1. System Manager File: “System Manager 1.c++”

1.1. General Information

1.1.1. File Role

Contains main() function for the system. Implements version 1 of system.

1.1.2. Standard Headers Required

SIOUX.h

1.1.3. Custom Headers Required

rCalculatorClass.h, calc_preprocessor.h, validator.h, CalculatorManager.h, newstring.h, complex.h, ulist.h, stringobject.h, name_object.h, IO_map_object.h, IO_map.h, set_input.h, set_input_object.h, Data Manager.h, extraclasses.h, graph_device.h, graph_spec.h, graph_spec_obj.h, Graph Manager.h, ulist.c++ (note this code file must be #included because it contains a template definition)

1.2. C-Type Definitions

1.2.1. Constants

const int preprocessor_array_length: length of the array ‘labels’ array
const int number_of_constants: number of constants to be preloaded into calculator class

1.2.2. Arrays

const user_label labels[preprocessor_array_length]: holds mappings from user name functions to OPERATION names
name_object calc_constants[number_of_constants] : holds calculator class global constants

1.2.3. Ulists:

ulist<IO_map_object> _t1; ulist<set_input_object> _t2; ulist<data_set_obj> _t3; ulist<record_object> _t4; ulist<string_object> _t5; ulist<name_object> _t6; ulist<graph_spec_obj> _t7 : these ulists are required to instantiate each version of templated ulist class

1.2.4. String_class globals: (these and respective double variables used only in this file)

string_class x_min_string : holds variable name looked up in ‘graph’ calculator for min x range val
string_class x_max_string : holds variable name looked up in ‘graph’ calculator for max x range val
string_class x_scale_string : holds variable name looked up in ‘graph’ calculator for x major scale
string_class x_div_string : holds variable name looked up in ‘graph’ calculator for x minticks
string_class y_min_string : holds variable name looked up in ‘graph’ calculator for min y range val
string_class y_max_string : holds variable name looked up in ‘graph’ calculator for max y range val
string_class y_scale_string : holds variable name looked up in ‘graph’ calculator for y major scale
string_class y_div_string : holds variable name looked up in ‘graph’ calculator for y minticks
string_class x_string : holds variable name looked up in ‘graph’ calculator for x graphing value
string_class y_string : holds variable name looked up in ‘graph’ calculator for y graphing value
string_class varmin : holds variable name looked up in ‘graph’ calculator for min sweep value
string_class varmax : holds variable name looked up in ‘graph’ calculator for max sweep value
string_class sample_res_string: holds var name in ‘graph’ calculator for graphing sample period

1.2.5. Global storage for results gained from using string_class globals to evaluate calculator contents:

double x_min, x_max, x_scale, x_tick, y_min, y_max, y_scale, y_tick, var_min, var_max, sample_res

1.3. Non-Class Function Prototypes

1.3.1. void main(void)

1.3.1.1. Role

1.3.1.1.1. Operator Type

Main function

1.3.1.1.2. Description

Entry point for program execution.Initialises terminal window. Initialises all relevant classes, instantiates a preprocessor object, calculator manager, data manager, graph manager and graph device. Draws default axes on graph. Allows singular access to data manager and graph manager intrefaces, and repeated access to calculator manager interface. Upon each exit from calculator manager interface, system attempts to draw graph using current calculator in calculator manager for evaluation and ‘graph’ calculator to get graph control variables.

1.3.1.2. Pre-conditions

None

1.3.1.3. Post-conditions

Program terminated

1.3.2. status extract_graph_specs(calculator_manager &calc_manager, graph_device &graph_)

1.3.2.1. Role

1.3.2.1.1. Operator Type

Extractor

1.3.2.1.2. Description

Extracts all graph control variables from ‘graph’ calculator in calc_manager. Variable names are stored in string_class globals - see above. For each variable: sends order to ‘graph’ calculator to return number stored with that variable name. Store results in respective global double variable - see above.

1.3.2.2. Pre-conditions

Calculator list stored in calculator manager contains calculator called ‘graph’ and this calculator contains ALL graph control variables in its variable list.

1.3.2.3. Post-conditions

Global doubles contain graph control variable values - graph_device graph_ loaded with graph control variable values.

1.3.2.4. Return Data

RETURN SUCCESS - if preconditions met - else RETURN ERROR

1.3.3. void display_graph(calculator_manager &calc_manager, graph_device &graph_);

1.3.3.1. Role

1.3.3.1.1. Operator Type

Operating System Modifier - draw to screen

1.3.3.1.2. Description

Assign ‘var’ variable in current calculator in calculator manager to values ranged from ‘varmin’ to ‘varmax’, incrementing in steps of ‘sample_res’. After each assignment, x_graph equation and y_graph equation stored in calculator are evaluated. These results are used as x,y co-ordinates for plotting on the graph. Each point is joined by a straight line. Uses graph_device to place points correctly in graphing window.

1.3.3.2. Pre-conditions

All graph controls must have been successfully extracted from ‘graph’ calculator by extract_graph_specs function. Equations in calculator being plotted must contain ‘var’ variable - equations must not generate errors - x_graph and y_graph equations must be present in current calculator for display_graph to extract equation results.

1.3.3.3. Post-conditions

See Description

1.3.4. void Initialize(void)

1.3.4.1. Role

1.3.4.1.1. Operator Type

Operating System Modifier

1.3.4.1.2. Description

Initialise all operating system managers - without which no window/graphing operations can take place.

1.3.4.2. Pre-conditions

None

1.3.4.3. Post-conditions

Mac OS ready to accept operating system function calls.

1.4. Class Definitions

None

2. System Manager File: “System Manager 2.c++”

2.1. General Information

2.1.1. File Role

Contains main() function for the system. Implements version 2 of system.

2.1.2. Standard Headers Required

Same as <System Manager 1.c++>

2.1.3. Custom Headers Required

Same as <System Manager 1.c++>, plus system_process.h

2.2. C-Type Definitions

2.2.1. Constants

Same as <System Manager 1.c++>

2.2.2. Arrays

Same as <System Manager 1.c++>

2.2.3. Ulists:

Same as <System Manager 1.c++>

2.2.4. String_class globals:

None

2.3. Non-Class Function Prototypes

2.3.1. void main(void)

2.3.1.1. Role

2.3.1.1.1. Operator Type

Main function

2.3.1.1.2. Description

Entry point for program execution.Initialises terminal window. Initialises all relevant classes, instantiates a preprocessor object, calculator manager, data manager, graph manager and graph device. Draws default axes on graph. Displays System Manager commands on screen. Calls system manager interface.

2.3.1.2. Pre-conditions

None

2.3.1.3. Post-conditions

Program terminated

2.3.2. void interface(calculator_manager &calc_man, data_manager &data_man,graph_manager &graph_man, system_process &process, graph_device &graph_);

2.3.2.1. Role

2.3.2.1.1. Operator Type

CLI interface

2.3.2.1.2. Description

Provides interface to System Manager and all sub-managers: calculator manager, data manager and graph manager.

2.3.2.2. Pre-conditions

None

2.3.2.3. Post-conditions

Dependent upon commands issued to system manager.

2.3.3. void display_help();

2.3.3.1. Role

2.3.3.1.1. Operator Type

Information Provider

2.3.3.1.2. Description

Outputs all system manager commands to standard output

2.3.3.2. Pre-conditions

None

2.3.3.3. Post-conditions

See Description

2.3.4. void process_help();

2.3.4.1. Role

2.3.4.1.1. Operator Type

Information Provider

2.3.4.1.2. Description

Outputs commands available when setting up process object from within system manager

2.3.4.2. Pre-conditions

None

2.3.4.3. Post-conditions

See Description

2.3.5. status set_process_object(system_process &process);

2.3.5.1. Role

2.3.5.1.1. Operator Type

Modifier

2.3.5.1.2. Description

Allow user to set process parameters.

2.3.5.2. Pre-conditions

None

2.3.5.3. Post-conditions

Process object updated according to user input

2.3.5.4. Return Data

RETURN SUCCESS if command and parameters processed ok.
RETURN ERROR if invalid command issued by user (command NOT d,g,s,m,c)

2.3.6. status calculate_process(calculator_manager &calc_man, data_manager &data_man, system_process &process);

2.3.6.1. Role

2.3.6.1.1. Operator Type

Primary Service

2.3.6.1.2. Description

Carries out process instructions contained in process object. Co-ordinates input data being retrieved from data_set in data_manager data_list, provides this to the relevant calculator in calc_list of calculator manager for evaluation, retrieves evaluation from calculator and stores results in data_set.

2.3.6.2. Pre-conditions

Process contains valid references to objects in data manager and calculator manager lists - references must also be compatible with each other.

2.3.6.3. Post-conditions

Calculator referenced in process.calc_name used for all evaluation: all set_input objects referenced by process.set_input_names applied to fields (referenced by process.set_input_fields) in data_set (referenced by process.data_set_name). All IO_map objects referenced by process.map_names used to select fields for inputting data to calculator - and for extracting evaulated data from calculator, storing in output fields (IO_map.output_fields) of data_set.

2.3.6.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

2.3.7. status display_rect_graph(data_manager &data_man, graph_manager &graph_man, system_process process, string_class x_axis, string_class y_axis, int start, int finish, graph_device &graph_);

2.3.7.1. Role

2.3.7.1.1. Operator Type

Operating System Modifier - draw to screen

2.3.7.1.2. Description

Plot x,y graph using graph_device. Graph axes and scales set up according to graph_spec in graph_manager referenced by process.graph_spec_name. X data retrieved from data_set referenced by process.data_set_name (field referenced by ‘x_axis’). Y data retrieved from data_set referenced by process.data_set_name (field referenced by ‘y_axis’). Only data lying within the index limits start to finish (inclusive) is plotted.

2.3.7.2. Pre-conditions

Process contains valid references to objects in data, calculator and graph managers’ lists - references must also be compatible with each other. Index limits are within range of data_set referenced by process.data_set_name.

2.3.7.3. Post-conditions

See Description

2.3.7.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

2.3.8. status display_cornu_graph(data_manager &data_man, graph_manager &graph_man, system_process process, string_class vector, int start, int finish, graph_device &graph_);

2.3.8.1. Role

2.3.8.1.1. Operator Type

Operating System Modifier - draw to screen

2.3.8.1.2. Description

Plot polar vector graph using graph_device. Data is retrieved from data_set referenced by process.data_set_name (field referenced by 'vector'). Only data lying within the index limits start to finish (inclusive) is plotted. Graph spec referenced by process.graph_spec_name used to set horizontal/vertical scaling and offsets for plot. horiz/vert.MajScale holds scaling (determines number of pixels per 1 unit of vector component); horiz/vert.MinTicks holds offset (determines x,y start origin relative to bottom left of graph area - measured in 1 unit vector components).

2.3.8.2. Pre-conditions

Process contains valid references to objects in data, calculator and graph managers' lists - references must also be compatible with each other. Index limits are within range of data_set referenced by process.data_set_name.

2.3.8.3. Post-conditions

See Description

2.3.8.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

2.3.9. void Initialize(void)

Same as for <System Manager 1.c++>

2.4. Class Definitions

2.4.1. None

3. Header File: “system_process.h”

3.1. General Information

3.1.1. Header File Role

Declares system_process class

3.1.2. Standard Headers Required

None

3.1.3. Custom Headers Required

ulist.h, stringobject.h, newstring.h

3.2. C-Type Definitions

None

3.3. Non-Class Function Prototypes

None

3.4. Class Definitions

3.4.1. Class “system_process”

3.4.1.1. Role

Holds all information required for evaluating and graphing a set of calculations - used exclusively by System Manager.

3.4.1.2. Class Initialisation:

None

3.4.1.3. Private Data Members

None

3.4.1.4. Public Data Members

3.4.1.4.1. string_class calc_name

Holds the name of the calculator (which must be present in the calculator list of calculator manager) with which all evaluations are carried out.

3.4.1.4.2. string_class data_set_name

Holds the name of the data_set (which must be present in the data list of data manager) used to store all input and output data, sent and received to/from the named calculator.

3.4.1.4.3. string_class graph_spec_name

Holds the name of the graph_spec (which must be present in the graph_spec list of data manager) used to set up axes for plotting a graph of data inside data_set array.

3.4.1.4.4. ulist<string_object> set_input_names

Holds a list of names (each stored in DATA field of string_object) of set_input objects (which must all be present in the set_input_list of data manager) used to initialise the named data_set prior to evaluation by the calculator. Set_inputs are applied to the data_set in the order they appear in the list.

3.4.1.4.5. ulist<string_object> set_input_fields

Holds a list of field names corresponding to data_set fields.The columns referenced by these fields are loaded with their respective set_input referenced in the set_input_names list. The set_input name at the head of the set_input_names list is applied to the column of data_set with field name at the head of the set_input_fields list.

3.4.1.4.6. ulist<string_object> map_names

Holds a list of names of IO_map objects (which must all be present in the map_list of data manager) used to tell the system manager which data_set fields are used as calculator inputs, and which are used to store calculator outputs. IO_maps are applied to the data_set in the order they are found in the map_names list.

3.4.1.5. Static Data Members

None

3.4.1.6. Private Member Functions

3.4.1.6. Private Member Functions

None

3.4.1.7. Public Member Functions

3.4.1.7.1. void load_list(istream &input_stream, int list)

3.4.1.7.1.1. Role

3.4.1.7.1.1.1. Operator Type

Modifier

3.4.1.7.1.1.2. Description

Reads names separated by spaces from input_stream, and stores them in the list indicated by int list. List ordering is dictated by the order names are read from the stream - note that no alphabetic ordering on name is performed.

3.4.1.7.1.2. Pre-conditions

input_stream contains data.
list is 0,1 or 2

3.4.1.7.1.3. Post-conditions

IF list==0 then set_input_names list loaded with names from input_stream
IF list==1 then set_input_fields list loaded with names from input_stream
IF list==2 then map_names list loaded with names from input_stream

3.4.1.8. Friend Member Functions

3.4.1.8.1. ostream& operator<<(ostream& output_stream, system_process& output_map)

3.4.1.8.1.1. Role

3.4.1.8.1.1.1. Operator Type

Overloaded Output Operator

3.4.1.8.1.1.2. Description

Output all data members with labels to output_stream

3.4.1.8.1.2. Pre-conditions

None

3.4.1.8.1.3. Post-conditions

See Description

3.4.1.8.1.4. Return Data

return output_stream

3.4.1.9. Static Member Functions

None

4. Header File: “CalculatorManager.h”

4.1. General Information

4.1.1. Header File Role

To declare calculator class

4.1.2. Standard Headers Required

iostream.h

4.1.3. Custom Headers Required

rCalculatorClass.h, calc_preprocessor.h, validator.h, calcobject.h, newstring.h

4.2. C-Type Definitions

None

4.3. Non-Class Function Prototypes

None

4.4. Class Definitions

4.4.1. Class “calculator_manager”

4.4.1.1. Role:

To maintain and allow access to a list of calc_objects, each consisting of a single calculator object and a unique identifying name. A CLI is defined for allowing user interaction with the list and its contents. The user is permitted to perform the following operations: adding and removing calculators from the list (searched by name), giving a particular calculator an order (ie to evaluate an expression, record an assignment in its variable list or define an equation in its equation_list), clearing memories in a particular calculator, setting auto-verify setting of individual calculators, viewing/clearing errors in individual calculators and viewing variable/equation lists stored in individual or all calculators in list.

4.4.1.2. Class Initialisation:

None

4.4.1.3. Public Data Members

4.4.1.3.1. ulist<calc_object> calc_list

Stores the list of calc_object nodes.

4.4.1.3.2. calc_object *current_calc

Points to the calculator node in the calc_list to which commands issued from the CLI are directed.

4.4.1.3.3. calc_preprocessor *preprocessor

References a preprocessor object required for validating equations.

4.4.1.3.4. validator *equation_checker

References a validator object which is used by all calculator objects for auto-verification of equations, and by calculator_manager for manual verification when auto-verify is turned off for a particular calculator.

4.4.1.4. Public Data Members

None

4.4.1.5. Static Data Members

None

4.4.1.6. Private Member Functions

None

4.4.1.7. Public Member Functions

4.4.1.7.1. calculator_manager(calc_preprocessor *preprocess)

4.4.1.7.1.1. Role

4.4.1.7.1.1.1. Operator Type

Foundation

4.4.1.7.1.1.2. Description

Parameterized Constructor

4.4.1.7.1.2. Pre-conditions

None

4.4.1.7.1.3. Post-conditions

4.4.1.7.1.3. Post-conditions

Preprocessor pointer initialised to ‘preprocess’.
Equation checker points to an instantiated validator object using ‘preprocessor’ to aid in validating.
current_calc set to no calculator, NULL

4.4.1.7.2. ~calculator_manager()

4.4.1.7.2.1. Role

4.4.1.7.2.1.1. Operator Type

Foundation

4.4.1.7.2.1.2. Description

Destructor

4.4.1.7.2.2. Pre-conditions

None

4.4.1.7.2.3. Post-conditions

equation_checker memory deallocated

4.4.1.7.3. calculator_manager(calculator_manager &original)

4.4.1.7.3.1. Role

4.4.1.7.3.1.1. Operator Type

Foundation

4.4.1.7.3.1.2. Description

Copy Constructor

4.4.1.7.3.2. Pre-conditions

None

4.4.1.7.3.3. Post-conditions

See overloaded = operator

4.4.1.7.4. calculator_manager& operator=(calculator_manager &source)

4.4.1.7.4.1. Role

4.4.1.7.4.1.1. Operator Type

Overloaded assignment operator

4.4.1.7.4.1.2. Description

Allows assignment between calculator manager objects

4.4.1.7.4.2. Pre-conditions

Object not to be assigned to itself

4.4.1.7.4.3. Post-conditions

All member data copied from’source’ to *this

4.4.1.7.4.4. Return Data

RETURN *this

4.4.1.7.5. status add_calculator(const string_class name)

4.4.1.7.5.1. Role

4.4.1.7.5.1.1. Operator Type

Modifier

4.4.1.7.5.1.2. Description

Adds a single calc_object to calc_list, with identifier ‘name’ - if name is unique to list.

4.4.1.7.5.2. Pre-conditions

Name is unique to this calc_list

4.4.1.7.5.3. Post-conditions

calc_list holds additional calc_object using ASCENDING lexical name ordering if name unique.
Otherwise, output error message.

4.4.1.7.5.4. Return Data

RETURN SUCCESS if preconditions satisfied, else RETURN ERROR

4.4.1.7.6. status remove_calculator(const string_class name)

4.4.1.7.6.1. Role

4.4.1.7.6.1.1. Operator Type

Modifier

4.4.1.7.6.1.2. Description

Remove a single calc_object from calc_list with identifier ‘name’.

4.4.1.7.6.2. Pre-conditions

Such a calc_object exists in calc_list.

4.4.1.7.6.3. Post-conditions

If pre-conditions met, named calc_object removed from calc_list.

4.4.1.7.6.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR.

4.4.1.7.7. complex process_order(const string_class &calc_order)

4.4.1.7.7.1. Role

4.4.1.7.7.1.1. Operator Type

Primary Service

4.4.1.7.7.1.2. Description

Send a string consisting of either an expression for evaluation, an assignment to a variable, or a definition of an equation to the calculator object stored in the calc_object pointed to by current_calc.

4.4.1.7.7.2. Pre-conditions

calc_list is not empty

4.4.1.7.7.3. Post-conditions

Any errors occurred will be stored in error report inside current_calc calculator object.
For an evalulation: no further post-conditions
For an assignment: depending upon the validity of the assignment - may have new variable stored in calculator variable_list
For a definition: depending upon the validity of the definition - may have new equation stored in calculator equation_list

4.4.1.7.7.4. Return Data

IF pre-conditions not met, RETURN NULLcomplex.
ELSE return complex number result of evaluation/assignment/definition.

4.4.1.7.8. status set_current_calc(const string_class name_string="")

4.4.1.7.8.1. Role

4.4.1.7.8.1.1. Operator Type

Modifier

4.4.1.7.8.1.2. Description

Sets current_calc to point to the calc_object with name specified by name_string. Specifying no name_string means point to the first calc_object in the calc_list.

4.4.1.7.8.2. Pre-conditions

calc_object with name name_string exists in calc_list.

4.4.1.7.8.3. Post-conditions

See description.

4.4.1.7.8.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

4.4.1.7.9. ostream& current_storage(ostream &output_stream)

4.4.1.7.9.1. Role

4.4.1.7.9.1.1. Operator Type

Extractor

4.4.1.7.9.1.2. Description

Output variable and equation lists contained for calculator object inside calc_object pointed to by current_calc.

4.4.1.7.9.2. Pre-conditions

calc_list is not empty

4.4.1.7.9.3. Post-conditions

See Description.

4.4.1.7.9.4. Return Data

RETURN output_stream

4.4.1.7.10. ostream& current_errors(ostream &output_stream)

4.4.1.7.10.1. Role

4.4.1.7.10.1.1. Operator Type

Extractor

4.4.1.7.10.1.2. Description

Output error report for calculator object inside calc_object pointed to by current_calc.

4.4.1.7.10.2. Pre-conditions

calc_list is not empty

4.4.1.7.10.3. Post-conditions

See Description.

4.4.1.7.10.4. Return Data
RETURN output_stream

4.4.1.7.11. string_class current_clear_errors()
4.4.1.7.11.1. Role
4.4.1.7.11.1.1. Operator Type
Modifier
4.4.1.7.11.1.2. Description
Extract and clear error_report for calculator inside calc_object pointed to by current_calc.
4.4.1.7.11.2. Pre-conditions
calc_list is not empty
4.4.1.7.11.3. Post-conditions
See Description.
4.4.1.7.11.4. Return Data
RETURN extracted error report, if pre-condition met, else RETURN null string

4.4.1.7.12. status validate_current()
4.4.1.7.12.1. Role
4.4.1.7.12.1.1. Operator Type
Decider
4.4.1.7.12.1.2. Description
Calls the validator object to verify all equations in the equation list for calculator inside calc_object pointed to by current_calc.
4.4.1.7.12.2. Pre-conditions
calc_list is not empty
4.4.1.7.12.3. Post-conditions
Output message reflecting state of equations.
4.4.1.7.12.4. Return Data
RETURN SUCCESS if pre-conditions met and validation returns SUCCESS.
Else RETURN ERROR

4.4.1.7.13. string_class clear_memory()
4.4.1.7.13.1. Role
4.4.1.7.13.1.1. Operator Type
Modifier
4.4.1.7.13.1.2. Description
Clear all variables and equations stored in calculator within calc_object pointed to by current_calc.
4.4.1.7.13.2. Pre-conditions
calc_list is not empty.
4.4.1.7.13.3. Post-conditions
See Description.
4.4.1.7.13.4. Return Data
RETURN error message if pre-conditions not met, else return “”

4.4.1.7.14. string_class clear_single_memory(string_class name)
4.4.1.7.14.1. Role
4.4.1.7.14.1.1. Operator Type
Modifier
4.4.1.7.14.1.2. Description
Clear variable or equation identified by name in calculator within calc_object pointed to by current_calc.
4.4.1.7.14.2. Pre-conditions
calc_list is not empty
4.4.1.7.14.3. Post-conditions
See Description.
4.4.1.7.14.4. Return Data
RETURN error message if pre-conditions not met, else return “Cleared”

4.4.1.7.15. void interface()

4.4.1.7.15.1. Role

4.4.1.7.15.1.1. Operator Type

CLI interface

4.4.1.7.15.1.2. Description

Allows user to interact with the calc_object list as described in the class role description, through the use of a CLI. In general, each user command corresponds accessing an equivalent public member function in this class, which in turn calls the corresponding public member function in current calcultator.

4.4.1.7.15.2. Pre-conditions

None

4.4.1.7.15.3. Post-conditions

calc_list updated according to users commands.

4.4.1.7.16. void display_help()

4.4.1.7.16.1. Role

4.4.1.7.16.1.1. Operator Type

Information provider

4.4.1.7.16.1.2. Description

Output list of user commands for this manager to standard output.

4.4.1.7.16.2. Pre-conditions

None

4.4.1.7.16.3. Post-conditions

See description

4.4.1.7.17. string_class get_current_calc()

4.4.1.7.17.1. Role

4.4.1.7.17.1.1. Operator Type

Extractor

4.4.1.7.17.1.2. Description

extract name of calc_object pointed to by current_calc

4.4.1.7.17.2. Pre-conditions

calc_list not empty

4.4.1.7.17.3. Post-conditions

None

4.4.1.7.17.4. Return Data

Return calc_object name if pre-conditions met, else return “”

4.4.1.7.18. void auto_verify_on()

4.4.1.7.18.1. Role

4.4.1.7.18.1.1. Operator Type

Modifier

4.4.1.7.18.1.2. Description

Set auto_verify for calculator object within calc_object pointed to by current_calc to on.

4.4.1.7.18.2. Pre-conditions

calc_list is not empty, calculator object equations validate okay.

4.4.1.7.18.3. Post-conditions

See description, where pre-conditions met, else output error message.

4.4.1.7.19. void auto_verify_off()

4.4.1.7.19.1. Role

4.4.1.7.19.1.1. Operator Type

Modifier

4.4.1.7.19.1.2. Description

Set auto_verify for calculator object within calc_object pointed to by current_calc to off.

4.4.1.7.19.2. Pre-conditions

calc_list is not empty.

4.4.1.7.19.3. Post-conditions

See description

4.4.1.7.20. void reset_manager()

4.4.1.7.20.1. Role

4.4.1.7.20.1.1. Operator Type

Modifier

4.4.1.7.20.1.2. Description

Clears calculator list, sets current_calc to NULL.

4.4.1.7.20.2. Pre-conditions

None

4.4.1.7.20.3. Post-conditions

See Description

4.4.1.8. Friend Member Functions

4.4.1.8.1. ostream& operator<<(ostream& output_stream, calculator_manager &output)

4.4.1.8.1.1. Role

4.4.1.8.1.1.1. Operator Type

Overloaded output operator

4.4.1.8.1.1.2. Description

Output names and calculator variable/equation lists for each calc_object in calc_list.

4.4.1.8.1.2. Pre-conditions

None

4.4.1.8.1.3. Post-conditions

If pre-conditions not met, output error message, otherwise see description.

4.4.1.8.1.4. Return Data

return output_stream

4.4.1.9. Static Member Functions

None

5. Header File: “rCalculatorUserTypes.h”

5.1. General Information

5.1.1. Header File Role

To declare enums token_value, macro_type, calculator_type. To declare classes token_name, calculator_symbol.

5.1.2. Standard Headers Required

string.h, iostream.h, strstream.h, ctype.h, math.h

5.1.3. Custom Headers Required

complex.h, newstring.h

5.2. C-Type Definitions

5.2.1. Enumerations

5.2.1.1. enum token_value

5.2.1.1.1. Enum values

{NAME, NUMBER, END, PLUS='+', MINUS='-', MUL='*', DIV='/', POW='^', FACTORIAL='!', SQRT='@', CBRT='£', ROOTX='$', SIN='¡', COS='™', TAN='#', ASIN='¢', ACOS='∞', ATAN='§', SINH='~',COSH='Ω',TANH='≈', ASINH='ç', ACOSH='√', ATANH='∫', LN='¶', LOG10='•', LOG2='ª', LOGX='º', PRINT=';', ASSIGN_CONSTANT='=', LP='(', RP=')', LM='[', RM=']', ARGUMENT='≠', MILLI='œ', MICRO='∑', NANO='®', PICO='†', FEMTO='¥', KILO='å', MEGA='ß', GIGA='∂', TERA='ƒ', PETA='©', EXA='Δ', DEFINE_EQUATION=':', SUMMATION='¬', PRODATION='∏', COMMA=',', REAL='≤', IMAGINARY='≥', WINDOW='÷’}

5.2.1.1.2. Role

Contains all tokens understood by calculator class. Each token name is associated with a single integer which is interpreted as an unsigned ASCII character, ranged over 0-255.

5.2.1.2. enum macro_type

5.2.1.2.1. Enum values

{ CONSTANT, EQUATION }

5.2.1.2.2. Role

Used to distinguish between the two types of data stored in complex_container class. 'CONSTANT' signifies a fixed complex_number is stored, whereas 'EQUATION' signifies a string_class object is stored.

5.2.1.3. enum calculator_type

5.2.1.3.1. Enum Values

{ SUPER_CALCULATOR, SUB_CALCULATOR }

5.2.1.3.2. Role

Used to determine between a calculator created by the user (SUPER_CALCULATOR - capable of assignment of variables/equations) and those created on-the-fly by a SUPER_CALCULATOR, called SUB_CALCULATORs, which are used to perform recursive calculations on behalf of the SUPER_CALCULATOR.

5.3. Non-Class Function Prototypes

none

5.4. Class Definitions

5.4.1. Class “token_name”

5.4.1.1. Role:

'token_name' is used to map an OPERATION string_class to a character token. OPERATIONs are common to both preprocessor and calculator. Character tokens are known only by the calculator and the preprocessor.

5.4.1.2. Class Initialisation:

None

5.4.1.3. Private Data Members

None

5.4.1.4. Public Data Members

5.4.1.4.1. string_class name;

Holds OPERATION string. Matches with identical OPERATION string in look-up table in preprocessor object.

5.4.1.4.2. char token

Holds character which calculator recognises on input as indicating particular OPERATION.

5.4.1.5. Static Data Members

None

5.4.1.6. Private Member Functions

None

5.4.1.7. Public Member Functions

5.4.1.7.1. token_name()

5.4.1.7.1.1. Role

5.4.1.7.1.1.1. Operator Type

Foundation

5.4.1.7.1.1.2. Description

Default Constructor

5.4.1.7.1.2. Pre-conditions

None

5.4.1.7.1.3. Post-conditions

token=0 (ie token=NAME)

5.4.1.7.2. token_name(const char *tokenname, const char token_character)

5.4.1.7.2.1. Role

5.4.1.7.2.1.1. Operator Type

Foundation

5.4.1.7.2.1.2. Description

Parameterized constructor

5.4.1.7.2.2. Pre-conditions

None

5.4.1.7.2.3. Post-conditions

name=tokenname

token=token_character

5.4.1.8. Friend Member Functions

5.4.1.8.1. ostream& operator<<(ostream& output_stream, const token_name output)

5.4.1.8.1.1. Role

5.4.1.8.1.1.1. Operator Type

Overloaded output operator

5.4.1.8.1.1.2. Description

Output to stream output.name followed by '->' followed by output.token

5.4.1.8.1.2. Pre-conditions

None

5.4.1.8.1.3. Post-conditions

Class info output to stream as above.

5.4.1.9. Static Member Functions

None

5.4.2. Class "calculator_symbol"

5.4.2.1. Role:

'calculator_symbol' is used to hold the current symbol read in by calculator class from the input stream. It holds a single token_value corresponding to the interpretation of the character token read in. It also holds either a complex number, or a string_class depending upon the particular token_value stored. Information is stored mutually exclusively between complex/string_class data members (cf C union).

5.4.2.2. Class Initialisation:

None

5.4.2.3. Private Data Members

None

5.4.2.4. Public Data Members

5.4.2.4.1. token_value token

Holds character token (enumerated)

5.4.2.4.2. complex number_value

If token requires complex number data, then this member is used to store that number

5.4.2.4.3. string_class name_string

If token requires string data, then this member used is store that string.

5.4.2.5. Static Data Members

None

5.4.2.6. Private Member Functions

None

5.4.2.7. Public Member Functions

5.4.2.7.1. void clear()

5.4.2.7.1.1. Role

5.4.2.7.1.1.1. Operator Type

Modifier

5.4.2.7.1.1.2. Description

Initialise all data members

5.4.2.7.1.2. Pre-conditions

None

5.4.2.7.1.3. Post-conditions

token=NAME;
name_string="";
number_value=complex (0,0);

5.4.2.8. Friend Member Functions

None

5.4.2.9. Static Member Functions

None

6. Header File: “extraclasses.h”

6.1. General Information

6.1.1. Header File Role

Contains class definition for complex_container class

6.1.2. Standard Headers Required

None

6.1.3. Custom Headers Required

rCalculatorUserTypes.h, complex.h, define_vars.h, newstring.h

6.1.4. Forward Declarations Required

class calc_preprocessor - used as public static data member

6.2. C-Type Definitions

None

6.3. Non-Class Function Prototypes

none

6.4. Class Definitions

6.4.1. Class “complex_container”

6.4.1.1. Role:

Used as the DATA member in name_object - which form the base node for variable and equation lists in calculator class. Holds either a complex number (indicator=CONSTANT) or a string, (indicator=EQUATION).

6.4.1.2. Class Initialisation:

’calc_preprocessor *postprocessing’ must be set to point to an existing calc_preprocessor object before any complex_container objects are instantiated.

6.4.1.3. Private Data Members

None

6.4.1.4. Public Data Members

6.4.1.4.1. macro_type indicator

Signifies whether container holds a complex number (indicator=CONSTANT) used with variable lists, or holds a string (indicator=EQUATION) used with equation lists.

6.4.1.4.2. complex complex_number

Holds complex number when indicator=CONSTANT

6.4.1.4.3. string_class equation

Holds an equation string (where math functions are represented by calculator token characters) when indicator=EQUATION.

6.4.1.5. Static Data Members

6.4.1.5.1. calc_preprocessor *postprocessing

Used when outputting the contents of the string_class data member. Postprocess member function of calc_preprocessor expands tokens stored in equation to full user function names.

6.4.1.6. Private Member Functions

None

6.4.1.7. Public Member Functions

6.4.1.7.1. complex_container()

6.4.1.7.1.1. Role

6.4.1.7.1.1.1. Operator Type

Foundation

6.4.1.7.1.1.2. Description

Default Constructor

6.4.1.7.1.2. Pre-conditions

None

6.4.1.7.1.3. Post-conditions

SET indicator to CONSTANT (ie complex number is the relevant data member).

6.4.1.7.2. complex_container(macro_type store_indicator, complex store_complex_number)

6.4.1.7.2. complex_container(macro_type store_indicator, complex store_complex_number)

6.4.1.7.2.1. Role

6.4.1.7.2.1.1. Operator Type

Foundation

6.4.1.7.2.1.2. Description

Parameterized Constructor for complex number storage

6.4.1.7.2.2. Pre-conditions

None

6.4.1.7.2.3. Post-conditions

indicator=CONSTANT.
complex_number=store_complex_number.
If store_indicator is not CONSTANT, output error message to COUT - as string must be stored when indicator is EQUATION.

6.4.1.7.3. complex_container(string_class equation_string)

6.4.1.7.3.1. Role

6.4.1.7.3.1.1. Operator Type

Foundation

6.4.1.7.3.1.2. Description

Parameterized Constructor - for string storage

6.4.1.7.3.2. Pre-conditions

None

6.4.1.7.3.3. Post-conditions

indicator=EQUATION.
equation=equation_string.

6.4.1.8. Friend Member Functions

6.4.1.8.1. compare compare_containers(complex_container left, complex_container right)

6.4.1.8.1.1. Role

6.4.1.8.1.1.1. Operator Type

Decider

6.4.1.8.1.1.2. Description

Compares two complex containers, and returns SMALLER, EQUAL or LARGER, depending upon the data stored in both containers. The result is applied thus: eg ‘left’ is SMALLER than ‘right’. First order on enum (ie CONSTANT < EQUATION), then for CONSTANT - compare real components then imaginary components of complex number - and for EQUATION - lexically compare equation strings.

6.4.1.8.1.2. Pre-conditions

None

6.4.1.8.1.3. Post-conditions

None

6.4.1.8.1.4. Return Data

RETURN SMALLER if left<right
RETURN EQUAL if left==right
RETURN LARGER if left>right

6.4.1.8.2. ostream& operator<<(ostream& output_stream, const complex_container container)

6.4.1.8.2.1. Role

6.4.1.8.2.1.1. Operator Type

Overloaded output operator

6.4.1.8.2.1.2. Description

For indicator=CONSTANT , output complex number.
For indicator=EQUATION, output “EQUATION=” followed by equation string.

6.4.1.8.2.2. Pre-conditions

None

6.4.1.8.2.3. Post-conditions

See Description for output on stream.

6.4.1.8.3. istream& operator>>(istream& input_stream, complex_container& container)

6.4.1.8.3.1. Role

6.4.1.8.3.1.1. Operator Type

Overloaded input operator

6.4.1.8.3.1.2. Description

Accepts input in one of two forms :

1) CONSTANT a+bj
2) EQUATION <equation_string>

6.4.1.8.3.2. Pre-conditions

None

6.4.1.8.3.3. Post-conditions

indicator set to CONSTANT/EQUATION according to form of input.
either complex_number or equation string set according to form of input.

6.4.1.9. Static Member Functions

None

7. Header File: “rCalculatorClass.h”

7.1. General Information

7.1.1. Header File Role

To declare calculator class.

7.1.2. Standard Headers Required

None

7.1.3. Custom Headers Required

rCalculatorUserTypes.h, complex.h, complex functions.h, iadditionalmath.h, extraclasses.h, ulist.h, name_object.h, validator.h

7.2. C-Type Definitions

7.2.1. Forward Declarations

class validator

7.3. Non-Class Function Prototypes

none

7.4. Class Definitions

7.4.1. Class “calculator”

7.4.1.1. Role:

Given an ‘order’ string, the calculator object will attempt to evaluate its algebraic interpretation using complex number arithmetic. If an assignment order is given, an attempt will be made to store the result of the expression in the variable list of the calculator. If a definition order for an equation is given, an attempt will be made to store the equation string in the equation list of the calculator (the new equation is cross-referenced against all other equations in the equation list for circular references - if auto verfication is active). Any algebraic errors, invalid variable/equation names or invalid operands (eg divide by zero) are stored in an on-going error report, which may be viewed or cleared by the user at will.

7.4.1.2. Class Initialisation:

Call to build_internal_constants member function allows the generic constants list for all calculator objects to be set. (implementation stores pi,e and j values)

7.4.1.3. Private Data Members

7.4.1.3.1. validator *equation_checker

Pointer to validator object which provides a verifying service for equations placed in the equation list.

7.4.1.3.2. int *number_of_errors

Points to a count of the number of errors occurred since the error stream was last cleared.

7.4.1.3.3. calculator_type rank

Determines whether the calculator defines its own error/input streams and variable/equation lists. A SUPER CALCULATOR does have this authority, whereas a SUB CALCULATOR does not, referencing streams and lists in a SUPER CALCULATOR.

7.4.1.3.4. calculator_symbol current_symbol

Holds the current token name and data read in from the input stream.

7.4.1.3.5. int binding_segment

Keeps track of the current level of parenthesis/modulus nesting.

7.4.1.3.6. ulist<name_object> *var_list

A list of variable memories, similar to the memories available on a pocket calculator. Each variable is referenced by an alphanumeric string name, and has tied to it a complex number. A variable name may only appear once in this list. The list is ordered in ascending alphabetical order on name. A variable name referenced in an expression for evaluation is expanded to its tied complex number.

7.4.1.3.7. ulist<name_object> *equation_list

A list of equation memories, each with a unique name. Each name has tied to it a string holding an expression. The expression is stored using calculator character tokens. An equation name referenced in an expression for evaluation has its expression/equation string calculated by a dynamically

in an expression for evaluation has its expression/equation string calculated by a dynamically created SUB CALCULATOR. The complex number result from this is substituted into the original expression. Equations may be nested to any depth, but must not contain a circular reference.

7.4.1.3.8. istrstream *input_stream

A string stream pointer used to extract characters from a tied character array (a calculator order or part of an order)

7.4.1.3.9. ostrstream *error_stream

A string stream pointer used to output characters into a tied character array. The error_stream contains all errors reported throughout evaluation thus far. The error_stream is cleared by a call to flush_errors.

7.4.1.3.10. char *input_char_array

The char array buffer which input_stream accepts characters from.

7.4.1.3.11. char *error_string

The char array buffer error_stream outputs characters to.

7.4.1.3.12. int auto_verify

Flag which forces a newly defined equation to be checked against all other equations in the equation list for circular definitions. Active=1, Inactive=0

7.4.1.4. Public Data Members

None

7.4.1.5. Static Data Members

7.4.1.5.1. int math_function_array_initialised (PRIVATE)

Flag which enables class to initialise its static math_func and terminator token arrays on first instantiation, but not for further instantiations.

7.4.1.5.2. double (*math_func[256])(double) (PRIVATE)

Look-up table matching a subset of calculator tokens to C ‘math.h’ library functions.

7.4.1.5.3. char terminator_table[256] (PRIVATE)

Derived look-up table (order 1) flagging a subset of calculator tokens as valid final tokens in an expression.

7.4.1.5.4. const token_value terminator_tokens[15] (PRIVATE)

Hard-coded array of valid terminator tokens. Is used for generating an ORDER 1 look-up table (see above).

7.4.1.5.5. const token_name token_names[50] (PUBLIC)

Hard-coded array of valid token names (OPERATION names) and associated single chars which are recognised by calculator as token characters.

7.4.1.5.6. ulist<name_object> constant_list (PUBLIC)

Externally provided list of constants (each with name and complex number) - generic to all calculator objects.

7.4.1.5.7. int ERROR_STREAM_SIZE (PUBLIC)

Size of error stream buffer, ie size of error_string.

7.4.1.6. Private Member Functions

7.4.1.6.1. void new_error_stream()

7.4.1.6.1.1. Role

7.4.1.6.1.1.1. Operator Type

Modifier

7.4.1.6.1.1.2. Description

Deletes old error stream and error_string. Allocates new error stream and error_string.

7.4.1.6.1.2. Pre-conditions

None

7.4.1.6.1.3. Post-conditions

*number_of_errors=0

*error_string=”” , new error_stream/error_string allocated

7.4.1.6.2. complex level1()

7.4.1.6.2.1. Role

7.4.1.6.2.1.1. Operator Type

7.4.1.6.2.1.2. Description

Recursive level1 - lowest precedence evaluation function. Deals with addition/subtraction.

7.4.1.6.2.2. Pre-conditions

current_symbol is loaded with data pertaining to last token read from input.

7.4.1.6.2.3. Post-conditions

current_symbol holds END or next token after PLUS/MINUS operation.

current_symbol unchanged if PLUS/MINUS not specified in current_symbol.

Report error if second term of binary operation missing from input.

7.4.1.6.2.4. Return Data

RETURN result of addition/subtraction or level2 call if current_symbol.token not PLUS/MINUS.

7.4.1.6.3. complex level2()

7.4.1.6.3.1. Role

7.4.1.6.3.1.1. Operator Type

7.4.1.6.3.1.2. Description

Recursive level2 - evaluation function. Deals with multiplication/division.

7.4.1.6.3.2. Pre-conditions

current_symbol is loaded with data pertaining to last token read from input.

7.4.1.6.3.3. Post-conditions

current_symbol holds END or next token after MUL/DIV operation.

current_symbol unchanged if MUL/DIV not specified in current_symbol.

Report error if second term of binary operation missing from input, or divide by 0.

7.4.1.6.3.4. Return Data

RETURN result of MUL/DIV or level3 call if current_symbol.token not MUL/DIV.

7.4.1.6.4. complex level3()

7.4.1.6.4.1. Role

7.4.1.6.4.1.1. Operator Type

7.4.1.6.4.1.2. Description

Recursive level3 - evaluation function. Deals with power/roots/logs.

7.4.1.6.4.2. Pre-conditions

current_symbol is loaded with data pertaining to last token read from input.

7.4.1.6.4.3. Post-conditions

current_symbol holds END or next token after POW/ROOTX/LOGX operation.

current_symbol unchanged if POW/ROOTX/LOGX not specified in current_symbol.

Report error if second term of binary operation missing from input, or invalid operands.

7.4.1.6.4.4. Return Data

RETURN result of POW/ROOTX/LOGX or level4 call if current_symbol.token not POW/ROOTX/LOGX.

7.4.1.6.5. complex level4()

7.4.1.6.5.1. Role

7.4.1.6.5.1.1. Operator Type

7.4.1.6.5.1.2. Description

Recursive level4 - evaluation function. Deals with FACTORIAL.

7.4.1.6.5.2. Pre-conditions

current_symbol is loaded with data pertaining to last token read from input.

7.4.1.6.5.3. Post-conditions

current_symbol holds next token after FACTORIAL operation.

current_symbol unchanged if FACTORIAL not specified in current_symbol.

Report error if operand is not a non-negative integer.

7.4.1.6.5.4. Return Data

RETURN result of FACTORIAL or level5 call if current symbol not FACTORIAL.

7.4.1.6.6. complex level5()

7.4.1.6.6.1. Role

7.4.1.6.6.1.1. Operator Type

7.4.1.6.6.1.2. Description

Recursive level5 - evaluation function. Deals with engineering symbols: MILLI, MICRO, NANO, PICO, FEMTO, KILO, MEGA, GIGA, TERA, PETA, EXA

7.4.1.6.6.2. Pre-conditions

current_symbol is loaded with data pertaining to last token read from input.

7.4.1.6.6.3. Post-conditions

current_symbol holds next token after FACTORIAL operation.
current_symbol unchanged if FACTORIAL not specified in current_symbol.

7.4.1.6.6.4. Return Data

RETURN result of multiplying specified by engineering symbol by prefixed value or 'primary' level call if current symbol not an engineering symbol.

7.4.1.6.7. complex primary()

7.4.1.6.7.1. Role

7.4.1.6.7.1.1. Operator Type

7.4.1.6.7.1.2. Description

Recursive level6 - highest precedence evaluation function. Deals with remaining functions, parentheses, modulus, etc. See post-conditions for details.

7.4.1.6.7.2. Pre-conditions

current_symbol is loaded with data pertaining to last token read from input.

7.4.1.6.7.3. Post-conditions

IF current_symbol.token==:

NUMBER: current_symbol.number_value (and return) set to complex number read in.
ERRORS: if name/number follows name in input_stream.

NAME: If name is known and this is an assignment to that name...
name must be present in variable list - otherwise return NULLcomplex
update complex number for this name in var list according to input.
Else if name is known and this is a definition of a name (ie set up as equation)
name must be present in equation list - otherwise return NULLcomplex
update equation string for this name in equation list according to input.
IF auto verify is on and circular definitions are found...
Remove new name and equation from equation list,
and return NULLcomplex
Else If name is not known and this is an assignment to that name...
add this name and corresponding complex number to var list
Else if name is not known and this is a definition of a name (ie equation)
add this name and corresponding equation string to equation list.
Else if name is known and this is neither definition or assignment - then reference
IF name is a variable or constant return corresponding complex number
ELSE evaluate equation string for name and return result.
Else if name is not known and this is a reference
return NULLcomplex - name not found

MINUS: Prefixing unary minus - return negative of following term in expression
LP: Left parenthesis: return expression inside this patenthesis level
ERRORS: if this parenthesis level is not terminated with RP : right parenthesis
LM: Left Modulus: return complex modulus of expression inside this parenthesis level
ERRORS: if this parenthesis level is not terminated with RM : right modulus
SQRT/CBRT: square/cube root: Evaluate following term and take square/cube root.
ERRORS: flagged by invalid_operands function

SIN/COS/TAN/ASIN/ACOS/ATAN/SINH/COSH/TANH/ASINH/ACOSH/ATANH/LN/LOG10 /LOG2 : IF illegal_operands function finds following operand illegal return NULL result.
return evaluation of function upon operand.
ARGUMENT: return complex argument of following term.
REAL: return real component of following term.
IMAGINARY: return imag component of following term.
WINDOW: return following term unchanged if inside window bounds
ELSE return NULLcomplex

SUMMATION: return SUMMATION of following term given summation parameters.
PRODATION: return PRODATION of following term given prodation parameters.
None of the above: RETURN NULLcomplex

7.4.1.6.7.4. Return Data
See Post-conditions

7.4.1.6.8. complex compound(token_value method)
7.4.1.6.8.1. Role
7.4.1.6.8.1.1. Operator Type
Internal utility
7.4.1.6.8.1.2. Description
Called when a SUMMATION/PRODATION token is detected by primary(). Acquires compound parameters from input_stream, and repeatedly evaluates the following expression forming a SUM total, or a PRODUCTS total, which is returned.
7.4.1.6.8.2. Pre-conditions
Next token to be read from input_stream is left parenthesis (containing compound parameters).
7.4.1.6.8.3. Post-conditions
IF method=PLUS then SUMMATION is performed upon expression following compound parameters.
IF method=MUL then PRODATION is performed upon expression following compound parameters.
7.4.1.6.8.4. Return Data
RETURN result of SUMMATION/PRODATION if pre-condition met.
ELSE return NULLcomplex.

7.4.1.6.9. status get_compound_parameters(token_value method, string_class &variable, int &lower_bound, int &upper_bound)
7.4.1.6.9.1. Role
7.4.1.6.9.1.1. Operator Type
Internal utility
7.4.1.6.9.1.2. Description
Reads in parameters required for carrying out SUMMATION/PRODATION on expression following compound parameters in input_stream.
7.4.1.6.9.2. Pre-conditions
Next token to be read from input_stream is left parenthesis (containing compound parameters).
7.4.1.6.9.3. Post-conditions
IF compound parameter format is correct, input_stream now points to beginning of expression for applying SUMMATION/PRODATION to, and variable parameters of function call are loaded with parameters read from input_stream. Otherwise, where format is incorrect input_stream points to character in error.
7.4.1.6.9.4. Return Data
RETURN SUCCESS if pre-conditions met and compound parameters had correct format.
Otherwise RETURN ERROR.

7.4.1.6.10. status get_window_parameters(string_class &variable, double &lower_bound, double &upper_bound)
7.4.1.6.10.1. Role
7.4.1.6.10.1.1. Operator Type
Internal Utility
7.4.1.6.10.1.2. Description
Reads in parameters required for carrying out windowing function upon following expression in input_stream.
7.4.1.6.10.2. Pre-conditions
Next token to be read from input_stream is left parenthesis (containing compound parameters).
7.4.1.6.10.3. Post-conditions
IF window parameter format is correct, input_stream now points to beginning of expression for applying windowing operation to, and variable parameters of function call are loaded with parameters read from input_stream. Otherwise, input_stream points to character in error.
7.4.1.6.10.4. Return Data
RETURN SUCCESS if pre-conditions met and windowing parameters had correct format.
Otherwise RETURN ERROR.

7.4.1.6.11. string_class get_definition()

7.4.1.6.11.1. Role

7.4.1.6.11.1.1. Operator Type

Internal utility

7.4.1.6.11.1.2. Description

Spools remainder of input_stream into a string.

7.4.1.6.11.2. Pre-conditions

None

7.4.1.6.11.3. Post-conditions

input_stream now empty.

7.4.1.6.11.4. Return Data

RETURN string consisting of chars stored in input_stream when this function was called.

7.4.1.6.12. void init_math_array()

7.4.1.6.12.1. Role

7.4.1.6.12.1.1. Operator Type

Class Initialiser

7.4.1.6.12.1.2. Description

Called by Default Constructor - calls function to initialise math_func and terminator_table arrays if they have not already been initialised.

7.4.1.6.12.2. Pre-conditions

None

7.4.1.6.12.3. Post-conditions

IF math_function_array_initialised=0, math_func and terminator_tables now initialised.
ELSE no change.

7.4.1.6.13. complex engineering_conversion(const token_value token, const complex x)

7.4.1.6.13.1. Role

7.4.1.6.13.1.1. Operator Type

Internal utility

7.4.1.6.13.1.2. Description

Multiplies complex number ‘x’ by the appropriate power of 10 corresponding to ‘token’.

7.4.1.6.13.2. Pre-conditions

’token’ is an engineering symbol, ie MILLI, MICRO, NANO, PICO, FEMTO, KILO, MEGA, GIGA, TERA, PETA or EXA.

7.4.1.6.13.3. Post-conditions

None

7.4.1.6.13.4. Return Data

IF pre-condition met, RETURN result of multiplication.
Else report error and RETURN NULLcomplex

7.4.1.6.14. int invalid_operands(const token_value token, const complex value1, const complex value2=complex(0,0))

7.4.1.6.14.1. Role

7.4.1.6.14.1.1. Operator Type

7.4.1.6.14.1.2. Description

Check operand value1, and if warranted check operand value2 for validity in accordance with ‘token’ function. eg flag ‘log 0’ as an error.

7.4.1.6.14.2. Pre-conditions

None

7.4.1.6.14.3. Post-conditions

None

7.4.1.6.14.4. Return Data

If token==

DIV:	RETURN 0 if: divide by 0+0j
POW:	RETURN 0 if: (0+0j)^0; exponent complex & base!=exp
ROOTX:	RETURN 0 if: attempt to take 0th root
FACTORIAL:	RETURN 0 if: value1 is not integer or negative
TAN:	RETURN 0 if: value1 is a multiple of $\pi/2$
ASIN/ACOS:	RETURN 0 if: outside the range -1 <= value1 <= 1
ACOSH:	RETURN 0 if: <1
ATANH:	RETURN 0 if: outside the range -1 < value1 < 1
LN/LOG10/LOG2:	RETURN 0 if: <=0

LOG X: RETURN 0 if: value1 or value2 is <=0

Also return 0 if a complex number is passed to any of these tokens below-and-including FACTORIAL.

If no error, RETURN 1.

7.4.1.6.15. token_value get_token()

7.4.1.6.15.1. Role

7.4.1.6.15.1.1. Operator Type

Internal utility

7.4.1.6.15.1.2. Description

Reads characters from input_stream and interprets them as tokens - storing the appropriate token name in current_symbol.token

7.4.1.6.15.2. Pre-conditions

None

7.4.1.6.15.3. Post-conditions

If end of stream reached set and return current_symbol.token=END. If an invalid terminator token was used - report ERROR.

IF character read in is ‘;’, or ‘\n’ - set and return current_symbol.token=PRINT.

IF character read in is any valid token EXCEPT digits, decimal point or alphanumeric name:
set and return current_symbol.token to the corresponding calculator token for this char.

IF character read in is a name, return current_symbol.token=NAME - store alphanumeric name read in current_symbol.name_string.

IF character is none of the above report ERROR “bad token”, return NULL token (PRINT)

7.4.1.6.15.4. Return Data

RETURN token matched to character/group of characters read in.

7.4.1.6.16. complex error(const char* s)

7.4.1.6.16.1. Role

7.4.1.6.16.1.1. Operator Type

Internal utiliry

7.4.1.6.16.1.2. Description

Add char array error report to error_stream.

7.4.1.6.16.2. Pre-conditions

None

7.4.1.6.16.3. Post-conditions

number_of_errors int incremented by 1.

Error message appended to error_stream.

7.4.1.6.16.4. Return Data

RETURN NULLcomplex (0+0j). Enables recursive evaluation functions to return errors - error is reported by this function and (0+0j) is the NULL result sent back to the calling function of the recursive evaluation function.

7.4.1.6.17. complex error(const string_class s)

7.4.1.6.17.1. Role

7.4.1.6.17.1.1. Operator Type

Internal utility

7.4.1.6.17.1.2. Description

Add string_class error report to error_stream.

7.4.1.6.17.2. Pre-conditions

None

7.4.1.6.17.3. Post-conditions

number_of_errors int incremented by 1.

Error message appended to error_stream.

7.4.1.6.17.4. Return Data

RETURN NULLcomplex (0+0j). Enables recursive evaluation functions to return errors - error is reported by this function and (0+0j) is the NULL result sent back to the calling function of the recursive evaluation function.

7.4.1.6.18. calculator(ulist<name_object> *var_list, ulist<name_object> *equation_list, ostrstream *errors, int *number_of_errors)

7.4.1.6.18.1. Role

7.4.1.6.18.1.1. Operator Type

Private Default Constructor.

7.4.1.6.18.1.2. Description

Instantiate a SUB_CALCULATOR calculator object - inherits lists and streams from SUPER.

7.4.1.6.18.2. Pre-conditions

Must be an instantiation from within the calculator class.

7.4.1.6.18.3. Post-conditions

SET *this var_list, equation_list, error_stream, and number_of_errors to the constructor parameters - inherited from the SUPER_CALCULATOR.
input_stream, input_char_array, error_string, equation_checker all set to NULL.
Disable auto-verification - no defining can occur in a SUB_CALCULATOR.
Set *this.rank to SUB_CALCULATOR.

7.4.1.6.19. status set_input(string_class input_string)

7.4.1.6.19.1. Role

7.4.1.6.19.1.1. Operator Type

Internal Modifier

7.4.1.6.19.1.2. Description

Delete currently used input_stream/input_string and allocate new input_stream to point to a dynamically created copy of char array stored in input_string.

7.4.1.6.19.2. Pre-conditions

None

7.4.1.6.19.3. Post-conditions

input_stream reads from a buffer containing copy of char array stored in input_string.

7.4.1.6.19.4. Return Data

RETURN SUCCESS

7.4.1.6.20. status reset_input()

7.4.1.6.20.1. Role

7.4.1.6.20.1.1. Operator Type

Internal Modifier

7.4.1.6.20.1.2. Description

Delete currently used input_stream/input_string - they will now point to NULL.

7.4.1.6.20.2. Pre-conditions

input_stream/input_string are already allocated, ie not pointing to NULL.

7.4.1.6.20.3. Post-conditions

Input_stream/input_string deleted, set to NULL.

7.4.1.6.20.4. Return Data

RETURN SUCCESS if precondition met, else RETURN ERROR.

7.4.1.7. Public Member Functions

7.4.1.7.1. calculator()

7.4.1.7.1.1. Role

7.4.1.7.1.1.1. Operator Type

Foundation

7.4.1.7.1.1.2. Description

Default Constructor (generates a SUPER CALCULATOR)

7.4.1.7.1.2. Pre-conditions

None

7.4.1.7.1.3. Post-conditions

Set equation_checker, input_stream, input_char_array, error_stream, error_string to NULL.
Set auto_verify on. Reset binding segment to 0. Set rank as SUPER_CALCULATOR.
Math function array initialised if this is first calculator object instantiated.
var_list and equation_list dynamically allocated.
new error_stream/error_string allocated

7.4.1.7.2. ~calculator()

7.4.1.7.2.1. Role

7.4.1.7.2.1.1. Operator Type

Foundation

7.4.1.7.2.1.2. Description

Destructor

7.4.1.7.2.2. Pre-conditions

None

7.4.1.7.2.3. Post-conditions

If destroying a SUPER CALCULATOR, delete var_list, equation_list, error_string, error_stream.
Do nothing if SUB CALCULATOR being destroyed.

7.4.1.7.3. calculator(const calculator &original)

7.4.1.7.3.1. Role

7.4.1.7.3.1.1. Operator Type

Foundation

7.4.1.7.3.1.2. Description

Copy Constructor

7.4.1.7.3.2. Pre-conditions

None

7.4.1.7.3.3. Post-conditions

*this=original (overloaded = operator used)

7.4.1.7.4. calculator& operator=(const calculator &source)

7.4.1.7.4.1. Role

7.4.1.7.4.1.1. Operator Type

Overloaded = Operator

7.4.1.7.4.1.2. Description

Allow assignment between calculator objects.

7.4.1.7.4.2. Pre-conditions

Assignment to oneself not permitted.

7.4.1.7.4.3. Post-conditions

Allocate new var_list and equation_list - copy lists from source to *this.
Copy all non-dynamic data from source into *this.
Copy across *number_of_errors from source into *this.
Set input_streams and input_char_array to NULL.
Allocate new error stream/error_string/number_of_errors, copy across error string from source and copy *number_of_errors if not empty error string.
Otherwise, allocate new error_stream/error_string with no errors.

7.4.1.7.4.4. Return Data

RETURN *this

7.4.1.7.5. ulist<name_object>* get_equation_list()

7.4.1.7.5.1. Role

7.4.1.7.5.1.1. Operator Type

Extractor

7.4.1.7.5.1.2. Description

Return pointer to equation list (only used by preprocessor)

7.4.1.7.5.2. Pre-conditions

None

7.4.1.7.5.3. Post-conditions

None

7.4.1.7.5.4. Return Data

RETURN equation_list

7.4.1.7.6. ulist<name_object>* get_var_list()

7.4.1.7.6.1. Role

7.4.1.7.6.1.1. Operator Type

Extractor

7.4.1.7.6.1.2. Description

Return pointer to variable list (only used by preprocessor)

7.4.1.7.6.2. Pre-conditions

None

7.4.1.7.6.3. Post-conditions

None

7.4.1.7.6.4. Return Data

RETURN variable_list

7.4.1.7.7. complex evaluate(string_class input_string)

7.4.1.7.7.1. Role

7.4.1.7.7.1.1. Operator Type

Primary service

7.4.1.7.7.1.2. Description

Evaluates the entire expression pointed to by input_string, resulting in a single complex number which is returned.

7.4.1.7.7.2. Pre-conditions

None

7.4.1.7.7.3. Post-conditions

String is parsed, and expression evaluated. The exact post conditions depend upon the instructions given within the input_string. An evaluation may take place, a variable may be added to the variable list or an equation may be added to the equation list.

7.4.1.7.7.4. Return Data

If evaluation proceeds without ERROR then return complex result of evaluation.
ELSE if error occurs or assignment of variable/definition of equation occurs return NULLcomplex result = 0+0j. Any errors that have occurred may be found in the error_stream. Number of errors that occurred is stored in the corresponding private data member, and outputted if *number_of_errors!=0

7.4.1.7.8. string_class flush_errors()

7.4.1.7.8.1. Role

7.4.1.7.8.1.1. Operator Type

Modifier

7.4.1.7.8.1.2. Description

Blanks error_stream/error_string, deletes old stream/string and allocates new stream/string for errors.

7.4.1.7.8.2. Pre-conditions

None

7.4.1.7.8.3. Post-conditions

Error_stream/error_string are empty - *number_of_errors=0

7.4.1.7.8.4. Return Data

RETURN copy of error_string that has been deleted.

7.4.1.7.9. ostream& peek_errors(ostream& output_stream)

7.4.1.7.9.1. Role

7.4.1.7.9.1.1. Operator Type

Extractor

7.4.1.7.9.1.2. Description

Returns current error_string using streamed output.

7.4.1.7.9.2. Pre-conditions

None

7.4.1.7.9.3. Post-conditions

error_string is placed on output stream.

7.4.1.7.9.4. Return Data

RETURN output_stream

7.4.1.7.10. int get_number_of_errors()

7.4.1.7.10.1. Role

7.4.1.7.10.1.1. Operator Type

Extractor

7.4.1.7.10.1.2. Description

Return *number_of_errors

7.4.1.7.10.2. Pre-conditions

None

7.4.1.7.10.3. Post-conditions

None

7.4.1.7.10.4. Return Data

RETURN *number_of_errors

7.4.1.7.11. void all_clear()

7.4.1.7.11.1. Role

7.4.1.7.11.1.1. Operator Type

Modifier

7.4.1.7.11.1.2. Description

Clears variable and equation lists of all names and data.

7.4.1.7.11.2. Pre-conditions

None

7.4.1.7.11.3. Post-conditions

Variable and equation lists are empty.

7.4.1.7.12. status clear_single_memory(string_class name)

7.4.1.7.12.1. Role

7.4.1.7.12.1.1. Operator Type

Modifier

7.4.1.7.12.1.2. Description

Clears a particular variable or equation name and data, specified by ‘name’.

7.4.1.7.12.2. Pre-conditions

variable or equation is known by ‘name’

7.4.1.7.12.3. Post-conditions

variable or equation known by ‘name’ is removed from the relevant list.

7.4.1.7.12.4. Return Data

RETURN SUCCESS if precondition met, else RETURN ERROR

7.4.1.7.13. void auto_verify_off()

7.4.1.7.13.1. Role

7.4.1.7.13.1.1. Operator Type

Modifier

7.4.1.7.13.1.2. Description

Deactivate auto-checking of equations for circular references. Mainly for testing.

7.4.1.7.13.2. Pre-conditions

None

7.4.1.7.13.3. Post-conditions

Equation with circular references to themselves or other equations already in the equation list may be added to the equation list by the user.

7.4.1.7.14. status auto_verify_on()

7.4.1.7.14.1. Role

7.4.1.7.14.1.1. Operator Type

Modifier

7.4.1.7.14.1.2. Description

Activate auto-checking of equations. Checks for circular references to other equations in the equation list for this calculator are made as equation definitions are received.

7.4.1.7.14.2. Pre-conditions

None of the current equations stored in the equation list for this calculator describe a circular definition with themselves or any other equation stored in the list.

7.4.1.7.14.3. Post-conditions

An equation cannot be added to the equation_list if it generates a circular reference.

7.4.1.7.14.4. Return Data

RETURN SUCCESS if pre-conditions are met, otherwise RETURN ERROR.

7.4.1.7.15. void set_validator(validator *checker)

7.4.1.7.15.1. Role

7.4.1.7.15.1.1. Operator Type

Modifier - essential for auto-verification to operate.

7.4.1.7.15.1.2. Description

Links a validator object to this calculator so that auto-verification may be done.

7.4.1.7.15.2. Pre-conditions

’checker’ points to a validator that will be in existence for the life of the calculator object.

7.4.1.7.15.3. Post-conditions

equation_checker=checker

7.4.1.8. Friend Member Functions

7.4.1.8.1. ostream& operator<<(ostream& output_stream, const calculator a)

7.4.1.8.1.1. Role

7.4.1.8.1.1.1. Operator Type

Overloaded output operator

7.4.1.8.1.1.2. Description

Output variable list and equation list for this calculator.

7.4.1.8.1.2. Pre-conditions

None

7.4.1.8.1.3. Post-conditions

Output is sent to output stream.

7.4.1.8.1.4. Return Data

RETURN output_stream.

7.4.1.8.2. complex sqrt_comp(const complex value)

Defined in complex functions header

7.4.1.8.3. complex cbrt(const complex value)

Defined in complex functions header

7.4.1.9. Static Member Functions

7.4.1.9.1. void initialise_math_function_array() (PRIVATE)

7.4.1.9.1.1. Role

7.4.1.9.1.1.1. Operator Type

Generic Initialiser

7.4.1.9.1.1.2. Description

Called only once - when the first calculator object is instantiated. Builds ORDER 1 look-up tables: 1) math_func array holds function pointers to trigonometric and logarithmic C library functions; 2) terminator_table holds all valid expression terminator tokens.

7.4.1.9.1.2. Pre-conditions

None (except only needs to be called once through the constructor of the first calculator object)

7.4.1.9.1.3. Post-conditions

math_func array and terminator_table array are initialised.

7.4.1.9.2. void build_internal_constants(const name_object *constants,const int length)

7.4.1.9.2.1. Role

7.4.1.9.2.1.1. Operator Type

Generic Modifier

7.4.1.9.2.1.2. Description

Sets constant list for all calculators

7.4.1.9.2.2. Pre-conditions

None

7.4.1.9.2.3. Post-conditions

Constant list for all calculators built from ‘constants’ array

8. Header File: “Data Manager.h”

8.1. General Information

8.1.1. Header File Role

Declares data_manager object

8.1.2. Standard Headers Required

iostream.h, strstream.h

8.1.3. Custom Headers Required

set_input_object.h, data_set_obj.h, record_object.h, name_object.h, stringobject.h, newstring.h, complex.h, ulist.h

8.2. C-Type Definitions

None

8.3. Non-Class Function Prototypes

None

8.4. Class Definitions

8.4.1. Class “data_manager”

8.4.1.1. Role:

To maintain lists of record_objects, data_set_objs, IO_map_objects and set_input_objects. Functions are also supplied to check if certain named objects are present in lists. A CLI is defined for allowing user interaction with all lists. User may add and remove individual objects from each list, specifying object data when necessary. User may also dump contents of each list, separately, to output.

8.4.1.2. Class Initialisation:

None

8.4.1.3. Private Data Members

8.4.1.3.1. ulist<record_object> record_list

Each record object in the list has a unique identifying name and a list of stringobjects (holding field names).

8.4.1.3.2. ulist<data_set_obj> data_list

Each data_set_obj in the list has a unique identifying name and an array of data for processing by a calculator.

8.4.1.3.3. ulist<IO_map_object> map_list

Each IO_map_object in the list has a unique identifying name, list of input field names,and a list of output field names.

8.4.1.3.4. ulist<set_input_object> set_input_list

Each set_input_object in the list has a unique identifying name and a specification for loading a column of a data_set object with data.

8.4.1.4. Public Data Members

None

8.4.1.5. Static Data Members

None

8.4.1.6. Private Member Functions

None

8.4.1.7. Public Member Functions

8.4.1.7.1. void interface()

8.4.1.7.1.1. Role

8.4.1.7.1.1.1. Operator Type

CLI interface

8.4.1.7.1.1.2. Description

Allows user to interact with all lists as described in the class role description through the use of a CLI. In general, each user command corresponds to an equivalent public member function in this class.

8.4.1.7.1.2. Pre-conditions

None

8.4.1.7.1.3. Post-conditions

Lists will be updated in accordance to user’s commands where no errors have occurred.

8.4.1.7.2. void display_help()

8.4.1.7.2. void display_help()

8.4.1.7.2.1. Role

8.4.1.7.2.1.1. Operator Type

CLI interface

8.4.1.7.2.1.2. Description

Output list of user commands for this manager to standard output.

8.4.1.7.2.2. Pre-conditions

None

8.4.1.7.2.3. Post-conditions

See Description

8.4.1.7.3. status new_record(string_class name, string_class record)

8.4.1.7.3.1. Role

8.4.1.7.3.1.1. Operator Type

Modifier

8.4.1.7.3.1.2. Description

Adds new record_object to record_list using ‘name’ and ‘record’ to build list node, ordered alphabetically on name.

8.4.1.7.3.2. Pre-conditions

’name’ for record_object is unique to record_list

8.4.1.7.3.3. Post-conditions

See Description

8.4.1.7.3.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

8.4.1.7.4. status delete_record(string_class name)

8.4.1.7.4.1. Role

8.4.1.7.4.1.1. Operator Type

Modifier

8.4.1.7.4.1.2. Description

Removes record_object identified by ‘name’ from record_list

8.4.1.7.4.2. Pre-conditions

Such a record_object exists in record_list

8.4.1.7.4.3. Post-conditions

See Description

8.4.1.7.4.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

8.4.1.7.5. void output_record_list()

8.4.1.7.5.1. Role

8.4.1.7.5.1.1. Operator Type

Extractor

8.4.1.7.5.1.2. Description

Output contents of record_list to standard output

8.4.1.7.5.2. Pre-conditions

None

8.4.1.7.5.3. Post-conditions

See Description.

8.4.1.7.6. status new_data(string_class name, string_class record, int length)

8.4.1.7.6.1. Role

8.4.1.7.6.1.1. Operator Type

Modifier

8.4.1.7.6.1.2. Description

Adds new data_set_obj to data_list using ‘name’, ‘record’ and ‘length’ to build list node, ordered alphabetically on name.

8.4.1.7.6.2. Pre-conditions

’name’ for data_set_obj is unique to data_list

8.4.1.7.6.3. Post-conditions

See Description

8.4.1.7.6.4. Return Data

RETURN SUCCSS if pre-conditions met, else RETURN ERROR

8.4.1.7.7. status delete_data(string_class name)

8.4.1.7.7.1. Role

8.4.1.7.7.1.1. Operator Type

Modifier

8.4.1.7.7.1.2. Description

Removes data_set_obj identifier by ‘name’ from data_list

8.4.1.7.7.2. Pre-conditions

Such a data_set_obj exists in data_list

8.4.1.7.7.3. Post-conditions

See Description

8.4.1.7.7.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

8.4.1.7.8. void output_data_list()

8.4.1.7.8.1. Role

8.4.1.7.8.1.1. Operator Type

Extractor

8.4.1.7.8.1.2. Description

Output contents of data_list to standard output

8.4.1.7.8.2. Pre-conditions

None

8.4.1.7.8.3. Post-conditions

See Description

8.4.1.7.9. status check_data_list(string_class data_name, ulist<string_object> inputfields, ulist<string_object> mapnames, data_set_obj* &target)

8.4.1.7.9.1. Role

8.4.1.7.9.1.1. Operator Type

Decider

8.4.1.7.9.1.2. Description

Check data_set_obj exists in data_set list with 'data_name' name, check all strings in inputfields are valid data_set field names, check all mapnames strings match to IO_map_objects in map_list with same names, check all field names referenced inside each IO_map_object are present in specified data_set and return pointer to data_set_obj with 'data_name' name in 'target'.

8.4.1.7.9.2. Pre-conditions

data_list is not empty

8.4.1.7.9.3. Post-conditions

None

8.4.1.7.9.4. Return Data

RETURN SUCCESS if pre-conditions met and all checks are ok, else RETURN ERROR.

8.4.1.7.10. status new_map(string_class map_name, string_class input_fields, string_class output_fields)

8.4.1.7.10.1. Role

8.4.1.7.10.1.1. Operator Type

Modifier

8.4.1.7.10.1.2. Description

Adds new IO_map_object to map_list using ‘name’, ‘input fields’ and ‘output_fields’ to build list node, ordered alphabetically on name.

8.4.1.7.10.2. Pre-conditions

’name’ for IO_map_object is unique to map_list

8.4.1.7.10.3. Post-conditions

See Description

8.4.1.7.10.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

8.4.1.7.11. status delete_map(string_class map_name)

8.4.1.7.11.1. Role

8.4.1.7.11.1.1. Operator Type

Modifier

8.4.1.7.11.1.2. Description

Removes IO_map_object identified by ‘name’ from map_list

8.4.1.7.11.2. Pre-conditions
Such an IO_map_object exists in map_list
8.4.1.7.11.3. Post-conditions
See Description
8.4.1.7.11.4. Return Data
RETURN SUCCESS if pre-conditions met, else RETURN ERROR

8.4.1.7.12. void output_map_list()
8.4.1.7.12.1. Role
8.4.1.7.12.1.1. Operator Type
Extractor
8.4.1.7.12.1.2. Description
Output contents of map_list to standard output
8.4.1.7.12.2. Pre-conditions
None
8.4.1.7.12.3. Post-conditions
See Description

8.4.1.7.13. status new_set_input(string_class set_input_name, int lower_i, int upper_i, float start_val, float increment)
8.4.1.7.13.1. Role
8.4.1.7.13.1.1. Operator Type
Modifier
8.4.1.7.13.1.2. Description
Adds new set_input_object to set_input_list using ‘name’, ‘lower_i’, ‘upper_i’, ‘start_val’ and ‘increment’ to build list node, ordered alphabetically on name.
8.4.1.7.13.2. Pre-conditions
’name’ for set_input_object is unique to set_input_list
8.4.1.7.13.3. Post-conditions
See Description
8.4.1.7.13.4. Return Data
RETURN SUCCESS if pre-conditions met, else RETURN ERROR

8.4.1.7.14. status delete_set_input(string_class set_input_name)
8.4.1.7.14.1. Role
8.4.1.7.14.1.1. Operator Type
Modifier
8.4.1.7.14.1.2. Description
Remove set_input_object identified by ‘name’ from set_input_list
8.4.1.7.14.2. Pre-conditions
Such a set_input_object exists in set_input_list
8.4.1.7.14.3. Post-conditions
See Description
8.4.1.7.14.4. Return Data
RETURN SUCCESS if pre-conditions met, else RETURN ERROR

8.4.1.7.15. void output_set_input_list()
8.4.1.7.15.1. Role
8.4.1.7.15.1.1. Operator Type
Extractor
8.4.1.7.15.1.2. Description
Output contents of set_input_list to standard output.
8.4.1.7.15.2. Pre-conditions
None
8.4.1.7.15.3. Post-conditions
See Description

8.4.1.7.16. status check_set_input_list(ulist<string_object> sourcelist)

8.4.1.7.16.1. Role

8.4.1.7.16.1.1. Operator Type

Decider

8.4.1.7.16.1.2. Description

Check that all names in sourcelist reference valid set_input_objs in set_input_list .

8.4.1.7.16.2. Pre-conditions

sourcelist is not an empty list

8.4.1.7.16.3. Post-conditions

None

8.4.1.7.16.4. Return Data

RETURN SUCCESS if pre-conditions met and all names are found in set_input_list.
else RETURN ERROR

8.4.1.7.17. void reset_manager()

8.4.1.7.17.1. Role

8.4.1.7.17.1.1. Operator Type

Modifier

8.4.1.7.17.1.2. Description

Clears all lists.

8.4.1.7.17.2. Pre-conditions

None

8.4.1.7.17.3. Post-conditions

See Description

8.4.1.8. Friend Member Functions

None

8.4.1.9. Static Member Functions

None

9. Header File: “data_set.h”

9.1. General Information

9.1.1. Header File Role

Contains class definition for data_set class.

9.1.2. Standard Headers Required

None

9.1.3. Custom Headers Required

newstring.h, stringobject.h, ulist.h, complex.h, record_object.h, set_input_object.h

9.2. C-Type Definitions

None

9.3. Non-Class Function Prototypes

none

9.4. Class Definitions

9.4.1. Class “data_set”

9.4.1.1. Role:

To hold a dynamically allocated array (dimensions ‘length’ x ’width’) which holds data for a calculator to use as input/output. A unique field name describes each column of data - the set of field names is stored in record_object with particular ‘record_name’. An array element is referenced associatively or by index across columns, and by index across rows.

9.4.1.2. Class Initialisation:

None

9.4.1.3. Private Data Members

9.4.1.3.1. string_class record_name

Reference to record which holds field names (in list format)

9.4.1.3.2. int length

No of rows in array.

9.4.1.3.3. int width

No of columns in array.

9.4.1.3.4. complex *data_array

Pointer to 2 Dimensional complex number array

9.4.1.3.5. string_class *fields

Pointer to 1 Dimensional array holding copy of field names found in record referenced by record_name.

9.4.1.4. Public Data Members

None

9.4.1.5. Static Data Members

None

9.4.1.6. Private Member Functions

None

9.4.1.7. Public Member Functions

9.4.1.7.1. data_set()

9.4.1.7.1.1. Role

9.4.1.7.1.1.1. Operator Type

Foundation

9.4.1.7.1.1.2. Description

Default Constructor

9.4.1.7.1.2. Pre-conditions

None

9.4.1.7.1.3. Post-conditions

9.4.1.7.1.3. Post-conditions

SET length & width to 0, set pointers to NULL

9.4.1.7.2. data_set(record_object record_ob, const int array_length)

9.4.1.7.2.1. Role

9.4.1.7.2.1.1. Operator Type

Foundation

9.4.1.7.2.1.2. Description

Parameterized Constructor - builds initial data_set array

9.4.1.7.2.2. Pre-conditions

None

9.4.1.7.2.3. Post-conditions

name of record_ob stored in 'record_name'.
field list in record_ob converted to array of string_class, stored in dynamically allocated 'fields'
data_array allocated of size array_length x number of fields - all elements initialised to 0+0j

9.4.1.7.3. data_set(const data_set &original)

9.4.1.7.3.1. Role

9.4.1.7.3.1.1. Operator Type

Foundation

9.4.1.7.3.1.2. Description

Copy Constructor

9.4.1.7.3.2. Pre-conditions

None

9.4.1.7.3.3. Post-conditions

all non-dynamic data copied from original to *this
all dynamic data copied into newly allocated memory referenced by *this

9.4.1.7.4. data_set& operator=(const data_set &original)

9.4.1.7.4.1. Role

9.4.1.7.4.1.1. Operator Type

Overloaded assignment operator

9.4.1.7.4.1.2. Description

Define assignment between data_set objects

9.4.1.7.4.2. Pre-conditions

Not assigning to oneself

9.4.1.7.4.3. Post-conditions

all non-dynamic data copied from original to *this
all dynamic data copied into newly allocated memory referenced by *this

9.4.1.7.4.4. Return Data

return *this

9.4.1.7.5. ~data_set()

9.4.1.7.5.1. Role

9.4.1.7.5.1.1. Operator Type

Foundation

9.4.1.7.5.1.2. Description

Destructor

9.4.1.7.5.2. Pre-conditions

None

9.4.1.7.5.3. Post-conditions

Delete data_array and fields.

9.4.1.7.6. string_class get_record_name()

9.4.1.7.6.1. Role

9.4.1.7.6.1.1. Operator Type

Extractor

9.4.1.7.6.1.2. Description

Extract record_name from data_set

9.4.1.7.6.2. Pre-conditions

None

9.4.1.7.6.3. Post-conditions

None

9.4.1.7.6.4. Return Data

return record_name

9.4.1.7.7. status get_element_assoc(complex &result, const int index, const string_class field)

9.4.1.7.7.1. Role

9.4.1.7.7.1.1. Operator Type

Extractor

9.4.1.7.7.1.2. Description

Extract complex number in data_array. Element defined by column labelled by ‘field’, row ‘index’.

9.4.1.7.7.2. Pre-conditions

field and index are within bounds of data_array.

9.4.1.7.7.3. Post-conditions

variable function parameter ‘result’ set to complex number requested

9.4.1.7.7.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

9.4.1.7.8. status set_element_assoc(const complex new_value, const int index, const string_class field)

9.4.1.7.8.1. Role

9.4.1.7.8.1.1. Operator Type

Modifier

9.4.1.7.8.1.2. Description

Set complex number in data_array to ‘new_value’. Element defined by column labelled by ‘field’, row ‘index’.

9.4.1.7.8.2. Pre-conditions

field and index are within bounds of data_array.

9.4.1.7.8.3. Post-conditions

element described holds ‘new_value’

9.4.1.7.8.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

9.4.1.7.9. status get_element(complex &result, const int index, const int field)

9.4.1.7.9.1. Role

9.4.1.7.9.1.1. Operator Type

Extractor

9.4.1.7.9.1.2. Description

Extract complex number in data_array. Element defined by column ‘field’, row ‘index’.

9.4.1.7.9.2. Pre-conditions

field and index are within bounds of data_array.

9.4.1.7.9.3. Post-conditions

variable function parameter ‘result’ set to complex number requested

9.4.1.7.9.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

9.4.1.7.10. status set_element(const complex new_value, const int index, const int field)

9.4.1.7.10.1. Role

9.4.1.7.10.1.1. Operator Type

Modifier

9.4.1.7.10.1.2. Description

Set complex number in data_array to ‘new_value’. Element defined by column ‘field’, row ‘index’.

9.4.1.7.10.2. Pre-conditions

field and index are within bounds of data_array.

9.4.1.7.10.3. Post-conditions

element described holds ‘new_value’

9.4.1.7.10.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

9.4.1.7.11. int get_length()

9.4.1.7.11.1. Role

9.4.1.7.11.1.1. Operator Type
Extractor
9.4.1.7.11.1.2. Description
Extract length of data array
9.4.1.7.11.2. Pre-conditions
None
9.4.1.7.11.3. Post-conditions
None
9.4.1.7.11.4. Return Data
return length

9.4.1.7.12. int get_width()
9.4.1.7.12.1. Role
9.4.1.7.12.1.1. Operator Type
Extractor
9.4.1.7.12.1.2. Description
Extract width of data array
9.4.1.7.12.2. Pre-conditions
None
9.4.1.7.12.3. Post-conditions
None
9.4.1.7.12.4. Return Data
return width

9.4.1.7.13. void get_fields(string_class *storage)
9.4.1.7.13.1. Role
9.4.1.7.13.1.1. Operator Type
Extractor
9.4.1.7.13.1.2. Description
Copies fields string_class array into memory pointed to by storage
9.4.1.7.13.2. Pre-conditions
data_array is set up and not pointing to NULL
9.4.1.7.13.3. Post-conditions
See description

9.4.1.7.14. status clear_data_array()
9.4.1.7.14.1. Role
9.4.1.7.14.1.1. Operator Type
Modifier
9.4.1.7.14.1.2. Description
Sets all complex elements in data_array to 0+0j.
9.4.1.7.14.2. Pre-conditions
data_array is set up and not pointing to NULL.
9.4.1.7.14.3. Post-conditions
See description
9.4.1.7.14.4. Return Data
RETURN SUCCESS if pre-conditions met, else RETURN ERROR

9.4.1.8. Friend Member Functions
9.4.1.8.1. ostream& operator<<(ostream& output_stream, const data_set output_set)
9.4.1.8.1.1. Role
9.4.1.8.1.1.1. Operator Type
Overloaded output operator
9.4.1.8.1.1.2. Description
Output with labels, record_name, dimensions of data_array, field_names labelling columns of data_array and entire contents of data_array object to output_stream.
9.4.1.8.1.2. Pre-conditions
data_array is set up and not pointing to NULL

9.4.1.8.1.3. Post-conditions
If pre-conditions met, see description; else output error message
9.4.1.8.1.4. Return Data

```
        return output_stream
```

9.4.1.9. Static Member Functions

None

10. Header File: “set_input.h”

10.1. General Information

10.1.1. Header File Role

Declares set_input class

10.1.2. Standard Headers Required

None

10.1.3. Custom Headers Required

newstring.h, ulist.h, stringobject.h, define_vars.h

10.2. C-Type Definitions

None

10.3. Non-Class Function Prototypes

none

10.4. Class Definitions

10.4.1. Class “set_input”

10.4.1.1. Role:

Holds information required to load a column of a data_set data array with real numbers. The range of elements affected in the column described is from lower_index to upper_index. The numbers stored in these elements are set according to start_value and incrementer values.

10.4.1.2. Class Initialisation:

None

10.4.1.3. Private Data Members

10.4.1.3.1. int lower_index

First element to be affected by this set_input object

10.4.1.3.2. int upper_index

Last element to be affected by this set_input object

10.4.1.3.3. float start_value

Number to be stored in first element.

10.4.1.3.4. float incrementer

Quantity start_value is incremented by - as element number increases down the column of data_set data_array.

10.4.1.4. Public Data Members

None

10.4.1.5. Static Data Members

None

10.4.1.6. Private Member Functions

None

10.4.1.7. Public Member Functions

10.4.1.7.1. set_input()

10.4.1.7.1.1. Role

10.4.1.7.1.1.1. Operator Type

Foundation

10.4.1.7.1.1.2. Description

Default Constructor

10.4.1.7.1.2. Pre-conditions

None

10.4.1.7.1.3. Post-conditions

All data members initialsed to 0

10.4.1.7.2. set_input(int lower_i,int upper_i,float start_val,float inc)

10.4.1.7.2.1. Role

10.4.1.7.2.1.1. Operator Type

Foundation

10.4.1.7.2.1.2. Description

Parameterized Constructor

10.4.1.7.2.2. Pre-conditions

None

10.4.1.7.2.3. Post-conditions

data members set to values specified in function call

10.4.1.7.3. set_input(set_input &original)

10.4.1.7.3.1. Role

10.4.1.7.3.1.1. Operator Type

Foundation

10.4.1.7.3.1.2. Description

Copy Constructor

10.4.1.7.3.2. Pre-conditions

None

10.4.1.7.3.3. Post-conditions

All data members from original copied to *this

10.4.1.7.4. set_input& operator=(set_input &original)

10.4.1.7.4.1. Role

10.4.1.7.4.1.1. Operator Type

Overloaded assignment operator

10.4.1.7.4.1.2. Description

Allow assignment between set_input objects

10.4.1.7.4.2. Pre-conditions

None

10.4.1.7.4.3. Post-conditions

All data members from original copied to *this

10.4.1.7.4.4. Return Data

return *this

10.4.1.7.5. ~set_input()

10.4.1.7.5.1. Role

10.4.1.7.5.1.1. Operator Type

Foundation

10.4.1.7.5.1.2. Description

Destructor

10.4.1.7.5.2. Pre-conditions

None

10.4.1.7.5.3. Post-conditions

None - destructor not required to do anything as no dynamic data present

10.4.1.7.6. void set_parameters(int lower_i, int upper_i, float start_val, float inc)

10.4.1.7.6.1. Role

10.4.1.7.6.1.1. Operator Type

Modifier

10.4.1.7.6.1.2. Description

Initialise all data members

10.4.1.7.6.2. Pre-conditions

None

10.4.1.7.6.3. Post-conditions

data members set according to function parameter values

10.4.1.7.7. void

10.4.1.7.8. get_parameters(int &lower_i, int &upper_i, float &start_val, float &inc)

10.4.1.7.8.1. Role

10.4.1.7.8.1.1. Operator Type

Extractor

10.4.1.7.8.1.2. Description

Variable parameters of function set to values of respective data members

10.4.1.7.8.2. Pre-conditions

None

10.4.1.7.8.3. Post-conditions

See Description

10.4.1.8. Friend Member Functions

10.4.1.8.1. ostream& operator<<(ostream& output_stream, const set_input output_set)

10.4.1.8.1.1. Role

10.4.1.8.1.1.1. Operator Type

Overloaded output operator

10.4.1.8.1.1.2. Description

Output all data members with labels to output_stream

10.4.1.8.1.2. Pre-conditions

None

10.4.1.8.1.3. Post-conditions

See Description

10.4.1.8.1.4. Return Data

return output_stream

10.4.1.9. Static Member Functions

None

11. Header File: “IO_map.h”

11.1. General Information

11.1.1. Header File Role

Contains class definition for IO_map class.

11.1.2. Standard Headers Required

iostream.h

11.1.3. Custom Headers Required

ulist.h, stringobject.h, newstring.h, define_vars.h

11.2. C-Type Definitions

None

11.3. Non-Class Function Prototypes

none

11.4. Class Definitions

11.4.1. Class “IO_map”

11.4.1.1. Role:

Defines which fields in a data_set are used as input parameters for a calculator order, and which are used as output parameters.

11.4.1.2. Class Initialisation:

None

11.4.1.3. Private Data Members

11.4.1.3.1. int input_length

Used to store number of elements in input_fields array.

11.4.1.3.2. int output_length

Used to store number of elements in output_fields array.

11.4.1.3.3. string_class *input_fields

Array of field_names (ie variable names) which reference fields in a corresponding data set. The numeric data contained within these fields is used to set variables in a calculator’s variable list. The variables take their names from the corresponding field_names.

11.4.1.3.4. string_class *output_fields

Array of field_names (ie variable names) which reference fields in a corresponding data set. These variables are evaluated by the calculator and the evaluated results are stored in the named fields of data_set.

11.4.1.4. Public Data Members

None

11.4.1.5. Static Data Members

None

11.4.1.6. Private Member Functions

None

11.4.1.7. Public Member Functions

11.4.1.7.1. IO_map()

11.4.1.7.1.1. Role

11.4.1.7.1.1.1. Operator Type

Foundation

11.4.1.7.1.1.2. Description

Default Constructor

11.4.1.7.1.2. Pre-conditions

None

11.4.1.7.1.3. Post-conditions

output_fields & input_fields point to NULL; input_length & output_length=0

11.4.1.7.2. IO_map(ulist<string_object> in_fields, ulist<string_object> out_fields);

11.4.1.7.2.1. Role

11.4.1.7.2.1.1. Operator Type

Foundation

11.4.1.7.2.1.2. Description

Parameterized Constructor

11.4.1.7.2.2. Pre-conditions

None

11.4.1.7.2.3. Post-conditions

input_fields holds array copy of ulist in_fields; input_length holds length of input_fields array
output_fields holds array copy of ulist out_fields; output_length holds length of output_fields array

11.4.1.7.3. IO_map(IO_map &original)

11.4.1.7.3.1. Role

11.4.1.7.3.1.1. Operator Type

Foundation

11.4.1.7.3.1.2. Description

Copy Constructor

11.4.1.7.3.2. Pre-conditions

None

11.4.1.7.3.3. Post-conditions

See overloaded = operator post-conditions

11.4.1.7.4. IO_map& operator=(const IO_map &original)

11.4.1.7.4.1. Role

11.4.1.7.4.1.1. Operator Type

Overloaded = operator

11.4.1.7.4.1.2. Description

Allows assignment between IO_map objects

11.4.1.7.4.2. Pre-conditions

None

11.4.1.7.4.3. Post-conditions

All non-dynamic private data members copied from original to '*this'
Copy of input_fields array is stored in *this.input_fields
Copy of output_fields array is stored in *this.output_fields

11.4.1.7.4.4. Return Data

RETURN *this

11.4.1.7.5. ~IO_map()

11.4.1.7.5.1. Role

11.4.1.7.5.1.1. Operator Type

Foundation

11.4.1.7.5.1.2. Description

Destructor

11.4.1.7.5.2. Pre-conditions

None

11.4.1.7.5.3. Post-conditions

Input_fields array is deallocated.

11.4.1.7.6. int get_number_of_input_fields()

11.4.1.7.6.1. Role

11.4.1.7.6.1.1. Operator Type

Extractor

11.4.1.7.6.1.2. Description

Provide read-access to input_length data member

11.4.1.7.6.2. Pre-conditions

None

11.4.1.7.6.3. Post-conditions

None

11.4.1.7.6.4. Return Data

RETURN input_length

11.4.1.7.7. void get_input_fields(string_class *storage)

11.4.1.7.7.1. Role

11.4.1.7.7.1.1. Operator Type

Extractor

11.4.1.7.7.1.2. Description

Copies input_fields array into array pointed to by ‘storage’

11.4.1.7.7.2. Pre-conditions

Memory pointed to by storage is of sufficient size to hold entire array

11.4.1.7.7.3. Post-conditions

Storage points to array with ‘input_length’ elements, containing input_field strings.

11.4.1.7.8. int get_number_of_output_fields()

11.4.1.7.8.1. Role

11.4.1.7.8.1.1. Operator Type

Extractor

11.4.1.7.8.1.2. Description

Provide read-access to output_length data member

11.4.1.7.8.2. Pre-conditions

None

11.4.1.7.8.3. Post-conditions

None

11.4.1.7.8.4. Return Data

RETURN output_length

11.4.1.7.9. void get_output_fields(string_class *storage)

11.4.1.7.9.1. Role

11.4.1.7.9.1.1. Operator Type

Extractor

11.4.1.7.9.1.2. Description

Copies output_fields array into array pointed to by ‘storage’

11.4.1.7.9.2. Pre-conditions

Memory pointed to by storage is of sufficient size to hold entire array

11.4.1.7.9.3. Post-conditions

Storage points to array with ‘output_length’ elements, containing output_field strings.

11.4.1.8. Friend Member Functions

11.4.1.8.1. friend ostream& operator<<(ostream& output_stream, const IO_map output_map)

11.4.1.8.1.1. Role

11.4.1.8.1.1.1. Operator Type

Overloaded output operator

11.4.1.8.1.1.2. Description

Outputs output_map.input_fields and output_map.outputs_fields to output_stream with labels.

11.4.1.8.1.2. Pre-conditions

Input_fields must not be NULL

11.4.1.8.1.3. Post-conditions

See Description

11.4.1.8.1.4. Return Data

RETURN output_stream

11.4.1.9. Static Member Functions

None

12. Header File: “Graph Manager.h”

12.1. General Information

12.1.1. Header File Role

Declares graph_manager class

12.1.2. Standard Headers Required

None

12.1.3. Custom Headers Required

ulist.h, graph_spec_obj.h, newstring.h

12.2. C-Type Definitions

None

12.3. Non-Class Function Prototypes

none

12.4. Class Definitions

12.4.1. Class “graph_manager”

12.4.1.1. Role:

To maintain list of graph_spec_objects. A CLI is defined for allowing user interaction with the list. User may add and remove individual objects from the list, specifying object data when necessary. User may also dump contents of the list to output.

12.4.1.2. Class Initialisation:

None

12.4.1.3. Private Data Members

12.4.1.3.1. ulist<graph_spec_obj> spec_list

Each graph_spec_object in the list contains a unique identifying name and a graph_spec object.

12.4.1.4. Public Data Members

None

12.4.1.5. Static Data Members

None

12.4.1.6. Private Member Functions

None

12.4.1.7. Public Member Functions

12.4.1.7.1. void interface()

12.4.1.7.1.1. Role

12.4.1.7.1.1.1. Operator Type

CLI interface

12.4.1.7.1.1.2. Description

Allows user to interact with the graph spec list as described in the class role description, through the use of a CLI. In general, each user command corresponds to an equivalent public member function in this class.

12.4.1.7.1.2. Pre-conditions

None

12.4.1.7.1.3. Post-conditions

Lists will be updated in accordance to user’s commands where no errors have occurred.

12.4.1.7.2. void display_help()

12.4.1.7.2.1. Role

12.4.1.7.2.1.1. Operator Type

Information Provider

12.4.1.7.2.1.2. Description

Output list of user commands for this manager to standard output.

12.4.1.7.2.2. Pre-conditions

None

12.4.1.7.2.3. Post-conditions

See Description

12.4.1.7.3. status new_spec(string_class name, graph_spec new_spec)

12.4.1.7.3. status new_spec(string_class name, graph_spec new_spec)

12.4.1.7.3.1. Role

12.4.1.7.3.1.1. Operator Type

Modifier

12.4.1.7.3.1.2. Description

Adds new graph_spec_obj to spec_list using ‘name’ and ‘new_spec’ to build list node.

12.4.1.7.3.2. Pre-conditions

’name’ for graph_spec_obj is unique to spec_list

12.4.1.7.3.3. Post-conditions

See Description

12.4.1.7.3.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

12.4.1.7.4. status delete_spec(string_class name)

12.4.1.7.4.1. Role

12.4.1.7.4.1.1. Operator Type

Modifier

12.4.1.7.4.1.2. Description

Removes graph_spec_obj identified by ‘name’ from spec_list

12.4.1.7.4.2. Pre-conditions

Such a graph_spec_obj exists in spec_list

12.4.1.7.4.3. Post-conditions

See Description

12.4.1.7.4.4. Return Data

RETURN SUCCESS if pre-conditions met, else RETURN ERROR

12.4.1.7.5. void output_spec_list()

12.4.1.7.5.1. Role

12.4.1.7.5.1.1. Operator Type

Extractor

12.4.1.7.5.1.2. Description

Output contents of spec_list to standard output

12.4.1.7.5.2. Pre-conditions

None

12.4.1.7.5.3. Post-conditions

See Description

12.4.1.7.6. void reset_manager()

12.4.1.7.6.1. Role

12.4.1.7.6.1.1. Operator Type

Modifier

12.4.1.7.6.1.2. Description

Clears graph_spec list of all nodes.

12.4.1.7.6.2. Pre-conditions

None

12.4.1.7.6.3. Post-conditions

See Description

12.4.1.8. Friend Member Functions

None

12.4.1.9. Static Member Functions

None

13. Header File: “graph_spec.h”

13.1. General Information

13.1.1. Header File Role

Declare graph_spec class

13.1.2. Standard Headers Required

None

13.1.3. Custom Headers Required

extragraphclasses.h

13.2. C-Type Definitions

None

13.3. Non-Class Function Prototypes

none

13.4. Class Definitions

13.4.1. Class “graph_spec”

13.4.1.1. Role:

Specifies all attributes required for specifying a graphing window. Size of window, size of border around graph area, horizontal/vertical scales (including number of minor division tick marks between each scale number labelled major division tick mark), horizontal/vertical ranges (min, max values), size of minor and major tick marks on axes and point size of scale numbering text are stored.

13.4.1.2. Class Initialisation:

None

13.4.1.3. Private Data Members

None

13.4.1.4. Public Data Members

13.4.1.4.1. port_info port

Holds global screen co-ordinates of top and left pixels, and width and height in pixels of graph port window.

13.4.1.4.2. border_info border

Holds width of top,bottom,left and right blank borders measured from graph boundary to port window in pixels.

13.4.1.4.3. scale_info horiz_scale

Scale information for horizontal axis of graph

13.4.1.4.4. scale_info vert_scale

Scale information for vertical axis of graph

13.4.1.4.5. range_info horiz_range

Range information for horizontal axis of graph

13.4.1.4.6. range_info vert_range

Range information for vertical axis of graph

13.4.1.4.7. tick_info ticks

Pixel lengths for major and minor tick marks on both horizontal and vertical axes of graph.

13.4.1.4.8. int scaleSize

Text size in points for scale labelling.

13.4.1.5. Static Data Members

None

13.4.1.6. Private Member Functions

None

13.4.1.7. Public Member Functions

13.4.1.7.1. graph_spec()

13.4.1.7.1.1. Role

13.4.1.7.1.1.1. Operator Type

Foundation

13.4.1.7.1.1.2. Description

Default Constructor

13.4.1.7.1.2. Pre-conditions

None

13.4.1.7.1.3. Post-conditions

scaleSize set to 10, all other data members initialised to default values by respective constructors.

13.4.1.7.2. spec(port_info port1, border_info border1, scale_info horiz_scale1, scale_info vert_scale1, range_info horiz_range1, range_info vert_range1, tick_info ticks1)

13.4.1.7.2.1. Role

13.4.1.7.2.1.1. Operator Type

Foundation

13.4.1.7.2.1.2. Description

Parameterized constructor

13.4.1.7.2.2. Pre-conditions

None

13.4.1.7.2.3. Post-conditions

All data members set to function parameter values.

13.4.1.8. Friend Member Functions

13.4.1.8.1. ostream& operator<<(ostream &output_stream, graph_spec spec)

13.4.1.8.1.1. Role

13.4.1.8.1.1.1. Operator Type

Overloaded output operator

13.4.1.8.1.1.2. Description

Output data members in this object

13.4.1.8.1.2. Pre-conditions

None

13.4.1.8.1.3. Post-conditions

Output only horizontal and vertical range and scale data.

13.4.1.8.1.4. Return Data

return output_stream

13.4.1.8.2. istream& operator>>(istream& input_stream, graph_spec& spec)

13.4.1.8.2.1. Role

13.4.1.8.2.1.1. Operator Type

Overloaded input operator

13.4.1.8.2.1.2. Description

Read input_stream for horizontal and vertical range and scale data.

13.4.1.8.2.2. Pre-conditions

None

13.4.1.8.2.3. Post-conditions

horiz and vert scale and range data members set to values read from input_stream

13.4.1.8.2.4. Return Data

return input_stream

13.4.1.9. Static Member Functions

None

14. Header File: “extragraphclasses.h”

14.1. General Information

14.1.1. Header File Role

Declare port_info, border_info, bound_info, scale_info, range_info and tick_info classes

14.1.2. Standard Headers Required

iostream.h

14.1.3. Custom Headers Required

none

14.2. C-Type Definitions

None

14.3. Non-Class Function Prototypes

None

14.4. Class Definitions

14.4.1. Class “port_info”

14.4.1.1. Role:

Holds global pixel co-ordinates for window displaying a graph.

14.4.1.2. Class Initialisation:

None

14.4.1.3. Private Data Members

None

14.4.1.4. Public Data Members

14.4.1.4.1. double top

Global co-ordinate for top window pixel

14.4.1.4.2. double left

Global co-ordinate for left window pixel

14.4.1.4.3. double width

Width of window in pixels

14.4.1.4.4. double height

Height of window in pixels

14.4.1.5. Static Data Members

None

14.4.1.6. Private Member Functions

None

14.4.1.7. Public Member Functions

14.4.1.7.1. port_info (double WIDTH=500, double HEIGHT=500)

14.4.1.7.1.1. Role

14.4.1.7.1.1.1. Operator Type

Foundation

14.4.1.7.1.1.2. Description

Parameterized Constructor - defaulting width to 500, height to 500

14.4.1.7.1.2. Pre-conditions

None

14.4.1.7.1.3. Post-conditions

Top&Left co-ordinates set within bounds of graphics display.
If WIDTH and HEIGHT are specified, they set respective data members.
Else default WIDTH=500, HEIGHT=500.

14.4.1.7.2. port_info (double TOP, double LEFT, double WIDTH, double HEIGHT)

14.4.1.7.2.1. Role

14.4.1.7.2.1.1. Operator Type

Foundation

14.4.1.7.2.1.2. Description

Parameterized Constructor - no defaults

14.4.1.7.2.2. Pre-conditions

14.4.1.7.2.2. Pre-conditions
None
14.4.1.7.2.3. Post-conditions
Set data members according to function parameters

14.4.1.8. Friend Member Functions
14.4.1.8.1. ostream& operator<<(ostream& output_stream, port_info port)
14.4.1.8.1.1. Role
14.4.1.8.1.1.1. Operator Type
Overloaded output operator
14.4.1.8.1.1.2. Description
Output all data members with labels.
14.4.1.8.1.2. Pre-conditions
None
14.4.1.8.1.3. Post-conditions
See Description

14.4.1.9. Static Member Functions
None

14.4.2. Class "border_info"
14.4.2.1. Role:
Border width for top, bottom, left and right edges of graph. Width is measured in pixels from window edge to graph area.
14.4.2.2. Class Initialisation:
None
14.4.2.3. Private Data Members
None

14.4.2.4. Public Data Members
14.4.2.4.1. double top
graph border width along top edge of graph area
14.4.2.4.2. double bottom
graph border width along bottom edge of graph area
14.4.2.4.3. double left
graph border width along left edge of graph area
14.4.2.4.4. double right
graph border width along right edge of graph area

14.4.2.5. Static Data Members
None

14.4.2.6. Private Member Functions
None

14.4.2.7. Public Member Functions
14.4.2.7.1. border_info (double TOP=10, double BOTTOM=20, double LEFT=30, double RIGHT=10)
14.4.2.7.1.1. Role
14.4.2.7.1.1.1. Operator Type
Foundation
14.4.2.7.1.1.2. Description
Parameterized Constructor - defaults according to function prototype
14.4.2.7.1.2. Pre-conditions
None
14.4.2.7.1.3. Post-conditions
IF parameters specified, set appropriate data members to specified value.
ELSE set data members to default values.

14.4.2.8. Friend Member Functions

14.4.2.8.1. ostream& operator<<(ostream& output_stream, border_info border)

14.4.2.8.1.1. Role

14.4.2.8.1.1.1. Operator Type

Overloaded output operator

14.4.2.8.1.1.2. Description

output all data members with labels.

14.4.2.8.1.2. Pre-conditions

None

14.4.2.8.1.3. Post-conditions

See Description

14.4.2.9. Static Member Functions

None

14.4.3. Class "bound_info"

14.4.3.1. Role:

Global co-ordinates of graph area within graph window.

14.4.3.2. Class Initialisation:

None

14.4.3.3. Private Data Members

none

14.4.3.4. Public Data Members

14.4.3.4.1. double top

Top edge of graph

14.4.3.4.2. double bottom

Bottom edge of graph

14.4.3.4.3. double left

Left edge of graph

14.4.3.4.4. double right

Right edge of graph

14.4.3.4.5. double width

Width of graph area

14.4.3.4.6. double height

Height of graph area

14.4.3.5. Static Data Members

None

14.4.3.6. Private Member Functions

None

14.4.3.7. Public Member Functions

14.4.3.7.1. bound_info()

14.4.3.7.1.1. Role

14.4.3.7.1.1.1. Operator Type

Foundation

14.4.3.7.1.1.2. Description

Default Constructor

14.4.3.7.1.2. Pre-conditions

None

14.4.3.7.1.3. Post-conditions

Initilaise all data members to 0

14.4.3.8. Friend Member Functions

14.4.3.8.1. ostream& operator<<(ostream& output_stream, bound_info bound)

14.4.3.8.1.1. Role

14.4.3.8.1.1.1. Operator Type

Overloaded output operator

14.4.3.8.1.1.2. Description

output all data members with labels.

14.4.3.8.1.2. Pre-conditions

None

14.4.3.8.1.3. Post-conditions

See Description

14.4.3.8.1.4. Return Data

return output_stream

14.4.3.9. Static Member Functions

None

14.4.4. Class "scale_info"

14.4.4.1. Role:

Holds magnitude of step in axis value between numerical labels and number of divisions(ticks) between each major divison.

14.4.4.2. Class Initialisation:

None

14.4.4.3. Private Data Members

None

14.4.4.4. Public Data Members

14.4.4.4.1. double MajScale

Magnitude of difference in axis value between numerical labels.

14.4.4.4.2. double MinTicks

Number of tick marks between each major divison (not numerically labelled)

14.4.4.5. Static Data Members

None

14.4.4.6. Private Member Functions

None

14.4.4.7. Public Member Functions

14.4.4.7.1. scale_info (double MAJ=1, double MIN=5)

14.4.4.7.1.1. Role

14.4.4.7.1.1.1. Operator Type

Foundation

14.4.4.7.1.1.2. Description

Parameterized Consructor

14.4.4.7.1.2. Pre-conditions

None

14.4.4.7.1.3. Post-conditions

IF specifed, data members set to values passed to function.

ELSE default values stored in data members.

14.4.4.8. Friend Member Functions

14.4.4.8.1. ostream& operator<<(ostream& output_stream, scale_info scale)

14.4.4.8.1.1. Role

14.4.4.8.1.1.1. Operator Type

Overloaded output operator

14.4.4.8.1.1.2. Description

output all data members with labels.

14.4.4.8.1.2. Pre-conditions

None

14.4.4.8.1.3. Post-conditions

See Description

14.4.4.8.1.4. Return Data

return output_stream

14.4.4.8.2. istream& operator>>(istream& input_stream, scale_info& scale)

14.4.4.8.2.1. Role

14.4.4.8.2.1.1. Operator Type

Overloaded input operator

14.4.4.8.2.1.2. Description

Reads in both data members from input_stream

14.4.4.8.2.2. Pre-conditions

input_stream contains data

14.4.4.8.2.3. Post-conditions

Prompt user with “Scale Division? “
Read in MajScale data member
Prompt user with “Inter-Divisions?”
Read in scale.MinTicks

14.4.4.8.2.4. Return Data

return input_stream

14.4.4.9. Static Member Functions

None

14.4.5. Class “range_info”

14.4.5.1. Role:

Holds minimum and maximum values represented along a graph axis.

14.4.5.2. Class Initialisation:

None

14.4.5.3. Private Data Members

None

14.4.5.4. Public Data Members

14.4.5.4.1. double Max

Larger axis value

14.4.5.4.2. double Min

Smaller axis value

14.4.5.5. Static Data Members

None

14.4.5.6. Private Member Functions

None

14.4.5.7. Public Member Functions

14.4.5.7.1. range_info (double MIN=-5, double MAX=5)

14.4.5.7.1.1. Role

14.4.5.7.1.1.1. Operator Type

Foundation

14.4.5.7.1.1.2. Description

Parameterized Constructor

14.4.5.7.1.2. Pre-conditions

None

14.4.5.7.1.3. Post-conditions

Data members set if specified in function parameters, else set to default values.

14.4.5.7.2. range()

14.4.5.7.2.1. Role

14.4.5.7.2.1.1. Operator Type

Extractor

14.4.5.7.2.1.2. Description

Return result of max-min

14.4.5.7.2.2. Pre-conditions

None

14.4.5.7.2.3. Post-conditions

None

14.4.5.7.2.4. Return Data

return max-min

14.4.5.8. Friend Member Functions

14.4.5.8.1. ostream& operator<<(ostream& output_stream, range_info range)

14.4.5.8.1.1. Role

14.4.5.8.1.1.1. Operator Type

Overloaded output operator

14.4.5.8.1.1.2. Description

Output data members with labels

14.4.5.8.1.2. Pre-conditions

None

14.4.5.8.1.3. Post-conditions

See Description

14.4.5.8.2. istream& operator>>(istream& input_stream, range_info& range)

14.4.5.8.2.1. Role

14.4.5.8.2.1.1. Operator Type

Overloaded input operator

14.4.5.8.2.1.2. Description

Reads both data members from input_stream

14.4.5.8.2.2. Pre-conditions

input_stream contains data

14.4.5.8.2.3. Post-conditions

Prompt user with "Range Min? "
Read in min data member
Prompt user with "Range Max? "
Read in max data member

14.4.5.8.2.4. Return Data

return input_stream

14.4.5.9. Static Member Functions

None

14.4.6. Class "tick_info"

14.4.6.1. Role:

Holds length in pixels of a major tick/divison (placed at every numerically labelled position along an axis), and length in pixels of a minor tick/division (placed between major ticks).

14.4.6.2. Class Initialisation:

None

14.4.6.3. Private Data Members

none

14.4.6.4. Public Data Members

14.4.6.4.1. double MajTickSize

Length for major tick

14.4.6.4.2. double MinTickSize

Length for minor tick

14.4.6.5. Static Data Members

None

14.4.6.6. Private Member Functions

None

14.4.6.7. Public Member Functions

14.4.6.7.1. tick_info(double MAJ=5, double MIN=2)

14.4.6.7.1.1. Role

14.4.6.7.1.1.1. Operator Type

Foundation

14.4.6.7.1.1.2. Description

Parameterized Constructor

14.4.6.7.1.2. Pre-conditions

None

14.4.6.7.1.3. Post-conditions

Data members set to function parameters if set, else data members set to default values.

14.4.6.8. Friend Member Functions

14.4.6.8.1. ostream& operator<<(ostream& output_stream, tick_info tick)

14.4.6.8.1.1. Role

14.4.6.8.1.1.1. Operator Type

Overloaded output operator

14.4.6.8.1.1.2. Description

Output data members with labels

14.4.6.8.1.2. Pre-conditions

None

14.4.6.8.1.3. Post-conditions

See Description

14.4.6.8.1.4. Return Data

return output_stream

14.4.6.9. Static Member Functions

none

15. Header File: “graph_device.h”

15.1. General Information

15.1.1. Header File Role

Declares graph_device class

15.1.2. Standard Headers Required

stdio.h

15.1.3. Custom Headers Required

extragraphclasses.h

15.2. C-Type Definitions

None

15.3. Non-Class Function Prototypes

None

15.4. Class Definitions

15.4.1. Class “graph_device”

15.4.1.1. Role:

Hold graph_spec data members (separately) for current graph. Handles acquistion of graphing window pointer from Mac OS. Draws all axes and scales for graphs - translates (x,y) graph co-ordinates to graph port window co-ordinates. Provides functions concerning the graphing window.

15.4.1.2. Class Initialisation:

None

15.4.1.3. Private Data Members

15.4.1.3.1. port_info port

15.4.1.3.2. border_info border

15.4.1.3.3. bound_info bound

15.4.1.3.4. scale_info horiz_scale

15.4.1.3.5. scale_info vert_scale

15.4.1.3.6. range_info horiz_range

15.4.1.3.7. range_info vert_range

15.4.1.3.8. tick_info ticks

15.4.1.3.9. int scaleSize

15.4.1.3.9.1. Role

See identical data members in graph_spec class.

15.4.1.3.10. WindowPtr mainPtr

Pointer to graphing window - used to tell OS which window to plot to.

15.4.1.4. Public Data Members

None

15.4.1.5. Static Data Members

None

15.4.1.6. Private Member Functions

None

15.4.1.7. Public Member Functions

15.4.1.7.1. graph_device()

15.4.1.7.1.1. Role

15.4.1.7.1.1.1. Operator Type

Foundation

15.4.1.7.1.1.2. Description

Default Constructor

15.4.1.7.1.2. Pre-conditions

None

15.4.1.7.1.3. Post-conditions

mainPtr set to NULL, scaleSize set to 10

All other data members automatically initialised to default values by respective constructor functions.

15.4.1.7.2. graph_device(port_info newport, border_info newborder, scale_info newhoriz_scale, scale_info newvert_scale, range_info newhoriz_range, range_info newvert_range, tick_info newticks)

15.4.1.7.2.1. Role

15.4.1.7.2.1.1. Operator Type

Foundation

15.4.1.7.2.1.2. Description

Parameterized Constructor

15.4.1.7.2.2. Pre-conditions

None

15.4.1.7.2.3. Post-conditions

mainPtr set to NULL, scaleSize set to 10

Function parameter values used to set respective data members.

15.4.1.7.3. void set_params(scale_info newhoriz_scale, scale_info newvert_scale, range_info newhoriz_range, range_info newvert_range)

15.4.1.7.3.1. Role

15.4.1.7.3.1.1. Operator Type

Modifier

15.4.1.7.3.1.2. Description

Set appropriate data members to values provided in respective function parameters.

15.4.1.7.3.2. Pre-conditions

None

15.4.1.7.3.3. Post-conditions

See Description

15.4.1.7.4. void showGraph()

15.4.1.7.4.1. Role

15.4.1.7.4.1.1. Operator Type

Operating System Modifier

15.4.1.7.4.1.2. Description

Hi-lites and selects graphing window - brings window to front.

15.4.1.7.4.2. Pre-conditions

mainPtr points to valid window

15.4.1.7.4.3. Post-conditions

See Description

15.4.1.7.5. void clear_window()

15.4.1.7.5.1. Role

15.4.1.7.5.1.1. Operator Type

Operating System Modifier - draw to screen

15.4.1.7.5.1.2. Description

Clears graphing window.

15.4.1.7.5.2. Pre-conditions

mainPtr points to valid window, all data members are set.

15.4.1.7.5.3. Post-conditions

See Description

15.4.1.7.6. void blank_graph()

15.4.1.7.6.1. Role

15.4.1.7.6.1.1. Operator Type

Operating System Modifier - draw to screen

15.4.1.7.6.1.2. Description

Clears graphing window, then draws horizontal / vertical axes/ticks/scales and other graph artefacts in graphing window. Prepares graph window for plotting.

15.4.1.7.6.2. Pre-conditions

mainPtr points to valid window, all data members are set.

15.4.1.7.6.3. Post-conditions

See Description

15.4.1.7.7. double translate_x(double x)

15.4.1.7.7.1. Role

15.4.1.7.7.1.1. Operator Type

Calculation based on data member values

15.4.1.7.7.1.2. Description

Translates an x-axis graph co-ordinate (in range specified by horiz_range) to the corresponding distance across the graphing window in pixels.

15.4.1.7.7.2. Pre-conditions

mainPtr points to valid window, all data members are set.

15.4.1.7.7.3. Post-conditions

None

15.4.1.7.7.4. Return Data

Result of translation

15.4.1.7.8. double translate_y(double y)

15.4.1.7.8.1. Role

15.4.1.7.8.1.1. Operator Type

Calculation based on data member values

15.4.1.7.8.1.2. Description

Translates a y-axis graph co-ordinate (in range specified by vert_range) to the corresponding distance down the graphing window in pixels.

15.4.1.7.8.2. Pre-conditions

mainPtr points to valid window, all data members are set.

Post-conditions

None

15.4.1.7.8.3. Return Data

Result of translation

15.4.1.7.9. void draw_x_line(double x, long int brightness)

15.4.1.7.9.1. Role

15.4.1.7.9.1.1. Operator Type

Operating System Modifier - draw to screen

15.4.1.7.9.1.2. Description

Draw a line down the graphing window at the location on the graph (within horiz_range) indicated by x.

15.4.1.7.9.2. Pre-conditions

mainPtr points to valid window, all data members are set.

Current drawing port is graphing window.

15.4.1.7.9.3. Post-conditions

See Description.

15.4.1.7.10. void draw_y_line(double y, long int brightness)

15.4.1.7.10.1. Role

15.4.1.7.10.1.1. Operator Type

Operating System Modifier - draw to screen

15.4.1.7.10.1.2. Description

Draw a line across the graphing window at the location on the graph (within vert_range) indicated by y.

15.4.1.7.10.2. Pre-conditions

mainPtr points to valid window, all data members are set.

Current drawing port is graphing window.

15.4.1.7.10.3. Post-conditions

See Description.

15.4.1.7.11. void draw_axes()

15.4.1.7.11.1. Role

15.4.1.7.11.1.1. Operator Type

Operating System Modifier - draw to screen

15.4.1.7.11.1.2. Description

Draw both sets of axes, tick divisions and scale numberings in graphing window.

15.4.1.7.11.2. Pre-conditions

mainPtr points to valid window, all data members are set.
Current drawing port is graphing window.

15.4.1.7.11.3. Post-conditions

See Description

15.4.1.7.12. void vertical_ticks()

15.4.1.7.12.1. Role

15.4.1.7.12.1.1. Operator Type

Operating System Modifier - draw to screen

15.4.1.7.12.1.2. Description

Draw minor and major tick marks with scale numbering along vertical axis of graph.

15.4.1.7.12.2. Pre-conditions

mainPtr points to valid window, all data members are set.
Current drawing port is graphing window.

15.4.1.7.12.3. Post-conditions

See Description

15.4.1.7.13. void horizontal_ticks()

15.4.1.7.13.1. Role

15.4.1.7.13.1.1. Operator Type

Operating System Modifier - draw to screen

15.4.1.7.13.1.2. Description

Draw minor and major tick marks with scale numbering along horizontal axis of graph.

15.4.1.7.13.2. Pre-conditions

mainPtr points to valid window, all data members are set.
Current drawing port is graphing window.

15.4.1.7.13.3. Post-conditions

See Description

15.4.1.7.14. void PlaceCross(int Width)

15.4.1.7.14.1. Role

15.4.1.7.14.1.1. Operator Type

Operating System Modifier - draw to screen

15.4.1.7.14.1.2. Description

Place a cross of size ‘Width’ pixels at the current pen position in graphing window.

15.4.1.7.14.2. Pre-conditions

mainPtr points to valid window, all data members are set.
Current drawing port is graphing window.

15.4.1.7.14.3. Post-conditions

See Description

15.4.1.7.15. void set_port(port_info newport)

15.4.1.7.15.1. Role

15.4.1.7.15.1.1. Operator Type

Modifier

15.4.1.7.15.1.2. Description

Change size of graphing window.

15.4.1.7.15.2. Pre-conditions

border is set

15.4.1.7.15.3. Post-conditions

Set port value to newport, reset bound values to compensate for any change in port values.
mainPtr points to new window pointer.

15.4.1.7.16. void set_border(border_info newborder)

15.4.1.7.16.1. Role

15.4.1.7.16.1.1. Operator Type

Modifier

15.4.1.7.16.1.2. Description

Set border values

15.4.1.7.16.2. Pre-conditions

port is set

15.4.1.7.16.3. Post-conditions

border updated to newborder.
bound updated to take account of borders.

15.4.1.7.17. void set_horiz_scales(scale_info newhoriz_scale)
15.4.1.7.18. void set_vert_scales(scale_info newvert_scale)
15.4.1.7.19. void set_horiz_range(range_info newhoriz_range)
15.4.1.7.20. void set_vert_range(range_info newvert_range)
15.4.1.7.21. void set_ticks(tick_info newticks)
All the above set individual data members accordingly. No pre-conditions

15.4.1.7.22. double x_origin(double x)
15.4.1.7.22.1. Role
15.4.1.7.22.1.1. Operator Type
Calculation based on data member values
15.4.1.7.22.1.2. Description
Return x-axis pixel location referencing ‘x’ pixels to the right of bottom-left corner of graph area.
15.4.1.7.22.2. Pre-conditions
bound data member is set
15.4.1.7.22.3. Post-conditions
None
15.4.1.7.22.4. Return Data
See Description

15.4.1.7.23. double y_origin(double y)
15.4.1.7.23.1. Role
15.4.1.7.23.1.1. Operator Type
Calculation based on data member values
15.4.1.7.23.1.2. Description
Return y-axis pixel location referencing ‘y’ pixels to above bottom-left corner of graph area.
15.4.1.7.23.2. Pre-conditions
bound data member is set
15.4.1.7.23.3. Post-conditions
None
15.4.1.7.23.4. Return Data
See Description

15.4.1.8. Friend Member Functions
None

15.4.1.9. Static Member Functions
None

16. Header File: “validator.h”

16.1. General Information

16.1.1. Header File Role

Declare validator class

16.1.2. Standard Headers Required

None

16.1.3. Custom Headers Required

define_vars.h, calc_preprocessor.h, rCalculatorClass.h, newstring.h, ulist.h, stringobject.h

16.2. C-Type Definitions

Forward Decalaration of calculator class - pointer to calculator object required.

16.3. Non-Class Function Prototypes

none

16.4. Class Definitions

16.4.1. Class “validator”

16.4.1.1. Role:

Checks equations stored in connected calculator for circular definitions. If such an error is found, an error trace indicating the equations at fault is recorded.

16.4.1.2. Class Initialisation:

None

16.4.1.3. Private Data Members

16.4.1.3.1. calculator *connected_calculator

Pointer to the calculator object holding the list of equations we wish to verify.

16.4.1.3.2. calc_preprocessor *connected_preprocessor

Pointer to a preprocessor object used to get a list of all unidentified user names (ie a list of variable and equation names which may cause circular definitions) for equation being checked.

16.4.1.4. Public Data Members

16.4.1.4.1. string_class error_trace

If an error occurred on the last call to validate, the error trace causing the circular definition is stored as a list of single space separated user names here. The error trace is empty “” if no error has occurred.

16.4.1.5. Static Data Members

None

16.4.1.6. Private Member Functions

16.4.1.6.1. status verify(const string_class name_string, string_class dependents="")

16.4.1.6.1.1. Role

16.4.1.6.1.1.1. Operator Type

Recursive

16.4.1.6.1.1.2. Description

Verifies that the expression stored in name_string contains no instances of the user names (separated by a single space) in dependents string. If dependents string is not specifed, there are assumed to be no dependents.

16.4.1.6.1.2. Pre-conditions

Same as for validator public member function.

16.4.1.6.1.3. Post-conditions

Same as for validator public member function.

16.4.1.6.1.4. Return Data

RETURN SUCCESS if no errors found, else RETURN ERROR.

16.4.1.7. Public Member Functions

16.4.1.7.1. validator(calc_preprocessor *preprocessor)

16.4.1.7.1.1. Role

16.4.1.7.1.1.1. Operator Type

Foundation

16.4.1.7.1.1.2. Description

Default Constructor

16.4.1.7.1.2. Pre-conditions

*preprocessor points to a calc_preprocessor object that will exist for the lifetime of the validator object and which contains correlator tables applicable to the connected calculator - ie they have compatible token tables.

16.4.1.7.1.3. Post-conditions

connected_preprocessor=preprocessor

16.4.1.7.2. status validate(calculator *connect_calculator)

16.4.1.7.2.1. Role

16.4.1.7.2.1.1. Operator Type

Primary Service

16.4.1.7.2.1.2. Description

Verifies every equation stored in the equation list of the connected calculator against itself and all other equations in the equation list, until an error is found or all equations checked.

16.4.1.7.2.2. Pre-conditions

connect_calculator and connected_preprocessor point to objects which are currently in existence and have compatible token tables.

16.4.1.7.2.3. Post-conditions

No errors found - error trace is empty.

Errors found - error trace is updated.

16.4.1.7.2.4. Return Data

RETURN SUCCESS if no errors found, else RETURN ERROR.

16.4.1.8. Friend Member Functions

None

16.4.1.9. Static Member Functions

None

17. Header File: “preprocessorTypes.h”

17.1. General Information

17.1.1. Header File Role

Declare user_label class

17.1.2. Standard Headers Required

None

17.1.3. Custom Headers Required

newstring.h

17.2. C-Type Definitions

None

17.3. Non-Class Function Prototypes

none

17.4. Class Definitions

17.4.1. Class “user_label”

17.4.1.1. Role:

Defines the mapping between a user function name (typed on the keyboard) with an OPERATION name. cf calculator_symbol which maps OPERATION to calculator token character.

17.4.1.2. Class Initialisation:

None

17.4.1.3. Private Data Members

None

17.4.1.4. Public Data Members

17.4.1.4.1. string_class input_string

Stores the string to be recognised from user input. This is the user function name.

17.4.1.4.2. string_class calc_string

The corresponding OPERATION string name for the user function name. OPERATION strings are known to both preprocessor and calculator - they are used to couple a user function name to a calculator token character.

17.4.1.5. Static Data Members

None

17.4.1.6. Private Member Functions

None

17.4.1.7. Public Member Functions

17.4.1.7.1. user_label()

17.4.1.7.1.1. Role

17.4.1.7.1.1.1. Operator Type

Foundation

17.4.1.7.1.1.2. Description

Default Constructor

17.4.1.7.1.2. Pre-conditions

None

17.4.1.7.1.3. Post-conditions

Both string data members initialised to “” (empty string)

17.4.1.7.2. user_label(string_class i_string, string_class c_string)

17.4.1.7.2.1. Role

17.4.1.7.2.1.1. Operator Type

Foundation

17.4.1.7.2.1.2. Description

Parameterized Constructor

17.4.1.7.2.2. Pre-conditions

None

17.4.1.7.2.3. Post-conditions

17.4.1.7.2.3. Post-conditions

input_string=i_string
calc_string=c_string

17.4.1.8. Friend Member Functions

17.4.1.8.1. ostream& operator<<(ostream& output_stream, const user_label label)

17.4.1.8.1.1. Role

17.4.1.8.1.1.1. Operator Type

Overloaded Output Operator

17.4.1.8.1.1.2. Description

Output contents of user_label object to output stream.

17.4.1.8.1.2. Pre-conditions

None

17.4.1.8.1.3. Post-conditions

Output label.input_string followed by "->" followed by label.calc_string to output_stream

17.4.1.8.1.4. Return Data

RETURN output_stream

17.4.1.9. Static Member Functions

None

18. Header File: “calc_preprocessor.h”

18.1. General Information

18.1.1. Header File Role

Declare calc_preprocesor class.

18.1.2. Standard Headers Required

iostream.h

18.1.3. Custom Headers Required

rCalculatorUserTypes.h, newstring.h, rCalculatorClass.h, preprocessorTypes.h, ulist.h, stringobject.h

18.2. C-Type Definitions

None

18.3. Non-Class Function Prototypes

none

18.4. Class Definitions

18.4.1. Class “calc_preprocessor”

18.4.1.1. Role:

To preprocess an expression for evaluation by a calculator object involving the substitution of calculator token characters in place of user function names. Additional service - to postprocess an expression containing calculator token characters by expanding tokens back to original user function names.

18.4.1.2. Class Initialisation:

None

18.4.1.3. Private Data Members

18.4.1.3.1. token_name *correlator

Points to dynamically allocated array look-up table (ORDER N) which performs the preprocessing function.

18.4.1.3.2. string_class inverse_correlator[256]

Points to dynamically allocated array look-up table (ORDER 1) which performs the postprocessing function.

18.4.1.3.3. int array_length

Length of correlator array.

18.4.1.3.4. istrstream *input_stream

Used for extracting characters from user input stored in input_char_array.

18.4.1.3.5. char *input_char_array

Stores user input expression string - contains user function names.

18.4.1.4. Public Data Members

None

18.4.1.5. Static Data Members

None

18.4.1.6. Private Member Functions

18.4.1.6.1. status set_input(string_class input_string)

18.4.1.6.1.1. Role

18.4.1.6.1.1.1. Operator Type

Modifier

18.4.1.6.1.1.2. Description

Delete old input_stream (and input_char_array) and create new input_stream that points to the first character of a copy of the char array stored in ‘input_string’.

18.4.1.6.1.2. Pre-conditions

None

18.4.1.6.1.3. Post-conditions

input_stream and input_char_array point to newly allocated memory blocks. Input_char array points to a copy of the char array stored in ‘input_string’.

18.4.1.6.1.4. Return Data

RETURN SUCCESS

RETURN SUCCESS

18.4.1.6.2. status reset_input()

18.4.1.6.2.1. Role

18.4.1.6.2.1.1. Operator Type

Modifier

18.4.1.6.2.1.2. Description

Delete input_stream and input_char_array objects.

18.4.1.6.2.2. Pre-conditions

input_stream and input_char_array are previously allocated and do not point to NULL.

18.4.1.6.2.3. Post-conditions

input_stream and input_char_array point to NULL

18.4.1.6.2.4. Return Data

RETURN SUCCESS if pre-conditions met,
else RETURN ERROR

18.4.1.7. Public Member Functions

18.4.1.7.1. calc_preprocessor(const user_label *input_labels, const token_name *token_mappings, const int array_length)

18.4.1.7.1.1. Role

18.4.1.7.1.1.1. Operator Type

Foundation

18.4.1.7.1.1.2. Description

Parameterized Constructor - builds correlator and inverse_correlator look-up tables by matching each input_mapping with its corresponding token_mapping, taken from the two arrays in the function parameter list.

18.4.1.7.1.2. Pre-conditions

None

18.4.1.7.1.3. Post-conditions

Input_stream/input_char_array initialised to zero.
Correlator table built - each entry consists of a user_name in the .name field and a calculator character token in the .token field. (Order N search)
Inverse Correlator table built - Order 1 search - this is a sparse array indexed from 0 -255. Index is equivalent to token character being looked-up. Data at each location is the user_name for that character token.

18.4.1.7.2. ~calc_preprocessor()

18.4.1.7.2.1. Role

18.4.1.7.2.1.1. Operator Type

Foundation

18.4.1.7.2.1.2. Description

Destructor

18.4.1.7.2.2. Pre-conditions

None

18.4.1.7.2.3. Post-conditions

Delete correlator array

18.4.1.7.3. void preprocess(string_class input_string, string_class &output_string)

18.4.1.7.3.1. Role

18.4.1.7.3.1.1. Operator Type

Primary Service

18.4.1.7.3.1.2. Description

Accepts an input_string containing user name functions - contracts all user names to equivalent calculator tokens leaving remainder of string intact - stores result in output_string variable function parameter.

18.4.1.7.3.2. Pre-conditions

None

18.4.1.7.3.3. Post-conditions

None - except output_string now holds the preprocessed version of input_string.

18.4.1.7.4. void preprocess(string_class input_string, string_class &output_string, ulist<string_object>

*unidentified)

18.4.1.7.4.1. Role

18.4.1.7.4.1.1. Operator Type

Primary Service

18.4.1.7.4.1.2. Description

Same as other ‘preprocess’ public member function - additionally a list of unmatched custom user names is returned by loading the *unidentified list with non-matched names. These names correspond to variable & equation names defined by the user - they are left untouched by the preprocessing operation.

18.4.1.7.4.2. Pre-conditions

*unidentifier must point to a valid ulist<string_object> object.

18.4.1.7.4.3. Post-conditions

None - except output_string now holds the preprocessed version of input_string.

18.4.1.7.5. void postprocess(const string_class &input_string, string_class &postprocessed_string)

18.4.1.7.5.1. Role

18.4.1.7.5.1.1. Operator Type

Primary Service

18.4.1.7.5.1.2. Description

Expands all calculator tokens found in input_string, in situ, to their equivalent user names. Resulting string returned by storing in postprecessed_string.

18.4.1.7.5.2. Pre-conditions

None

18.4.1.7.5.3. Post-conditions

None - except postprocessed_string now holds the postprocessed version of input_string

18.4.1.8. Friend Member Functions

None

18.4.1.9. Static Member Functions

None

19. Header File: “ulist.h”

19.1. General Information

19.1.1. Header File Role

Contains class definition for ulist class.

19.1.2. Standard Headers Required

none

19.1.3. Custom Headers Required

define_vars.h

19.2. C-Type Definitions

none

19.3. Non-Class Function Prototypes

none

19.4. Class Definitions

19.4.1. Class “ulist<node>”

19.4.1.1. Role:

A templated list container class. Facilities for maintaining a list of unique nodes of any type (containing a base configuration), automated ordering of nodes using either INDEX or DATA information within each node when adding or changing nodes in list, removal of node from list based upon INDEX or DATA information and external traversal of list. A facility for appending nodes to the end of a list without regard for ordering is available.

19.4.1.2. Class Initialisation:

none

19.4.1.3. Private Data Members

19.4.1.3.1. node *head

Pointer to the first node in the list - if list is empty then head==NULL.

19.4.1.3.2. node *tail

Pointer to the last node in the list - if list is empty then tail==NULL.

19.4.1.3.3. node *current

Pointer used to navigate list by member functions.

19.4.1.3.4. node *transverse

Pointer which is manipulated externally from class by using transverse member functions.

19.4.1.4. Public Data Members

none

19.4.1.5. Static Data Members

none

19.4.1.6. Private Member Functions

19.4.1.6.1. node* search(node searchitem, ordering method)

19.4.1.6.1.1. Role

19.4.1.6.1.1.1. Operator Type

Internal Utility

19.4.1.6.1.1.2. Description

The search is conducted on either the index or data value of searchitem - NOT BOTH. This search criteria is determined by 'method'. If method = COUNTER, an index check is made. If method = DATA, a data check is made. The node found will be the FIRST match found, searching from the start of the list. Comparisons between index values or data values are done using the compare_index and compare_data functions present in the node object.

If the node is found, current is set to point to the required node. Else, current is set to NULL. A pointer to the node previous to the required node is returned. If the required node is not found, previous points to NULL.

19.4.1.6.1.2. Pre-conditions

none

19.4.1.6.1.3. Post-conditions

If node found:

previous->get_pointer() = Current

(previous=NULL if current at head of list)

Current = &searchitem

Current = &searchitem

Else

previous = NULL = CURRENT

19.4.1.6.1.4. Return Data

value of ‘previous’

19.4.1.6.2. void find_neighbours(node* target, node* &preceding, node* &proceding, ordering comparisontype, orderproperty direction)

19.4.1.6.2.1. Role

19.4.1.6.2.1.1. Operator Type

Internal Utility

19.4.1.6.2.1.2. Description

The search may be on index or data value, specified by ‘comparisontype’ (cf SEARCH FUNCTION). The orderproperty 'direction' tells the function whether the search should assume an ASCENDING or DESCENDING order method. The function looks for an insertion point for the target node. The insertion point is between preceding and proceding pointed nodes, and is returned by reference through these function parameters.

19.4.1.6.2.2. Pre-conditions

none

19.4.1.6.2.3. Post-conditions

Preceding and proceding points to the two nodes which target must be inserted between. A NULL pointer for either of these indicates insertion at head/tail respectively.

19.4.1.7. Public Member Functions

19.4.1.7.1. ulist()

19.4.1.7.1.1. Role

19.4.1.7.1.1.1. Operator Type

foundation

19.4.1.7.1.1.2. Description

default constructor

19.4.1.7.1.2. Pre-conditions

none

19.4.1.7.1.3. Post-conditions

head=tail=current=transverse == NULL

19.4.1.7.2. ulist(ulist &original)

19.4.1.7.2.1. Role

19.4.1.7.2.1.1. Operator Type

foundation

19.4.1.7.2.1.2. Description

Copy Constructor

19.4.1.7.2.2. Pre-conditions

none

19.4.1.7.2.3. Post-conditions

copy of original list pointed to by 'this'

19.4.1.7.3. void clearlist(void)

19.4.1.7.3.1. Role

19.4.1.7.3.1.1. Operator Type

Reset List

19.4.1.7.3.1.2. Description

Remove all nodes from ulist

19.4.1.7.3.2. Pre-conditions

none

19.4.1.7.3.3. Post-conditions

List empty

19.4.1.7.4. ~ulist(void)

19.4.1.7.4.1. Role

19.4.1.7.4.1.1. Operator Type

foundation

19.4.1.7.4.1.2. Description

Destructor

19.4.1.7.4.2. Pre-conditions

none

19.4.1.7.4.3. Post-conditions

All nodes in list deleted, all memory deallocated.

19.4.1.7.5. status add(node newitem, ordering order, orderproperty direction)

19.4.1.7.5.1. Role

19.4.1.7.5.1.1. Operator Type

19.4.1.7.5.1.2. Description

This function adds a copy of newitem to the list. The insertion position is determined using: (index/data values below are taken from newitem node)

1) ordering order : order=COUNTER - Item is added according to index value
   order=DATA - Item is added according to data value
2) orderproperty direction :
   direction=ASCENDING - ordered item increases down list
   direction=DESCENDING - ordered item decreases down list

The addition occurs in such a position as to keep the ordering of the list consistent with the above conditions. Addition is not made if a node already exists in the list with ‘order’ value equal to that in ‘newitem’. In this case ‘newitem’ has been found to be non-unique within the ulist.

19.4.1.7.5.2. Pre-conditions

Newitem contains valid data and index values.

19.4.1.7.5.3. Post-conditions

If node is unique (see above), node is added in position according to above ordering criteria.
Otherwise, when non-unique addition is attempted, no change is made to the list.

19.4.1.7.5.4. Return Data

RETURN ERROR if out of memory, or ‘newitem’ is non-unique within the list.
RETURN SUCCESS otherwise.

19.4.1.7.6. status remove(node item, ordering method)

19.4.1.7.6.1. Role

19.4.1.7.6.1.1. Operator Type

19.4.1.7.6.1.2. Description

This function will search for item node in the list and remove the item from the list. It will match either the index or the data item when applying the search algorithm. The search method is determined by the 'ordering method' :
method=COUNTER -> search on index for node to remove
method=DATA -> search on data for node to remove

19.4.1.7.6.2. Pre-conditions

none

19.4.1.7.6.3. Post-conditions

If removeitem is in the list, remove the first occurrence of it.
Otherwise, make no change to the list.

19.4.1.7.6.4. Return Data

RETURN SUCCESS if node successfully removed
RETURN ERROR if node not found

19.4.1.7.7. status update(node reference, node new_info, ordering update_field, ordering structure_order, orderproperty direction)

19.4.1.7.7.1. Role

19.4.1.7.7.1.1. Operator Type

19.4.1.7.7.1.2. Description

This function updates a node in ulist with new_info. The node for updating is found by searching on the COUNTER or DATA field of 'reference'. The found node is referred to as the 'working node'. The field for searching on is stored in 'structure_order'. The field that we want

to update is specified in 'update_field'. This field is copied across from 'reference' to the 'working node'. If the resulting 'working node' corrupts ulist ordering, specified by 'structure_order' (COUNTER/DATA) and 'direction' (ASCENDING/DESCENDING) - it is moved to the correct position in the ulist. The list is assumed to be unique on 'structure_order' field. If an update is performed which breaks the unique property of the list, the update request is refused and ERROR returned.

19.4.1.7.7.2. Pre-conditions

None

19.4.1.7.7.3. Post-conditions

List has been updated in accordance with the description above, if no error has occurred.
Otherwise, no change is made to the list.

19.4.1.7.7.4. Return Data

RETURN ERROR : if empty list, invalid enum values are used or update breaks unique property of ulist.
RETURN SUCCESS : all other instances, when update has been successful.

19.4.1.7.8. status maintain(node new_info, ordering order_method, orderproperty direction)

19.4.1.7.8.1. Role

19.4.1.7.8.1.1. Operator Type

19.4.1.7.8.1.2. Description

This function checks to see if a node with index==new_info.index or data=new_info.data (depending upon value of structure_order (COUNTER/DATA)) is present in the ulist. If it is, then this node has its member that IS NOT SPECIFIED by structure_order, updated with the appropriate value in new_info node. If the node is not present in the list, newinfo node is added to the ulist. Unique list is maintained, ordering is maintained.

19.4.1.7.8.2. Pre-conditions

None

19.4.1.7.8.3. Post-conditions

Information from Node ‘new_info’ has been placed in ulist, if no error occurs.
Otherwise, ulist is left unchanged.

19.4.1.7.8.4. Return Data

RETURN ERROR: if update fails, add fails, or invalid enum values are given.

19.4.1.7.9. status add_to_end(node newitem)

19.4.1.7.9.1. Role

19.4.1.7.9.1.1. Operator Type

19.4.1.7.9.1.2. Description

This function appends a copy of newitem node to the end of the ulist. If the index of newitem has not been set the index of the new node is derived by calling set_to_next_index on the index of the last node of the ulist.

19.4.1.7.9.2. Pre-conditions

Memory is not full.

19.4.1.7.9.3. Post-conditions

Node is appended to end of list. If its index is not defined, it is set to the value following that of the previous node in the list.

19.4.1.7.9.4. Return Data

RETURN ERROR if memory full.

19.4.1.7.10. status search_node(node &search_item, ordering method)

19.4.1.7.10.1. Role

19.4.1.7.10.1.1. Operator Type

19.4.1.7.10.1.2. Description

This function searches the ulist for a node with ordering ‘method’ field matching the corresponding field in ‘search_item’.

19.4.1.7.10.2. Pre-conditions

None

19.4.1.7.10.3. Post-conditions

If a matching node is found in the ulist, its contents is copied into ‘search_item’ and returned back to the calling function.
Otherwise, no change is made to ‘search_item’.

19.4.1.7.10.4. Return Data

RETURN ERROR: if matching node not found

RETURN SUCCESS: if matching node found

19.4.1.7.11. status reset_transverse()

19.4.1.7.11.1. Role

19.4.1.7.11.1.1. Operator Type

19.4.1.7.11.1.2. Description

This function sets transverse pointer to the head of the list.

19.4.1.7.11.2. Pre-conditions

List is not empty

19.4.1.7.11.3. Post-conditions

Transverse points to the head of the list

19.4.1.7.11.4. Return Data

RETURN SUCCESS if list is not empty
RETURN ERROR if list is empty.

19.4.1.7.12. status get_transverse(node &gotitem)

19.4.1.7.12.1. Role

19.4.1.7.12.1.1. Operator Type

19.4.1.7.12.1.2. Description

Returns copy of node pointed to by transverse

19.4.1.7.12.2. Pre-conditions

list is not empty

19.4.1.7.12.3. Post-conditions

node pointed to by transverse is copied to gotitem

19.4.1.7.12.4. Return Data

RETURN SUCCESS if gotitem has been copied to
RETURN ERROR if list is empty

19.4.1.7.13. status progress_transverse()

19.4.1.7.13.1. Role

19.4.1.7.13.1.1. Operator Type

19.4.1.7.13.1.2. Description

This function will move transverse to point to the next node in the list, if one exists.

19.4.1.7.13.2. Pre-conditions

List not empty.
Transverse doesn't point to last node in the list.

19.4.1.7.13.3. Post-conditions

Transverse now points to the next node in the list.

19.4.1.7.13.4. Return Data

RETURN SUCCESS if transverse has been moved to the next node.
RETURN ERROR if the list is empty, or if transverse already points to last node in ulist.

19.4.1.7.14. status remove_transverse()

19.4.1.7.14.1. Role

19.4.1.7.14.1.1. Operator Type

19.4.1.7.14.1.2. Description

This function deletes the node pointed to by the transverse pointer.

19.4.1.7.14.2. Pre-conditions

list is not empty

19.4.1.7.14.3. Post-conditions

Node is removed from the list.
If removed node was at the tail of the list, transverse now points to the new tail.
Otherwise transverse points to the node that immediately proceded the removed node

19.4.1.7.14.4. Return Data

RETURN SUCCESS if list not empty, and node removed.
RETURN ERROR if list empty.

19.4.1.7.15. node* get_transverse_node_pointer()

19.4.1.7.15.1. Role

19.4.1.7.15.1.1. Operator Type

19.4.1.7.15.1.2. Description

Allows access to the transverse pointer. For allowing access to member functions associated with transverse node by an external class, which wants to operate upon nodes in the ulist, in situ.

19.4.1.7.15.2. Pre-conditions

none

19.4.1.7.15.3. Post-conditions

none

19.4.1.7.15.4. Return Data

'Transverse' pointer value

19.4.1.7.16. ulist<node>& operator=(ulist<node> &source)

19.4.1.7.16.1. Role

19.4.1.7.16.1.1. Operator Type

19.4.1.7.16.1.2. Description

This function permits the use of '=' operator on list classes. The target ist is emptied of any nodes, and then all nodes in the source list are copied to the target preserving ordering.

19.4.1.7.16.2. Pre-conditions

none

19.4.1.7.16.3. Post-conditions

list pointed to by 'this' is an identical copy of source. This applies to both static and dynamic data within the node.

19.4.1.7.16.4. Return data

return *this

19.4.1.8. Friend Member Functions

19.4.1.8.1. Friend ostream& operator<<(ostream& output_stream, const ulist a)

19.4.1.8.1.1. Role

19.4.1.8.1.1.1. Operator Type

19.4.1.8.1.1.2. Description

Prints out entire list, beginning with head to standard output using the print_node member function of node.

19.4.1.8.1.2. Pre-conditions

none

19.4.1.8.1.3. Post-conditions

output stream is loaded with list information. If list is empty, an empty-list message is outputted to the stream.

19.4.1.8.1.4. Return Data

output_stream

19.4.1.9. Static Member Functions

none

20. Header File: “record_object.h”

20.1. General Information

20.1.1. Header File Role

Contains class definition for record_object class

20.1.2. Standard Headers Required

iostream.h

20.1.3. Custom Headers Required

newstring.h; stringobject.h; ulist.h; define_vars.h

20.2. C-Type Definitions

None

20.3. Non-Class Function Prototypes

None

20.4. Class Definitions

20.4.1. Class “record_object”

20.4.1.1. Role:

Node of ulist which is INDEXed by string_class name and contains a ulist of string_objects as DATA.

20.4.1.2. Class Initialisation:

None

20.4.1.3. Private Data Members

20.4.1.3.1. string_class name

Index or ‘COUNTER’ field for node - primary ordering field - Holds name for ulist<string_object>

20.4.1.3.2. ulist<string_object> data

DATA field for node

20.4.1.3.3. record_object *next_record_object

Linking pointer field for linked list (ulist)

20.4.1.4. Public Data Members

None

20.4.1.5. Static Data Members

20.4.1.5.1. string_class undefined_index

Holds undefined_index=””

20.4.1.6. Private Member Functions

None

20.4.1.7. Public Member Functions

20.4.1.7.1. record_object()
20.4.1.7.2. record_object(string_class name_string)
20.4.1.7.3. record_object(string_class name_string, ulist<string_object> data_list)
20.4.1.7.4. record_object(record_object &original)
20.4.1.7.5. record_object& operator=(record_object &original)
20.4.1.7.6. string_class get_index()
20.4.1.7.7. status set_index(const string_class setting)
20.4.1.7.8. void set_to_first_index()
20.4.1.7.9. void set_to_next_index(const string_class ref)
20.4.1.7.10. compare compare_index(const string_class index1, const string_class index2)
20.4.1.7.11. ulist<string_object> get_data()
20.4.1.7.12. status set_data(ulist<string_object> setting)
20.4.1.7.13. compare compare_data(ulist<string_object> name1, ulist<string_object> name2)
20.4.1.7.14. void set_pointer_to(record_object *p)
20.4.1.7.15. record_object* get_pointer()
20.4.1.7.16. void print_node()

See equivalent functions in name_object class

20.4.1.8. Friend Member Functions

20.4.1.8.1. friend ostream& operator<<(ostream& output_stream, record_object a)

See equivalent functions in name_object class

See equivalent functions in name_object class

20.4.1.9. Static Member Functions

None

21. Header File: “data_set_obj.h”

21.1. General Information

21.1.1. Header File Role

Contains class definition for data_set_obj class

21.1.2. Standard Headers Required

iostream.h

21.1.3. Custom Headers Required

newstring.h, data_set.h, define_vars.h

21.2. C-Type Definitions

None

21.3. Non-Class Function Prototypes

none

21.4. Class Definitions

21.4.1. Class “data_set_obj”

21.4.1.1. Role:

Node of ulist which is INDEXed by string_class name and contains a data_set object as DATA.

21.4.1.2. Class Initialisation:

None

21.4.1.3. Private Data Members

21.4.1.3.1. string_class name

Index or ‘COUNTER’ field for node - primary ordering field - Holds name of calculator object

21.4.1.3.2. data_set data

DATA field of node

21.4.1.3.3. data_set_obj *next_data_set_obj

Linking pointer field for linked list (ulist)

21.4.1.4. Public Data Members

None

21.4.1.5. Static Data Members

21.4.1.5.1. string_class undefined_index

21.4.1.5.1.1. Role

Holds undefined_index=””

21.4.1.6. Private Member Functions

None

21.4.1.7. Public Member Functions

21.4.1.7.1. data_set_obj()

21.4.1.7.2. data_set_obj(string_class name_string)

21.4.1.7.3. data_set_obj(string_class name_string, data_set data_list)

21.4.1.7.4. data_set_obj(data_set_obj &original)

21.4.1.7.5. data_set_obj& operator=(data_set_obj &original)

21.4.1.7.6. string_class get_index()

21.4.1.7.7. status set_index(const string_class setting)

21.4.1.7.8. void set_to_first_index()

21.4.1.7.9. void set_to_next_index(const string_class ref)

21.4.1.7.10. compare compare_index(const string_class index1, const string_class index2)

21.4.1.7.11. data_set get_data()

21.4.1.7.12. status set_data(const data_set setting)

21.4.1.7.13. compare compare_data(const data_set name1, const data_set name2)

21.4.1.7.14. void set_pointer_to(data_set_obj *p)

21.4.1.7.14. void set_pointer_to(data_set_obj *p)
21.4.1.7.15. data_set_obj* get_pointer()
21.4.1.7.16. void print_node()
See equivalent functions in name_object class.

21.4.1.7.17. status get_element_assoc(complex &result, const int index, const string_class field)
21.4.1.7.18. status set_element_assoc(const complex new_value, const int index, const string_class field)
21.4.1.7.19. status get_element(complex &result, const int index, const int field_index)
21.4.1.7.20. status set_element(const complex new_value, const int index, const int field_index)
21.4.1.7.21. int get_length()
21.4.1.7.22. int get_width()
21.4.1.7.23. void get_fields(string_class *storage)
21.4.1.7.24. status clear_data_array()
These functions allow access and carry out identical operations to the eqivalent public member functions of the DATA field, data_set 'data'.

21.4.1.7.25. status load_real_input(set_input &input, string_class field)
21.4.1.7.25.1. Role
21.4.1.7.25.1.1. Operator Type
Modifier
21.4.1.7.25.1.2. Description
Applies a set_input object to field 'field' of private member data_set 'data'.
21.4.1.7.25.2. Pre-conditions
'field' is valid for this data_set.
Element bounds in 'input' are within bounds of data_set.
21.4.1.7.25.3. Post-conditions
The particular field of data_set is loaded with numbers used for calculator input, according to the state of set_input 'input'.
21.4.1.7.25.4. Return Data
RETURN ERROR preconditions not met.
RETURN SUCCESS otherwise.

21.4.1.8. Friend Member Functions
21.4.1.8.1. ostream& operator<<(ostream& output_stream, const data_set_obj a);
See equivalent function in name_object class.

21.4.1.9. Static Member Functions
None

22. Header File: "set_input_object.h"

22.1. General Information

22.1.1. Header File Role
Contains class definition for set_input_object class
22.1.2. Standard Headers Required
<iostream.h>
22.1.3. Custom Headers Required
newstring.h, data_set.h, define_vars.h

22.2. C-Type Definitions

None

22.3. Non-Class Function Prototypes

none

22.4. Class Definitions

22.4.1. Class "set_input_object"
22.4.1.1. Role:
Node of ulist which is INDEXed by string_class name and contains a set_input object as DATA.
22.4.1.2. Class Initialisation:
None
22.4.1.3. Private Data Members

22.4.1.3. Private Data Members

22.4.1.3.1. string_class name

Index or ‘COUNTER’ field for node - primary ordering field - Holds name of calculator object

22.4.1.3.2. set_input load_data

DATA field for node

22.4.1.3.3. set_input_object *next_set_input_object

Linking pointer field for linked list (ulist)

22.4.1.4. Public Data Members

None

22.4.1.5. Static Data Members

22.4.1.5.1. string_class undefined_index

Holds undefined_index=””

22.4.1.6. Private Member Functions

None

22.4.1.7. Public Member Functions

22.4.1.7.1. set_input_object()
22.4.1.7.2. set_input_object(string_class name_string)
22.4.1.7.3. set_input_object(string_class name_string, set_input data)
22.4.1.7.4. set_input_object(set_input_object &original)
22.4.1.7.5. set_input_object& operator=(set_input_object &original)
22.4.1.7.6. string_class get_index()
22.4.1.7.7. status set_index(const string_class setting)
22.4.1.7.8. void set_to_first_index()
22.4.1.7.9. void set_to_next_index(const string_class ref)
22.4.1.7.10. compare compare_index(const string_class index1,const string_class index2)
22.4.1.7.11. set_input get_data()
22.4.1.7.12. status set_data(const set_input setting)
22.4.1.7.13. compare compare_data(const set_input name1, const set_input name2)
22.4.1.7.14. void set_pointer_to(set_input_object *p)
22.4.1.7.15. set_input_object* get_pointer()
22.4.1.7.16. void print_node()

See equivalent functions in name_object class.

22.4.1.8. Friend Member Functions

22.4.1.8.1. ostream& operator<<(ostream& output_stream, const set_input_object a)

See equivalent function in name_object class.

22.4.1.9. Static Member Functions

None

23. Header File: “IO_map_object.h”

23.1. General Information

23.1.1. Header File Role

Contains class definition for IO_map_object class

23.1.2. Standard Headers Required

none

23.1.3. Custom Headers Required

newstring.h, IO_map.h, define_vars.h

23.2. C-Type Definitions

None

23.3. Non-Class Function Prototypes

23.3. Non-Class Function Prototypes

None

23.4. Class Definitions

23.4.1. Class "IO_map_object"

23.4.1.1. Role:

Node of ulist which is INDEXed by string_class name and contains an IO_map object as DATA.

23.4.1.2. Class Initialisation:

None

23.4.1.3. Private Data Members

23.4.1.3.1. string_class name

Index or 'COUNTER' field for node - primary ordering field - Holds name of IO_map object.

23.4.1.3.2. IO_map map

DATA field for node - secondary ordering field - Holds IO_map object

23.4.1.3.3. IO_map_object *next_IO_map_object

Linking pointer field for linked list (ulist)

23.4.1.4. Public Data Members

None

23.4.1.5. Static Data Members

23.4.1.5.1. string_class undefined_index

Holds undefined_index=""

23.4.1.6. Private Member Functions

None

23.4.1.7. Public Member Functions

23.4.1.7.1. IO_map_object()
23.4.1.7.2. IO_map_object(string_class map_name)
23.4.1.7.3. IO_map_object(string_class map_name, IO_map new_map)
23.4.1.7.4. IO_map_object(IO_map_object &original)
23.4.1.7.5. IO_map_object& operator=(IO_map_object &original)
23.4.1.7.6. string_class get_index();
23.4.1.7.7. status set_index(const string_class setting)
23.4.1.7.8. void set_to_first_index()
23.4.1.7.9. void set_to_next_index(const string_class ref)
23.4.1.7.10. compare compare_index(const string_class index1, const string_class index2);
23.4.1.7.11. IO_map get_data()
23.4.1.7.12. status set_data(const IO_map setting)
23.4.1.7.13. compare compare_data(const IO_map name1, const IO_map name2)
23.4.1.7.14. void set_pointer_to(IO_map_object *p)
23.4.1.7.15. IO_map_object* get_pointer()
23.4.1.7.16. void print_node()

See equivalent functions in name_object class.

23.4.1.7.17. void get_input_fields(string_class *stored_input_fields)
23.4.1.7.18. int get_number_of_input_fields()
23.4.1.7.19. void get_output_fields(string_class *stored_output_fields)
23.4.1.7.20. int get_number_of_output_fields()

These 4 functions allow access and carry out identical operations to the eqivalent public member functions of the DATA field, IO_map 'map'.

23.4.1.8. Friend Member Functions

23.4.1.8.1. ostream& operator<<(ostream& output_stream, const calc_object a)

See equivalent functions in name_object class.

23.4.1.9. Static Member Functions

None

24. Header File: "name_object.h"

24. Header File: “name_object.h”

24.1. General Information

24.1.1. Header File Role

Contains class definition for name_object class.

24.1.2. Standard Headers Required

none

24.1.3. Custom Headers Required

newstring.h, extraclasses.h, define_vars.h

24.2. C-Type Definitions

None.

24.3. Non-Class Function Prototypes

None.

24.4. Class Definitions

24.4.1. Class “name_object”

24.4.1.1. Role:

Node of ulist which provides way of storing a VARIABLE/EQUATION name, together with the definition for that name (complex number (VARIABLE) or char string (EQUATION)), in complex_container data.

24.4.1.2. Class Initialisation:

None

24.4.1.3. Private Data Members

24.4.1.3.1. string_class name

Index or ‘COUNTER’ field for node - primary ordering field - Holds name of VARIABLE/EQUATION

24.4.1.3.2. complex_container data

DATA field for node - secondary ordering field - Holds complex_number/string in complex_container

24.4.1.3.3. name_object *next_name_object

Linking pointer field for linked list (ulist)

24.4.1.4. Public Data Members

none

24.4.1.5. Static Data Members

24.4.1.5.1. string_class undefined_index

24.4.1.5.1.1. Role

Holds undefined_index=””

24.4.1.5.1.2. Range

Any string permitted by string_class

24.4.1.6. Private Member Functions

none

24.4.1.7. Public Member Functions

24.4.1.7.1. name_object()

24.4.1.7.1.1. Role

24.4.1.7.1.1.1. Operator Type

Foundation

24.4.1.7.1.1.2. Description

Default Constructor

24.4.1.7.1.2. Pre-conditions

none

24.4.1.7.1.3. Post-conditions

next_name_object=NULL, name=name_object::undefined_index

24.4.1.7.2. name_object(const string_class string, const complex_container

24.4.1.7.2. name_object(const string_class string, const complex_container complex_field=complex_container(CONSTANT,complex(0, 0)))

24.4.1.7.2.1. Role

24.4.1.7.2.1.1. Operator Type

Foundation

24.4.1.7.2.1.2. Description

(INDEX=string_class, DATA=complex_container) Parameterized Constructor
If DATA is unspecified, then default values of (CONSTANT,complex(0,0)) are assumed.

24.4.1.7.2.2. Pre-conditions

None

24.4.1.7.2.3. Post-conditions

name=string, data=complex_field, next_name_object=NULL

24.4.1.7.3. name_object(const char *string, const complex_container complex_field=complex_container(CONSTANT, complex(0, 0)))

24.4.1.7.3.1. Role

24.4.1.7.3.1.1. Operator Type

Foundation

24.4.1.7.3.1.2. Description

(INDEX=char array, DATA=complex_container) Parameterized Constructor
If DATA is unspecified, then default values of (CONSTANT,complex(0,0)) are assumed.

24.4.1.7.3.2. Pre-conditions

none

24.4.1.7.3.3. Post-conditions

name=string, data=complex_field, next_name_object=NULL

24.4.1.7.4. name_object(const name_object &original)

24.4.1.7.4.1. Role

24.4.1.7.4.1.1. Operator Type

Foundation

24.4.1.7.4.1.2. Description

Copy Constructor

24.4.1.7.4.2. Pre-conditions

none

24.4.1.7.4.3. Post-conditions

All private data members from original copied into ‘this’

24.4.1.7.4.4. Return Data

RETURN *this

24.4.1.7.5. name_object& operator=(name_object &source)

24.4.1.7.5.1. Role

24.4.1.7.5.1.1. Operator Type

Overloaded = operator

24.4.1.7.5.1.2. Description

Allows assignment between name_objects

24.4.1.7.5.2. Pre-conditions

none

24.4.1.7.5.3. Post-conditions

All private data members from source are copied into ‘this’

24.4.1.7.5.4. Return Data

RETURN *this

24.4.1.7.6. string_class get_index()

24.4.1.7.6.1. Role

24.4.1.7.6.1.1. Operator Type

Extractor

24.4.1.7.6.1.2. Description

Return INDEX value, ie return string_class ‘name’

24.4.1.7.6.2. Pre-conditions

none

24.4.1.7.6.3. Post-conditions

none

24.4.1.7.6.4. Return Data

RETURN name

24.4.1.7.7. status set_index(const string_class setting)

24.4.1.7.7.1. Role

24.4.1.7.7.1.1. Operator Type

Modifier

24.4.1.7.7.1.2. Description

Set INDEX value, ie string_class ‘name’ to ‘setting’

24.4.1.7.7.2. Pre-conditions

none

24.4.1.7.7.3. Post-conditions

name=setting

24.4.1.7.7.4. Return Data

RETURN SUCCESS

24.4.1.7.8. void set_to_first_index()

24.4.1.7.8.1. Role

24.4.1.7.8.1.1. Operator Type

Modifier

24.4.1.7.8.1.2. Description

Set INDEX, ie string_class name, to first permissible index=”a”

24.4.1.7.8.2. Pre-conditions

none

24.4.1.7.8.3. Post-conditions

name=”a”

24.4.1.7.9. void set_to_next_index(const string_class ref)

24.4.1.7.9.1. Role

24.4.1.7.9.1.1. Operator Type

Modifier

24.4.1.7.9.1.2. Description

Set INDEX, ie string_class name, to next permissible index -> ref+”a“

24.4.1.7.9.2. Pre-conditions

none

24.4.1.7.9.3. Post-conditions

name=name+”a”

24.4.1.7.10. compare compare_index(const string_class index1, const string_class index2);

24.4.1.7.10.1. Role

24.4.1.7.10.1.1. Operator Type

Decider

24.4.1.7.10.1.2. Description

Lexical comparison between two INDEX values, index1 and index2.

24.4.1.7.10.2. Pre-conditions

none

24.4.1.7.10.3. Post-conditions

none

24.4.1.7.10.4. Return Data

RETURN SMALLER if index1 < index2

RETURN EQUAL if index1 = index2

RETURN LARGER if index1 > index2

24.4.1.7.11. complex_container get_data()

24.4.1.7.11.1. Role

24.4.1.7.11.1.1. Operator Type

Extractor

24.4.1.7.11.1.2. Description

Return DATA value, ie return complex_container ‘data’

24.4.1.7.11.2. Pre-conditions

none

24.4.1.7.11.3. Post-conditions

none

24.4.1.7.11.4. Return Data

RETURN data

24.4.1.7.12. status set_data(const complex_container setting);

24.4.1.7.12.1. Role

24.4.1.7.12.1.1. Operator Type

Modifier

24.4.1.7.12.1.2. Description

Set DATA value, ie complex_container ‘data’ to ‘setting’

24.4.1.7.12.2. Pre-conditions

none

24.4.1.7.12.3. Post-conditions

data=setting

24.4.1.7.12.4. Return Data

RETURN SUCCESS

24.4.1.7.13. compare compare_data(const complex_container name1, const complex_container name2)

24.4.1.7.13.1. Role

24.4.1.7.13.1.1. Operator Type

Decider

24.4.1.7.13.1.2. Description

Define comparison operation between two DATA values, name1 and name2.
Ordering is defined on complex_container.indicator, then either .complex_number or .equation.
CONSTANT<VARIABLE<EQUATION; complex_numbers are ordered according to real component and then imag component; equations are lexically ordered.

24.4.1.7.13.2. Pre-conditions

none

24.4.1.7.13.3. Post-conditions

none

24.4.1.7.13.4. Return Data

RETURN SMALLER if name1 < name2
RETURN EQUAL if name1 = name2
RETURN LARGER if name1 > name2

24.4.1.7.14. void set_pointer_to(name_object *p)

24.4.1.7.14.1. Role

24.4.1.7.14.1.1. Operator Type

Modifier

24.4.1.7.14.1.2. Description

Set next_name_object pointer to new address in Application Heap space.

24.4.1.7.14.2. Pre-conditions

none

24.4.1.7.14.3. Post-conditions

next_name_object=p

24.4.1.7.15. name_object* get_pointer()

24.4.1.7.15.1. Role

24.4.1.7.15.1.1. Operator Type

Extractor

24.4.1.7.15.1.2. Description

Return value of next_name_object pointer

24.4.1.7.15.2. Pre-conditions

none

24.4.1.7.15.3. Post-conditions

none

24.4.1.7.15.4. Return Data

RETURN ‘next_name_object’

24.4.1.7.16. void print_node()
24.4.1.7.16.1. Role
24.4.1.7.16.1.1. Operator Type
Extractor
24.4.1.7.16.1.2. Description
Output INDEX and DATA fields of node.
24.4.1.7.16.2. Pre-conditions
none
24.4.1.7.16.3. Post-conditions
Output to cout ‘name’+“->”+’data’
24.4.1.7.16.4. Return Data
none

24.4.1.7.17. complex get_complex()
24.4.1.7.17.1. Role
24.4.1.7.17.1.1. Operator Type
Extractor
24.4.1.7.17.1.2. Description
Allow read-access to data.complex_number
24.4.1.7.17.2. Pre-conditions
none
24.4.1.7.17.3. Post-conditions
none
24.4.1.7.17.4. Return Data
RETURN data.complex_number

24.4.1.7.18. void set_complex(const complex new_complex)
24.4.1.7.18.1. Role
24.4.1.7.18.1.1. Operator Type
Modifier
24.4.1.7.18.1.2. Description
Allow write-access to data.complex_number
24.4.1.7.18.2. Pre-conditions
none
24.4.1.7.18.3. Post-conditions
data.complex_number=new_complex

24.4.1.7.19. macro_type get_indicator()
24.4.1.7.19.1. Role
24.4.1.7.19.1.1. Operator Type
Extractor
24.4.1.7.19.1.2. Description
Allow read-access to data.indicator
24.4.1.7.19.2. Pre-conditions
none
24.4.1.7.19.3. Post-conditions
none
24.4.1.7.19.4. Return Data
RETURN data.indicator

24.4.1.7.20. void set_indicator(const macro_type new_indicator)
24.4.1.7.20.1. Role
24.4.1.7.20.1.1. Operator Type
Modifier
24.4.1.7.20.1.2. Description
Allow write-access to data.indicator
24.4.1.7.20.2. Pre-conditions
none
24.4.1.7.20.3. Post-conditions
data.indicator=new_indicator

24.4.1.7.21. string_class get_equation()
24.4.1.7.21.1. Role
24.4.1.7.21.1.1. Operator Type
Extractor
24.4.1.7.21.1.2. Description
Allow read-access to data.equation
24.4.1.7.21.2. Pre-conditions
none
24.4.1.7.21.3. Post-conditions
none
24.4.1.7.21.4. Return Data
RETURN data.equation

24.4.1.7.22. void set_equation(const string_class new_equation)
24.4.1.7.22.1. Role
24.4.1.7.22.1.1. Operator Type
Modifier
24.4.1.7.22.1.2. Description
Allow write-access to data.equation
24.4.1.7.22.2. Pre-conditions
none
24.4.1.7.22.3. Post-conditions
data.equation=new_equation

24.4.1.8. Friend Member Functions
24.4.1.8.1. friend ostream& operator<<(ostream& output_stream, const name_object a)
24.4.1.8.1.1. Role
24.4.1.8.1.1.1. Operator Type
Overloaded output operator
24.4.1.8.1.1.2. Description
Streamed output INDEX and DATA fields of node.
24.4.1.8.1.2. Pre-conditions
none
24.4.1.8.1.3. Post-conditions
Output to output_stream ‘name’+“->”+’data’
24.4.1.8.1.4. Return Data
RETURN output_stream

24.4.1.9. Static Member Functions
None

25. Header File: “graph_spec_obj”

25.1. General Information

25.1.1. Header File Role
Contains class definition for graph_spec_obj class
25.1.2. Standard Headers Required
iostream.h
25.1.3. Custom Headers Required
newstring.h, data_set.h, define_vars.h

25.2. C-Type Definitions

None

25.3. Non-Class Function Prototypes

None

25.4. Class Definitions

25.4.1. Class “graph_spec_obj”
25.4.1.1. Role:
Node of ulist which is INDEXed by string_class name and contains a graph_spec object as DATA.

25.4.1.2. Class Initialisation:
None

25.4.1.3. Private Data Members

25.4.1.3.1. string_class name
Index or ‘COUNTER’ field for node - primary ordering field - Holds name for graph_spec object

25.4.1.3.2. graph_spec spec
DATA field for node

25.4.1.3.3. graph_spec_obj *next_graph_spec_obj
Linking pointer field for linked list (ulist)

25.4.1.4. Public Data Members
None

25.4.1.5. Static Data Members

25.4.1.5.1. string_class undefined_index
Holds undefined_index=””

25.4.1.6. Private Member Functions
None

25.4.1.7. Public Member Functions

25.4.1.7.1. graph_spec_obj()
25.4.1.7.2. graph_spec_obj(string_class name)
25.4.1.7.3. graph_spec_obj(string_class name, graph_spec new_spec)
25.4.1.7.4. graph_spec_obj(graph_spec_obj &original)
25.4.1.7.5. graph_spec_obj& operator=(graph_spec_obj &original)
25.4.1.7.6. string_class get_index()
25.4.1.7.7. status set_index(const string_class setting)
25.4.1.7.8. void set_to_first_index()
25.4.1.7.9. void set_to_next_index(const string_class ref)
25.4.1.7.10. compare compare_index(const string_class index1, const string_class index2)
25.4.1.7.11. graph_spec get_data()
25.4.1.7.12. status set_data(const graph_spec setting)
25.4.1.7.13. compare compare_data(const graph_spec name1, const graph_spec name2)
25.4.1.7.14. void set_pointer_to(graph_spec_obj *p)
25.4.1.7.15. graph_spec_obj* get_pointer()
25.4.1.7.16. void print_node()
See equivalent functions in name_object class.

25.4.1.8. Friend Member Functions

25.4.1.8.1. ostream& operator<<(ostream& output_stream, const graph_spec_obj a)
See equivalent function in name_object class.

25.4.1.9. Static Member Functions
None

26. Header File: “calcobject.h”

26.1. General Information

26.1.1. Header File Role
Contains class definition for calc_object class

26.1.2. Standard Headers Required
none

26.1.3. Custom Headers Required
newstring.h, rCalcuatorClass.h, define_vars.h

26.2. C-Type Definitions
None

26.3. Non-Class Function Prototypes

26.3. Non-Class Function Prototypes

None

26.4. Class Definitions

26.4.1. Class "calc_object"

26.4.1.1. Role:

Node of ulist which is INDEXed by string_class name and contains a calculator object as DATA.

26.4.1.2. Class Initialisation:

None

26.4.1.3. Private Data Members

26.4.1.3.1. string_class identifier

Index or 'COUNTER' field for node - primary ordering field - Holds name of calculator object

26.4.1.3.2. calculator data

DATA field for node

26.4.1.3.3. calc_object *next_calc_object

Linking pointer field for linked list (ulist)

26.4.1.4. Public Data Members

none

26.4.1.5. Static Data Members

26.4.1.5.1. string_class undefined_index

Holds undefined_index=""

26.4.1.6. Private Member Functions

None

26.4.1.7. Public Member Functions

26.4.1.7.1. calc_object()
26.4.1.7.2. calc_object(const string_class name)
26.4.1.7.3. calc_object(const calc_object &original)
26.4.1.7.4. calc_object& operator=(calc_object &source)
26.4.1.7.5. string_class get_index()
26.4.1.7.6. status set_index(const string_class setting)
26.4.1.7.7. void set_to_first_index()
26.4.1.7.8. void set_to_next_index(const string_class ref)
26.4.1.7.9. compare compare_index(const string_class index1, const string_class index2)
26.4.1.7.10. calculator get_data()
26.4.1.7.11. status set_data(const calculator setting)
26.4.1.7.12. compare compare_data(const calculator name1, const calculator name2)
26.4.1.7.13. void set_pointer_to(calc_object *p)
26.4.1.7.14. calc_object* get_pointer()
26.4.1.7.15. void print_node()

See equivalent functions in name_object class.

26.4.1.7.16. complex evaluate(string_class input_string)
26.4.1.7.17. string_class flush_errors()
26.4.1.7.18. ostream& peek_errors(ostream& output_stream)
26.4.1.7.19. int get_number_of_errors()
26.4.1.7.20. void all_clear()
26.4.1.7.21. status clear_single_memory(string_class name)
26.4.1.7.22. void auto_verify_off()
26.4.1.7.23. status auto_verify_on()
26.4.1.7.24. void set_validator(validator *checker)

These 9 functions allow access and carry out identical operations to the eqivalent public member functions of the DATA field, calculator 'data'.

26.4.1.8. Friend Member Functions

26.4.1.8.1. ostream& operator<<(ostream& output_stream, const calc_object a)

See equivalent function in name_object class.

26.4.1.9. Static Member Functions

None

None

27. Header File: “stringobject.h”

27.1. General Information

27.1.1. Header File Role

Contains class definition for string_object class.

27.1.2. Standard Headers Required

none

27.1.3. Custom Headers Required

define_vars.h, newstring.h

27.2. C-Type Definitions

none

27.3. Non-Class Function Prototypes

none

27.4. Class Definitions

27.4.1. Class “string_object”

27.4.1.1. Role:

Node of ulist which provides way of storing a single string (DATA field) and an associated integer INDEX.

27.4.1.2. Class Initialisation:

None

27.4.1.3. Private Data Members

27.4.1.3.1. int index

COUNTER field for node - primary ordering field - Holds integer ordering

27.4.1.3.2. string_class string

Data field for node - secondary ordering field - Holds arbitrary string

27.4.1.3.3. string_object *string_name_object

Linking pointer field for linked list (ulist)

27.4.1.4. Public Data Members

none

27.4.1.5. Static Data Members

27.4.1.5.1. int undefined_index

27.4.1.5.1.1. Role

Holds undefined_index=0

27.4.1.5.1.2. Range

Integer range

27.4.1.6. Private Member Functions

none

27.4.1.7. Public Member Functions

27.4.1.7.1. string_object()

27.4.1.7.1.1. Role

27.4.1.7.1.1.1. Operator Type

Foundation

27.4.1.7.1.1.2. Description

Default Constructor

27.4.1.7.1.2. Pre-conditions

none

27.4.1.7.1.3. Post-conditions

next_string_object=NULL, index=string_object::undefined_index, string=””

27.4.1.7.2. string_object(const string_class original)

27.4.1.7.2.1. Role

27.4.1.7.2.1.1. Operator Type

Foundation

27.4.1.7.2.1.2. Description

Parameterized constructor

27.4.1.7.2.2. Pre-conditions

none

27.4.1.7.2.3. Post-conditions

next_string_object=NULL, index=string_object::undefined_index, string=original

27.4.1.7.3. string_object(const string_object &original)

27.4.1.7.3.1. Role

27.4.1.7.3.1.1. Operator Type

Foundation

27.4.1.7.3.1.2. Description

Copy Constructor

27.4.1.7.3.2. Pre-conditions

none

27.4.1.7.3.3. Post-conditions

All private data members from original copied into ‘this’

27.4.1.7.4. string_object& operator=(string_object &source)

27.4.1.7.4.1. Role

27.4.1.7.4.1.1. Operator Type

Overloaded = operator

27.4.1.7.4.1.2. Description

Allows assignment between string_objects

27.4.1.7.4.2. Pre-conditions

none

27.4.1.7.4.3. Post-conditions

All private data members from source are copied into ‘this’

27.4.1.7.5. int get_index()

27.4.1.7.5.1. Role

27.4.1.7.5.1.1. Operator Type

Extractor

27.4.1.7.5.1.2. Description

Return INDEX value

27.4.1.7.5.2. Pre-conditions

none

27.4.1.7.5.3. Post-conditions

none

27.4.1.7.5.4. Return Data

RETURN index

27.4.1.7.6. status set_index(const int setting)

27.4.1.7.6.1. Role

27.4.1.7.6.1.1. Operator Type

Modifier

27.4.1.7.6.1.2. Description

Set INDEX value to ‘setting’

27.4.1.7.6.2. Pre-conditions

none

27.4.1.7.6.3. Post-conditions

index=setting

27.4.1.7.6.4. Return Data

RETURN SUCCESS

27.4.1.7.7. void set_to_first_index()
27.4.1.7.7.1. Role
27.4.1.7.7.1.1. Operator Type
Modifier
27.4.1.7.7.1.2. Description
Set INDEX to first permissible index=1
27.4.1.7.7.2. Pre-conditions
none
27.4.1.7.7.3. Post-conditions
index=1

27.4.1.7.8. status set_to_next_index(const int ref)
27.4.1.7.8.1. Role
27.4.1.7.8.1.1. Operator Type
Modifier
27.4.1.7.8.1.2. Description
Set INDEX to next permissible index -> ref+1
27.4.1.7.8.2. Pre-conditions
none
27.4.1.7.8.3. Post-conditions
index=ref+1
27.4.1.7.8.4. Return Data
RETURN SUCCESS

27.4.1.7.9. compare compare_index(const int index1, const int index2)
27.4.1.7.9.1. Role
27.4.1.7.9.1.1. Operator Type
Decider
27.4.1.7.9.1.2. Description
Compare two indexes, real number comparison
27.4.1.7.9.2. Pre-conditions
none
27.4.1.7.9.3. Post-conditions
none
27.4.1.7.9.4. Return Data
RETURN SMALLER if index1<index2
RETURN EQUAL if index1=index2
RETURN LARGER if index1>index2

27.4.1.7.10. string_class get_data()
27.4.1.7.10.1. Role
27.4.1.7.10.1.1. Operator Type
Extractor
27.4.1.7.10.1.2. Description
Return DATA value, ie return string_class ‘string’
27.4.1.7.10.2. Pre-conditions
none
27.4.1.7.10.3. Post-conditions
none
27.4.1.7.10.4. Return Data
RETURN string

27.4.1.7.11. status set_data(const string_class set_string)
27.4.1.7.11.1. Role
27.4.1.7.11.1.1. Operator Type
Modifier
27.4.1.7.11.1.2. Description
Set DATA value, ie string_class ‘string’ to ‘set_string’
27.4.1.7.11.2. Pre-conditions
none
27.4.1.7.11.3. Post-conditions
string=set_string

27.4.1.7.12. compare compare_data(const string_class data1, const string_class data2)

27.4.1.7.12.1. Role

27.4.1.7.12.1.1. Operator Type

Decider

27.4.1.7.12.1.2. Description

Lexical comparison between two DATA values, data1 and data2

27.4.1.7.12.2. Pre-conditions

none

27.4.1.7.12.3. Post-conditions

none

27.4.1.7.12.4. Return Data

RETURN SMALLER if data1<data2

RETURN EQUAL if data1=data2

RETURN LARGER if data1>data2

27.4.1.7.13. void set_pointer_to(string_object *p)

27.4.1.7.13.1. Role

27.4.1.7.13.1.1. Operator Type

Modifier

27.4.1.7.13.1.2. Description

Set next_string_object to new address in Application Heap space.

27.4.1.7.13.2. Pre-conditions

none

27.4.1.7.13.3. Post-conditions

next_string_object=p

27.4.1.7.14. string_object* get_pointer()

27.4.1.7.14.1. Role

27.4.1.7.14.1.1. Operator Type

Extractor

27.4.1.7.14.1.2. Description

Return value of next_string_object pointer

27.4.1.7.14.2. Pre-conditions

none

27.4.1.7.14.3. Post-conditions

none

27.4.1.7.14.4. Return Data

RETURN ‘next_string_object’

27.4.1.7.15. void print_node()

27.4.1.7.15.1. Role

27.4.1.7.15.1.1. Operator Type

Extractor

27.4.1.7.15.1.2. Description

Output INDEX and DATA fields of node.

27.4.1.7.15.2. Pre-conditions

none

27.4.1.7.15.3. Post-conditions

Output to cout ‘index’+“ ”+’string’

27.4.1.8. Friend Member Functions

27.4.1.8.1. ostream& operator<<(ostream& output_stream, const string_object a)

27.4.1.8.1.1. Role

27.4.1.8.1.1.1. Operator Type

Overloaded output operator

27.4.1.8.1.1.2. Description

Output to stream INDEX and DATA fields of node.

27.4.1.8.1.2. Pre-conditions

none

27.4.1.8.1.3. Post-conditions

Output to output_stream ‘index’+“ ”+’string’

27.4.1.8.1.4. Return Data

RETURN output_stream

27.4.1.9. Static Member Functions

None

28. Header File: “iadditionalmath.h”

28.1. General Information

28.1.1. Header File Role

Provides additional real mathematical functions

28.1.2. Standard Headers Required

math.h

28.1.3. Custom Headers Required

none

28.2. C-Type Definitions

None

28.3. Non-Class Function Prototypes

28.3.1. double log2(const double value)

28.3.1.1. Role

To calculate log to base 2 of ‘value’ and return the result

28.3.1.2. Pre-conditions

value!=0

28.3.1.3. Post-conditions

None

28.3.2. double logx(const double value, const double x)

28.3.2.1. Role

To calculate log to base ‘base’ of ‘value’ and return the result

28.3.2.2. Pre-conditions

value!=0; x!=0

28.3.2.3. Post-conditions

None

28.3.3. double asinh(const double x)

28.3.3.1. Role

To calculate the inverse hyperbolic sine of ‘x’ and return the result

28.3.3.2. Pre-conditions

None

28.3.3.3. Post-conditions

None

28.3.4. double acosh(const double x)

28.3.4.1. Role

To calculate the inverse hyperbolic cosine of ‘x’ and return the result

28.3.4.2. Pre-conditions

None

28.3.4.3. Post-conditions

None

28.3.5. double atanh(const double x)

28.3.5.1. Role

To calculate the inverse hyperbolic tangent of ‘x’ and return the result

28.3.5.2. Pre-conditions

|x|!=1

28.3.5.3. Post-conditions

None

28.3.6. int factorial(const int x)

28.3.6.1. Role

To calculate the factorial of x and return the result

28.3.6.2. Pre-conditions

x>=0

28.3.6.3. Post-conditions

None

29. Header File: “complex functions.h”

29.1. General Information

29.1.1. Header File Role

Provides additional functions for manipulating complex numbers

29.1.2. Standard Headers Required

none

29.1.3. Custom Headers Required

complex.h

29.2. C-Type Definitions

29.2.1. Constants

double PI

π to 20 significant figures

29.3. Non-Class Function Prototypes

29.3.1. double magnitude(const complex a)

29.3.1.1. Role

To calculate and return the magnitude of the complex number ‘a’.

29.3.1.2. Pre-conditions

none

29.3.1.3. Post-conditions

none

29.3.2. double arg(const complex a)

29.3.2.1. Role

To calculate and return the argument of the complex number ‘a’, in the range 0->2π

29.3.2.2. Pre-conditions

none

29.3.2.3. Post-conditions

none

29.3.3. complex sqrt_comp(const complex a)

29.3.3.1. Role

To calculate and return the square root of the complex number ‘a’ in complex form

29.3.3.2. Pre-conditions

None

29.3.3.3. Post-conditions

None

29.3.4. complex cbrt(const complex a)

29.3.4.1. Role

To calculate and return the cube root of the complex number ‘a’ in complex form

29.3.4.2. Pre-conditions

None

29.3.4.3. Post-conditions

None

29.3.5. complex polar_rect(const double mag, const double arg)

29.3.5.1. Role

To calculate and return the rectangular conversion of the complex number (magnitude*exp^arg)

29.3.5.2. Pre-conditions

None

29.3.5.3. Post-conditions

None

30. Header File: “complex.h”

30.1. General Information

30.1.1. Header File Role

Contains class definition for complex class

30.1.2. Standard Headers Required

iostream.h, math.h

30.1.3. Custom Headers Required

none

30.2. C-Type Definitions

none

30.3. Non-Class Function Prototypes

none

30.4. Class Definitions

30.4.1. Class “complex”

30.4.1.1. Role:

Complex number class providing +,-,*,/ and power operations, as well as input/output operators.

30.4.1.2. Class Initialisation:

none

30.4.1.3. Private Data Members

none

30.4.1.4. Public Data Members

30.4.1.4.1. double re

store real part of complex number

30.4.1.4.2. double im

store imaginary part of complex number

30.4.1.5. Private Member Functions

none

30.4.1.6. Public Member Functions

30.4.1.6.1. complex()

30.4.1.6.1.1. Role

30.4.1.6.1.1.1. Operator Type

foundation

30.4.1.6.1.1.2. Description

default constructor

30.4.1.6.1.2. Pre-conditions

none

30.4.1.6.1.3. Post-conditions

’re’=0

’im’=0

30.4.1.6.2. complex(const double real, const double imag)

30.4.1.6.2.1. Role

30.4.1.6.2.1.1. Operator Type

foundation

30.4.1.6.2.1.2. Description

real/imag paremateriz ed constructor - allows initialisation of object

30.4.1.6.2.2. Pre-conditions

none

30.4.1.6.2.3. Post-conditions

’re’=real

’im’=imag

30.4.1.6.3. void set(const double real, const double imag)

30.4.1.6.3.1. Role

30.4.1.6.3.1.1. Operator Type

modifier

30.4.1.6.3.1.2. Description

Alllow setting of both real and imag members at once

30.4.1.6.3.2. Pre-conditions

30.4.1.6.3.2. Pre-conditions
none
30.4.1.6.3.3. Post-conditions
're'=real
'im'=imag

30.4.1.6.4. complex& operator+=(const complex a)
30.4.1.6.4.1. Role
30.4.1.6.4.1.1. Operator Type
overloaded += operator
30.4.1.6.4.1.2. Description
complex addition:
allows the construct 'a+=b' where a,b are complex objects
30.4.1.6.4.2. Pre-conditions
none
30.4.1.6.4.3. Post-conditions
*this=*this+a

30.4.1.6.5. complex& operator-=(const complex a)
30.4.1.6.5.1. Role
30.4.1.6.5.1.1. Operator Type
overloaded -= operator
30.4.1.6.5.1.2. Description
complex subtraction
allows the construct 'a-=b' where a,b are complex objects
30.4.1.6.5.2. Pre-conditions
none
30.4.1.6.5.3. Post-conditions
*this=*this-a

30.4.1.6.6. complex operator-()
30.4.1.6.6.1. Role
30.4.1.6.6.1.1. Operator Type
overloaded unary - operator
30.4.1.6.6.1.2. Description
complex unary minus:
allows the construct '-a' where a is a complex object
30.4.1.6.6.2. Pre-conditions
none
30.4.1.6.6.3. Post-conditions
*this=-*this

30.4.1.6.7. complex& operator*=(const complex a)
30.4.1.6.7.1. Role
30.4.1.6.7.1.1. Operator Type
overloaded *= operator
30.4.1.6.7.1.2. Description
complex multiplication:
allows the construct 'a*=b' where a,b are complex objects
30.4.1.6.7.2. Pre-conditions
none
30.4.1.6.7.3. Post-conditions
*this=*this*a

30.4.1.6.8. complex& operator/=(const complex a)
30.4.1.6.8.1. Role
30.4.1.6.8.1.1. Operator Type
overloaded /= operator
30.4.1.6.8.1.2. Description
complex division:
allows the construct 'a/=b' where a,b are complex objects
30.4.1.6.8.2. Pre-conditions

none

30.4.1.6.8.3. Post-conditions

*this=*this/a

30.4.1.7. Friend Member Functions

30.4.1.7.1. friend complex operator+(const complex a, const complex b)

30.4.1.7.1.1. Role

30.4.1.7.1.1.1. Operator Type

overloaded + operator

30.4.1.7.1.1.2. Description

complex addition:
allows the construct ‘a+b’ where a,b are complex objects

30.4.1.7.1.2. Pre-conditions

none

30.4.1.7.1.3. Post-conditions

none

30.4.1.7.1.4. Return Data

a+b

30.4.1.7.2. friend complex operator-(const complex k, const complex b)

30.4.1.7.2.1. Role

30.4.1.7.2.1.1. Operator Type

overloaded - operator

30.4.1.7.2.1.2. Description

complex subtraction:
allows the construct ‘a-b’ where a,b are complex objects

30.4.1.7.2.2. Pre-conditions

none

30.4.1.7.2.3. Post-conditions

none

30.4.1.7.2.4. Return Data

a-b

30.4.1.7.3. friend complex operator*(const complex a, const complex b)

30.4.1.7.3.1. Role

30.4.1.7.3.1.1. Operator Type

overloaded * operator

30.4.1.7.3.1.2. Description

complex multiplication:
allows the construct ‘a*b’ where a,b are complex numbers

30.4.1.7.3.2. Pre-conditions

none

30.4.1.7.3.3. Post-conditions

none

30.4.1.7.3.4. Return Data

a*b

30.4.1.7.4. friend complex operator/(const complex a, const complex b)

30.4.1.7.4.1. Role

30.4.1.7.4.1.1. Operator Type

overloaded / operator

30.4.1.7.4.1.2. Description

complex division
allows the construct ‘a/b’ where a,b are complex objects

30.4.1.7.4.2. Pre-conditions

none

30.4.1.7.4.3. Post-conditions

none

30.4.1.7.4.4. Return Data

a/b

30.4.1.7.5. friend complex operator^(const complex a, const complex exponent)

30.4.1.7.5.1. Role

30.4.1.7.5.1.1. Operator Type

complex power operator
overloaded ^ operator

30.4.1.7.5.1.2. Description

calculates the result of raising a complex number to a real power, in rectangular form.
calculates the rectangular form of exponential-form complex numbers, ie exp^(a+bj)

30.4.1.7.5.2. Pre-conditions

(a.re==0 && a.im==0 && b.re==0 && b.im==0) IS FALSE
imaginary component of exponent assumed zero if base!=exp(1)

30.4.1.7.5.3. Post-conditions

None

30.4.1.7.5.4. Return Data

a to the power b in rectangular form - in the case of taking roots the primary root is returned

30.4.1.7.6. friend ostream& operator<<(ostream& output_stream, const complex a)

30.4.1.7.6.1. Role

30.4.1.7.6.1.1. Operator Type

overloaded output operator

30.4.1.7.6.1.2. Description

outputs the real/imag components in rectangular form: ‘a+bj’ to ostream

30.4.1.7.6.2. Pre-conditions

none

30.4.1.7.6.3. Post-conditions

none

30.4.1.7.7. friend istream& operator>>(istream& input_stream, complex& complex_number)

30.4.1.7.7.1. Role

30.4.1.7.7.1.1. Operator Type

overloaded input operator

30.4.1.7.7.1.2. Description

extracts formatted character input from input_stream in the form : ‘a+bj’

30.4.1.7.7.2. Pre-conditions

format must be ‘a+bj’ including the ‘j’ place-holder, a and b are floating point numbers

30.4.1.7.7.3. Post-conditions

complex_number.re=a
complex_number.im=b

30.4.1.8. Static Member Functions

none

31. Header File: “newstring.h”

31.1. General Information

31.1.1. Header File Role

Contains class definition for string_class class

31.1.2. Standard Headers Required

iostream.h, string.h, ctype.h

31.1.3. Custom Headers Required

define_vars.h

31.2. C-Type Definitions

31.2.1. Constants

int MAX_INPUT_STRING_LENGTH

Size of buffer used for overloaded input operator

31.3. Non-Class Function Prototypes

none

31.4. Class Definitions

31.4.1. Class “string_class”

31.4.1.1. Role:

Provides a string data structure using a private dynamic char array to hold the string. All memory allocation/deallocation is encapsulated within the scope of the class. Overloaded comparison operators and append operators specified.

31.4.1.2. Class Initialisation:

None

31.4.1.3. Private Data Members

31.4.1.3.1. char *letters

Pointer to dynamically allocated memory holding character array

31.4.1.3.2. int string_length

Holds current character length of string

31.4.1.4. Public Data Members

None

31.4.1.5. Static Data Members

None

31.4.1.6. Private Member Functions

None

31.4.1.7. Public Member Functions

31.4.1.7.1. string_class()

31.4.1.7.1.1. Role

31.4.1.7.1.1.1. Operator Type

foundation

31.4.1.7.1.1.2. Description

default constructor

31.4.1.7.1.2. Pre-conditions

none

31.4.1.7.1.3. Post-conditions

’letters’=NULL (no memory allocated)

’string_length’=1 (including NULL terminator)

31.4.1.7.2. string_class(const char *source)

31.4.1.7.2.1. Role

31.4.1.7.2.1.1. Operator Type

foundation

31.4.1.7.2.1.2. Description

constructor - copy source char array into new string object

31.4.1.7.2.2. Pre-conditions

31.4.1.7.2.2. Pre-conditions
source!=NULL && source is NULL terminated
31.4.1.7.2.3. Post-conditions
'letters' points to a new copy of the char array pointed to by source
'string_length' holds length of source string (including NULL terminator)

31.4.1.7.3. string_class(const string_class &original);
31.4.1.7.3.1. Role
31.4.1.7.3.1.1. Operator Type
foundation
31.4.1.7.3.1.2. Description
copy constructor
31.4.1.7.3.2. Pre-conditions
none
31.4.1.7.3.3. Post-conditions
a copy of 'original' is created, pointed to by 'this'

31.4.1.7.4. ~string_class()
31.4.1.7.4.1. Role
31.4.1.7.4.1.1. Operator Type
foundation
31.4.1.7.4.1.2. Description
destructor
31.4.1.7.4.2. Pre-conditions
none
31.4.1.7.4.3. Post-conditions
string_class object pointed to by 'this' has all data members deallocated and itself deallocated

31.4.1.7.5. status string_copy(char *target)
31.4.1.7.5.1. Role
31.4.1.7.5.1.1. Operator Type
extractor
31.4.1.7.5.1.2. Description
copies 'this.letters' into 'target'
31.4.1.7.5.2. Pre-conditions
target!=NULL && target points to allocated memory large enough to hold 'this.letters'
31.4.1.7.5.3. Post-conditions
'target' is loaded with the NULL terminated char array pointed to by 'this.letters'
31.4.1.7.5.4. Return Data
status==ERROR if target==NULL, else status==SUCCESS

31.4.1.7.6. string_class& operator=(const string_class &source)
31.4.1.7.6.1. Role
31.4.1.7.6.1.1. Operator Type
foundation
31.4.1.7.6.1.2. Description
Overloaded equals operator enabling assignment between string_class objects.
31.4.1.7.6.2. Pre-conditions
Not assigning an object to itself
31.4.1.7.6.3. Post-conditions
*this contains copy of all data in source
31.4.1.7.6.4. Return Data
return *this

31.4.1.7.7. string_class& operator=(const char *source)
31.4.1.7.7.1. Role
31.4.1.7.7.1.1. Operator Type
foundation
31.4.1.7.7.1.2. Description
Overloaded equals operator enabling assignment of a char array to string_class
31.4.1.7.7.2. Pre-conditions
char array is NULL terminated
31.4.1.7.7.3. Post-conditions

*this contains copy of all data in source

31.4.1.7.7.4. Return Data

return *this

31.4.1.7.8. string_class& operator=(const char &source)

31.4.1.7.8.1. Role

31.4.1.7.8.1.1. Operator Type

foundation

31.4.1.7.8.1.2. Description

Overloaded equals operator enabling assignment of a single char to string_class

31.4.1.7.8.2. Pre-conditions

None

31.4.1.7.8.3. Post-conditions

*this contains copy of single character followed by NULL terminator, string length 2.

31.4.1.7.8.4. Return Data

return *this

31.4.1.7.9. char& operator[](const int index)

31.4.1.7.9.1. Role

31.4.1.7.9.1.1. Operator Type

overloaded subscript operator for non-const string_class objects

31.4.1.7.9.1.2. Description

Allows access to individual characters within the ‘letters’ array.

31.4.1.7.9.2. Pre-conditions

’index’ >0 && index<string_length

31.4.1.7.10. char& operator[](const int index) const

31.4.1.7.10.1. Role

31.4.1.7.10.1.1. Operator Type

overloaded subscript operator for const string_class objects

31.4.1.7.10.1.2. Description

Allows access to individual characters within the ‘letters’ array.

31.4.1.7.10.2. Pre-conditions

’index’ >0 && index<string_length

31.4.1.7.11. int length()

31.4.1.7.11.1. Role

31.4.1.7.11.1.1. Operator Type

extractor

31.4.1.7.11.1.2. Description

returns length of ‘letters’ char array - not including NULL terminator

31.4.1.7.11.2. Pre-conditions

none

31.4.1.7.11.3. Post-conditions

none

31.4.1.7.11.4. Return Data

return string_length data member - 1

31.4.1.8. Friend Member Functions

31.4.1.8.1. friend ostream& operator<<(ostream& output_stream, const string_class output_string)

31.4.1.8.1.1. Role

31.4.1.8.1.1.1. Operator Type

overloaded output stream operator

31.4.1.8.1.1.2. Description

formatted stream output of ‘output_string.letters’ char array sent to output_stream

31.4.1.8.1.2. Pre-conditions

none

31.4.1.8.1.3. Post-conditions

See Description

31.4.1.8.1.4. Return data

return output_stream

31.4.1.8.2. friend istream& operator>>(istream& input_stream, string_class& input_string)

31.4.1.8.2.1. Role

31.4.1.8.2.1.1. Operator Type

overloaded input stream operator

31.4.1.8.2.1.2. Description

Formatted stream input is stored in ‘input_string.letters’ char array.
Char input is read up to and not including the first non-alphanumeric character found in the input stream. This does not apply to initial white space which is skipped.

31.4.1.8.2.2. Pre-conditions

stream input length < MAX_INPUT_STRING_LENGTH

31.4.1.8.2.3. Post-conditions

’input_string.letters’ contains the inputted alphanumeric char data

31.4.1.8.2.4. Return data

return input_stream

31.4.1.8.3. friend int operator==(const string_class &string1, const string_class &string2)

31.4.1.8.3.1. Role

31.4.1.8.3.1.1. Operator Type

overloaded == operator

31.4.1.8.3.1.2. Description

lexically compares two string_class objects

31.4.1.8.4. friend int operator==(const string_class &string1, const char *string2)

31.4.1.8.4.1. Role

31.4.1.8.4.1.1. Operator Type

overloaded == operator

31.4.1.8.4.1.2. Description

lexically compares a string_class object and a following char array

31.4.1.8.4.2. Pre-conditions

string2 is NULL terminated

31.4.1.8.5. friend int operator==(const char *string1, const string_class &string2)

31.4.1.8.5.1. Role

31.4.1.8.5.1.1. Operator Type

overloaded == operator

31.4.1.8.5.1.2. Description

lexically compares a char array and a following string_class object

31.4.1.8.5.2. Pre-conditions

string1 is NULL terminated

31.4.1.8.6. friend int operator!=(const string_class &string1, const string_class &string2)

31.4.1.8.6.1. Role

31.4.1.8.6.1.1. Operator Type

overloaded != operator

31.4.1.8.6.1.2. Description

lexically compares two string_class objects

31.4.1.8.7. friend int operator!=(const string_class &string1, const char *string2)

31.4.1.8.7.1. Role

31.4.1.8.7.1.1. Operator Type

overloaded != operator

31.4.1.8.7.1.2. Description

lexically compares a string_class object and a following char array

31.4.1.8.7.2. Pre-conditions

string2 is NULL terminated

31.4.1.8.8. friend int operator!=(const char *string1, const string_class &string2)

31.4.1.8.8.1. Role

31.4.1.8.8.1.1. Operator Type

overloaded != operator

31.4.1.8.8.1.2. Description

lexically compares a char array and a following string_class object

31.4.1.8.8.2. Pre-conditions

string1 is NULL terminated

31.4.1.8.9. friend int operator>=(const string_class &string1, const string_class &string2)

31.4.1.8.9.1. Role

31.4.1.8.9.1.1. Operator Type

overloaded >= operator

31.4.1.8.9.1.2. Description

lexically compares two string_class objects

31.4.1.8.10. friend int operator<=(const string_class &string1, const string_class &string2)

31.4.1.8.10.1. Role

31.4.1.8.10.1.1. Operator Type

overloaded <= operator

31.4.1.8.10.1.2. Description

lexically compares two string_class objects

31.4.1.8.11. friend int operator>(const string_class &string1, const string_class &string2)

31.4.1.8.11.1. Role

31.4.1.8.11.1.1. Operator Type

overloaded > operator

31.4.1.8.11.1.2. Description

lexically compares two string_class objects

31.4.1.8.12. friend int operator<(const string_class &string1, const string_class &string2)

31.4.1.8.12.1. Role

31.4.1.8.12.1.1. Operator Type

overloaded < operator

31.4.1.8.12.1.2. Description

lexically compares two string_class objects

31.4.1.8.13. friend string_class operator+(const string_class source1, const string_class source2)

31.4.1.8.13.1. Role

31.4.1.8.13.1.1. Operator Type

overloaded + operator

31.4.1.8.13.1.2. Description

returns a string_class object which holds the concatenation of string_class source2 to string_class source1

31.4.1.8.14. friend string_class operator+(const string_class source1, const char *source2)

31.4.1.8.14.1. Role

31.4.1.8.14.1.1. Operator Type

overloaded + operator

31.4.1.8.14.1.2. Description

returns a string_class object which holds the concatenation of char array source2 to string_class source1

31.4.1.8.14.2. Pre-conditions

source2 is NULL terminated

31.4.1.8.15. friend string_class operator+(const char *source1, const string_class source2)

31.4.1.8.15.1. Role

31.4.1.8.15.1.1. Operator Type

overloaded + operator

31.4.1.8.15.1.2. Description

returns a string_class object which holds the concatenation of string_class source2 to char array source1

31.4.1.8.15.2. Pre-conditions

source1 is NULL terminated

31.4.1.8.16. friend string_class operator+(const string_class source1, const char input_char)

31.4.1.8.16.1. Role

31.4.1.8.16.1.1. Operator Type

overloaded + operator

31.4.1.8.16.1.2. Description

returns a string_class object which holds the concatenation of char source2 to string_class source1

31.4.1.8.17. friend void append_name(string_class &target, const string_class &source)

31.4.1.8.17.1. Role

31.4.1.8.17.1.1. Operator Type

modifier

31.4.1.8.17.1.2. Description

Concatenates source.letters char array to target.letters char array.
Concatenates a single space char to the resulting array.

31.4.1.8.18. friend status search_string(const string_class &source, const string_class &target)

31.4.1.8.18.1. Role

31.4.1.8.18.1.1. Operator Type

extractor

31.4.1.8.18.1.2. Description

searches for an instance of the ‘target’ string (followed by a space character) within the ‘source’ string.

31.4.1.8.18.2. Pre-conditions

’source’ has a space (ASCII 32) as its final char before the NULL terminator

31.4.1.8.18.3. Post-conditions

none

31.4.1.8.18.4. Return Data

RETURN SUCCESS if target followed by space is found, else RETURN ERROR

31.4.1.9. Static Member Functions

none