正　本

令和元年（ワ）第１０９４０号　損害賠償請求事件

原告　森　次　茂　廣

被告　株式会社[redacted]

第４準備書面

令和２年１１月９日

大阪地方裁判所第２６民事部合議係　御中

原告訴訟代理人弁護士　[redacted]

同　[redacted]

同　[redacted]

（担当）同　[redacted]

１　本件プログラム１の著作物性に関する主張の補充

本件プログラム１の全体に著作物性が認められることは，第２準備書面のとおりであるが，以下，データベースを操作するクラスである「clsAccessMdb.vb」のソースコード（第２準備書面別紙２の１頁乃至６頁参照）を例として具体的に述べる。

（１）　プログラムに選択の幅があること

ア　「clsAccessMdb.vb」のソースコードは，アクセスデータベースファイルの操作に関するものをクラス（オブジェクトを作る設計書）

として定義して記載したものである。

そして，「clsAccessMdb.vb」のクラスの最初の処理として，タイトルバーに表示する工事名称の取得処理が挙げられる。工事名称の取得処理の解法（アルゴリズム）としては，

①データベースをオープン

②検索命令を発する。

③検索結果からデータを取る。

④データベースをクローズ

が挙げられる。このアルゴリズムをプログラム言語及び規約で示すための選択肢の１つとして以下の＜構成１＞が考えられる。

＜構成１＞

```
メインルーチン
1: _con.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & FileName
2: _con.Open()
3: _sqlCommand = New OleDbCommand("select * from `ManlockInfo` WHERE `ID`=1", _con, _trn)
4: _adapter = New OleDbDataAdapter(_sqlCommand)
5: _adapter.Fill(_Table)
6: gManlockInfo.Title = _Table.Rows(0).Item("Title")
7: _con.Close()
```

処理解説

- 1～2のコードで、アルゴリズム①の「データベースをオープン」を行います。
- 3～5のコードで、アルゴリズム②の「検索命令を発する」を行います。
- 6のコードで、アルゴリズム③の「検索結果からデータを取る」を行います。
- 7のコードで、アルゴリズム④の「データベースをクローズ」を行います。

また，コードの解りやすさを重視して，以下の＜構成２＞のようなソースコードにする選択肢もあり得る。以下のソースコードは，オブジェクト化したもので，令文で並んだコードを処理毎（開く→取得→閉じる）に分かりやすく配列している。

＜構成２＞

```
メインルーチン
    1：dbOpen
    2：dbGetTitle()
    3：dbClose
サブルーチン
    4: Public Function dbOpen()
    5:   _con.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & FileName
    6:   _con.Open()
    7: End Function
    8: Public Function dbGetTitle()
    9:   _sqlCommand = New OleDbCommand("select * from `ManlockInfo` WHERE `ID`=1", _con, _trn)
   10:   _adapter = New OleDbDataAdapter(_sqlCommand)
   11:   _adapter.Fill(_Table)
   12:   gManlockInfo.Title = _Table.Rows(0).Item("Title")
   13: End Function
   14: Public Function dbClose()
   15:   _con.Close()
   16: End Function
処理解説
    処理を内容ごとに細分化し、動作を解り易く著しています。
    メインルーチンの1～3で、動作の流れを表記し、サブルーチンの4～16で、それぞれの動作を
    コードで記述しています。
    メインルーチンで、開く → 取得 → 閉じる と処理の流れが明確に表現されています。
    サブルーチンでは、構成1の処理解説と同様に詳細が記されています。
```

さらに，上記＜構成２＞をさらに解りやすくするために，以下の＜構成３＞のようなソースコードとする選択肢もある。＜構成３＞は，内部の処理に関わらず結果が正しければ良いという考えのもと処理をカプセル化し，目的のみを明示したものである。大きなプログラムでは，細部の処理にとらわれると全体を把握することができないことから，個々の処理を完結して，大きな流れを把握するために＜構成３＞のような記載をするという方法がある。

＜構成３＞

```
メインルーチン
   1: GetTitle()
サブルーチン
   2: Public Function GetTitle()
   3:   dbOpen
   4:   gManlockInfo.Title =dbGetTitle()
   5:   dbClose
   6: End Function
   7: Public Function dbOpen()
   8:   _con.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & FileName
   9:   _con.Open()
  10: End Function
  11: Public Function dbGetTitle()
  12:   _sqlCommand = New OleDbCommand("select * from `ManlockInfo` WHERE `ID`=1", _con, _trn)
  13:   _adapter = New OleDbDataAdapter(_sqlCommand)
  14:   _adapter.Fill(_Table)
  15:   gManlockInfo.Title = _Table.Rows(0).Item("Title")
  16: End Function
  17: Public Function dbClose()
  18:   _con.Close()
  19: End Function
```

処理解説
処理をカプセル化し、メインルーチンからは目的のみが解る様にした。
処理を階層化し、詳細はサブルーチンを探って見れるようにしています。

この他にも，処理スピード，使用メモリ，例外処理等の事情に合わせて，上記＜構成１＞ないし＜構成３＞とは別の記載方法でソースコードを表現することもできる。また個々のプログラマーによって，記載の方法は異なることから，「clsAccessMdb.vb」のソースコードだけでも膨大な選択肢があることは明らかである。

イ　このような膨大な選択肢の中，原告が選択したソースコードが第２準備書面別紙２の１頁乃至６頁のソースコード及び第２準備書面別紙２の２１頁１７行目～４０行目の部分である。

（２）　原告の表現上の創作性があふれていること

上記（１）アで述べたアルゴリズムのうち，①のアルゴリズムを

実現するためには，＜構成２＞の４乃至７のソースコードを記載するだけでも対応が可能である。もっとも，原告は，より精度の高い結果を実現するために，以下の＜原告が選択したソースコード＞を作成した。

＜原告が作成したソースコード＞

第２準備書面別紙２　１頁１３行目～３６行目

```
Public Function Connect(Optional ByVal dsn As String = "db1.mdb", _
                Optional ByVal tot As Integer = -1) As Boolean
  Try
    If _con Is Nothing Then
      _con = New OleDbConnection
    End If
    Dim cst As String = ""
    cst = cst & "Provider=Microsoft.Jet.OLEDB.4.0"
    cst = cst & ";Data Source=" & dsn
    ' データベースパスワードが設定されている場合
    ' cst = cst & ";Jet OLEDB:Database Password=xxxxx"
    If tot > -1 Then
      '_con.ConnectionTimeout = tot
      cst = cst & ";Connect Timeout=" & tot.ToString
    End If
    _con.ConnectionString = cst
    _con.Open()
    Connect = True
  Catch ex As Exception
    '     Throw New Exception("Connect Error", ex)
    Beep()
    Connect = False
  End Try
End Function
```

第２準備書面別紙２　２１頁１７行目～４０行目

```
Private Function dbOpen(ByVal vRecNo As Integer) As Boolean
  Dim mFileName As String
  mFileName = FileName
  If vRecNo >= 1 And vRecNo <= 10 Then
    mFileName = FileName & (vRecNo - 1).ToString("0")
```

```
    Else
      mFileName = FileName
    End If
    Return dbFileOpen(mFileName)
  End Function
  Private Function dbFileOpen(ByVal vFileName As String) As Boolean
    Dim mOpenFlag As Boolean, mFileName As String
    Dim libFF As New clsFolderFile
    If libFF.ExistsFile(vFileName) = False Then Return False
    Try
      mFileName = vFileName
      pvDB = New clsAccessMdb
      mOpenFlag = pvDB.Connect(mFileName)
      Return mOpenFlag
    Catch
      pvDB = Nothing
      Return False
    End Try
  End Function
```

処理解説

データベースの基本的な命令に、例外処理を加えています。

巨大なデータを扱うため、ファイルの切り替えに対応しています。

上記の４０行を超える大量のソースコードで，原告は，①のアルゴリズムを実現するために，ファイルの正当性のチェックや，様々なアクセスに対応するためのファイルの切り替えの処理を行えるようにソースコードを組み合わせている。

加えて，＜原告が選択したソースコード＞に複数記載のある「Ｔｒｙ～」，「Ｃａｔｃｈ～」の箇所で，原告は，例外処理を行えるようにソースコードを組み合わせている。具体的には，低レベルのファイル操作を行うにあたって，エラーが発生した場合は，それが致命的なエラーとなり，プログラムを停止することを余儀なくされることから，このような事態を回避するために，原告は上記「Ｔｒｙ～」，「Ｃａｔｃｈ～」の箇所でエラー処理を行っている。エラー処理は，システムに任せるという方法も取れるところ，原告は，独自の考えから，プログラムでエラー処理ができるようにソースコ

ードを組み合わせている。

以上より，原告は，①を実現するために，ファイルの正当性チェック，ファイルの切り替え，エラー処理等の複雑な機能を実現するソースコードを組み合わせており，これらについて原告の独自の考えによる組み合わせを織り交ぜていることから，原告の個性が発揮されていることは明らかである。

よって，創作性が認められる。

（３）　小括

第２準備書面別紙２の１頁から始まる「clsAccessMdb.vb」のソースコードだけでも，上記のように多様な選択肢の中から，原告の個性が発揮されたソースコードが構成されている。原告は，その他のクラスについても，同様の方法でソースコードを作成していることから，本件プログラム１全体に選択肢の幅があり，原告の表現上の創作性が表れているといえる。

よって，本件プログラム１に著作物性が認められる。

２　本件プログラム３の著作物性

（１）　本件プログラム３の概要

本件プログラム３は，工事現場等で常時騒音及び振動を測定し，規定以上の騒音及び振動が発生していないか監視するプログラムである。具体的には，騒音計及び振動計から発信されるアナログ信号をリアルタイムに表示して警戒値の判定を行い，騒音及び振動が警戒値を超えた場合は，指定時間に遡ってデータ収録を行う機能が備えられている。

そして，騒音計及び振動計から発信されるアナログ信号の入力処

理については，コンテック社のアナログ入力ボード（以下，「ＡＤＣ」という）を制御することによって行われている。

（２）　ソースコードの構造について

ア　ＡＤＣ制御のアルゴリズムは，①初期化，②設定，③変換開始，④変換終了，⑤データ取得，⑥データ保存というように行うことが一般的であるため，ソースコードについても上記①～⑥を前提として作成されるのが一般的である。

もっとも，本件プログラム３においては，騒音計及び振動計から無限にサンプリングを行うことが予定されているため，上記⑤，⑥を無限に繰り返す処理が必要であった。そのため，何かのタイミングでデータを取得するようにして，メモリーオーバーのエラーの発生を防止することや，処理時間を短縮するために取得したデータをその都度処理することができるようにソースコードを作成しなければならなかった。したがって，原告は，独自の処理ルーチンを作成して，これらに対応できるようにソースコードを工夫している。以下，詳述する。

イ　データ取得処理

例えば，継続的に何かのタイミングでデータ取得処理をするソースコードを作成する場合，ボードメーカーから割込信号が発せられたタイミングでデータを取得する方法を選択して，以下の＜構成１＞のソースコードを選択することもできる。

＜構成１＞

```
'-----------------------------------------------------------------------------
  フォームのロード
'-----------------------------------------------------------------------------
Private Sub Ai_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles MyBase.Load
        '
        '
        '
  'イベント通知用デリゲートの作成
  pdelegate_func = New PAICALLBACK(AddressOf CallBackProc)
   デリゲートが解放されないように設定します.
  If gCh.IsAllocated = False Then
    gCh = GCHandle.Alloc(pdelegate_func)
  End If
End Sub

'-----------------------------------------------------
 コールバックルーチン
'-----------------------------------------------------
Protected Sub CallBackProc(ByVal lc As Short, ByVal Message As Short, ByVal wParam As Integer, ByVal lParam As Integer, ByVal Param As IntPtr)
  Dim I As Integer
  Dim AiSamplingCount As Integer
  If Message = AIOM_AIE_END Then
    AiSamplingCount = lParam
    TextString = "デバイス動作終了イベント発生"
        '
        '
        '
    Delegate処理，サンプリングしたデータを上方します.
    Invoke(New SetDataDelegate(AddressOf SetDataText), New Object() {TextString})
  End If
End Sub

Private Sub AiCall_FormClosed(ByVal sender As System.Object, ByVal e As System.Windows.Forms.FormClosedEventArgs) Handles MyBase.FormClosed
  'デバイスの終了処理
  Ret = AioExit(lDid)
   取り込み処理用デリゲートのガベージコレクションハンドルを開放
  If gCh.IsAllocated = True Then
    gCh.Free()
  End If
End Sub
```

上記の他にも，何を重要視するかという点（速度や最適化等）や，プログラマーとしてのスタイル等により，ソースコードの選択肢は多岐にわたる。

本件では，原告は，無限にデータを取得する過程で，収録したデータと時刻を結びつける際に誤差が生じ，リアルタイム性に欠ける懸念して，多様な選択肢の中から，以下のソースコード（原告第３準備書面　別紙３　５９頁３２行目以下）を作成した。

＜原告が作成したソースコード１＞

```
Private Sub Timer1_Timer()
  Dim Ret As Integer
  Dim ii As Integer, jj As Integer, DataCompare As Long
  Dim a As Double, kk As Integer, Volt As Double
  Dim DataTime As Date
  GetAiStatus_Click

  If AcxAio1.GetAiStatus <> 1 Then
    cmdRun_Click
  End If
  Dim ScanN As Long, aaa As Long
  AcxAio1.GetScanNumber ScanN
  prbAdBuf.Value = ScanN
  If ScanN >= 1000 Then
    Ret = AcxAio1.GetAiData(1000, Buffer)
    For jj = 0 To 999
      For ii = 0 To gMaxChNo
        DataSec(jj, ii) = ConvInt(Buffer(jj * (gMaxChNo + 1) + ii))
      Next
    Next
    DataTime = (Date + Time) - Int(ScanN / 1000) / 86400
    MstData.PutbData DataSec, 1000, DataTime
    For ii = 0 To gMaxChNo
      Volt = ConvData(DataSec(0, ii), ii)
      lblAdData(ii) = Right("      " + Format(Volt, "##0.0000"), 8) & "V"
      prbAdData(ii).Value = DataSec(0, ii) + 32768
    Next
  End If
  If ScanN >= 10000 Then
    Me.Show
    If Me.WindowState = vbMinimized Then
      Me.WindowState = vbNormal
    End If
  End If

  DispStatus

  txtActMode = MstData.ActModeStr
  If Format(Time, "hh:ss") = "00:00" Then
    ZCalender1.Year = Format(Date, "yyyy")
    ZCalender1.Month = Format(Date, "mm")
    ZCalender1.Day = Format(Date, "dd")
    ZCalender1.Refresh
  End If
End Sub
```

上記のソースコードで，原告は，タイマーイベントを利用して，プログラム側で定期的にメモリ状態を監視し，取得処理を行うという複雑な機能を実現しており，原告の個性が発揮されているといえる。

ウ　データファイル収録処理について

ファイルの出力は，ハードディスクのシークタイムが重要な要素となり，少ないデータを頻繁に書き込むと，シークタイムが嵩張り，かつ多くのデータを一度に書き込むと他の処理に遅延が生じ，障害が発生する。

そのため，原告は，経験に基づく独自の考えから，シークタイムを考慮した最適な度合を検討し，データファイルの収録処理に関して，以下のソースコードを作成した。

＜原告が作成したソースコード２＞

原告第３準備書面　別紙３　１９頁　９行目～２０頁　４行目

```
Public Sub RecStart(ByVal vStartTime As Date)
 Dim pos As Long
 CreateFileName vStartTime
' FileOpen
 HeadData.観測開始_年 = CInt(Format(vStartTime, "yyyy"))
 HeadData.観測開始_月 = CByte(Format(vStartTime, "mm"))
 HeadData.観測開始_日 = CByte(Format(vStartTime, "dd"))
 HeadData.観測開始_時 = CByte(Format(vStartTime, "hh"))
 HeadData.観測開始_分 = CByte(Format(vStartTime, "nn"))
 HeadData.観測開始_秒 = CByte(Format(vStartTime, "ss"))

 HeadData.観測に装着号 = MeasPosNo

 RecDataN = 0
 AppendLog "観測開始"
 Grig_AppendStatus "観測開始"
 ActMode = eMode.収録中
 For pos = DataPoint - 2000 To DataPoint - 1
  RecDataPut pos
 Next

End Sub
Private Sub RecDataPut(vPos As Long)
 Dim DimPos As Long, i As Integer
 DimPos = (vPos + DataBufSize) Mod DataBufSize
 For i = 0 To AllInpChN - 1
  DataLine(RecDataN, i) = DataADI(DimPos, i)
 Next
 RecDataN = RecDataN + 1
 If RecDataN = RecAllTime Then
  RecEnd
End Sub

Public Sub RecEnd()
 DataFilePut
 AppendLog "観測終了"
 Grig_AppendStatus "観測終了"
 ActMode = eMode.起動中
 DataFileListUpdate
 If MoveFileN >= 100 Then
  MoveFile(MoveFileN) = mvarFileName
  MoveFileN = MoveFileN + 1
 End If
End Sub
Public Function PathData(rData() As Integer, vCount As Long, vTime As Date) As Boolean
 Dim i As Integer, j As Integer, Offset As Long, mMeasTimeFlag As Boolean
 Dim mData As Integer
```

上記のソースコードのヘッダ部の構造体の要素名に，「観測開始_＊」というような記載がある。一般的に，漢字を変数名等に使用す

ることは推奨されていないが，原告はソースコード上でプログラム設計書の役割も果たすという方針で，上記のような記載としている。

また，観測開始時刻を１つの変数で管理するというソースコードの記載とする選択肢もある中で，原告はこれまでの経験に基づいて年，月，日，時，分，秒をそれぞれ一つの変数に分けて管理することを選択し，それを第３準備書面　別紙３　１９頁の１３行目から１８行目の部分で表現している。

したがって，上記ソースコードには，変数名の記載方法や，変数の管理等について，プログラマー毎の方針（何を重視するか，どこまで許容するか）に従って多様な選択肢があるといえる。

そして，原告は，当時被告が利用していたパソコン（ハードディスク）のスペックも考慮した上で，独自に設計したデータフォーマットに合わせてデータを書き込む形で上記ソースコードを作成していることから，上記ソースコードは，ありふれたものではなく，原告の個性が発揮されている。

エ　処理速度の向上について

原告は，本件プログラム３について，連続的に長時間安定した処理ができることを目的として，個々の処理を細分化し，同時実行に見せかけるようタイムシェアリングを行っている。このような同時実行に見せかけるための要素として，処理速度の向上が挙げられる。

処理速度は，実数値演算よりも整数値演算の方が早いという性質があることから，原告は，独自の考えに基づき，整数値演算の関数を作成し，以下のソースコードを作成した。

＜原告が作成したソースコード３＞

原告第３準備書面　別紙３　１頁　４行目以下

```
Public Const AD_MaxVolt = 10    'バイポーラ -10～+10V

Public Function ConvInt(vinData As Single) As Integer
 If vinData >= 10 Or vinData <= -10 Then
  MsgBox "電圧が範囲外", vbOKOnly
 End If
 If vinData >= 10 Then vinData = 10
 If vinData <= -10 Then vinData = -10
 ConvInt = vinData * 3276.7 'vinData - IIf(vinData >= 32767, 65536, 0)
End Function

Public Function ConvData(vinData As Integer, vChNo As Integer) As Single
          .
          .
          .
```

上記のソースコードは，整数値演算の点を重視して作成されているところ，実数値演算にするか否かという点や関数の式の組立方法等について，多様な選択肢があり，個々のプログラマーによって，ソースコードの記載方法は異なる。

そして，原告は，上記のソースコードにおいて，独自の考えに基づき，入力される電圧値（実数値）を整数値に変換して，その逆を行う関数を定義していることから，精度を維持したまま処理を高速化することが実現できるようになっている。

加えて，原告は，上記演算自体も極力少なくするために，トリガー判定の対象値の演算をトリガー判定時ではなく，設定時に行うようにソースコードを作成した。これによって，全体の演算回数が減少し，処理速度が向上している。

以上より，＜原告が作成したソースコード３＞には，選択の幅があり，原告の独自の考えに基づく演算等が表現されているため，原告の個性が発揮されているといえる。

オ　小括

上記イ乃至エにより，本件プログラム３は，全体的に選択肢の幅があり，原告の独自の考えや経験等に基づく表現上の創作性が表れ

ているといえる。

よって，本件プログラム３に著作物性が認められる。

３　サイレントロボについて

被告は，サイレントロボを制作したのは本件プログラム３よりも前の平成１５年１０月であったため，サイレントロボは本件プログラム３を基礎にしていないと主張する（被告準備書面１　第３参照）。

しかしながら，本件プログラム３はそもそも平成１４年６月２４日に納品した高山トンネル工事の振動騒音測定プログラムを改良して作成した志津見ダムの発破振動測定プログラム（平成１５年７月３１日納品）を更に改良して作成したものである。そのため，本件プログラム３と高山トンネル工事の振動騒音測定プログラム及び志津見ダムの発破振動測定プログラムのソースコードは，多くの部分で共通している（甲２５）。

したがって，サイレントロボによって，原告の同一性保持権は侵害されている。なお，仮に被告の主張するとおり，サイレントロボが平成１５年１０月に制作されているのであれば，原告が被告に納品した高山トンネル工事の振動騒音測定プログラム又は志津見ダムの発破振動測定プログラムのソースコードがサイレントロボに用いられていると考えるのが自然である（甲２６－１，２６－２）。

被告は，サイレントロボによって著作権侵害がないというのであれば，平成１５年１０月当時のサイレントロボのソースコードを開示されたい。

以上