Expr

Expr — Parent class containing all expression objects available in msdscript

Methods

bool equals()

PTR(Val) interp()

void print()

std::string to_string()

void pretty_print()

void step_interp()

NumExpr

NumExpr – subclass of Expr, integer expression object

AddExpr

AddExpr – subclass of Expr, holds two expression objects being added together

MultExpr

MultExpr – subclass of Expr, holds two expression objects being multiplied together

VarExpr

VarExpr – subclass of Expr, variable expression object

LetExpr

LetExpr – subclass of Expr, enables the use of a defined variable in an expression object

EqExpr

EqExpr – subclass of Expr, equation expression object, used for comparing the equality of two expression objects

BoolExpr

BoolExpr – subclass of Expr, Boolean expression object, holds BoolVal object

IfExpr

IfExpr – subclass of Expr, if expression object used for if, then, else logic.

FunExpr

FunExpr – subclass of Expr, function expression object, contains an “unbound” variable, and a function expression. An unbound variable is a variable that has not been set to a definite value.

CallExpr

CallExpr – subclass of Expr, represents a function call object. Contains a function expression and an argument expression.

Includes

#include "expr.hpp"

Val

Val — Parent class containing all value objects available in msdscript

Methods

bool equals()

PTR(Expr) to_expr()

PTR(Val) add_to()

PTR(Val) mult_to()

PTR(Val) call()

std::string to_string()

void call_step()

NumVal

NumVal – subclass of Val, object representing integer values. A NumVal can be added or multiplied. A negative sign will make a NumVal a negative integer value. There is no subtraction in msdscript, to do so, a negative NumVal must be added.

BoolVal

BoolVal – subclass of Val, Boolean value object. Can be true or false.

FunVal

FunVal – subclass of Val, identical to FunExpr expressions except with an additional environment argument used when interpreting function calls.

Includes

#include "val.hpp"

Env

Env — Parent class containing all environment objects in msdscript. An environment represents a set of substitutions to perform. An environment can either be empty (EmptyEnv), or extended (ExtendedEnv).

Methods

PTR(Val) lookup()

EmptyEnv

EmptyEnv – subclass of Env, an empty environment object, meaning there are no substitutions to perform.

ExtendedEnv

ExtendedEnv – subclass of Env, an extended environment object, meaning there are a stack of substitutions to perform.

Includes

#include "env.hpp"

Step

Step — A class containing static variables and a struct to store information needed for continuations.

Member Variables

typedef enum { interp_mode, continue_mode } mode_t

static mode_t mode

static PTR(Expr) expr

static PTR(Env) env

static PTR(Val) val

static PTR(Cont) cont

Methods

static PTR(Val) interp_by_steps()

Includes

#include "step.hpp"

Cont

Cont — Parent class containing all continuation objects in msdscript. Continuation objects remember data needed for continuation steps.

Member Variables

static PTR(Cont) done

Methods

void step_continue()

DoneCont

DoneCont – subclass of Cont, a done continuation object

RightThenAddCont

RightThenAddCont – subclass of Cont,