

=====GDS2_BIN2TXT_main.cpp=====

```

#include          <iostream>
#include          <fstream>
#include          <math.h>
using namespace std;
ifstream::pos_type
char             *memblock;
unsigned         Int_2u(   unsigned hi, unsigned lo );
int             Numb_Bytes(int hi, int lo );
int             Int_Sign4( int first, int second, int third, int forth );
double          Real_8(   int X0, int X1, int X2, int X3, int X4, int X5, int X6, int X7 );
string          CodeID(   int hi, int lo );
char            str2[60];
const char      *fname;

int             main (int argc, char * const argv[])
{ fname =       "TEST.SF";           //Copy a StreamFile and rename it.
  cout         <<"<< fname << " is the file being read \n" ;
  ifstream     file (fname, ios::in|ios::binary|ios::ate);
  if           (file.is_open())
  { getPointer = file.tellg();       //Get position of the get pointer
    memblock =  new char [getPointer];
    file.seekg ( 0, ios::beg);      // repositions get pointer to beginning
    getPointer = file.tellg();     //find new position of the get pointer
    file.read ( memblock, getPointer);

    string     ID_str, refl_str;
    int        numb, val , j , numbData;
    double     realVal;

    do
  { file.read(str2,2); numb = Numb_Bytes( str2[0], str2[1] ); //first read numb bytes in block
    file.read(str2,2); ID_str = CodeID( str2[0], str2[1] ); //read block ID
    numb =      numb - 4; //rest of bytes to read
    file.read(str2,numb); //read rest of bytes into str2

    if           ( ID_str == "HEADER Release # ")
  { val =       Int_2u( str2[0] , str2[1] ); //2_data_bytes=version
    cout         <<" " << ID_str << val << " \n" ;
  } if           ( ID_str == "BGNLIB")
  { cout        <<" " << ID_str << " " ; //data_bytes=dates
  } if           ( ID_str == "LIBNAME = ")
  { str2[numb]='\0'; // null terminate str //data_bytes=string
    cout         <<" " << ID_str << str2 << " " ;
  } if           ( ID_str == "FONTS = ")
  { str2[numb]='\0'; // null terminate str //data_bytes=string
    cout         <<" " << ID_str << str2 << " \n" ;
  } if           ( ID_str == "GENERATIONS")
  { val =       Int_2u( str2[0] , str2[1] ); //2_data_bytes=generation
  }
}

```

```

cout << ID_str << " " << val << " " ;
} if ( ID_str == "BGNSTR" ) //data_bytes=dates
{ cout << " " << ID_str << " " ;
} if ( ID_str == "STRNAME = " ) //data_bytes=string
{ str2[numb]='\0'; // null terminate str
cout << " " << ID_str << str2 << " \n" ;
} if ( ID_str == "BOUNDARY" ) //no data_bytes
{ cout << " " << ID_str << " " ;
} if ( ID_str == "LAYER = " ) //2_data_bytes=layer
{ val = Int_2u( str2[0] , str2[1] );
cout << " " << ID_str << val << " " ;
} if ( ID_str == "DATATYPE = " ) //2_data_bytes=datatype
{ val = Int_2u( str2[0] , str2[1] );
cout << " " << ID_str << val << " \n" ;
} if ( ID_str == "ENDEL" ) //no data_bytes
{ cout << " " << ID_str << " \n" ;
} if ( ID_str == "ENDSTR" ) //no data_bytes
{ cout << " " << ID_str << " \n" ;
} if ( ID_str == "ENDLIB" ) //no data_bytes
{ cout << " " << ID_str << " \n" ;
} if ( ID_str == "SREF" ) //no data_bytes
{ cout << " " << ID_str << " " ;
} if ( ID_str == "SNAME" ) //data_bytes=string
{ str2[numb]='\0'; // null terminate str
cout << " " << ID_str << " " << str2 << " " ;
} if ( ID_str == "STRANS" )
{ refl_str = "none";
val = Int_2u( str2[0] , str2[1] ); //2_data_bytes=reflect
if ( val != 0 ) refl_str = "reflect";
cout << " " << ID_str << " " << refl_str << " " ;
} if ( ID_str == "XY = " )
{ numbData = numb/4;
cout << " " << ID_str << " " ; //data_bytes(int_4) =positions
for (j= 0; j< numbData; j=j+1)
{ val = Int_Sign4( str2[0+4*j], str2[1+4*j], str2[2+4*j], str2[3+4*j] );
cout << " " << val << " " ;
} cout << "\n " ;
} if ( ID_str == "UNITS = " ) //data_bytes(real_8) =scale
{ realVal = Real_8(str2[0],str2[1],str2[2],str2[3],str2[4],str2[5],str2[6],str2[7] );
cout << " " << ID_str << " " << realVal << " " ;
realVal = Real_8(str2[8],str2[9],str2[10],str2[11],str2[12],str2[13],str2[14],str2[15] );
cout << realVal << " \n" ;
} if ( ID_str == "TEXT" ) //no data_bytes
{ cout << " " << ID_str << " " ;
} if ( ID_str == "TEXTTYPE" ) //2_data_bytes=texttype
{ val = Int_2u( str2[0] , str2[1] );
cout << " " << ID_str << " " << val << " " ;
} if ( ID_str == "PRESENTATION" ) //2_data_bytes=presentation
{ val = Int_2u( str2[0] , str2[1] );
cout << " " << ID_str << " " << val << " " ;
} if ( ID_str == "MAG" ) //data_bytes(real_8) =magnitude
{ realVal = Real_8(str2[0],str2[1],str2[2],str2[3],str2[4],str2[5],str2[6],str2[7] );

```

```

    cout << " " << ID_str << " " << realVal << "\n" ;
} if ( ID_str == "ANGLE" ) //data_bytes(real_8) =angle
{ realVal = Real_8(str2[0],str2[1],str2[2],str2[3],str2[4],str2[5],str2[6],str2[7] );
  cout << " " << ID_str << " " << realVal << " " ;
} if ( ID_str == "STRING" ) //data_bytes=string
{ str2[numb]='\0'; // null terminate str
  cout << " " << ID_str << " " << str2 << " \n " ;
} if ( ID_str == "PATHTYPE" ) //2_data_bytes=pathtype
{ val = Int_2u( str2[0] , str2[1] );
  cout << " " << ID_str << " " << val << " " ;
} if ( ID_str == "WIDTH" ) //data_bytes(int_4) =width
{ val = Int_Sign4( str2[0], str2[1], str2[2], str2[3] );
  cout << " " << ID_str << " " << val << " \n" ;
} if ( ID_str == "PATH" ) //no data_bytes
{ cout << " " << ID_str << " " ;
}

} while ( !file.eof()); /* do readfile until eof */
  file.close();

} else cout << "Unable to open " << fname << " \n"; //std::cout << "Hello, World!\n";
return 0;
}

```

```

//=====
string CodeID( int hi, int lo )
{ string str = "" ;
  if ( hi == 00 && lo == 02 ) str = "HEADER release # " ;
  if ( hi == 00 && lo == 02 ) str = "HEADER Release # " ;
  if ( hi == 01 && lo == 02 ) str = "BGNLIB"; // 0101
  if ( hi == 02 && lo == 06 ) str = "LIBNAME = "; // 0206
  if ( hi == 32 && lo == 06 ) str = "FONTS = "; // 2006
  if ( hi == 34 && lo == 02 ) str = "GENERATIONS"; // 2202
  if ( hi == 03 && lo == 05 ) str = "UNITS = "; // 0305
  if ( hi == 05 && lo == 02 ) str = "BGNSTR"; // 0502
  if ( hi == 06 && lo == 06 ) str = "STRNAME = "; // 0606
  if ( hi == 8 && lo == 00 ) str = "BOUNDARY"; // 0800
  if ( hi == 13 && lo == 02 ) str = "LAYER = "; // 0d02
  if ( hi == 14 && lo == 02 ) str = "DATATYPE = "; // 0e02
  if ( hi == 16 && lo == 03 ) str = "XY = "; // 1003
  if ( hi == 17 && lo == 0 ) str = "ENDEL"; // 1100
  if ( hi == 7 && lo == 0 ) str = "ENDSTR"; // 0700
  if ( hi == 4 && lo == 0 ) str = "ENDLIB"; // 0400
  if ( hi == 10 && lo == 0 ) str = "SREF"; // 0a00
  if ( hi == 18 && lo == 6 ) str = "SNAME"; // 1206
  if ( hi == 26 && lo == 1 ) str = "STRANS"; // 1a01
  if ( hi == 12 && lo == 0 ) str = "TEXT"; // 0c00
  if ( hi == 22 && lo == 2 ) str = "TEXTTYPE"; // 1602
  if ( hi == 23 && lo == 1 ) str = "PRESENTATION"; // 1701
  if ( hi == 27 && lo == 5 ) str = "MAG"; // 1b05
  if ( hi == 25 && lo == 6 ) str = "STRING"; // 1906
  if ( hi == 33 && lo == 2 ) str = "PATHTYPE"; // 2102
}

```

```

if          ( hi == 15 && lo == 3 )   str = "WIDTH";           // 0f03
if          ( hi == 9  && lo == 0 )   str = "PATH";             // 0900
if          ( hi == 28 && lo == 5 )   str = "ANGLE";           // 1C05
return str;
}

```

```

//=====
double      Real_8(int X0, int X1, int X2, int X3,int X4, int X5, int X6, int X7 )
{ if        ( X0 <0 ) X0 = X0+256;
  if        ( X1 <0 ) X1 = X1+256;
  if        ( X2 <0 ) X2 = X2+256;
  if        ( X3 <0 ) X3 = X3+256;
  if        ( X4 <0 ) X4 = X4+256;
  if        ( X5 <0 ) X5 = X5+256;
  if        ( X6 <0 ) X6 = X6+256;
  if        ( X7 <0 ) X7 = X7+256;
  int       exp , i ,bit,testbit, scalebit;
  exp       = (X0 & 127);           // will bit mask to = 01111111
  exp       = 0;
  for       (i=1; i< 8 ; i++)
{ bit      = (int) pow(2,7-i);
  exp       = exp + (X0 & bit);
} exp      = exp-65;               // add offset
  double    matval,matvalb , value;
  matval    = 0;
  for       (i=0; i< 8 ; i++)
{ bit      = pow(2,7-i);
  scalebit = pow(2,i);
  testbit  = (X1 & bit);
  matvalb  = (double) 8*(X1 & bit)/(bit*scalebit);
  matval   = matval + matvalb;
  matvalb  = (double) 8*(X2 & bit)/(256*bit*scalebit);
  matval   = matval + matvalb;
  matvalb  = (double) 8*(X3 & bit)/(256*256*bit*scalebit);
  matval   = matval + matvalb;
} value    = matval*pow(16,exp);
  return value;
}

```

```

//=====
int         Int_Sign4( int first, int second, int third, int forth )
{ int      numb ;
  if       ( second <0 ) second = second+256;
  if       ( third  <0 ) third  = third +256;
  if       ( forth  <0 ) forth  = forth +256;
  numb     = 256*(256*(256*first + second)+third )+forth;
  return numb;
}

```

```

//=====
int         Numb_Bytes( int hi, int lo )
{ int      numb ;

```

```
if ( hi < 0 ) hi = hi+256;
if ( lo < 0 ) lo = lo+256;
numb =256*hi + lo;
return numb;
}
```

```
//=====
unsigned Int_2u( unsigned hi, unsigned lo )
{ int numb ;
  numb =256*hi + lo;
  // cout <<hi<<" "<<lo<<" "<<numb<<" \n" ;
  return numb;
}
```