ファイル名　　LuDecomp.bas
モジュール名　　LuDecomp（標準モジュール）

サブルーチン名　`LuDcmp`

引数　`A() As Double, Indx() As Long, D as Long`

説明　行列 **A**（n 行 n 列）の LU 分解を行う。

A(n×n): `A(1 to n, 1 to n) as Double`

サブルーチンの実行の結果、`A()`は、**L** 行列と **U** 行列を組み合わせたもので上書きされる。`A()`の対各要素とその上側の要素には **U** 行列の要素が書き込まれ、残りには **L** 行列の要素が書き込まれる。**L** 行列の対各要素は 1 に規格化されているものとし、`A()`には書き込まれない。

`Indx(1 to n)`には、部分ピボット選択の際の行交換の履歴が戻る。ただし、`Indx()`は動的配列であること。`D` には、行交換の回数が偶数なら 1、奇数なら-1 が戻る。

`A()`に Singular Matrix が与えられた場合、エラーが発生する。

```
Err.Number = 4103
Err.Description = "Singular Matrix in LuDcmp"
```

キーワード　LU 分解

数学　行列 **A** を以下のように下三角行列 **L** と上三角行列 **U** の積で表現する。

$$\mathbf{L}\cdot\mathbf{U}=\mathbf{A}$$

例えば、4 行 4 列の場合、

$$\begin{bmatrix}\alpha_{11} & 0 & 0 & 0\\ \alpha_{21} & \alpha_{22} & 0 & 0\\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0\\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44}\end{bmatrix}\cdot\begin{bmatrix}\beta_{11} & \beta_{12} & \beta_{13} & \beta_{14}\\ 0 & \beta_{22} & \beta_{23} & \beta_{24}\\ 0 & 0 & \beta_{33} & \beta_{34}\\ 0 & 0 & 0 & \beta_{44}\end{bmatrix}=\begin{bmatrix}a_{11} & a_{12} & a_{13} & a_{14}\\ a_{21} & a_{22} & a_{23} & a_{24}\\ a_{31} & a_{32} & a_{33} & a_{34}\\ a_{41} & a_{42} & a_{43} & a_{44}\end{bmatrix}$$

となるが、任意に定めることができる要素が存在する。ここでは、下三角行列 **L** の対角要素α_{ii}はすべて 1 にできる。

サブルーチン名　LuBksb

引数　A() As Double, Indx() As Long, B() as Double

説明　LU 分解法により連立 1 次方程式 **Ax**=**b** の解を求める。ただし、**A** は n 行 n 列の行列、**b** は n 行のベクトルである。解 **x** も n 行のベクトルである。なお、**A** は LuDcmp で前もって LU 分解しておく。この時に生成された Indx()も必要である。

A(n×n)の LU 分解（LuDcmp で返されたもの）:

```
A(1 to n, 1 to n) as Double
```

LuDcmp が返した Indx():

```
Indx(1 to n) as Long
```

b(n):　`B(1 to n) as Double`

サブルーチンの実行の結果、A()と Indx()は、保存される。一方、B()は、解 **x** で上書きされる。

連立 1 次方程式が配列 A()と B()で与えられている場合、以下の手順で解を求める。

```
Dim A() as Double, B() as Double
Dim Indx() as long, D as long
      ....
LuDcmp A, Indx, B, D
LuBksb A, Indx, B
```

この結果 B()は解で上書きされる。A()は LuDcmp の結果で上書きされる。しかしながら、元の A()に対して、異なった B()で別の解を求めたい場合は、LuDcmp から返された A()と Indx()を用いて

```
LuBksb A, Indx, B
```

だけを実行すればよい。

キーワード　LU 分解、連立 1 次方程式、前進代入、後退代入

数学　行列 **A** を以下のように下三角行列 **L** と上三角行列 **U** に LU 分解されたとすると、方程式 :

$$\mathbf{A}\cdot\mathbf{x}=(\mathbf{L}\cdot\mathbf{U})\cdot\mathbf{x}=\mathbf{L}(\mathbf{U}\cdot\mathbf{x})=\mathbf{b}$$

は、

$\mathbf{L}\cdot\mathbf{y}=\mathbf{b}$　（前進代入）

$\mathbf{U}\cdot\mathbf{x}=\mathbf{y}$　（後退代入）

で解くことができる。異なった **b** に対して別の解を求める場合でも、LU 分解をやり直す必要はない。

サブルーチン名　`LuInv`

引数　`A() As Double, B() as Double`

説明　LU 分解法により行列 **A**（n 行 n 列）の逆行列 $\mathbf{A}^{-1}$（n 行 n 列）を求める。

A(n×n): `A(1 to n, 1 to n) as Double`

$\mathbf{A}^{-1}$(n×n): `B(1 to n) as Double`

サブルーチンの実行の結果、`A()`の逆行列が `B()`でもどる。なお、`B()`は動的配列でなければならない。`A()`は LU 分解の結果で上書きされる。

キーワード　LU 分解、逆行列

関数名　　`LuDet`

戻り値型　　`Double`

引数　　`A() As Double`

説明　　行列 **A**（n 行 n 列）の行列式（determinant）を LU 分解法で得られた下三角行列 **L** の対角要素から求め、戻り値として返す。

A(n×n): `A(1 to n, 1 to n) as Double`

実行の結果、`A()`は LU 分解の結果で上書きされる。

キーワード　　LU 分解、行列式

数学　　行列 **A**（n 行 n 列）が下三角行列 **L** と上三角行列 **U** に LU 分解されたとすると、行列 **A** の行列式 det は、上三角行列 **U** の対角要素を用いて、

$$\det \mathbf{A} = \prod_{j=1}^{n} \beta_{jj}$$

と表せる。ただし、下三角行列 **L** の対角要素は 1 とする。

サブルーチン名　Mprove

引数　A() As Double, Alud() As Double, Indx() As Long,
B() As Double, X() As Double

説明　連立 1 次方程式 **AX**=**b** の解 **x** を改良する。ただし、**A** は n 行 n 列の行列、**b** は n 行のベクトルである。解 **x** も n 行のベクトルである。

A(n×n): A(1 to n, 1 to n) as Double

A(n×n)の LU 分解（LuDcmp で返されたもの）:

Alud(1 to n, 1 to n) as Double

LuDcmp が返した Indx():

Indx(1 to n) as Long

b(n): B(1 to n) as Double

x(n): x(1 to n) as Double

A()には LU 分解前のオリジナルの行列を与える。LU 分解（LuDcmp による）後の行列は、Alud()に与える。LuDcmp が返した Indx()も必要である。また、改良前の解 x()も、別の方法であらかじめ（例えば LuBksb で）求めておく。サブルーチンの実行の結果、x()は、改良後の解で上書きされる。

解を求めるための演算回数は n^3 のオーダーであり、一方、解の改良に必要な演算回数は n^2 のオーダーであるため、改良を実施する価値は十分にある。

改良は、通常、Mprove を 1 回呼び出すだけで十分であるが、必要であれば、収束を監視しながら Mprove を複数回呼び出してもよい。

キーワード　LU 分解、連立 1 次方程式、解の改良