

NAME

Class `ALPSdata` – Constructs an object that has three instance variable vectors that hold the following: problem description input data, parameter solutions from an ALP program, and any user-defined functions that may have been used in the ALP. An object can write the information to disk in the form of a specifically formatted `.alps` file. An object can also be instantiated from an existing `.alps` file and can return each of the three components.

SYNOPSIS

```
#include <iostream>
#include <fstream>
#include <ALPSdata.h>

ALPSdata();
ALPSdata(const std::string alpsFile);
ALPSdata(std::vector<std::string> inputData,
          std::vector<std::string> paramData);
ALPSdata(std::vector<std::string> inputData,
          std::vector<std::string> paramData,
          std::vector<std::string> functionData);
ALPSdata(std::vector<std::string> inputData,
          std::vector<std::string> paramData,
          std::string functionDataFile);
ALPSdata(std::vector<std::string> inputData, int numCoef,
          double var[], std::vector<char*> name, double obj);
ALPSdata(std::vector<std::string> inputData, int numCoef,
          double var[], std::vector<char*> name, double obj,
          std::vector<std::string> functionData);
ALPSdata(std::vector<std::string> inputData, int numCoef,
          double var[], std::vector<char*> name, double obj,
          std::string functionFileName);
void setInput(const std::string inputDataFile);
void setInput(const std::vector<std::string> inputData);
void setALPSolution(const std::string paramDataFile);
void setALPSolution(const std::vector<std::string> paramData);
void setBasisInfo(const std::string functionDataFile);
void setBasisInfo(const std::vector<std::string> functionData);
std::vector<std::string> getInput();
std::vector<std::string> getALPSolution();
std::vector<std::string> getBasisInfo();
void writeALPSFile(std::string fileName);
void writeBasisInfoFile(std::string fileName);
```

DESCRIPTION

Provides an interface for the user to create and access structured `.alps` solution files, which contain problem description input parameters, coefficient parameter solutions, and user-defined functions. There are many constructors and methods to allow easy creation of new and access of previous `.alps` files. This class, used in conjunction with the `TaggedValues` class, can allow for versatile storage, access, and manipulation of data.

There are eight `ALPSdata()` constructors for the object. The first creates an empty object with no data.

The second takes one parameter, the path for a previous `.alps` file, with which the object will populate its input, parameter, and function vectors.

The third takes two vector parameters. The first is a vector of strings which represent the input data. The second is a vector of strings which represent the parameters solutions. This constructor is useful when the

user did not use any user-defined functions.

The fourth constructor is the same as the third except that there is a third parameter, a vector of strings representing the user-defined basis functions.

The fifth constructor is the same as the fourth with one difference being that the third parameter, the user-defined functions, is a file path in the form of a string, not a vector of strings.

The final three constructors mimic the preceding three except that that are passed the raw ALP solution parameter information. The new parameters are: *numCoef* - the number of coefficients in the ALP solution (same as the number of basis functions), *var[]* - the array of coefficient values, *name* - a vector of character strings holding the variable names, and *obj* - the objective value.

The **setInput()**, **setALPSolution()**, and **setBasisInfo()** methods are all very similar. There are two versions of each. One version is to provide the data as a vector of strings, and the second provides a string path to a file which contains the data.

The **getInput()**, **getALPSolution()**, and **getBasisInfo()** are also similar. They return their respective data in the form of a vector of strings.

writeALPSFile() takes one parameter, a file path, and writes the data currently stored in the class to the file path on the disk with the provided name.

writeBasisInfoFile() also takes one parameter, a file path, and writes the user-defined function data that is currently stored to a file on disk with the given name. This method can be used when compiling a shared object to be used with an ALP program.

Files written to disk with the two previous methods overwrite any already existing files with the provided names.

SEE ALSO

TaggedValues(3), **ALPSdata(5)**

AUTHOR

The class documented here was written by Taylor Carr and Jonathan Senning, who also wrote the manual page. Copyright © 2009 Department of Mathematics and Computer Science, Gordon College, 255 Grapevine Road, Wenham MA, 01984.