

General Functions

General Functions	
APNDYN	Append/Create dynamic array
ApnDyn	Append/Create dynamic array
mtd	convert 1-4(max) bytes of mem to data by order reversal of memory bytes
dtom	convert 1-4(max) bytes of data to memory by order reversal of data bytes
acopy	copy array Source->Destination
MakDir	Make directories and sub-directories if they don't exist
gline	get line from specified position in text file
Radix 10 Math Functions	
slice	# of Slices of destination number in source number
remn	Remainder of division
rnd	Rounded division(num & den have same sign)
roun	Round number to whole number
round	Round float number to specified # of decimal places
nti	Decrement number (limited to 0)
prs	Determine if number is greater than zero
fmax	Return larger number
low	Return smaller number
POW	Return m raised to power of n
fact	Return Factorial of a number
bmux	Select Source number if Select>0 else Destination number
leq	Determine if source number <= destination number
equ	Determine if numbers are equal
geq	Determine if source number >= destination number
odd	Determine if number is Odd
evn	Determine if number is Even
solve	Samultaneous equation solver
General Radix Math	
MSB	Index of Most Significant radix 2 digit in unsigned integer
rdx2	Determine if a number is radix 2
exc_r2	Number,if not Radix 2, to higher Radix 2 number
und_r2	Number,if not Radix 2, to lower Radix 2 number
rnd_r2	Number rounded closest Radix 2 number
nbt	Number of Radix 2 digits in an unsigned integer number
ndc	Number of Radix 10 digits in an integer number

Common Function Inputs

n Numeric Operand
m Numeric Operand
n1 Source Numeric Operand
n2 Destination Numeric Operand

Data Structures Used

```

struct TXFL{
    FILE      *hand;           //txt file structure(for gline)
    DWORD     bcnt;           //File Handle
    DWORD     lcnt;           //byte offset from where to get next line
    DWORD     lenl;           //Count of lines read since 'lcnt' made zero
    char      mark;           //line length
                                //if mark>0,anything, beyond & incl this mark are excluded
};
  
```

General Functions

APNDYN

<Data type> APNDYN(DWORD adp, <Data type> *src, <Data type>data)

Description

If the *<adp>* input is zero a dynamic array is Created otherwise it is appended. Data is appended by first creating a new array with space for *<data>* then copying data from *<src>* and *<data>* into it. *<src>* is then deleted and the address of the new array is returned.

Parameters

adp -Count of elements in array *<src[]>* before the call to APNDYN().

src -Address of array *src[]* before the call to APNDYN()

data -Data of type **<Data type>**

Note

<Data type> may be :FLDS, VALID, POINTI, POINTF, DWORD, double, char, void*, FOBJ, PEOB, ENTITY, PTHO, TNGT, PTHS and DBOX

Returns

Address of dynamic array *<src[]>* after appending *<data>*. The returned array must be deleted in the calling program

ApnDyn

<Data type> APNDYN(DWORD *adp, DWORD *udp, <Data type> *src, <Data type>data)

Description

If the count at *<adp>* is less than or equal to the count at *<udp>*, there is insufficient space in *<src[]>* to accomodate *<data>* which is then appended by first creating a new array with space for *<data>* then copying data from *<src>* and *<data>* into it. *<src>* is then deleted and the address of the new array is returned after incrementing the counts at *<adp>* and *<udp>*.

If the count at *<adp>* is greater than the count at *<udp>*, there is sufficient space in *<src[]>* to accomodate *<data>* which is then written to element *<src[udp]>*. The address of *<src[]>* is returned after incrementing the count at *<udp>*.

Parameters

adp -Address of a variable with the count of relevant and irrelevant elements in array *<src[]>* before the call to ApnDyn(). The variable is incremented if required within ApnDyn()

udp -Address of a variable with the count of relevant elements in array *<src[]>* before the call to ApnDyn(). The variable is incremented if required within ApnDyn()

src -Address of array *src[]* before the call to ApnDyn()

data -Data of type **<Data type>**

Note

<Data type> may be :POINTF, FOBJ and ENTITY.

Returns

Address of dynamic array *<src[]>* after appending *<data>*. The returned array must be deleted in the calling program

mtod

DWORD mtod(BYTE *str,DWORD cnt)

Description

convert 1-4(max) bytes of contiguous memory to numeric data by order reversal of memory bytes

Parameters

str -Array of memory bytes to convert into data

cnt -Number of bytes in *<str[]>*

Returns

Memory data in array converted to Numeric data.

dtom

void dtom(DWORD n, BYTE *str, DWORD cnt)

Description

convert 1-4(max) bytes of data to contiguous memory by order reversal of data bytes

Parameters

n -Data to be converted to memory bytes
str -Array of memory bytes converted from data
cnt -Byte width of <*n*> and <*str*[]>

Result

Data converted to contiguous memory bytes are placed in <*str*[]>.

acopy

#	Function Prototype	Description
1	char *acopy(char *dst, char *src, DWORD cnt, DWORD p, DWORD n)	copy array <i>src</i> []-> <i>dst</i> [] .Type 'char'
2	PTHO *acopy(PTHO *dst, PTHO *src, DWORD cnt, DWORD p, DWORD n)	copy array <i>src</i> []-> <i>dst</i> [] .Type PTHO. See PthFunc.doc for definition of PTHO

Parameters

src -Address of Source array to be copied to **specified**<*dst*[]> or **new destination array**
dst -Address of **specified destination array** into which Source array<*src*[]> is copied
cnt -Number of elements in source array<*src*[]> and **specified**<*dst*[]> or **new destination array**
n -Create a dynamic **new destination array** to hold the result.
p 0-*dst*[] not dynamic or dynamic but not deleted. *src*[] not dynamic or dynamic but not deleted.
1-*dst*[] is dynamic and deleted in function. *src*[] not dynamic or dynamic but not deleted.
2-*dst*[] not dynamic or dynamic but not deleted. *src*[] is dynamic and deleted in function.
3-*dst*[] is dynamic and deleted in function. *src*[] is dynamic and deleted in function.

Note

If <*p*>=1 or 3, *dst*[] is existant(not NULL), dynamic and deleted in function and a dynamic **new destination array** is created to hold the result. Otherwise, if <*n*>=1, the **specified destination array**<*dst*[]>, is unused or non-existent(NULL) and a dynamic **new destination array** is created to hold the result.

Returns

If <*p*>=1 or 3 or <*n*>=1, the Address of dynamic **new destination array**, used by the calling function to delete it. Otherwise address of the **specified destination array**<*dst*[]>.

MakDir

void MakDir(char *str)

Description

Make directories and sub-directories, if they don't exist.

Parameters

str -Address of Path name string

Example

```
Assuming program is run from d:\Robocup IP>
MakDir("c:\x\y\z\k"); //Make c:\x\y\z\k>
MakDir("\x\y\z\k"); //Make d:\x\y\z\k>
MakDir("\x\y.txt\z\k"); //Make d:\x>
MakDir("\x\y\z\k.txt"); //Make d:\x\y\z>
MakDir("x\y\z\k"); //Make d:\Robocup IP\x\y\z\k>
MakDir("x\y\z\k\"); //Make d:\Robocup IP\x\y\z\k>
```

gline

char *gline(TXFL *txt)

Description

Get line from text file. Tab characters are replaced w/ eight spaces & non-displayable characters omitted

Parameters

txt->hand -Handle of file to read
txt->bcnt -byte offset in file from where to get next line
txt->lcnt -Number of lines read before call to gline()
txt->mark -A character, after and including which, the read line is truncated. Ignored if 0

Result

txt->lenl -Char length of line read
txt->bcnt -byte offset in file from where to get next line
txt->lcnt -Incremented by one if atleast one byte was encountered before EOF

Returns

Address of dynamic **new result string**, containing the line read. Used by calling function to delete it.

Example

```
TXFL txt;  
char *xyz;
```

```
txt.hand=fopen(txt.sfnm,"rb+"); //File Handle  
txt.bcnt=0; //byte offset in file from where to get next line  
txt.lcnt=0; //Count of lines read since 'lcnt' made zero  
txt.mark=';';(or txt.mark=0;)  
  
while(txt.bcnt<=fsize(txt.hand)){ //Examine remaining bytes in file  
    xyz=gline(&txt); //txt->lenl contains Char length of line read  
    ... //txt->bcnt contains next byte to be read  
    ... //txt->lcnt is incremented if atleast 1 byte read  
    delete []xyz; //delete dynamic array after line is processed  
}
```

Radix 10 Math Functions

slice

DWORD slice(double n1,double n2)

Description

Number of Slices of Destination number(*n2*) in Source number(*n1*)

remn

double remn (double n1,double n2)

Description

Return Remainder of division of *n1* with *n2*

rnd

DWORD rnd(double n1,double n2)

Description

Rounded division of *n1* with *n2* (*n1* & *n2* have same sign)

roun

DWORD roun (double n)

Description

Round number(n) to nearest whole number

round

double round(double n1,DWORD n2)

Description

Round $n1$ to $n2$ decimal places

nti

DWORD nti (DWORD n)

Description

Return $n-1$ (limited to 0)

prs

DWORD prs (double n)

Description

Return 1 if number(n) is Greater than zero, else return 0

fmax

double fmax (double n1,double n2)

Description

Return larger number

low

double low (double n1,double n2)

Description

Return smaller number

POW

double POW (int m,DWORD n)

Description

Return m raised to power of n

fact

DWORD fact(DWORD n)

Description

Return Factorial of a number

bmux

DWORD bmux(DWORD n1,DWORD n2,DWORD s)

Description

Return Source number($n1$) if Select(s)>0 else Destination number($n2$)

leq

DWORD leq(DWORD n1,DWORD n2)

Description

Return 1 if Source number($n1$) is less than or equal to Destination number($n2$), else return 0

equ

DWORD equ(DWORD n1,DWORD n2)

Description

Return 1 if Source number($n1$) is equal to Destination number($n2$), else return 0

geq

DWORD geq(DWORD n1,DWORD n2)

Description

Return 1 if Source number($n1$) is greater than or equal to Destination number($n2$), else return 0

odd

DWORD odd(DWORD n)

Description

Return 1 number(n) is Odd else 0

evn

DWORD evn(DWORD n)

Description

Return 1 number(n) is Even else 0

solve

DWORD solve(double **a,DWORD n,double *x)

Description

Samultaneous equations are solved by Gaussian Elimination with Partial Pivoting. The process, **Elimination w/ Partial Pivoting** followed by **Back Substitution**, is described here w/ an example: -

$$\begin{aligned} 1x_0+1x_1+1x_2 &= 0 \\ 2x_0+1x_1+1x_2 &= 1 \\ 1x_0+2x_1+1x_2 &= 15 \end{aligned}$$

Elimination

The aim is to make all the elements below the diagonal zero. The first row above is multiplied by +2 and then subtracted from the second row to give a new set of equations (but with the same solution)

$$\begin{aligned} 1x_0+1x_1+1x_2 &= 0 \\ 0x_0-1x_1-1x_2 &= 1 \\ 1x_0+2x_1+1x_2 &= 15 \end{aligned}$$

The rows and factor chosen were designed to eliminate the first unknown x_0 from the second equation. If we now take -1 times the first row and add the last row we get

$$\begin{aligned} 1x_0+1x_1+1x_2 &= 0 \\ 0x_0-1x_1-1x_2 &= 1 \\ 0x_0+1x_1+0x_2 &= 15 \end{aligned}$$

And finally, adding the last two rows gives

$$\begin{aligned} 1x_0+1x_1+1x_2 &= 0 \\ 0x_0-1x_1-1x_2 &= 1 \\ 0x_0+0x_1-1x_2 &= 16 \end{aligned}$$

The division operation in 'elimination' may result in a divide by zero. The solution is to swap the rows around so that the divisor element has the largest magnitude(**Partial Pivoting**). If this element is still zero, all elements in the column below the element are zero then this reflects a "singular" matrix, that is, the n equations were not all independent and therefore there isn't a unique solution.

Back Substitution

Eqn3 gives $x_2 = -16$. Substituting into eqn2 gives $x_1 = 15$. Substituting x_1 & x_2 into eqn1 gives $x_0 = 1$.

Matrix Notation

A series of 'n' samultaneous equations with 'n' unknowns, in matrix notation are: -

$$\begin{pmatrix} a_{0,0} & a_{1,0} & a_{2,0} & \dots & a_{n-1,0} \\ a_{0,1} & a_{1,1} & a_{2,1} & \dots & a_{n-1,1} \\ a_{0,2} & a_{1,2} & a_{2,2} & \dots & a_{n-1,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{0,n-1} & a_{1,n-1} & a_{2,n-1} & \dots & a_{n-1,n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{pmatrix}$$

This is of the form: -

A x = b, where **A** is a matrix, **x** and **b** are vectors.

$$\begin{pmatrix} a_{0,0} & a_{1,0} & a_{2,0} & \dots & a_{n-1,0} \\ a_{0,1} & a_{1,1} & a_{2,1} & \dots & a_{n-1,1} \\ a_{0,2} & a_{1,2} & a_{2,2} & \dots & a_{n-1,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{0,n-1} & a_{1,n-1} & a_{2,n-1} & \dots & a_{n-1,n-1} \end{pmatrix} \begin{matrix} \boxed{\times} \\ \\ \\ \\ \end{matrix}$$

Placing vector **b** in the n^{th} column of the **A** matrix we get the modified **A** matrix

Parameters

- a* -Pointer to the modified **A** matrix, a 2D array of $(n+1)$ columns each w/ (n) row elements.
x -Pointer to the **x** vector, a 1D array of n result elements.
n -Number of unknowns & equations, # of rows & columns in the **A** matrix, # of rows in **x** vector.

Returns

- 0 -If the matrix **A** is "Singular" ie a unique solution doesn't exist or the value of an unknown exceeds the limit of a double precision float number.
1 -The equations were successfully solved.

Example

Given the equations:-

$$1x_0 + 1x_1 + 1x_2 = 0$$

$$2x_0 + 1x_1 + 1x_2 = 1$$

$$1x_0 + 2x_1 + 1x_2 = 15$$

The Modified **A** matrix is made from the coefficients of unknowns on the LHS of the equations and the numbers on the RHS. The **x** vector is an empty array to hold results and $n=3$. Thus: -

```
DWORD i,n=3;
double x[3]={0},**a;

a=new double*[n+1]; //Create modified A matrix(dynamic 2D array)
for (i=0;i<n+1;i++) a[i]=new double[n];

a[0][0] = 1; a[1][0] = 1; a[2][0] = 1; a[3][0] = 0; //Fill the modified A matrix:-
a[0][1] = 2; a[1][1] = 1; a[2][1] = 1; a[3][1] = 1; //a[col index][row index]
a[0][2] = 1; a[1][2] = 2; a[2][2] = 1; a[3][2] = 15;
solve(a,n,x); //Results: x[0]=1, x[1]=15, x[2]=-16
```

General Radix Math Functions

MSB

DWORD MSB(DWORD n)

Description

Return index of Most Significant Radix 2 digit in unsigned integer(n)

rdx2

DWORD rdx2(DWORD n)

Description

Determine if a number(n) is Radix 2

exc_r2

DWORD exc_r2(DWORD n)

Description

If number(n) is not Radix 2, return higher Radix 2 number, else return(n)

und_r2

DWORD und_r2(DWORD n)

Description

If number(n) is not Radix 2, return lower Radix 2 number, else return(n)

rnd_r2

DWORD rnd_r2(DWORD n)

Description

Return number(n) rounded closest Radix 2 number

nbt

DWORD nbt(DWORD n)

Description

Return number of Radix 2 digits in an unsigned integer number(n)

nbc

DWORD nbc(int n)

Description

Return number of Radix 10 digits in an integer number(n)