NAME

readValueFile − Read cost value data from a file

SYNOPSIS

#include <qnet.h>

int readValueFile(string *path***, int** *ndim***, int** *nval***, int** *trunc[]***, void*** *h***);**

DESCRIPTION

The **readValueFile**() reads cost value information for the entire state space from the file whose name is *path*. The dimension of the state space must match *ndim*, the number values per state must match *nval*, and the state space truncations must match those in *trunc[]*. The cost data is returned in the memory pointed to by *h* which should have been allocated with one of the **makeValueArray**(), **allocateArray**(), or **allocateArrayV**() functions.

RETURN VALUE

On success, the value **QNET_NORMAL** (defined to be zero) is returned, otherwise one of the following negative values is returned:

QNET_BAD_DIMENSION
: The expected dimension of the state space (specified in *ndim*) does not match the dimension in the file.

QNET_BAD_NVALUES
: The expected number of cost values per state does not match the number stored in the file.

QNET_BAD_TRUNCATION
: At least one of the expected state space truncations (specified in *trunc[]*) does not match the corresponding truncation in the file.

QNET_BAD_FILE
: The input file could not be opened or read from.

EXAMPLE

The following sequence can be used to read cost data from the file *value.h* into the memory pointed to by *h*. First the header information is read

```
int maxdim = 3;    // largest state space dimension expected
int ndim;          // actual dimension of state space
int nval;          // number of cost values per state
int N[maxdim + 1]; // state space truncations

getValueFileInfo(string("value.h"), maxdim, &ndim, &nval, &N[1]);
```

and the values of *ndim* and *nval* can be checked to make sure they match the expected values. Assuming that *ndim*=3 and *nval*=1 the appropriately sized arrays can be declared and populated with cost data

```
double*** h = (double***) makeValueArray(ndim, nval, &N[1]);
readValueFile(string("value.h"), ndim, nval, &N[1], h);
```

Note that N[0] is not used; this follows the convention used in most of the QNET DP code. If, on the other hand, *nval*=2 (or any number greater than 1), an additional level of indirection is required:

```
double**** h = (double****) makeValueArray(ndim, nval, &N[1]);
readValueFile(string("value.h"), ndim, nval, &N[1], h);
```

In this case h[x1][x2][x3][0] is the first value for state (x1,x2,x3) and h[x1][x2][x3][1] is the second value for the same state.

SEE ALSO

getValueFileInfo(3), **writeValueFile**(3), **makeValueArray**(3)

AUTHOR