Program One CMPS 5243 –

Keno Probability Simulation

Mark L. Short
Version 1
October 1, 2014

Main Page

Author:

Mark L. Short

Date:

October 1, 2014

The casino game 'KENO' involves the selection of 20 balls from 80 balls numbered 1 ... 80. The player selects k numbers (1-20). The payoff depends on how many of the player's numbers that the casino machine selects. For example, if the player selects 9 numbers and 5 of them are generated by the machine, he/she wins $4.00; if 6 are generated, the player wins $43.00.

This program is to determine the probability of each possible situation and place these probabilities in a 20 x 21 array of real values. The entries in the [ith] row assume that the player has selected i numbers. The entry in the [jth] column is the probability that the player catches j spots out of i possible. Of course, if j > i, the probability is 0.0. In general, if i <= j, the probability is given by:

```
P (catch j out of i numbers) = C ( i, j ) * P1 * P2 / P3, where
C (i, j) = i! / ((i-j)! * j!)

P1 ( factors ) = 20 * 19 * 18 ... factors, where factors = j
P2 ( factors ) = 60 * 59 * 58 ... factors, where factors = i - j
P3 ( factors ) = 80 * 79 * 78 ... factors, where factors = i
```

Together with this program specification there is a sheet of payoffs for between 1 & 9 spots marked. Calculate for each number of spots marked the "expected value" of a $1 bet.

File Index

File List

Here is a list of all files with brief descriptions:

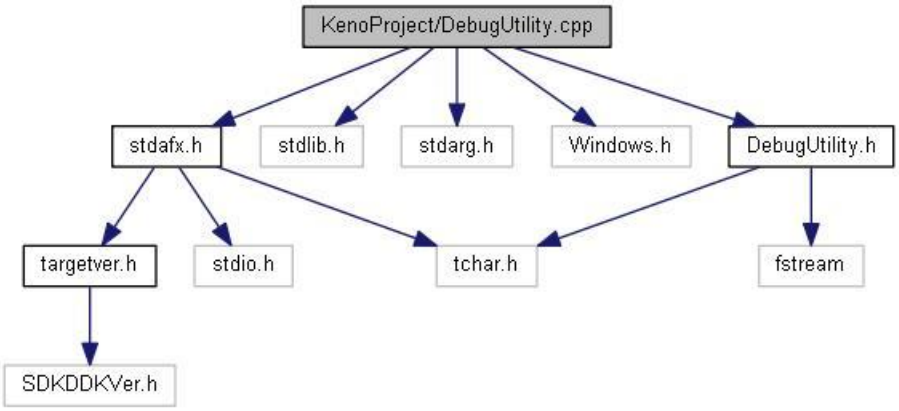
File Documentation

KenoProject/DebugUtility.cpp File Reference

Implementation of DebugUtility.cpp.

```
#include "stdafx.h"
#include "stdlib.h"
#include "stdarg.h"
#include "Windows.h"
#include "DebugUtility.h"
```

Include dependency graph for DebugUtility.cpp:



Functions

- int DebugTrace (const TCHAR *szFmt,...)
  Function used for debugging and tracing diagnostics with output directed to the IDE output window.
- TCHAR * GetModulePath (TCHAR *szModulePath, size_t cchLen)
  Function retrieves the current executable directory.

Detailed Description

Implementation of DebugUtility.cpp.

Author:
Mark L. Short

Date:
October 1, 2014

Definition in file DebugUtility.cpp.

Function Documentation

int DebugTrace (const TCHAR * *szMsg*, ...)

Function used for debugging and tracing diagnostics with output directed to the IDE output window.

Parameters:

in	*szMsg*	format string

Returns:

the number of characters written

TCHAR* GetModulePath (TCHAR * *szModulePath*, size_t *cchLen*)

Function retrieves the current executable directory.

Parameters:

out	*szModulePath*	destination memory address used to write application's directory path
in	*cchLen*	count of charecters in available to be written in destination buffer

Returns:

destination address NULL on error

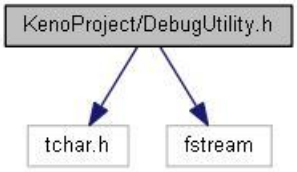
Definition at line 36 of file DebugUtility.cpp.

KenoProject/DebugUtility.h File Reference

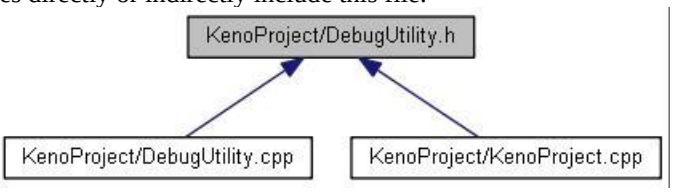
Debugging method declarations.

```
#include "tchar.h"
#include <fstream>
```

Include dependency graph for DebugUtility.h:



This graph shows which files directly or indirectly include this file:



Functions

- int DebugTrace (const TCHAR *szMsg,...)
  Function used for debugging and tracing diagnostics with output directed to the IDE output window.
- TCHAR * GetModulePath (TCHAR *szModulePath, size_t cchLen)
  Function retrieves the current executable directory.

Detailed Description

Debugging method declarations.

Author:

Mark L. Short

Date:

October 1, 2014

Definition in file DebugUtility.h.

Function Documentation

int DebugTrace (const TCHAR * *szMsg*, ...)

Function used for debugging and tracing diagnostics with output directed to the IDE output window.

Parameters:

in	*szMsg*	format string

Returns:

the number of characters written

TCHAR* GetModulePath (TCHAR * *szModulePath*, size_t *cchLen*)

Function retrieves the current executable directory.

Parameters:

out	*szModulePath*	destination memory address used to write application's directory path
in	*cchLen*	count of characters in available to be written in destination buffer

Returns:

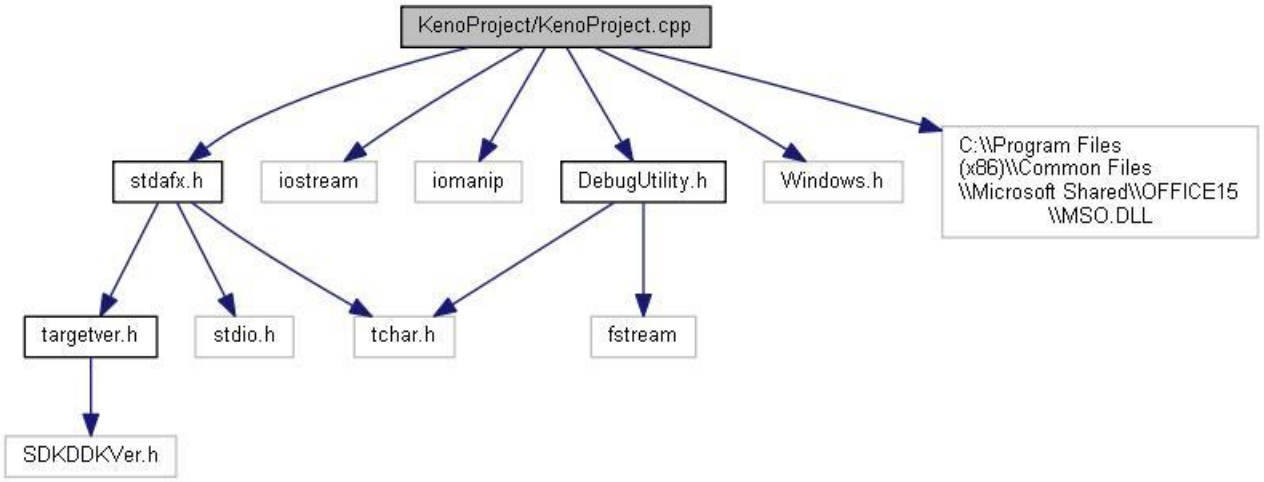
destination address NULL on error

KenoProject/KenoProject.cpp File Reference

Implementation of Keno Project.

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include "DebugUtility.h"
#include "Windows.h"
#import "C:\\Program Files (x86)\\Common Files\\Microsoft
Shared\\OFFICE15\\MSO.DLL"
```

Include dependency graph for KenoProject.cpp:



Typedefs

- typedef unsigned __int64 QWORD

Functions

- QWORD calcFactorial (WORD wN)
  calcFactorial
- double calcPartialFactorial (WORD wN, WORD wNumTerms)
  calcPartialFactorial
- DWORD calcCombinations (DWORD dwN, DWORD dwR)
  calcCombinations - "N things taken R at a time, without repetition"
- double calcKenoProbability (DWORD dwNumMarked, DWORD dwCaught)
  calcKenoProbability
- int ExportKenoProbabilityDataSheet (Excel::_WorkbookPtr &pWorkBook)
  Exports Keno Probability data to and Excel spreadsheet.
- int ExportKenoPayOutDataSheet (Excel::_WorkbookPtr &pWorkBook)
  Exports Expected Value data to an Excel spreadsheet.
- int ExportDataToExcel (void)
  ExportDataToExcel.
- int _tmain (int argc, _TCHAR *argv[])

Variables

- const int g_MAX_ROWS = 20
- const int g_MAX_COLS = 21
- const int g_TOTAL_BALLS = 80
- const int g_MAX_SELECTABLE_BALLS = 20
- double g_rgProbability [g_MAX_ROWS][g_MAX_COLS] = { 0.0 }
  The entries in the [ith] row assumes that the player has 'marked' i numbers The entry in the [jth] column is the probability that the player catches j spots out of i possible.
- const int g_MAX_PAYOUT_ROWS = 9
- const int g_MAX_PAYOUT_COLS = 9
- const int g_MAX_SPOTS_MARKED = 9
- const double g_rgCatchPayOut [g_MAX_PAYOUT_ROWS][g_MAX_PAYOUT_COLS]
- double g_rgExpectedValue [g_MAX_SPOTS_MARKED] = { 0.0 }

Detailed Description

Implementation of Keno Project.

Definition in file KenoProject.cpp.

Typedef Documentation

typedef unsigned __int64 QWORD

Function Documentation

int _tmain (int *argc*, _TCHAR * *argv*[])

Here is the call graph for this function:

DWORD calcCombinations (DWORD *dwN*, DWORD *dwR*)

calcCombinations - "N things taken R at a time, without repetition"

Combination is the quantity of subgroups of a size 'R' that can be formed out of a group of a size 'N' in which the order is NOT important. For example given 3 fruits (an apple, an orange and a pear), there are 3 combinations of 2 that can be drawn from this set: {apple, pear}, {apple, orange}, {pear, orange}. This expression is often written mathematically as C (N, R) where R is less than or equal to N, calculated as N! / R!(N-R)! and which is 0 when R > N.

Parameters:

in	*dwN*	Group Size - number of things to choose from
in	*dwR*	Subgroup Size - number of things chosen

Returns:

The number of 'dwR' sized subgroups that can be formed from a set containing 'dwN' number of elements.

See also:

http://en.wikipedia.org/wiki/Combination

QWORD calcFactorial (WORD *wN*)

calcFactorial

A recursive factorial implimentation

Parameters:

in	*wN*	value used for factorial operation

Returns:

computed results

Definition at line 91 of file KenoProject.cpp.

double calcKenoProbability (DWORD *dwNumMarked*, DWORD *dwCaught*)

calcKenoProbability

Calulates the probability of a 'dwCaught' sized catch from any set of 'dwNumMarked' number of player picked balls, based on a total of 80 KENO balls with the maximum number of balls selectable being 20.

Parameters:

in	*dwNumMarked*	The number of KENO ball spots a player has 'marked' or selected
in	*dwCaught*	The catch size of interest which probability is calculated against

Returns:

The calculated probability of a matching a subset of potential 'dwCaught' sized number of balls from a set consisting of 'dwNumMarked' number of spots from the possible 20 selectable balls.

Definition at line 194 of file KenoProject.cpp.

double calcPartialFactorial (WORD *wN*, WORD *wNumTerms*)

calcPartialFactorial

This performs a factorial computation of 'wN' utilizing only a 'wNumTerms' number of the highest valued terms of in traditional factorial computation.

For example, calcPartialFactorial(10, 4) would initiate a factorial computation, but would stop after evaluating the top 4 terms as follows: (10 * 9 * 7 * 6)

If 'wNumTerms' == 0, then a '1' is immediately returned.

Parameters:

in	*wN*	initial term
in	*wNumTerms*	number of terms used in partial factorial

Returns:

calculated partial factorial

int ExportDataToExcel (void)

ExportDataToExcel.

This method creates an instance of an Excel Component Object Model (COM) object and proceeds to export the computed data to an Excel spreadsheet.

int ExportKenoPayOutDataSheet (Excel::_WorkbookPtr & *pWorkBook*)

Exports Expected Value data to an Excel spreadsheet.

Parameters:

in	*pWorkBook*	A reference to an existing Excel Workbook Object

Definition at line 294 of file KenoProject.cpp.

References g_MAX_SPOTS_MARKED.

int ExportKenoProbabilityDataSheet (Excel::_WorkbookPtr & *pWorkBook*)

Exports Keno Probability data to and Excel spreadsheet.

Parameters:

in	*pWorkBook*	A reference to an existing Excel Workbook Object

Definition at line 233 of file KenoProject.cpp.

References g_MAX_COLS, and g_MAX_ROWS.

Variable Documentation

const int g_MAX_COLS = 21

Referenced by _tmain(), and ExportKenoProbabilityDataSheet().

const int g_MAX_PAYOUT_COLS = 9

const int g_MAX_PAYOUT_ROWS = 9

const int g_MAX_ROWS = 20

Referenced by _tmain(), and ExportKenoProbabilityDataSheet().

const int g_MAX_SELECTABLE_BALLS = 20

const int g_MAX_SPOTS_MARKED = 9

Referenced by _tmain(), and ExportKenoPayOutDataSheet().

const double g_rgCatchPayOut[g_MAX_PAYOUT_ROWS][g_MAX_PAYOUT_COLS]

```
Initial value:=
```

```
{ { 3.0,  0.0,  0.0,  0.0,    0.0,    0.0,    0.0,     0.0,     0.0 },
  { 0.0, 12.0,  0.0,  0.0,    0.0,    0.0,    0.0,     0.0,     0.0 },
  { 0.0,  1.0, 42.0,  0.0,    0.0,    0.0,    0.0,     0.0,     0.0 },
  { 0.0,  1.0,  3.0, 120.0,   0.0,    0.0,    0.0,     0.0,     0.0 },
  { 0.0,  0.0,  1.0,   9.0, 800.0,    0.0,    0.0,     0.0,     0.0 },
  { 0.0,  0.0,  1.0,   4.0,  88.0, 1500.0,    0.0,     0.0,     0.0 },
  { 0.0,  0.0,  0.0,   2.0,  20.0,  350.0,  700.0,     0.0,     0.0 },
  { 0.0,  0.0,  0.0,   0.0,   9.0,   90.0, 1500.0, 20000.0,     0.0 },
  { 0.0,  0.0,  0.0,   0.0,   4.0,   43.0, 3000.0,  4000.0, 25000.0 } }
```

Referenced by _tmain().

double g_rgExpectedValue[g_MAX_SPOTS_MARKED] = { 0.0 }

See also:

http://en.wikipedia.org/wiki/Expected_value

Referenced by _tmain().

double g_rgProbability[g_MAX_ROWS][g_MAX_COLS] = { 0.0 }

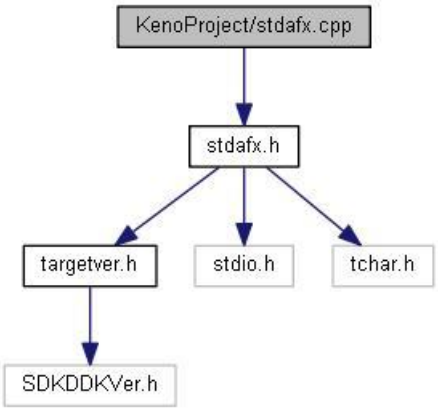
The entries in the [ith] row assumes that the player has 'marked' i numbers The entry in the [jth] column is the probability that the player catches j spots out of i possible.

Referenced by _tmain().

const int g_TOTAL_BALLS = 80

KenoProject/stdafx.cpp File Reference

source file that includes just the standard includes
`#include "stdafx.h"`
Include dependency graph for stdafx.cpp:



Detailed Description

source file that includes just the standard includes

KenoProject.pch will be the pre-compiled header stdafx.obj will contain the pre-compiled type information

Author:

Mark L. Short

Date:

October 1, 2014

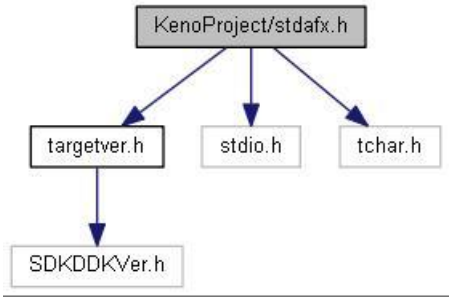
Definition in file stdafx.cpp.

KenoProject/stdafx.h File Reference

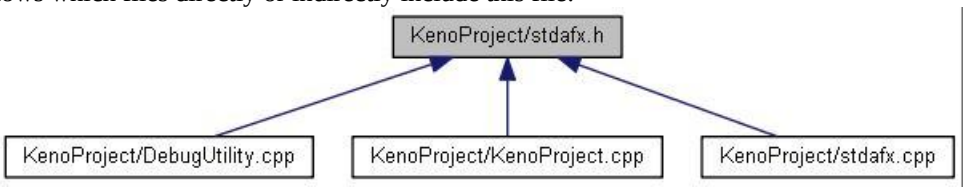
Application header file.

```
#include "targetver.h"
#include <stdio.h>
#include <tchar.h>
```

Include dependency graph for stdafx.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define _CRT_SECURE_NO_WARNINGS

Detailed Description

Application header file.

include file for standard system include files, or project specific include files that are used frequently, but are changed infrequently

Author:

Mark L. Short

Date:

Sept 19, 2014

Definition in file stdafx.h.

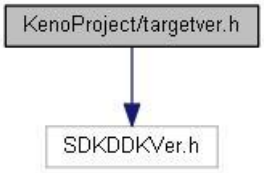
Macro Definition Documentation

#define _CRT_SECURE_NO_WARNINGS

KenoProject/targetver.h File Reference

```
#include <SDKDDKVer.h>
```

Include dependency graph for targetver.h:



This graph shows which files directly or indirectly include this file: