TermFile &stream, int length)



/*****************************************************************************
 * scanint(...) [ body ]
 *****************************************************************************/
inline
bool, int scanint()



/*****************************************************************************
 * scanfloat(...) [ body ]
 *****************************************************************************/
inline
bool, float scanfloat()



/*****************************************************************************
 * scandouble(...) [ body ]
 *****************************************************************************/
inline
bool, double scandouble()



/*****************************************************************************
 * scanbool(...) [ body ]
 *****************************************************************************/
bool, bool scanbool()
```

```
/*****************************************************************************
 * scanchar(...) [ body ]
 *****************************************************************************/
inline
bool, char scanchar()



/*****************************************************************************
 * scanstring(...) [ body ]
 *****************************************************************************/
inline
bool, string scanstring( int length)


/*-------------------------------------------------*/
```

4.12 StdIO.sac

```
/*-------------------------------------------------*/

module StdIO;

import File : all;
import TermFile : all;
import BinFile : all;
import ScalarIO : all;
import ArrayIO : all;
export all;

/*-------------------------------------------------*/
```

4.13 TermFile.sac

```
/*-------------------------------------------------*/

class TermFile;

use String : { string};
use Terminal : { TheTerminal};
export all except { createStdIn, createStdOut, createStdErr};


/*
 *  type definitions
 */

external classtype TermFile::TermFile;


/*
 *  prototypes for externals (FUNDECS)
 */

external TermFile createStdIn();
#pragma linkname "SAC_create_stdin"
#pragma linkobj "src/TermFile/stdstreams.o"
#pragma linksign []
```

```
#pragma effect TheTerminal

external TermFile createStdOut();
#pragma linkname "SAC_create_stdout"
#pragma linkobj "src/TermFile/stdstreams.o"
#pragma linksign []
#pragma effect TheTerminal

external TermFile createStdErr();
#pragma linkname "SAC_create_stderr"
#pragma linkobj "src/TermFile/stdstreams.o"
#pragma linksign []
#pragma effect TheTerminal

external void  fputc( char C, TermFile &STREAM);
#pragma linkname "SACfputc_TF"
#pragma linkobj "src/TermFile/fputc.o"
#pragma effect TheTerminal

external char fgetc( TermFile &STREAM);
#pragma linkname "SACfgetc_TF"
#pragma linkobj "src/TermFile/fgetc.o"
#pragma linksign []
#pragma effect TheTerminal

external void  fputs( string S, TermFile &STREAM);
#pragma linkname "SACfputs_TF"
#pragma linkobj "src/TermFile/fputs.o"
#pragma effect TheTerminal

external void  puts( string S);
#pragma linkname "SACputs_TF"
#pragma linkobj "src/TermFile/puts.o"
#pragma effect TheTerminal

external void  ungetc( char C, TermFile &STREAM);
#pragma linkname "SACungetc_TF"
#pragma linkobj "src/TermFile/ungetc.o"
#pragma effect TheTerminal

external void  fprintf( TermFile &STREAM, string FORMAT, ...);
#pragma linkname "SACfprintf_TF"
#pragma linkobj "src/TermFile/fprintf.o"
#pragma effect TheTerminal

external void  printf( string FORMAT, ...);
#pragma linkname "SACprintf_TF"
#pragma linkobj "src/TermFile/printf.o"
#pragma effect TheTerminal, stdout

external int, ... fscanf( TermFile &STREAM, string FORMAT);
#pragma linkname "SACfscanf_TF"
#pragma linkobj "src/TermFile/fscanf.o"
#pragma linksign []
#pragma effect TheTerminal

external int, ... scanf( string FORMAT);
#pragma linkname "SACscanf_TF"
#pragma linkobj "src/TermFile/scanf.o"
#pragma linksign []
#pragma effect TheTerminal, stdin
```

```
external string fscans( TermFile &STREAM, int MAX);
#pragma linkname "term_fscans"
#pragma linkobj "src/TermFile/fscans.o"
#pragma linksign []
#pragma effect TheTerminal

external string fscanl( TermFile &STREAM, int MAX);
#pragma linkname "term_fscanl"
#pragma linkobj "src/TermFile/fscanl.o"
#pragma linksign []
#pragma effect TheTerminal

external void  fflush( TermFile &STREAM);
#pragma linkname "SACfflush_TF"
#pragma linkobj "src/TermFile/fflush.o"
#pragma effect TheTerminal

external bool feof( TermFile &STREAM);
#pragma linkname "SACfeof_TF"
#pragma linkobj "src/TermFile/feof.o"
#pragma linksign []
#pragma effect TheTerminal



/*
 *  global objects
 */

TermFile stderr = createStdErr() ;

TermFile stdout = createStdOut() ;

TermFile stdin = createStdIn() ;



/*
 *  function definitions (FUNDEFS)
 */



/*****************************************************************************
 * putc(...) [ body ]
 *****************************************************************************/
inline
void  putc( char C)




/*****************************************************************************
 * getc(...) [ body ]
 *****************************************************************************/
inline
char getc()



/*****************************************************************************
```

```
 * ungetc(...) [ body ]
 ******************************************************************************/
inline
void  ungetc( char C)



/******************************************************************************
 * scans(...) [ body ]
 ******************************************************************************/
inline
string scans( int MAX)



/******************************************************************************
 * scanl(...) [ body ]
 ******************************************************************************/
inline
string scanl( int MAX)



/******************************************************************************
 * flush(...) [ body ]
 ******************************************************************************/
inline
void  flush()



/******************************************************************************
 * eof(...) [ body ]
 ******************************************************************************/
inline
bool eof()


/*-----------------------------------------------*/
```

5 structures

5.1 Array.sac

```
/*-----------------------------------------------*/

module Array;

import ArrayBasics : all;
import ArrayArith : all;
import ArrayTransform : all;
export all;

/*-----------------------------------------------*/
```

5.2 Bits.sac

```
/*------------------------------------------------*/

module Bits;

export all;


/*
 *  prototypes for externals (FUNDECS)
 */

external int BitShiftRight( int k, int val);
#pragma linkname "SAC_Bits_BitShiftRight"
#pragma linkobj "src/Bits/BitShiftRight.o"
#pragma linksign []

external int BitShiftLeft( int k, int val);
#pragma linkname "SAC_Bits_BitShiftLeft"
#pragma linkobj "src/Bits/BitShiftLeft.o"
#pragma linksign []

external int BitRotateRight( int k, int val);
#pragma linkname "SAC_Bits_BitRotateRight"
#pragma linkobj "src/Bits/BitRotateRight.o"
#pragma linksign []

external int BitRotateLeft( int k, int val);
#pragma linkname "SAC_Bits_BitRotateLeft"
#pragma linkobj "src/Bits/BitRotateLeft.o"
#pragma linksign []

external int BitAND( int mask, int val);
#pragma linkname "SAC_Bits_BitAND"
#pragma linkobj "src/Bits/BitAND.o"
#pragma linksign []

external int BitOR( int mask, int val);
#pragma linkname "SAC_Bits_BitOR"
#pragma linkobj "src/Bits/BitOR.o"
#pragma linksign []

external int BitXOR( int mask, int val);
#pragma linkname "SAC_Bits_BitXOR"
#pragma linkobj "src/Bits/BitXOR.o"
#pragma linksign []

external int BitOnesComplement( int val);
#pragma linkname "SAC_Bits_BitOnesComplement"
#pragma linkobj "src/Bits/BitOnesComplement.o"
#pragma linksign []

external int BitTwosComplement( int val);
#pragma linkname "SAC_Bits_BitTwosComplement"
#pragma linkobj "src/Bits/BitTwosComplement.o"
#pragma linksign []


/*------------------------------------------------*/
```

5.3 Bool.sac

```
/*-------------------------------------------*/

module Bool;

export all;


/*
 *  function definitions (FUNDEFS)
 */



/*****************************************************************************
 * toi(...) [ body ]
 *****************************************************************************/
inline
int toi( bool b)




/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( bool i)




/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( byte i)




/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( short i)




/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( int i)




/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
```

```
bool to_bool( long i)



/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( longlong i)



/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( ubyte i)



/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( ushort i)



/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( uint i)



/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( ulong i)



/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
inline
bool to_bool( ulonglong i)



/*****************************************************************************
 * to_bool(...) [ body ]
 *****************************************************************************/
```

```
inline
bool to_bool( float i)



/******************************************************************************
 * to_bool(...) [ body ]
 ******************************************************************************/
inline
bool to_bool( double i)



/*--------------------------------------------*/
```

5.4 Char.sac

```
/*--------------------------------------------*/

module Char;

use ArrayBasics : { shape, sel};
export all;


/*
 *  prototypes for externals (FUNDECS)
 */

external bool isalpha( char C);
#pragma linkname "SACisalpha"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool isupper( char C);
#pragma linkname "SACisupper"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool islower( char C);
#pragma linkname "SACislower"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool isdigit( char C);
#pragma linkname "SACisdigit"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool isxdigit( char C);
#pragma linkname "SACisxdigit"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool isspace( char C);
#pragma linkname "SACisspace"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool ispunct( char C);
#pragma linkname "SACispunct"
```

```
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool isalnum( char C);
#pragma linkname "SACisalnum"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool isprint( char C);
#pragma linkname "SACisprint"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool isgraph( char C);
#pragma linkname "SACisgraph"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool iscntrl( char C);
#pragma linkname "SACiscntrl"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external bool isascii( int N);
#pragma linkname "SACisascii"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external char toascii( int N);
#pragma linkname "SACtoascii"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external char tolower( char C);
#pragma linkname "SACtolower"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external char toupper( char C);
#pragma linkname "SACtoupper"
#pragma linkobj "src/Char/ctype.o"
#pragma linksign []

external char tochar( int N);
#pragma linkname "tochar_int"
#pragma linkobj "src/Char/Char.o"
#pragma linksign []

external char tochar( long N);
#pragma linkname "tochar_long"
#pragma linkobj "src/Char/Char.o"
#pragma linksign []

external char tochar( float N);
#pragma linkname "tochar_float"
#pragma linkobj "src/Char/Char.o"
#pragma linksign []

external char tochar( double N);
#pragma linkname "tochar_double"
#pragma linkobj "src/Char/Char.o"
#pragma linksign []
```

```
external char tochar( char N);
#pragma linkname "tochar_char"
#pragma linkobj "src/Char/Char.o"
#pragma linksign []


/*
 *  function definitions (FUNDEFS)
 */


/******************************************************************************
 * tochar(...) [ body ]
 ******************************************************************************/
inline
char[*] tochar( int[+] a)



/******************************************************************************
 * tochar(...) [ body ]
 ******************************************************************************/
inline
char[*] tochar( long[+] a)



/******************************************************************************
 * tochar(...) [ body ]
 ******************************************************************************/
inline
char[*] tochar( float[+] a)



/******************************************************************************
 * tochar(...) [ body ]
 ******************************************************************************/
inline
char[*] tochar( double[+] a)



/******************************************************************************
 * tochar(...) [ body ]
 ******************************************************************************/
inline
char[*] tochar( bool[+] a)



/******************************************************************************
 * tochar(...) [ body ]
 ******************************************************************************/
```

```
inline
char[*] tochar( char[+] a)


/*--------------------------------------------*/
```

5.5 Color8.sac

```
/*--------------------------------------------*/

module Color8;

use Array : all except { *, -, +, reshape, dim, shape, sel, tod, toi};
import Array : { *, -, +, reshape, dim, shape, sel, tod, toi};
use MathArray : all;
export all except { Weights2Clut, genSteps};


/*
 *  type definitions
 */

typedef int[3] color;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * black(...) [ body ]
 ******************************************************************************/
inline
color black()




/******************************************************************************
 * white(...) [ body ]
 ******************************************************************************/
inline
color white()




/******************************************************************************
 * red(...) [ body ]
 ******************************************************************************/
inline
color red()




/******************************************************************************
 * green(...) [ body ]
 ******************************************************************************/
```

```
inline
color green()



/*****************************************************************************
 * blue(...) [ body ]
 *****************************************************************************/
inline
color blue()



/*****************************************************************************
 * red(...) [ body ]
 *****************************************************************************/
inline
int red( color col)



/*****************************************************************************
 * green(...) [ body ]
 *****************************************************************************/
inline
int green( color col)



/*****************************************************************************
 * blue(...) [ body ]
 *****************************************************************************/
inline
int blue( color col)



/*****************************************************************************
 * newColor(...) [ body ]
 *****************************************************************************/
inline
color newColor( int[3] col)



/*****************************************************************************
 * newColor(...) [ body ]
 *****************************************************************************/
inline
color newColor( int x, int y, int z)



/*****************************************************************************
 * toi(...) [ body ]
```

```
 ******************************************************************************/
inline
int[3] toi( color c)



/******************************************************************************
 * toi(...) [ body ]
 ******************************************************************************/
inline
int[*] toi( color[*] c)



/******************************************************************************
 * tod(...) [ body ]
 ******************************************************************************/
inline
double[3] tod( color c)



/******************************************************************************
 * tod(...) [ body ]
 ******************************************************************************/
inline
double[*] tod( color[*] c)



/******************************************************************************
 * Weights2Clut(...) [ body ]
 ******************************************************************************/
inline
color[256] Weights2Clut( double[256] w, color c1, color c2)



/******************************************************************************
 * genSteps(...) [ body ]
 ******************************************************************************/
inline
double[256] genSteps( double from, double to)



/******************************************************************************
 * genLinearClut(...) [ body ]
 ******************************************************************************/
inline
color[256] genLinearClut( color c1, color c2)



/******************************************************************************
```

```
 * genLogarithmicClut(...) [ body ]
 ************************************************************************/
inline
color[256] genLogarithmicClut( double from, double to, color c1, color c2)



/************************************************************************
 * genExponentialClut(...) [ body ]
 ************************************************************************/
inline
color[256] genExponentialClut( double from, double to, color c1, color c2)



/************************************************************************
 * genAlternatingClut(...) [ body ]
 ************************************************************************/
inline
color[256] genAlternatingClut( color c1, color c2)



/************************************************************************
 * Hsb2Rgb(...) [ body ]
 ************************************************************************/
inline
color Hsb2Rgb( int h_in, int s_in, int b_in)



/************************************************************************
 * Hsb2Rgb(...) [ body ]
 ************************************************************************/
inline
color[*] Hsb2Rgb( int[*] h_in, int s_in, int b_in)



/************************************************************************
 * Hsb2Rgb(...) [ body ]
 ************************************************************************/
inline
color[*] Hsb2Rgb( int[*] h_in, int[*] s_in, int[*] b_in)



/************************************************************************
 * sel(...) [ body ]
 ************************************************************************/
inline
color[*] sel( int idx, color[*] clut)
```

```
/******************************************************************************
 * sel(...) [ body ]
 ******************************************************************************/
inline
color[*] sel( int[.] idx, color[*] a)



/******************************************************************************
 * dim(...) [ body ]
 ******************************************************************************/
inline
int dim( color[*] a)



/******************************************************************************
 * shape(...) [ body ]
 ******************************************************************************/
inline
int[.] shape( color[*] a)



/******************************************************************************
 * reshape(...) [ body ]
 ******************************************************************************/
inline
color[*] reshape( int[.] shp, color[*] a)



/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
color +( color a, color b)



/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
color[*] +( color[*] a, color[*] b)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
color -( color a, color b)
```

```
/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
color[*] -( color[*] a, color[*] b)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
color *( color c, double x)



/******************************************************************************
 * *(...) [ body ]
 ******************************************************************************/
inline
color[*] *( color[*] c, double x)



/*--------------------------------------------------*/
```

5.6 Complex.sac

```
/*--------------------------------------------------*/

module Complex;

import ComplexBasics : all;
import ComplexScalarArith : all;
import ComplexArrayBasics : all;
import ComplexArrayArith : all;
import ComplexArrayTransform : all;
export all;

/*--------------------------------------------------*/
```

5.7 ComplexBasics.sac

```
/*--------------------------------------------------*/

module ComplexBasics;

export all;
use ArrayBasics : { sel};
use Math : { atan2};


/*
 *  type definitions
 */

typedef double[2] complex;
```

```
/*
 *  function definitions (FUNDEFS)
 */


/*****************************************************************************
 * real(...) [ body ]
 *****************************************************************************/
inline
double real( complex CPX)



/*****************************************************************************
 * imag(...) [ body ]
 *****************************************************************************/
inline
double imag( complex CPX)



/*****************************************************************************
 * toc(...) [ body ]
 *****************************************************************************/
inline
complex toc( int REAL, int IMAG)



/*****************************************************************************
 * toc(...) [ body ]
 *****************************************************************************/
inline
complex toc( float REAL, float IMAG)



/*****************************************************************************
 * toc(...) [ body ]
 *****************************************************************************/
inline
complex toc( double REAL, double IMAG)



/*****************************************************************************
 * toc(...) [ body ]
 *****************************************************************************/
inline
complex toc( double[2] COMP)



/*****************************************************************************
```

```
 * toc(...) [ body ]
 ******************************************************************************/
inline
complex toc( int REAL)



/******************************************************************************
 * toc(...) [ body ]
 ******************************************************************************/
inline
complex toc( float REAL)



/******************************************************************************
 * toc(...) [ body ]
 ******************************************************************************/
inline
complex toc( double REAL)



/******************************************************************************
 * toi(...) [ body ]
 ******************************************************************************/
inline
int, int toi( complex C)



/******************************************************************************
 * tof(...) [ body ]
 ******************************************************************************/
inline
float, float tof( complex C)



/******************************************************************************
 * tod(...) [ body ]
 ******************************************************************************/
inline
double, double tod( complex C)



/******************************************************************************
 * todv(...) [ body ]
 ******************************************************************************/
inline
double[2] todv( complex C)
```

```
/*****************************************************************************
 * tos(...) [ body ]
 *****************************************************************************/
inline
String::string tos( complex c)



/*****************************************************************************
 * ptoi(...) [ body ]
 *****************************************************************************/
inline
int, int ptoi( complex C)



/*****************************************************************************
 * ptof(...) [ body ]
 *****************************************************************************/
inline
float, float ptof( complex C)



/*****************************************************************************
 * ptod(...) [ body ]
 *****************************************************************************/
inline
double, double ptod( complex C)



/*****************************************************************************
 * polar(...) [ body ]
 *****************************************************************************/
inline
complex polar( double MAG, double ANGLE)



/*****************************************************************************
 * polar(...) [ body ]
 *****************************************************************************/
inline
complex polar( double MAG)



/*****************************************************************************
 * normSq(...) [ body ]
 *****************************************************************************/
inline
double normSq( complex CPX)
```

```
/**************************************************************************
 * norm(...) [ body ]
 **************************************************************************/
inline
double norm( complex CPX)



/**************************************************************************
 * arg(...) [ body ]
 **************************************************************************/
inline
double arg( complex CPX)



/**************************************************************************
 * i(...) [ body ]
 **************************************************************************/
inline
complex i()



/**************************************************************************
 * zero(...) [ body ]
 **************************************************************************/
inline
complex zero()



/**************************************************************************
 * zero(...) [ body ]
 **************************************************************************/
inline
complex zero( complex[*] a)



/**************************************************************************
 * one(...) [ body ]
 **************************************************************************/
inline
complex one()



/**************************************************************************
 * one(...) [ body ]
 **************************************************************************/
inline
complex one( complex[*] a)
```

```
/*--------------------------------------------*/
```

5.8 ComplexScalarArith.sac

```
/*--------------------------------------------*/

module ComplexScalarArith;

export all;
use ArrayBasics : { sel};
use ScalarArith : { &, |};
use ComplexBasics : all;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * +(...) [ body ]
 *****************************************************************************/
inline
complex +( complex X1, complex X2)




/******************************************************************************
 * -(...) [ body ]
 *****************************************************************************/
inline
complex -( complex X1, complex X2)




/******************************************************************************
 * -(...) [ body ]
 *****************************************************************************/
inline
complex -( complex X1)




/******************************************************************************
 * *(...) [ body ]
 *****************************************************************************/
inline
complex *( complex X1, complex X2)




/******************************************************************************
 * /(...) [ body ]
 *****************************************************************************/
inline
complex /( complex X1, complex X2)
```

```
/*****************************************************************************
 * conj(...) [ body ]
 *****************************************************************************/
inline
complex conj( complex X)



/*****************************************************************************
 * abs(...) [ body ]
 *****************************************************************************/
inline
complex abs( complex X)



/*****************************************************************************
 * ==(...) [ body ]
 *****************************************************************************/
inline
bool ==( complex X1, complex X2)



/*****************************************************************************
 * !=(...) [ body ]
 *****************************************************************************/
inline
bool !=( complex X1, complex X2)



/*****************************************************************************
 * <(...) [ body ]
 *****************************************************************************/
inline
bool <( complex X1, complex X2)



/*****************************************************************************
 * <=(...) [ body ]
 *****************************************************************************/
inline
bool <=( complex X1, complex X2)



/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
```

```
bool >( complex X1, complex X2)



/******************************************************************************
 * >=(...) [ body ]
 ******************************************************************************/
inline
bool >=( complex X1, complex X2)



/******************************************************************************
 * min(...) [ body ]
 ******************************************************************************/
inline
complex min( complex X1, complex X2)



/******************************************************************************
 * max(...) [ body ]
 ******************************************************************************/
inline
complex max( complex X1, complex X2)



/*-----------------------------------------------*/
```

5.9 Constants.sac

```
/*-----------------------------------------------*/

module Constants;

export all;


/*
 *  prototypes for externals (FUNDECS)
 */

external byte minbyte();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external byte maxbyte();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external short minshort();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external short maxshort();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []
```

```
external int minint();
#pragma linkobj "src/Constants/minint.o"
#pragma linksign []

external int maxint();
#pragma linkobj "src/Constants/maxint.o"
#pragma linksign []

external long minlong();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external long maxlong();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external longlong minlonglong();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external longlong maxlonglong();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external ubyte minubyte();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external ubyte maxubyte();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external ushort minushort();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external ushort maxushort();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external uint minuint();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external uint maxuint();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external ulong minulong();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external ulong maxulong();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external ulonglong minulonglong();
#pragma linkobj "src/Constants/minmax.o"
#pragma linksign []

external ulonglong maxulonglong();
#pragma linkobj "src/Constants/minmax.o"
```

```
#pragma linksign []

external float minfloat();
#pragma linkobj "src/Constants/minfloat.o"
#pragma linksign []

external float maxfloat();
#pragma linkobj "src/Constants/maxfloat.o"
#pragma linksign []

external double mindouble();
#pragma linkobj "src/Constants/mindouble.o"
#pragma linksign []

external double tinydouble();
#pragma linkobj "src/Constants/tinydouble.o"
#pragma linksign []

external double maxdouble();
#pragma linkobj "src/Constants/maxdouble.o"
#pragma linksign []

external double epidouble();
#pragma linkobj "src/Constants/epidouble.o"
#pragma linksign []


/*-----------------------------------------------*/
```

5.10 Grey.sac

```
/*-----------------------------------------------*/

module Grey;

use Array : { min, max};
import ScalarArith : { tod, toi};
export all;


/*
 *  type definitions
 */

typedef int grey;


/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * newGrey(...) [ body ]
 ******************************************************************************/
inline
grey newGrey( int g)
```

```
/******************************************************************************
 * newGrey(...) [ body ]
 ******************************************************************************/
inline
grey newGrey( double g)



/******************************************************************************
 * toi(...) [ body ]
 ******************************************************************************/
inline
int[*] toi( grey g)



/******************************************************************************
 * tod(...) [ body ]
 ******************************************************************************/
inline
double[*] tod( grey g)



/******************************************************************************
 * sel(...) [ body ]
 ******************************************************************************/
inline
grey[*] sel( int[.] idx, grey[*] a)



/******************************************************************************
 * dim(...) [ body ]
 ******************************************************************************/
inline
int dim( grey[*] a)



/******************************************************************************
 * shape(...) [ body ]
 ******************************************************************************/
inline
int[.] shape( grey[*] a)



/******************************************************************************
 * reshape(...) [ body ]
 ******************************************************************************/
inline
grey[*] reshape( int[.] shp, grey[*] a)
```

```
/******************************************************************************
 * +(...) [ body ]
 ******************************************************************************/
inline
grey +( grey g1, grey g2)



/******************************************************************************
 * -(...) [ body ]
 ******************************************************************************/
inline
grey -( grey g1, grey g2)


/*--------------------------------------------*/
```

5.11 List.sac

```
/*--------------------------------------------*/

module List;

export all;


/*
 *  type definitions
 */

external typedef list;


/*
 *  prototypes for externals (FUNDECS)
 */

external list nil();
#pragma linkname "SAC_List_nil"
#pragma linkobj "src/List/nil.o"

external list cons( int ELEM, list LIST);
#pragma linkname "SAC_List_cons"
#pragma linkobj "src/List/cons.o"

external int hd( list LIST);
#pragma linkname "SAC_List_hd"
#pragma linkobj "src/List/hd.o"
#pragma linksign []

external list tl( list LIST);
#pragma linkname "SAC_List_tl"
#pragma linkobj "src/List/tl.o"

external bool empty( list LIST);
#pragma linkname "SAC_List_empty"
#pragma linkobj "src/List/empty.o"
#pragma linksign []
```

```
external list append( list LIST1, list LIST2);
#pragma linkname "SAC_List_append"
#pragma linkobj "src/List/append.o"

external int nth( int N, list LIST);
#pragma linkname "SAC_List_nth"
#pragma linkobj "src/List/nth.o"
#pragma linksign []

external int length( list LIST);
#pragma linkname "SAC_List_length"
#pragma linkobj "src/List/length.o"
#pragma linksign []

external list drop( int N, list LIST);
#pragma linkname "SAC_List_drop"
#pragma linkobj "src/List/drop.o"

external list take( int N, list LIST);
#pragma linkname "SAC_List_take"
#pragma linkobj "src/List/take.o"


/*------------------------------------------------*/
```

5.12 String.sac

```
/*------------------------------------------------*/

module String;

import Char : all;
import ScalarArith : { &, !=, ==, +, >=, >, <=, <, ++};
import ArrayBasics : { shape, sel};
export all except { indent};


/*
 *  type definitions
 */

external typedef string;


/*
 *  prototypes for externals (FUNDECS)
 */

external string to_string( char[.] A, int LENGTH);
#pragma linkobj "src/String/tostring.o"

external string strmod( string S, int P, char C);
#pragma linkobj "src/String/strmod.o"

external string strins( string S1, int P, string S2);
#pragma linkobj "src/String/strins.o"
#pragma linksign []

external string strovwt( string S1, int P, string S2);
#pragma linkobj "src/String/strovwt.o"

external char strsel( string S, int P);
```

```
#pragma linkobj "src/String/strsel.o"
#pragma linksign []

external string strcat( string S1, string S2);
#pragma linkname "SACstrcat"
#pragma linkobj "src/String/strcat.o"
#pragma linksign []

external string +( string S1, string S2);
#pragma linkname "SACstrcat"
#pragma linkobj "src/String/strcat.o"
#pragma linksign []

external string strncat( string S1, string S2, int N);
#pragma linkname "SACstrncat"
#pragma linkobj "src/String/strncat.o"
#pragma linksign []

external int strcmp( string S1, string S2);
#pragma linkname "SACstrcmp"
#pragma linkobj "src/String/strcmp.o"
#pragma linksign []

external int strncmp( string S1, string S2, int N);
#pragma linkname "SACstrncmp"
#pragma linkobj "src/String/strncmp.o"
#pragma linksign []

external int strcasecmp( string S1, string S2);
#pragma linkname "SACstrcasecmp"
#pragma linkobj "src/String/strcasecmp.o"
#pragma linksign []

external int strncasecmp( string S1, string S2, int N);
#pragma linkname "SACstrncasecmp"
#pragma linkobj "src/String/strncasecmp.o"
#pragma linksign []

external int strlen( string S);
#pragma linkname "SACstrlen"
#pragma linkobj "src/String/strlen.o"
#pragma linksign []

external string strtake( string S, int N);
#pragma linkobj "src/String/strtake.o"

external string strdrop( string S, int N);
#pragma linkobj "src/String/strdrop.o"
#pragma linksign []

external string strext( string S, int FIRST, int LEN);
#pragma linkobj "src/String/strext.o"
#pragma linksign []

external string sprintf( string FORMAT, ...);
#pragma linkname "SACsprintf"
#pragma linkobj "src/String/sprintf.o"
#pragma linksign []

external int, ... sscanf( string S, string FORMAT);
#pragma linkname "SACsscanf"
#pragma linkobj "src/String/sscanf.o"
```

```
#pragma linksign []

external string sscanf_str( string S, string FORMAT);
#pragma linkobj "src/String/sscanfstr.o"
#pragma linksign []

external int strchr( string S, char C);
#pragma linkname "SACstrchr"
#pragma linkobj "src/String/strchr.o"

external int strrchr( string S, char C);
#pragma linkname "SACstrrchr"
#pragma linkobj "src/String/strrchr.o"
#pragma linksign []

external int strcspn( string S, string REJECT);
#pragma linkname "SACstrcspn"
#pragma linkobj "src/String/strcspn.o"
#pragma linksign []

external int strspn( string S, string ACCEPT);
#pragma linkname "SACstrspn"
#pragma linkobj "src/String/strspn.o"
#pragma linksign []

external int strstr( string HAYSTACK, string NEEDLE);
#pragma linkname "SACstrstr"
#pragma linkobj "src/String/strstr.o"
#pragma linksign []

external string, string strtok( string S, string SEP);
#pragma linkname "SACstrtok"
#pragma linkobj "src/String/strtok.o"
#pragma linksign []

external string chomp( string S);
#pragma linkname "SACchomp"
#pragma linkobj "src/String/trim.o"
#pragma linksign []

external string rtrim( string S);
#pragma linkname "SACrtrim"
#pragma linkobj "src/String/trim.o"
#pragma linksign []

external string ltrim( string S);
#pragma linkname "SACltrim"
#pragma linkobj "src/String/trim.o"
#pragma linksign []

external string trim( string S);
#pragma linkname "SACtrim"
#pragma linkobj "src/String/trim.o"
#pragma linksign []

external int, string strtoi( string S, int BASE);
#pragma linkname "SACstrtoi"
#pragma linkobj "src/String/strtoi.o"
#pragma linksign []

external float, string strtof( string S);
#pragma linkname "SACstrtof"
```

```
#pragma linkobj "src/String/strtof.o"
#pragma linksign []

external double, string strtod( string S);
#pragma linkname "SACstrtod"
#pragma linkobj "src/String/strtod.o"
#pragma linksign []

external int toi( string S);
#pragma linkname "SACtoi"
#pragma linkobj "src/String/strtoi.o"
#pragma linksign []

external float tof( string S);
#pragma linkname "SACtof"
#pragma linkobj "src/String/strtof.o"
#pragma linksign []

external double tod( string S);
#pragma linkname "SACtod"
#pragma linkobj "src/String/strtod.o"
#pragma linksign []

external string tos( int N);
#pragma linkname "SACitos"
#pragma linkobj "src/String/itos.o"
#pragma linksign []

external string tos( float N);
#pragma linkname "SACftos"
#pragma linkobj "src/String/ftos.o"
#pragma linksign []

external string tos( double N);
#pragma linkname "SACdtos"
#pragma linkobj "src/String/dtos.o"
#pragma linksign []

external string tos( bool B);
#pragma linkname "SACbtos"
#pragma linkobj "src/String/btos.o"
#pragma linksign []



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * indent(...) [ body ]
 *****************************************************************************/
inline
string indent( int indent, string str)



/******************************************************************************
 * tos(...) [ body ]
```

```
 ******************************************************************************/
inline
string tos( int[+] in)



/******************************************************************************
 * tos(...) [ body ]
 ******************************************************************************/
inline
string tos( int in, int indent)



/******************************************************************************
 * tos(...) [ body ]
 ******************************************************************************/
inline
string tos( int[+] in, int indent)



/******************************************************************************
 * tos(...) [ body ]
 ******************************************************************************/
inline
string tos( float[+] in)



/******************************************************************************
 * tos(...) [ body ]
 ******************************************************************************/
inline
string tos( float in, int indent)



/******************************************************************************
 * tos(...) [ body ]
 ******************************************************************************/
inline
string tos( float[+] in, int indent)



/******************************************************************************
 * tos(...) [ body ]
 ******************************************************************************/
inline
string tos( double[+] in)



/******************************************************************************
```

```
 * tos(...) [ body ]
 ****************************************************************************/
inline
string tos( double in, int indent)



/****************************************************************************
 * tos(...) [ body ]
 ****************************************************************************/
inline
string tos( double[+] in, int indent)



/****************************************************************************
 * tos(...) [ body ]
 ****************************************************************************/
inline
string tos( bool[+] in)



/****************************************************************************
 * tos(...) [ body ]
 ****************************************************************************/
inline
string tos( bool in, int indent)



/****************************************************************************
 * tos(...) [ body ]
 ****************************************************************************/
inline
string tos( bool[+] in, int indent)



/****************************************************************************
 * to_string(...) [ body ]
 ****************************************************************************/
inline
string to_string( char[.] arr)



/****************************************************************************
 * tochar(...) [ body ]
 ****************************************************************************/
inline
char[.] tochar( string s)
```

```
/*****************************************************************************
 * sel(...) [ body ]
 *****************************************************************************/
inline
char sel( int[.] index, string s)



/*****************************************************************************
 * modarray(...) [ body ]
 *****************************************************************************/
inline
string modarray( string s, int[1] index, char c)



/*****************************************************************************
 * modarray(...) [ body ]
 *****************************************************************************/
inline
string modarray( string s1, int[1] index, string s2)



/*****************************************************************************
 * ==(...) [ body ]
 *****************************************************************************/
inline
bool ==( string A, string B)



/*****************************************************************************
 * !=(...) [ body ]
 *****************************************************************************/
inline
bool !=( string A, string B)



/*****************************************************************************
 * <(...) [ body ]
 *****************************************************************************/
inline
bool <( string A, string B)



/*****************************************************************************
 * <=(...) [ body ]
 *****************************************************************************/
inline
bool <=( string A, string B)
```

```
/*****************************************************************************
 * >(...) [ body ]
 *****************************************************************************/
inline
bool >( string A, string B)



/*****************************************************************************
 * >=(...) [ body ]
 *****************************************************************************/
inline
bool >=( string A, string B)



/*****************************************************************************
 * isalpha(...) [ body ]
 *****************************************************************************/
inline
bool isalpha( string S)



/*****************************************************************************
 * isupper(...) [ body ]
 *****************************************************************************/
inline
bool isupper( string S)



/*****************************************************************************
 * islower(...) [ body ]
 *****************************************************************************/
inline
bool islower( string S)



/*****************************************************************************
 * isdigit(...) [ body ]
 *****************************************************************************/
inline
bool isdigit( string S)



/*****************************************************************************
 * isxdigit(...) [ body ]
 *****************************************************************************/
inline
bool isxdigit( string S)
```

```
/*****************************************************************************
 * isspace(...) [ body ]
 *****************************************************************************/
inline
bool isspace( string S)



/*****************************************************************************
 * ispunct(...) [ body ]
 *****************************************************************************/
inline
bool ispunct( string S)



/*****************************************************************************
 * isalnum(...) [ body ]
 *****************************************************************************/
inline
bool isalnum( string S)



/*****************************************************************************
 * isprint(...) [ body ]
 *****************************************************************************/
inline
bool isprint( string S)



/*****************************************************************************
 * isgraph(...) [ body ]
 *****************************************************************************/
inline
bool isgraph( string S)



/*****************************************************************************
 * iscntrl(...) [ body ]
 *****************************************************************************/
inline
bool iscntrl( string S)



/*****************************************************************************
 * tolower(...) [ body ]
 *****************************************************************************/
inline
string tolower( string S)
```

```
/******************************************************************************
 * toupper(...) [ body ]
 ******************************************************************************/
inline
string toupper( string S)


/*--------------------------------------------*/
```

5.13 StringArray.sac

```
/*--------------------------------------------*/

module StringArray;

use String : { string};
use Array : { *, all, -, abs, <, >=, +, zero, ++};
export all except { eq_SxS, eq_SxA, eq_AxS, eq_AxA, getIndicies, modarray_AxVxS,
   ↪  sel_VxA, stringArrayCreator, drop};


/*
 *  type definitions
 */

external typedef stringArray;


/*
 *  prototypes for externals (FUNDECS)
 */

external stringArray modarray_AxVxS( stringArray labs, int[.] idx, string str);
#pragma linkname "SAC_StringArray_modarray"
#pragma linkobj "src/StringArray/index2offset.o" "src/StringArray/modarray.o"

external string sel_VxA( int[.] idx, stringArray labs);
#pragma linkname "SAC_StringArray_sel"
#pragma linkobj "src/StringArray/index2offset.o" "src/StringArray/sel.o"

external int dim( stringArray s);
#pragma linkname "SAC_StringArray_dim"
#pragma linkobj "src/StringArray/dim.o"

external int[.] shape( stringArray s);
#pragma linkname "SAC_StringArray_shape"
#pragma linkobj "src/StringArray/shape.o"

external stringArray stringArrayCreator( int[.] shp, string s);
#pragma linkname "SAC_StringArray_genarray"
#pragma linkobj "src/StringArray/genarray.o"



/*
 *  function definitions (FUNDEFS)
 */
```

```
/*****************************************************************************
 * drop(...) [ body ]
 *****************************************************************************/
stringArray drop( int[.] v, stringArray array)



/*****************************************************************************
 * where(...) [ body ]
 *****************************************************************************/
inline
int[+] where( bool[+] p, int A, int[+] B)



/*****************************************************************************
 * modarray(...) [ body ]
 *****************************************************************************/
stringArray modarray( stringArray array, int[.] idx, stringArray val)



/*****************************************************************************
 * sel(...) [ body ]
 *****************************************************************************/
stringArray sel( int[.] idx, stringArray array)



/*****************************************************************************
 * to_string(...) [ body ]
 *****************************************************************************/
string to_string( stringArray a)



/*****************************************************************************
 * to_stringArray(...) [ body ]
 *****************************************************************************/
stringArray to_stringArray( string s)



/*****************************************************************************
 * genarray(...) [ body ]
 *****************************************************************************/
stringArray genarray( int[.] shp, stringArray s)



/*****************************************************************************
 * eq_SxS(...) [ body ]
 *****************************************************************************/
```

```
inline
bool eq_SxS( stringArray a, stringArray b)



/******************************************************************************
 * eq_SxA(...) [ body ]
 ******************************************************************************/
inline
bool[*] eq_SxA( stringArray a, stringArray b)



/******************************************************************************
 * eq_AxS(...) [ body ]
 ******************************************************************************/
inline
bool[*] eq_AxS( stringArray a, stringArray b)



/******************************************************************************
 * eq_AxA(...) [ body ]
 ******************************************************************************/
inline
bool[*] eq_AxA( stringArray a, stringArray b)



/******************************************************************************
 * ==(...) [ body ]
 ******************************************************************************/
inline
bool[*] ==( stringArray a, stringArray b)



/******************************************************************************
 * getIndicies(...) [ body ]
 ******************************************************************************/
int[.,.] getIndicies( int[.] shp)



/******************************************************************************
 * concatStringArrays(...) [ body ]
 ******************************************************************************/
stringArray concatStringArrays( stringArray m1, stringArray m2)



/******************************************************************************
 * in(...) [ body ]
 ******************************************************************************/
bool in( string needle, stringArray haystack)
```

```
/******************************************************************************
 * indexOf(...) [ body ]
 *****************************************************************************/
int[.] indexOf( string needle, stringArray haystack)


/*--------------------------------------------------*/
```

5.14 Structures.sac

```
/*--------------------------------------------------*/

module Structures;

import Array : all;
import Char : all;
import Bits : all;
import String : all;
export all;

/*--------------------------------------------------*/
```

6 system

6.1 Clock.sac

```
/*--------------------------------------------------*/

class Clock;

use String : { string};
export all;


/*
 *  type definitions
 */

external classtype Clock::Clock;
external typedef time;


/*
 *  prototypes for externals (FUNDECS)
 */

external Clock create_TheClock();
#pragma linkobj "src/Clock/clock.o"
#pragma linksign []
#pragma effect World::TheWorld

external time to_time( int secs);
#pragma linkname "SACto_time"
#pragma linkobj "src/Clock/to_time.o"
#pragma linksign []
```

```
external time gettime();
#pragma linkname "SACgettime"
#pragma linkobj "src/Clock/gettime.o"
#pragma linksign []
#pragma effect TheClock

external time, bool mktime( int YEAR, int MON, int DAY, int HOUR, int MIN, int
   ↪ SEC);
#pragma linkname "SACmktime"
#pragma linkobj "src/Clock/mktime.o"
#pragma linksign []

external int sec( time T);
#pragma linkname "SACsec"
#pragma linkobj "src/Clock/extracttime.o"
#pragma linksign []

external int min( time T);
#pragma linkname "SACmin"
#pragma linkobj "src/Clock/extracttime.o"
#pragma linksign []

external int hour( time T);
#pragma linkname "SAChour"
#pragma linkobj "src/Clock/extracttime.o"
#pragma linksign []

external int mday( time T);
#pragma linkname "SACmday"
#pragma linkobj "src/Clock/extracttime.o"
#pragma linksign []

external int mon( time T);
#pragma linkname "SACmon"
#pragma linkobj "src/Clock/extracttime.o"
#pragma linksign []

external int year( time T);
#pragma linkname "SACyear"
#pragma linkobj "src/Clock/extracttime.o"
#pragma linksign []

external int wday( time T);
#pragma linkname "SACwday"
#pragma linkobj "src/Clock/extracttime.o"
#pragma linksign []

external int yday( time T);
#pragma linkname "SACyday"
#pragma linkobj "src/Clock/extracttime.o"
#pragma linksign []

external int, int, int clock( time T);
#pragma linkname "SACclock"
#pragma linkobj "src/Clock/date.o"
#pragma linksign []

external int, int, int date( time T);
#pragma linkname "SACdate"
#pragma linkobj "src/Clock/date.o"
#pragma linksign []
```

```
external int isdst( time T);
#pragma linkname "SACisdst"
#pragma linkobj "src/Clock/isdst.o"
#pragma linksign []

external bool isleap( int YEAR);
#pragma linkname "SACisleap"
#pragma linkobj "src/Clock/isleap.o"
#pragma linksign []

external bool isleap( time T);
#pragma linkname "SACisleapt"
#pragma linkobj "src/Clock/isleap.o"
#pragma linksign []

external double difftime( time T1, time T0);
#pragma linkname "SACdifftime"
#pragma linkobj "src/Clock/difftime.o"
#pragma linksign []

external string ctime( time T);
#pragma linkname "SACctime"
#pragma linkobj "src/Clock/ctime.o"
#pragma linksign []

external string strftime( int LEN, string FORMAT, time T);
#pragma linkname "SACstrftime"
#pragma linkobj "src/Clock/strftime.o"
#pragma linksign []

external time, string strptime( string S, string FORMAT);
#pragma linkname "SACstrptime"
#pragma linkobj "src/Clock/strptime.o"
#pragma linksign []

external void  sleep( int SECONDS);
#pragma linkname "SACsleep"
#pragma linkobj "src/Clock/sleep.o"
#pragma linksign []
#pragma effect World::TheWorld



/*
 *  global objects
 */

Clock TheClock = create_TheClock() ;


/*-----------------------------------------------*/
```

6.2 CommandLine.sac

```
/*-----------------------------------------------*/

class CommandLine;

use String : { string};
export all;
```

```
/*
 *  type definitions
 */

external classtype CommandLine::CommandLine;


/*
 *  prototypes for externals (FUNDECS)
 */

external CommandLine create_TheCommandLine();
#pragma linkobj "src/CommandLine/CommandLine.o"
#pragma linksign []
#pragma effect World::TheWorld

external int argc();
#pragma linkname "SACargc"
#pragma linkobj "src/CommandLine/CommandLine.o"
#pragma linksign []
#pragma effect TheCommandLine

external string argv();
#pragma linkname "SACargvall"
#pragma linkobj "src/CommandLine/CommandLine.o"
#pragma linksign []
#pragma effect TheCommandLine

external string argv( int N);
#pragma linkname "SACargv"
#pragma linkobj "src/CommandLine/CommandLine.o"
#pragma linksign []
#pragma effect TheCommandLine



/*
 *  global objects
 */

CommandLine TheCommandLine = create_TheCommandLine() ;


/*-----------------------------------------------*/
```

6.3 Dir.sac

```
/*-----------------------------------------------*/

class Dir;

use SysErr : { syserr};
use String : { string};
use FileSystem : { TheFileSystem};
export all;


/*
 *  type definitions
 */

external classtype Dir::Dir;
```

```
/*
 *  prototypes for externals (FUNDECS)
 */

external syserr, Dir opendir( string NAME);
#pragma linkname "SACopendir"
#pragma linkobj "src/Dir/opendir.o"
#pragma linksign []
#pragma effect TheFileSystem

external void  closedir( Dir &DIR);
#pragma linkname "SACclosedir"
#pragma linkobj "src/Dir/closedir.o"
#pragma linksign []
#pragma effect TheFileSystem

external string readdir( Dir &DIR);
#pragma linkname "SACreaddir"
#pragma linkobj "src/Dir/readdir.o"
#pragma linksign []
#pragma effect TheFileSystem

external void  rewinddir( Dir &DIR);
#pragma linkname "SACrewinddir"
#pragma linkobj "src/Dir/rewinddir.o"
#pragma linksign []
#pragma effect TheFileSystem

external long telldir( Dir &DIR);
#pragma linkname "SACtelldir"
#pragma linkobj "src/Dir/telldir.o"
#pragma linksign []
#pragma effect TheFileSystem

external void  seekdir( Dir &DIR, long POS);
#pragma linkname "SACseekdir"
#pragma linkobj "src/Dir/seekdir.o"
#pragma linksign []
#pragma effect TheFileSystem


/*--------------------------------------------------*/
```

6.4 Environment.sac

```
/*--------------------------------------------------*/

class Environment;

use String : { string};
export all;


/*
 *  type definitions
 */

external classtype Environment::Environment;
```

```
/*
 *  prototypes for externals (FUNDECS)
 */

external Environment create_TheEnvironment();
#pragma linkobj "src/Environment/Env.o"
#pragma linksign []
#pragma effect World::TheWorld

external string GetEnv( string ENVVAR);
#pragma linkobj "src/Environment/GetEnv.o"
#pragma linksign []
#pragma effect TheEnvironment

external bool ExistEnv( string ENVVAR);
#pragma linkobj "src/Environment/ExistEnv.o"
#pragma linksign []
#pragma effect TheEnvironment

external bool SetEnv( string ENVVAR, string VALUE, bool OVERWRITE);
#pragma linkobj "src/Environment/SetEnv.o"
#pragma linksign []
#pragma effect TheEnvironment

external void  UnsetEnv( string ENVVAR);
#pragma linkobj "src/Environment/UnsetEnv.o"
#pragma effect TheEnvironment

external int EnvCount();
#pragma linkobj "src/Environment/Environ.o"
#pragma linksign []
#pragma effect TheEnvironment

external string IndexEnv( int N);
#pragma linkobj "src/Environment/Environ.o"
#pragma linksign []
#pragma effect TheEnvironment



/*
 *  global objects
 */

Environment TheEnvironment = create_TheEnvironment() ;


/*-----------------------------------------------*/
```

6.5 FileSystem.sac

```
/*-----------------------------------------------*/

class FileSystem;

use SysErr : { syserr};
use String : { string};
export all;


/*
 *  type definitions
```

```
 */

external classtype FileSystem::FileSystem;


/*
 *  prototypes for externals (FUNDECS)
 */

external FileSystem create_TheFileSystem();
#pragma linkobj "src/FileSystem/filesys.o"
#pragma linksign []
#pragma effect World::TheWorld

external syserr rename( string OLDNAME, string NEWNAME);
#pragma linkname "SACrename"
#pragma linkobj "src/FileSystem/rename.o"
#pragma linksign []
#pragma effect TheFileSystem

external syserr remove( string PATHNAME);
#pragma linkname "SACremove"
#pragma linkobj "src/FileSystem/remove.o"
#pragma linksign []
#pragma effect TheFileSystem

external syserr symlink( string PATHNAME, string LINKNAME);
#pragma linkname "SACsymlink"
#pragma linkobj "src/FileSystem/symlink.o"
#pragma linksign []
#pragma effect TheFileSystem

external bool, syserr access( string PATHNAME, int HOW);
#pragma linkname "SACaccess"
#pragma linkobj "src/FileSystem/access.o"
#pragma linksign []
#pragma effect TheFileSystem

external bool, syserr isdir( string PATHNAME);
#pragma linkname "SACisdir"
#pragma linkobj "src/FileSystem/testfile.o"
#pragma linksign []
#pragma effect TheFileSystem

external bool, syserr isreg( string PATHNAME);
#pragma linkname "SACisreg"
#pragma linkobj "src/FileSystem/testfile.o"
#pragma linksign []
#pragma effect TheFileSystem

external bool, syserr islnk( string PATHNAME);
#pragma linkname "SACislnk"
#pragma linkobj "src/FileSystem/testfile.o"
#pragma linksign []
#pragma effect TheFileSystem

external ulonglong, syserr filesize( string PATHNAME);
#pragma linkname "SACfilesize"
#pragma linkobj "src/FileSystem/testfile.o"
#pragma linksign []
#pragma effect TheFileSystem
```

```
external string getcwd();
#pragma linkname "SACgetcwd"
#pragma linkobj "src/FileSystem/dir.o"
#pragma linksign []
#pragma effect TheFileSystem

external syserr chdir( string PATHNAME);
#pragma linkname "SACchdir"
#pragma linkobj "src/FileSystem/dir.o"
#pragma linksign []
#pragma effect TheFileSystem

external syserr mkdir( string PATHNAME);
#pragma linkname "SACmkdir"
#pragma linkobj "src/FileSystem/dir.o"
#pragma linksign []
#pragma effect TheFileSystem

external syserr rmdir( string PATHNAME);
#pragma linkname "SACrmdir"
#pragma linkobj "src/FileSystem/dir.o"
#pragma linksign []
#pragma effect TheFileSystem

external string P_tmpdir();
#pragma linkname "SACPtmpdir"
#pragma linkobj "src/FileSystem/pltmp.o"
#pragma linksign []

external int L_tmpnam();
#pragma linkname "SACLtmpnam"
#pragma linkobj "src/FileSystem/pltmp.o"
#pragma linksign []

external string mktemp( string TEMPLATE);
#pragma linkname "SACmktemp"
#pragma linkobj "src/FileSystem/mktemp.o"
#pragma linksign []
#pragma effect TheFileSystem

external string tmpnam();
#pragma linkname "SACtmpnam"
#pragma linkobj "src/FileSystem/tmpnam.o"
#pragma linksign []
#pragma effect TheFileSystem

external string tmpnam( string DIR, string PREFIX);
#pragma linkname "SACtempnam"
#pragma linkobj "src/FileSystem/tempnam.o"
#pragma linksign []
#pragma effect TheFileSystem



/*
 *  global objects
 */

FileSystem TheFileSystem = create_TheFileSystem() ;


/*-----------------------------------------------*/
```

6.6 GetOpt.sac

```
/*------------------------------------------------*/

module GetOpt;

use String : { string};
use CommandLine : { TheCommandLine};
export all;


/*
 *  prototypes for externals (FUNDECS)
 */

external char getopt( string opts);
#pragma linkname "getopt_sac"
#pragma linkobj "src/GetOpt/getopt.o"
#pragma linksign []
#pragma effect TheCommandLine

external char optEND();
#pragma linkname "optEND"
#pragma linkobj "src/GetOpt/getopt.o"
#pragma linksign []

external string optarg();
#pragma linkname "get_optarg"
#pragma linkobj "src/GetOpt/getopt.o"
#pragma linksign []
#pragma effect TheCommandLine

external char optopt();
#pragma linkname "get_optopt"
#pragma linkobj "src/GetOpt/getopt.o"
#pragma linksign []
#pragma effect TheCommandLine

external int optind();
#pragma linkname "get_optind"
#pragma linkobj "src/GetOpt/getopt.o"
#pragma linksign []
#pragma effect TheCommandLine

external void  optind( int set);
#pragma linkname "set_optind"
#pragma linkobj "src/GetOpt/getopt.o"
#pragma linksign []
#pragma effect TheCommandLine

external bool opterr();
#pragma linkname "get_opterr"
#pragma linkobj "src/GetOpt/getopt.o"
#pragma linksign []
#pragma effect TheCommandLine

external void  opterr( bool set);
#pragma linkname "set_opterr"
#pragma linkobj "src/GetOpt/getopt.o"
#pragma linksign []
#pragma effect TheCommandLine
```

```
/*-----------------------------------------------*/
```

6.7 MTClock.sac

```
/*-----------------------------------------------*/

class MTClock;

export { TheMTClock, touch, gettime};


/*
 *  type definitions
 */

external classtype MTClock::MTClock;


/*
 *  prototypes for externals (FUNDECS)
 */

external MTClock createTheMTClock();
#pragma linkname "SAC_MTClock_createTheMTClock"
#pragma linkobj "src/MTClock/mtclock.o"
#pragma linksign []
#pragma effect World::TheWorld

external void  touch( MTClock &mtclock);
#pragma linkname "SAC_MTClock_touch"
#pragma linkobj "src/MTClock/mtclock.o"

external long, long gettime();
#pragma linkname "SAC_MTClock_gettime"
#pragma linkobj "src/MTClock/mtclock.o"
#pragma linksign []
#pragma effect MTClock::TheMTClock



/*
 *  global objects
 */

MTClock TheMTClock = createTheMTClock() ;


/*-----------------------------------------------*/
```

6.8 Process.sac

```
/*-----------------------------------------------*/

module Process;

use FileSystem : { TheFileSystem};
use File : { File};
use String : { string};
use SysErr : { syserr};
export all;
```

```
/*
 *  prototypes for externals (FUNDECS)
 */

external syserr, File popen( string COMMAND, string MODE);
#pragma linkname "SACpopen"
#pragma linkobj "src/Process/process.o"
#pragma linksign []
#pragma effect TheFileSystem

external void  pclose( File STREAM);
#pragma linkname "SACpclose"
#pragma linkobj "src/Process/pclose.o"
#pragma effect TheFileSystem

external int system( string COMMAND);
#pragma linkname "SACsystem"
#pragma linkobj "src/Process/system.o"
#pragma linksign []
#pragma effect TheFileSystem


/*-----------------------------------------------*/
```

6.9 RTClock.sac

```
/*-----------------------------------------------*/

class RTClock;

export { TheRTClock, touch, gettime};


/*
 *  type definitions
 */

external classtype RTClock::RTClock;


/*
 *  prototypes for externals (FUNDECS)
 */

external RTClock createTheRTClock();
#pragma linkname "SAC_RTClock_createTheRTClock"
#pragma linkobj "src/RTClock/rtclock.o"
#pragma linksign []
#pragma effect World::TheWorld

external void  touch( RTClock &rtclock);
#pragma linkname "SAC_RTClock_touch"
#pragma linkobj "src/RTClock/rtclock.o"

external long, long gettime();
#pragma linkname "SAC_RTClock_gettime"
#pragma linkobj "src/RTClock/rtclock.o"
#pragma linksign []
#pragma effect RTClock::TheRTClock
```

```
/*
 *  global objects
 */

RTClock TheRTClock = createTheRTClock() ;


/*-----------------------------------------------*/
```

6.10 RTimer.sac

```
/*-----------------------------------------------*/

class RTimer;

use RTClock : all;
export all;


/*
 *  type definitions
 */

external classtype RTimer::RTimer;


/*
 *  prototypes for externals (FUNDECS)
 */

external RTimer createRTimer();
#pragma linkname "SAC_RTimer_createRTimer"
#pragma linkobj "src/RTimer/rtimer.o"
#pragma linksign []
#pragma effect RTClock::TheRTClock

external void  destroyRTimer( RTimer rtimer);
#pragma linkname "SAC_RTimer_destroyRTimer"
#pragma linkobj "src/RTimer/rtimer.o"
#pragma effect RTClock::TheRTClock

external void  startRTimer( RTimer &rtimer);
#pragma linkname "SAC_RTimer_startRTimer"
#pragma linkobj "src/RTimer/rtimer.o"
#pragma effect RTClock::TheRTClock

external void  stopRTimer( RTimer &rtimer);
#pragma linkname "SAC_RTimer_stopRTimer"
#pragma linkobj "src/RTimer/rtimer.o"
#pragma effect RTClock::TheRTClock

external void  resetRTimer( RTimer &rtimer);
#pragma linkname "SAC_RTimer_resetRTimer"
#pragma linkobj "src/RTimer/rtimer.o"

external int, int getRTimerInts( RTimer &rtimer);
#pragma linkname "SAC_RTimer_getRTimerInts"
#pragma linkobj "src/RTimer/rtimer.o"
#pragma linksign []
```

```
external double getRTimerDbl( RTimer &rtimer);
#pragma linkname "SAC_RTimer_getRTimerDbl"
#pragma linkobj "src/RTimer/rtimer.o"
#pragma linksign []


/*-----------------------------------------------*/
```

6.11 RuntimeError.sac

```
/*-----------------------------------------------*/

module RuntimeError;

export all;


/*
 *  prototypes for externals (FUNDECS)
 */

external void  error( int result, String::string message, ...);
#pragma linkname "SAC__RUNTIMEERROR_error"
#pragma linkobj "src/RuntimeError/error.o"
#pragma effect Terminal::TheTerminal


/*-----------------------------------------------*/
```

6.12 SysErr.sac

```
/*-----------------------------------------------*/

module SysErr;

use String : { string};
export all;


/*
 *  type definitions
 */

typedef int syserr;


/*
 *  prototypes for externals (FUNDECS)
 */

external bool fail( syserr ERROR);
#pragma linkobj "src/SysErr/failsucc.o"
#pragma linksign []

external bool clear( syserr ERROR);
#pragma linkobj "src/SysErr/failsucc.o"
#pragma linksign []

external string strerror( syserr ERROR);
#pragma linkname "SACstrerror"
#pragma linkobj "src/SysErr/strerror.o"
```

```
#pragma linksign []

external syserr Eperm();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enoent();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Esrch();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eintr();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eio();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enxio();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr E2big();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enoexec();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Ebadf();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Echild();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eagain();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enomem();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eacces();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Efault();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enotblk();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []
```

```
external syserr Ebusy();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eexist();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Exdev();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enodev();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enotdir();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eisdir();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Einval();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enfile();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Emfile();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enotty();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Etxtbsy();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Efbig();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enospc();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Espipe();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Erofs();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Emlink();
#pragma linkobj "src/SysErr/errorcode.o"
```

```
#pragma linksign []

external syserr Epipe();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eloop();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enametoolong();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enetdown();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enetunreach();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enetreset();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Econnaborted();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Econnreset();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enobufs();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eisconn();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enotconn();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eshutdown();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Etoomanyrefs();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Etimedout();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Econnrefused();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []
```

```
external syserr Ehostdown();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Ehostunreach();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Enotempty();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eusers();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Edquot();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Estale();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []

external syserr Eremote();
#pragma linkobj "src/SysErr/errorcode.o"
#pragma linksign []


/*-----------------------------------------------*/
```

6.13 System.sac

```
/*-----------------------------------------------*/

module System;

import World : all;
import Terminal : all;
import FileSystem : all;
import CommandLine : all;
import Environment : all;
import Clock : all;
import SysErr : all;
import RuntimeError : all;
export all;

/*-----------------------------------------------*/
```

6.14 Terminal.sac

```
/*-----------------------------------------------*/

class Terminal;

export all except { create_TheTerminal};


/*
 *  type definitions
 */
```

```
external classtype Terminal::Terminal;


/*
 *  prototypes for externals (FUNDECS)
 */

external Terminal create_TheTerminal();
#pragma linkobj "src/Terminal/terminal.o"
#pragma linksign []
#pragma effect World::TheWorld



/*
 *  global objects
 */

Terminal TheTerminal = create_TheTerminal() ;


/*-----------------------------------------------*/
```

6.15 TimeStamp.sac

```
/*-----------------------------------------------*/

module TimeStamp;

export all;


/*
 *  prototypes for externals (FUNDECS)
 */

external void  timeStamp();
#pragma linkobj "src/TimeStamp/TimeStamp.o"
#pragma effect World::TheWorld


/*-----------------------------------------------*/
```

6.16 World.sac

```
/*-----------------------------------------------*/

class World;

export all except { create_TheWorld};


/*
 *  type definitions
 */

external classtype World::World;


/*
```

```
 *  prototypes for externals (FUNDECS)
 */

external World create_TheWorld();
#pragma linkobj "src/World/World.o"
#pragma linksign []



/*
 *  global objects
 */

World TheWorld = create_TheWorld() ;


/*-----------------------------------------------*/
```

7 utrace

7.1 Indent.sac

```
/*-----------------------------------------------*/

class Indent;

export all;
use StdIO : all;
use Structures : all;


/*
 *  type definitions
 */

classtype int Indent::Indent;


/*
 *  global objects
 */

Indent indent = to_Indent( 0) ;



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * newIndent(...) [ body ]
 ******************************************************************************/
Indent newIndent( int value)



/******************************************************************************
 * getIndent(...) [ body ]
```

```
 ******************************************************************************/
int getIndent()



/******************************************************************************
 * getIndent(...) [ body ]
 ******************************************************************************/
int getIndent( Indent &i)



/******************************************************************************
 * setIndent(...) [ body ]
 ******************************************************************************/
void  setIndent( Indent &i, int val)



/******************************************************************************
 * doIndent(...) [ body ]
 ******************************************************************************/
void  doIndent( string pattern)



/******************************************************************************
 * doIndent(...) [ body ]
 ******************************************************************************/
void  doIndent( Indent &i, string pattern)



/******************************************************************************
 * incIndent(...) [ body ]
 ******************************************************************************/
void  incIndent( Indent &i)



/******************************************************************************
 * incIndent(...) [ body ]
 ******************************************************************************/
void  incIndent( Indent &i, int offset)



/******************************************************************************
 * incIndent(...) [ body ]
 ******************************************************************************/
void  incIndent()
```

```
/******************************************************************************
 * decIndent(...) [ body ]
 ******************************************************************************/
void  decIndent( Indent &i)



/******************************************************************************
 * decIndent(...) [ body ]
 ******************************************************************************/
void  decIndent( Indent &i, int offset)



/******************************************************************************
 * decIndent(...) [ body ]
 ******************************************************************************/
void  decIndent()


/*-----------------------------------------------*/
```

7.2 UTrace.sac

```
/*-----------------------------------------------*/

module UTrace;

export all except { PrintSeparator, PrintHeader, Indent};
use StdIO : all;
use Structures : all;
use ArrayFormat : all;
use Indent : all;


/*
 *  global objects
 */

Indent offset = newIndent( 0) ;



/*
 *  function definitions (FUNDEFS)
 */



/******************************************************************************
 * PrintSeparator(...) [ body ]
 ******************************************************************************/
void  PrintSeparator( string pattern)



/******************************************************************************
 * PrintHeader(...) [ body ]
```

```
 ******************************************************************************/
void  PrintHeader( string modName, int line)



/******************************************************************************
 * PrintFunEnter(...) [ body ]
 ******************************************************************************/
void  PrintFunEnter( string modName, int line, string funName)



/******************************************************************************
 * PrintFunLeave(...) [ body ]
 ******************************************************************************/
void  PrintFunLeave( string modName, int line, string funName)



/******************************************************************************
 * PrintArgsDone(...) [ body ]
 ******************************************************************************/
void  PrintArgsDone( string modName, int line, string funName)



/******************************************************************************
 * indentedShow(...) [ body ]
 ******************************************************************************/
void  indentedShow( int[+] arr)



/******************************************************************************
 * indentedShow(...) [ body ]
 ******************************************************************************/
void  indentedShow( int[.,.] arr)



/******************************************************************************
 * indentedShow(...) [ body ]
 ******************************************************************************/
void  indentedShow( int[.] arr)



/******************************************************************************
 * indentedShow(...) [ body ]
 ******************************************************************************/
void  indentedShow( int arr)
```

```
/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( float[+] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( float[.,.] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( float[.] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( float arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( double[+] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( double[.,.] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( double[.] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( double arr)
```

```
/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( bool[+] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( bool[.,.] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( bool[.] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( bool arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( char[+] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( char[.,.] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( char[.] arr)



/*****************************************************************************
 * indentedShow(...) [ body ]
 *****************************************************************************/
void  indentedShow( char arr)
```

```
/**************************************************************************
 * PrintArg(...) [ body ]
 **************************************************************************/
void  PrintArg( string modName, int line, string var, int[*] x)


/**************************************************************************
 * PrintArg(...) [ body ]
 **************************************************************************/
void  PrintArg( string modName, int line, string var, float[*] x)


/**************************************************************************
 * PrintArg(...) [ body ]
 **************************************************************************/
void  PrintArg( string modName, int line, string var, double[*] x)


/**************************************************************************
 * PrintArg(...) [ body ]
 **************************************************************************/
void  PrintArg( string modName, int line, string var, bool[*] x)


/**************************************************************************
 * PrintArg(...) [ body ]
 **************************************************************************/
void  PrintArg( string modName, int line, string var, char[*] x)


/**************************************************************************
 * PrintAssign(...) [ body ]
 **************************************************************************/
void  PrintAssign( string modName, int line, string var, int[*] x)


/**************************************************************************
 * PrintAssign(...) [ body ]
 **************************************************************************/
void  PrintAssign( string modName, int line, string var, float[*] x)


/**************************************************************************
 * PrintAssign(...) [ body ]
```

```
 ********************************************************************************/
void  PrintAssign( string modName, int line, string var, double[*] x)



/********************************************************************************
 * PrintAssign(...) [ body ]
 ********************************************************************************/
void  PrintAssign( string modName, int line, string var, bool[*] x)



/********************************************************************************
 * PrintAssign(...) [ body ]
 ********************************************************************************/
void  PrintAssign( string modName, int line, string var, char[*] x)



/********************************************************************************
 * PrintReturn(...) [ body ]
 ********************************************************************************/
void  PrintReturn( string modName, int line, int[*] x)



/********************************************************************************
 * PrintReturn(...) [ body ]
 ********************************************************************************/
void  PrintReturn( string modName, int line, float[*] x)



/********************************************************************************
 * PrintReturn(...) [ body ]
 ********************************************************************************/
void  PrintReturn( string modName, int line, double[*] x)



/********************************************************************************
 * PrintReturn(...) [ body ]
 ********************************************************************************/
void  PrintReturn( string modName, int line, bool[*] x)



/********************************************************************************
 * PrintReturn(...) [ body ]
 ********************************************************************************/
void  PrintReturn( string modName, int line, char[*] x)



/*-----------------------------------------------*/
```